

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Уральский федеральный университет имени первого Президента России  
Б. Н. Ельцина»

УТВЕРЖДАЮ

Директор по образовательной деятельности

  
« 7 » сентября 2023 г.



**Аналитика и визуализация данных**  
Учебно-методические материалы по направлению подготовки  
**09.03.03 Прикладная информатика**  
Образовательная программа «Прикладной искусственный интеллект»

Екатеринбург


## РАЗРАБОТЧИКИ УЧЕБНО-МЕТОДИЧЕСКИХ МАТЕРИАЛОВ

Доцент Базовой кафедры  
«Аналитика больших данных и  
методы видеоанализа»



М.А. Медведева

Ассистент Базовой кафедры  
«Аналитика больших данных и  
методы видеоанализа»



Д. Балунгу

## СОДЕРЖАНИЕ

1.	Введение в природу данных (большие данные) .....	5
1.1	Концепция и природа данных.....	5
1.2	Хранение данных .....	6
1.3.	Источники данных для аналитики .....	7
1.5	5 V больших данных.....	8
1.6	Жизненный цикл аналитики данных .....	10
1.7	Качественные, количественные и смешанные методы.....	13
2	Набор инструментов аналитиков данных.....	15
2.1	Бизнес-аналитика .....	15
2.2	Введение в науку о данных .....	16
2.3	Управление версиями с помощью Git и GitHub .....	17
3	Основы аналитики и визуализации с помощью Power BI .....	20
3.1	Общие сведения о Power BI .....	20
3.2	Моделирование и преобразования данных (DAX).....	21
3.3	Визуализация данных и отчеты Power BI.....	22
3.4	Обмен данными и совместная работа .....	23
5	Аналитика данных с помощью Python .....	25
5.1	Python (структура данных, функции, списки, словари и т. д.).....	25
5.2	Манипулирование данными и их анализ с помощью Pandas.....	26
5.3	Предварительная обработка данных.....	27
5.4	Визуализация данных с помощью Matplotlib и Seaborn .....	28
5.5	Статистический анализ с помощью NumPy и SciPy .....	29

Практическое занятие 1. Установка программной среды PYTHON, ПАКЕТОВ NUMPY, PANDAS, IPYTHON .....	32
Практическое занятие 2. Предварительная обработка данных в Pandas .....	40
Практическое занятие 3. Первичный анализ данных в PANDAS.....	51
Практическое занятие 4. Визуализация данных с MATPLOTLIB, SEABORN .....	58
Практическое занятие 5. Диаграммы в SEABORN .....	72
Практическое занятие 6. Парсинг и анализ данных из интернета...	103
Практическое занятие 7. Работа с данными в Power BI Desktop .....	110
Практическое занятие 8. Визуализация данных в Power BI Desktop .....	132
Контрольная работа .....	146
Домашняя работа .....	150
Зачет .....	152



## 1. Введение в природу данных (большие данные)

### 1.1 Концепция и природа данных

Данные — это набор фактов, цифр и статистических данных, которые используются для представления информации. Данные могут быть в различных формах, включая числа, текст, изображения, аудио, видео и многое другое. Данные необходимы для принятия обоснованных решений, выявления тенденций и понимания закономерностей. Данные можно разделить на два типа: структурированные и неструктурированные. Структурированные данные организованы и отформатированы определенным образом, что упрощает поиск, сортировку и анализ. Примерами структурированных данных являются электронные таблицы, базы данных и таблицы. С другой стороны, неструктурированные данные никак не организованы, что затрудняет поиск и анализ. Примеры неструктурированных данных включают электронные письма, сообщения в социальных сетях и изображения. Данные также можно разделить на два типа в зависимости от их источника: первичные и вторичные данные. Первичные данные собираются непосредственно из их источника, в то время как вторичные данные собираются из других источников, таких как книги, статьи и отчеты. Первичные данные могут быть собраны с помощью опросов, экспериментов, наблюдений и интервью.

Природа данных постоянно развивается, данные становятся все больше, сложнее и разнообразнее. Большие данные относятся к наборам данных, которые слишком велики и сложны, чтобы их можно было обрабатывать традиционными методами обработки данных. Развитие Интернета вещей (IoT) привело к созданию большего количества данных, чем когда-либо прежде, с датчиками и устройствами, собирающими данные в режиме реального времени. Эти данные можно анализировать, чтобы получить представление о поведении потребителей, характеристиках продукта и тенденциях рынка.

Вот некоторые дополнительные сведения о концепции и характере данных:

- Данные могут быть качественными или количественными: качественные данные — это нечисловые данные, носящие описательный характер, такие как текст или изображения. Количественные данные, с другой стороны, представляют собой числовые данные, которые можно измерить и проанализировать математически.
- Данные могут быть необработанными или обработанными: необработанные данные — это необработанные данные, которые не были проанализированы или организованы, в то время как обработанные данные — это данные, которые были проанализированы, организованы и представлены осмысленным образом.
- Данные могут быть статическими или динамическими: статические данные фиксированы и не изменяются, в то время как динамические данные изменяются со временем. Например, цены на акции — это динамические данные, которые меняются в режиме реального времени, в то время как список сотрудников в компании — это статические данные, которые могут время от времени меняться.
- Данные могут быть публичными или частными: общедоступные данные доступны каждому, в то время как личные данные являются конфиденциальными и могут быть доступны только уполномоченным лицам.
- Данные можно использовать для различных целей, включая описательную, диагностическую, прогнозную и предписывающую аналитику. Описательная аналитика включает в себя анализ исторических данных, чтобы понять, что произошло. Диагностическая аналитика включает в себя анализ данных, чтобы

понять, почему что-то произошло. Предиктивная аналитика включает в себя использование данных для прогнозирования того, что может произойти в будущем. Предписывающая аналитика включает в себя использование данных, чтобы рекомендовать или предписывать курс действий.

- Качество данных имеет решающее значение для точного анализа и принятия решений. Низкое качество данных может привести к неточным результатам и ошибочному принятию решений. Качество данных можно улучшить, обеспечив точность, полноту, согласованность и своевременность данных.
- Управление данными — еще один важный аспект управления данными. Управление данными включает в себя управление активами данных, включая политики, процессы и стандарты управления данными. Это гарантирует, что данные используются этично, безопасно и в соответствии с нормативными требованиями.
- Визуализация данных является мощным инструментом для представления и передачи данных. Он включает в себя создание визуальных представлений данных, таких как диаграммы, графики и карты, чтобы помочь пользователям понять и интерпретировать данные.
- Аналитика данных — это процесс анализа и интерпретации данных для получения информации и принятия обоснованных решений. Он включает в себя использование инструментов и методов, таких как статистический анализ, машинное обучение и интеллектуальный анализ данных, для извлечения информации из данных.

Таким образом, данные представляют собой набор фактов и цифр, которые можно использовать для представления информации. Его можно разделить на структурированные и неструктурированные данные, а также первичные и вторичные данные. Природа данных постоянно развивается, а рост больших данных и Интернета вещей приводит к созданию большего количества данных, чем когда-либо прежде. Понимание концепции и характера данных необходимо предприятиям и организациям для принятия обоснованных решений и получения информации о своей деятельности.

## 1.2 Хранение данных

Хранение данных относится к процессу хранения данных в структурированном и организованном виде для облегчения поиска и анализа. Существуют различные типы технологий хранения данных, начиная от традиционного файлового хранилища и заканчивая облачными решениями для хранения.

- Традиционное файловое хранилище: Этот тип хранилища предполагает хранение данных в файлах в локальной или сетевой файловой системе, такой как жесткий диск, флэш-накопитель или общее сетевое хранилище. Этот тип хранилища подходит для небольших нужд хранения, но по мере увеличения объема данных им может стать трудно управлять.
- Реляционные базы данных: Этот тип хранилища предполагает хранение данных в таблицах, где каждая таблица представляет определенную сущность, такую как клиенты, продукты или транзакции. Реляционные базы данных широко используются на предприятиях и в организациях из-за их способности хранить большие объемы структурированных данных и управлять ими.
- Нереляционные базы данных: Нереляционные базы данных, также известные как базы данных NoSQL, предназначены для обработки неструктурированных

данных, таких как сообщения в социальных сетях, изображения и видео. Эти базы данных обладают высокой масштабируемостью и могут обрабатывать большие объемы данных.

- Хранилища данных: Хранилища данных предназначены для хранения больших объемов данных из различных источников и поддержки аналитических запросов. Они часто используются в бизнес-аналитике, отчетности и аналитике.
- Облачное хранилище: Облачное хранилище включает в себя хранение данных в удаленном месте, к которому можно получить доступ через Интернет. Поставщики облачных хранилищ, такие как Amazon Web Services (AWS), Microsoft Azure и Google Cloud, предлагают масштабируемые решения для хранения данных, которые можно настроить в соответствии с конкретными потребностями предприятий и организаций. Облачное хранилище отличается высокой гибкостью, и к нему можно получить доступ из любого места, что делает его идеальным решением для удаленной работы.
- Объектное хранилище: Объектное хранилище — это тип хранилища, в котором данные хранятся в виде объектов, а не файлов. Каждый объект содержит как данные, так и метаданные, описывающие данные. Этот тип хранилища обладает высокой масштабируемостью и может использоваться для хранения больших объемов неструктурированных данных, таких как изображения, видео и сообщения в социальных сетях.
- Озера данных: озера данных предназначены для хранения больших объемов данных из различных источников, включая структурированные и неструктурированные данные. Они обладают высокой масштабируемостью и могут обрабатывать большие объемы данных. Озера данных часто используются в аналитике больших данных и машинном обучении.

Таким образом, хранение данных является важнейшим компонентом управления данными. Существуют различные типы технологий хранения данных, начиная от традиционного файлового хранилища и заканчивая облачными решениями для хранения. Выбор технологии хранения данных зависит от объема, типа и структуры данных, а также конкретных потребностей организации. Эффективно управляя своими решениями для хранения данных, предприятия и организации могут обеспечить безопасность, доступность и пригодность своих данных для анализа и принятия решений.

### 1.3. Источники данных для аналитики

Источники данных для аналитики относятся к различным типам данных, которые можно использовать для анализа и аналитических сведений. Вот некоторые распространенные источники данных для аналитики:

- Базы данных: Базы данных являются одним из наиболее распространенных источников данных для аналитики. К ним могут относиться базы данных SQL, базы данных NoSQL и хранилища данных. Базы данных могут хранить структурированные, полуструктурированные и неструктурированные данные и используются для хранения больших объемов данных в структурированном виде.
- Электронные таблицы: Электронные таблицы, такие как Microsoft Excel или Google Sheets, обычно используются для хранения и анализа данных. Их можно использовать для небольших задач анализа данных, таких как вычисление средних значений или создание простых диаграмм.

- Облачные сервисы: Облачные сервисы, такие как Amazon Web Services (AWS), Google Cloud и Microsoft Azure, предоставляют решения для хранения данных и аналитики, к которым можно получить удаленный доступ. Эти сервисы можно использовать для хранения и анализа больших объемов данных масштабируемым и экономичным способом.
- Поточковые данные: Поточковые данные относятся к данным, которые генерируются в режиме реального времени, таким как каналы социальных сетей, данные датчиков и журналы сервера. Этот тип данных можно анализировать в режиме реального времени с помощью таких инструментов, как Apache Kafka или Apache Spark Streaming.
- Социальные сети: Платформы социальных сетей, такие как Twitter, Facebook и Instagram, генерируют большие объемы данных, которые можно использовать для аналитики. Эти данные можно использовать для отслеживания настроений клиентов, мониторинга репутации бренда и выявления тенденций и закономерностей.
- Устройства Интернета вещей (IoT): устройства IoT, такие как датчики, носимые устройства и устройства умного дома, генерируют огромные объемы данных, которые можно использовать для аналитики. Эти данные можно использовать для профилактического обслуживания, удаленного мониторинга и других приложений.
- Веб-аналитика: Инструменты веб-аналитики, такие как Google Analytics или Adobe Analytics, предоставляют данные о посещаемости веб-сайта, поведении пользователей и других показателях. Эти данные могут быть использованы для оптимизации производительности веб-сайта, улучшения пользовательского опыта и выявления возможностей для роста.
- Открытые источники данных: существует множество открытых источников данных, таких как правительственные порталы данных, которые предоставляют доступ к общедоступным данным. Эти данные могут быть использованы для ряда аналитических приложений, таких как экономический анализ, демографический анализ и мониторинг окружающей среды.
- Данные, генерируемые компьютером: Данные, создаваемые компьютером, относятся к данным, создаваемым машинами или устройствами, таким как журналы, данные телеметрии и обмен данными между компьютерами. Эти данные можно использовать для профилактического обслуживания, обнаружения аномалий и других приложений.

Таким образом, существует множество различных источников данных для аналитики, начиная от традиционных баз данных и заканчивая потоковыми данными и устройствами IoT. Выбор источника данных зависит от конкретных потребностей организации и типа выполняемого анализа. Эффективно используя эти источники данных, предприятия и организации могут получить представление о своей деятельности, определить возможности для роста и принять обоснованные решения.

### 1.5 5 V больших данных

Пятью основными и врожденными характеристиками больших данных являются 5 V (скорость, объем, значение, разнообразие и достоверность). Знание 5 V позволяет специалистам по обработке и анализу данных извлекать больше пользы из своих данных, а также позволяет компании ученых стать более ориентированной на клиента.

Большие данные обсуждались только в начале этого века с точки зрения трех V: объема, скорости и разнообразия. Еще два V (ценность и достоверность) были разработаны с течением времени, чтобы помочь специалистам по обработке и анализу данных лучше сформулировать и передать важные качества больших данных. Число пять представляет собой пять фундаментальных вопросов, которые должны быть затронуты в любой новости.

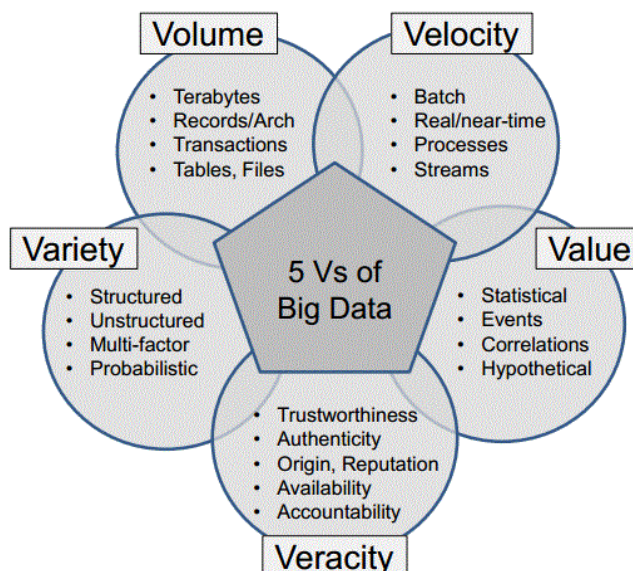


Рисунок 1 – 5 V больших данных

#### - Объём

Первым из пяти V больших данных является объем, который относится к количеству существующих данных. Объем является основой больших данных, потому что это начальное количество и объем собранных данных. Когда объем данных достаточно огромен, его называют большими данными. Однако то, что считается большими данными, является относительным и будет колебаться в зависимости от доступной вычислительной мощности на рынке.

#### - Скорость

Это относится к скорости, с которой данные генерируются и перемещаются. Это ключевое соображение для компаний, которым требуется, чтобы их данные текли быстро, чтобы они были доступны в нужные моменты для принятия наилучших бизнес-решений.

Организация больших данных будет иметь обширный и непрерывный поток данных, создаваемых и отправляемых в конечный пункт назначения. Данные могут поступать из различных источников, включая машины, сети, смартфоны и социальные сети. Эти данные должны быть быстро обработаны и проанализированы, иногда в режиме, близком к реальному времени.

В здравоохранении, например, в настоящее время доступно несколько медицинских устройств для мониторинга пациентов и сбора данных. От больничного медицинского оборудования до носимых гаджетов — полученные данные должны быстро передаваться и обрабатываться.

Однако в других обстоятельствах наличие ограниченного объема собранных данных может быть предпочтительнее, чем сбор большего количества данных, чем организация может обработать, поскольку это может привести к снижению скорости передачи данных.

#### - Разнообразие

Термин «разнообразие» относится к разнообразию типов данных. Организация может собирать данные из множества различных источников данных, ценность которых может варьироваться. Данные могут поступать как изнутри, так и из-за пределов компании. Стандартизация и обмен всеми собранными данными сопряжены с трудностями в разнообразии.

Собранные данные могут быть неструктурированными, полуструктурированными или структурированными. Неструктурированные данные — это неорганизованные данные, которые поступают во многих файлах или типах. Неструктурированные данные, как правило, не подходят для стандартной реляционной базы данных, поскольку они не вписываются в традиционные модели данных. Полуструктурированные данные — это информация, которая не была организована в определенной репозитории, но имеет связанную информацию, например метаданные. В результате их легче обрабатывать, чем неструктурированные данные. Структурированные данные, с другой стороны, представляют собой информацию, которая была организована в подготовленное хранилище. Это означает, что данные более доступны для эффективной обработки и анализа данных.

- Правдивость

Это относится к качеству и точности данных. Собранные данные могут быть неполными, ошибочными или неспособными предоставить правдивую, ценную информацию. В целом, правдивость относится к уровню доверия к полученным данным. Иногда данные могут стать беспорядочными и трудными в использовании. Если данные неполные, это может привести к большей путанице, чем к пониманию. В области медицины, например, если отсутствуют данные о том, какие лекарства принимает пациент, жизнь пациента может оказаться под угрозой.

Как ценность, так и правдивость вносят свой вклад в определение качества данных и понимания.

- Ценность

Это относится к ценности, которую могут дать большие данные, и тесно связано с тем, что организации могут делать с данными, которые они собирают. Требуется способность извлекать выгоду из больших данных, поскольку ценность больших данных значительно возрастает в зависимости от того, что из них можно почерпнуть. Организации могут получать и анализировать данные, используя одни и те же методы работы с большими данными, но то, как они извлекают выгоду из этих данных, должно быть уникальным для них.

## 1.6 Жизненный цикл аналитики данных

Жизненный цикл аналитики данных — это циклический процесс, который в шесть этапов описывает, как информация создается, собирается, обрабатывается, применяется и анализируется для различных целей.

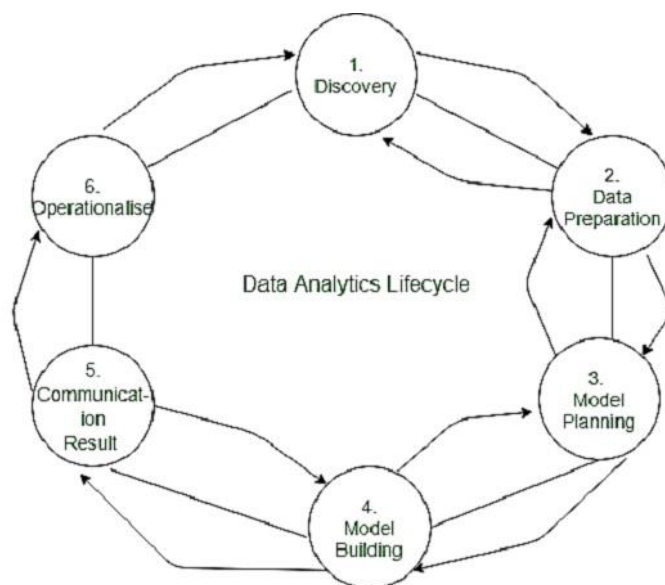


Рисунок 2 – Жизненный цикл аналитики данных

#### - Обнаружение данных

Это первый шаг к определению целей проекта и определению того, как завершить жизненный цикл аналитики данных. Начните с определения области вашего бизнеса и убедитесь, что у вас достаточно ресурсов (времени, технологий, данных и людей) для достижения ваших целей.

Самая сложная трудность на этом этапе — сбор достаточного количества информации. Вы должны создать план анализа, который потребует большого количества исследований.

Во-первых, вы должны проанализировать модели, которые вы намеревались разработать. Затем определите, какой объем знаний в предметной области вам необходимо приобрести для выполнения этих моделей.

Следующая важная вещь, которую нужно сделать, это оценить, достаточно ли у вас навыков и ресурсов, чтобы воплотить свои проекты в жизнь.

*Сформулируйте проблему:* проблемы, скорее всего, возникнут при удовлетворении ожиданий вашего клиента. Поэтому вам необходимо определить проблемы, связанные с проектом, и объяснить их своим клиентам. Этот процесс называется «кадрированием». Вы должны подготовить постановку проблемы, объясняющую текущую ситуацию и проблемы, которые могут возникнуть в будущем. Вам также необходимо определить цель проекта, включая критерии успеха и неудачи проекта.

*Сформулируйте первоначальную гипотезу:* После того, как вы соберете все требования клиентов, вы должны разработать первоначальные гипотезы после изучения исходных данных.

#### - Подготовка и обработка данных

Прежде чем перейти к процессу построения модели, этап подготовки и обработки данных включает в себя сбор, обработку и кондиционирование данных.

*Определите источники данных:* необходимо обнаружить несколько источников данных и оценить, сколько и какого типа данных можно собрать за определенный период времени. Оцените структуры данных, исследуйте их свойства и получите необходимые инструменты.

*Сбор данных:* Вы можете собирать данные тремя способами:

- Сбор данных: Вы можете собирать данные из внешних источников.

- Ввод данных: Вы можете подготовить точки данных с помощью цифровых систем или ручного ввода.
- Прием сигнала: Вы можете накапливать данные с цифровых устройств, таких как устройства IoT и системы управления.

#### - Планирование модели

Это этап, на котором вы должны оценить качество данных и выбрать подходящую модель для вашего проекта.

#### *Загрузка данных в песочнице Google Analytics*

Песочница аналитики — это компонент архитектуры озера данных, который позволяет хранить и обрабатывать огромные объемы данных. Он может эффективно обрабатывать широкий спектр данных, включая большие данные, транзакционные данные, данные социальных сетей, онлайн-данные и многое другое. Это параметр, в котором аналитики могут планировать и обрабатывать ресурсы данных с помощью выбранных ими инструментов обработки данных. Адаптивность аналитической песочницы — ее лучшая особенность. Это позволяет аналитикам обрабатывать данные в режиме реального времени и получать важную информацию за короткий промежуток времени.

Данные загружаются в песочницу тремя способами:

- ETL – Специалисты команды приводят данные в соответствие с бизнес-правилами перед их загрузкой в песочницу.
- ELT – данные загружаются в песочницу, а затем преобразуются в соответствии с бизнес-правилами.
- ETLT – Он включает в себя два уровня преобразования данных, включая ETL и ELT.

#### - Построение модели

Построение модели — это процесс развертывания запланированной модели в контексте реального времени. Это позволяет аналитикам консолидировать процесс принятия решений, предоставляя глубокие аналитические данные. Это повторяющийся процесс, потому что вы должны постоянно добавлять новые функции по запросу ваших клиентов. Ваша цель здесь — прогнозировать решения компании, настраивать рыночные стратегии и генерировать уникальные для вас интересы клиентов. Это достигается путем включения модели в существующий рабочий домен.

В определенных обстоятельствах конкретная модель точно соответствует бизнес-целям / данным, а в других требуется более одной попытки. Когда вы начнете изучать данные, вы должны запустить определенные алгоритмы и сравнить результаты со своими целями. В некоторых случаях может потребоваться запустить несколько вариантов модели одновременно, пока не будут получены требуемые результаты.

#### - Сообщение и публикация результатов

Это этап, на котором вы должны сообщить об анализе данных своим клиентам. Это требует различных сложных процессов, в которых вы должны четко доставлять им информацию. У ваших клиентов недостаточно времени, чтобы определить, какая информация является критической. В результате, чтобы привлечь внимание ваших клиентов, вы должны работать безупречно.

*Проверьте точность данных:* предоставляют ли данные информацию, как ожидалось? Если нет, то вам нужно запустить некоторые другие процессы, чтобы решить эту проблему. Вы должны убедиться, что данные, которые вы обрабатываете, предоставляют согласованную информацию. Это поможет вам построить убедительный аргумент при обобщении ваших выводов.

*Выделите важные выводы:* что ж, каждая база данных играет важную роль в построении эффективного проекта. Тем не менее, некоторые данные наследуют более мощную



информацию, которая действительно может принести пользу вашей аудитории. Подводя итоги, постарайтесь классифицировать данные по разным ключевым моментам.

*Определите наиболее подходящий формат общения:* то, как вы сообщаете о своих выводах, многое говорит о вас как о профессионале. Мы рекомендуем вам использовать визуальные эффекты, презентацию и анимацию, так как это поможет вам передать информацию намного быстрее. Тем не менее, иногда вам также нужно пойти на старую школу. Например, вашим клиентам, возможно, придется иметь результаты в физическом формате. Возможно, им также придется собирать определенную информацию и делиться ею с другими.

- Ввод в эксплуатацию

Жизненный цикл анализа данных почти завершен, как только вы создадите полный отчет, включающий основные результаты, документы и брифинги. Следующим шагом является оценка успеха вашего анализа перед отправкой окончательных отчетов заинтересованным сторонам.

Во время этой процедуры необходимо перенести данные песочницы и выполнить их в динамической среде. Затем вы должны регулярно отслеживать результаты, чтобы убедиться, что они соответствуют вашим ожиданиям. Вы можете завершить отчет, если результаты полностью соответствуют вашей цели. В противном случае необходимо изменить жизненный цикл аналитики данных и внести определенные коррективы.

## 1.7 Качественные, количественные и смешанные методы

Качественные, количественные и смешанные методы — это три широких подхода к исследованию и анализу данных.

- Качественное исследование: Качественное исследование — это исследовательский подход, который используется для понимания основных причин, мнений и мотивов. Этот тип исследований направлен на получение информации о мыслях, чувствах и переживаниях людей. Качественные данные обычно собираются с помощью таких методов, как интервью, фокус-группы, наблюдение и тематические исследования. Качественные данные часто носят описательный характер и анализируются с использованием таких методов, как контент-анализ, тематический анализ и обоснованная теория.
- Количественные исследования: Количественные исследования — это структурированный подход, который используется для измерения и анализа числовых данных. Этот тип исследований направлен на выявление закономерностей, взаимосвязей и тенденций в данных. Количественные данные обычно собираются с помощью таких методов, как опросы, эксперименты и статистический анализ существующих данных. Количественные данные часто анализируются с использованием статистических методов, таких как регрессионный анализ, корреляционный анализ и проверка гипотез.
- Исследование смешанных методов: Исследование смешанных методов представляет собой комбинацию качественных и количественных исследований. Этот тип исследования используется для получения более полного понимания вопроса исследования с использованием как качественных, так и количественных данных. Исследование смешанных методов может включать в себя одновременный сбор как качественных, так и количественных данных или сбор одного типа данных с последующим использованием другого типа данных для дальнейшего изучения вопроса исследования. Исследование смешанными

методами обычно включает сбор и анализ как числовых, так и описательных данных.

Подводя итог, можно сказать, что качественные, количественные и смешанные методы — это три широких подхода к исследованию и анализу данных. Качественные исследования сосредоточены на получении информации о мыслях, чувствах и опыте людей с использованием описательных данных, в то время как количественные исследования сосредоточены на выявлении закономерностей, отношений и тенденций с использованием числовых данных. Исследование со смешанными методами представляет собой комбинацию как качественных, так и количественных исследований, используемых для получения более полного понимания вопроса исследования. Выбор подхода зависит от вопроса исследования и типа собираемых данных. Эффективно используя эти подходы, исследователи могут получить представление о сложных явлениях и принимать обоснованные решения на основе результатов своих исследований.

## 2. Набор инструментов аналитиков данных

### 2.1 Бизнес-аналитика

Бизнес-аналитика относится к процессам и инструментам, используемым для анализа бизнес-данных, превращения их в действенные идеи и помощи всем сотрудникам организации в принятии более обоснованных решений. Система бизнес-аналитики, также известная как система поддержки принятия решений (DSS), анализирует текущие и исторические данные и представляет результаты в виде удобных для восприятия отчетов, информационных панелей, графиков, диаграмм и карт, которые можно совместно использовать в компании. Бизнес-аналитику иногда называют «описательной аналитикой», потому что она описывает, как бизнес работает сегодня и как он работал в прошлом. Он отвечает на такие вопросы, как «Что произошло?» и «Что нужно изменить?», но не вникает, почему что-то произошло или что может произойти дальше.

#### *Основные преимущества бизнес-аналитики*

Успешная программа бизнес-аналитики проливает свет на способы увеличения прибыли и производительности, выявления проблем, оптимизации операций и многого другого. Вот лишь некоторые из многих преимуществ бизнес-аналитики:

- Получите поддержку в принятии решений на основе фактов. Инструменты бизнес-аналитики помогают руководителям, менеджерам и работникам выявлять идеи, относящиеся к их ролям и сферам ответственности, и использовать их для принятия решений, основанных на фактах, а не на догадках.
- Получите и сохраните конкурентное преимущество. Благодаря своевременной бизнес-аналитике организации могут быстро выявлять новые тенденции и возможности и действовать в соответствии с ними. Они также могут оценить свои собственные возможности, сильные и слабые стороны по сравнению с конкурентами и использовать эту информацию в своих интересах.
- Измеряйте и отслеживайте производительность. Панели мониторинга бизнес-аналитики упрощают мониторинг ключевых показателей эффективности (KPI), отслеживание прогресса в достижении целевых показателей и настройку предупреждений, чтобы знать, где и когда следует сосредоточить инициативы по улучшению.
- Определите и установите контрольные показатели. Решения бизнес-аналитики позволяют организациям сравнивать свои процессы и показатели производительности с отраслевыми стандартами, определять, где необходимы улучшения, устанавливать значимые контрольные показатели и отслеживать прогресс в достижении целей.
- Выявляйте проблемы, чтобы их можно было решить. С помощью бизнес-аналитики пользователи могут выявлять потенциальные бизнес-проблемы до того, как они нанесут финансовый ущерб, такие как узкие места в производстве или дистрибуции, тенденции к росту оттока клиентов, рост затрат на рабочую силу и многое другое.
- Работайте более эффективно. Системы бизнес-аналитики позволяют каждому тратить меньше времени на поиск информации, анализ данных и создание отчетов. Они также могут выявлять области дублирования, дублирования или неэффективности между отделами или дочерними компаниями, чтобы оптимизировать операции.
- Сделайте данные и отчетность доступными для всех. Программное обеспечение бизнес-аналитики предлагает интуитивно понятные интерфейсы, отчеты с функцией перетаскивания и панели мониторинга на основе ролей, которые члены команды могут использовать сами — без необходимости кодирования или других технических навыков.
- Повысьте качество обслуживания клиентов и сотрудников. Пользователи бизнес-аналитики могут анализировать данные, чтобы выявлять закономерности в поведении

клиентов и сотрудников, анализировать отзывы и использовать аналитические данные для адаптации и улучшения опыта.

- Увеличьте выручку и прибыльность. В конечном счете, данные бизнес-аналитики приводят к лучшему пониманию того, где существуют риски и возможности, чтобы команды могли вносить выгодные коррективы.

## 2.2 Введение в науку о данных

В науке о данных сочетаются различные инструменты, алгоритмы и принципы машинного обучения. В своей основной форме это извлечение ценной информации или идей из организованных или неструктурированных данных с использованием навыков бизнеса, программирования и анализа. Это область с множеством различных компонентов, включая арифметику, статистику, информатику и т. Д. Специалисты по обработке и анализу данных — это те, кто преуспевает в своих дисциплинах и имеет полное представление об отрасли, в которой они хотят работать. Хотя это и не просто, но и не невозможно.

Наука о данных — это область, которая включает использование статистических и вычислительных методов для извлечения идей и знаний из данных. Это междисциплинарная область, которая охватывает аспекты информатики, статистики и предметно-ориентированных знаний. Специалисты по обработке и анализу данных используют различные инструменты и методы, такие как машинное обучение, статистическое моделирование и визуализация данных, для анализа и составления прогнозов на основе данных. Они работают как со структурированными, так и с неструктурированными данными и используют полученные знания для принятия обоснованных решений и поддержки бизнес-операций. Наука о данных применяется в широком спектре отраслей, включая финансы, здравоохранение, розничную торговлю и другие. Это помогает организациям принимать решения на основе данных и получать конкурентное преимущество.

*Как это работает?*

Наука о данных — это не одноэтапный процесс, поэтому вы сможете изучить его за короткое время и назвать нас специалистом по данным. Он проходит через множество этапов, и каждый элемент важен. Всегда нужно следовать правильным шагам, чтобы добраться до лестницы. Каждый шаг имеет свою ценность, и он учитывается в вашей модели. Пристегнитесь на своих местах и приготовьтесь узнать об этих шагах.

- Постановка проблемы: ни одна работа не начинается без мотивации; Однако наука о данных не является исключением. Важно очень четко и точно сформулировать или сформулировать свою задачу. Вся ваша модель и ее работа зависят от вашего утверждения. Многие ученые считают это главным и очень важным шагом науки о данных. Итак, убедитесь, какова ваша постановка проблемы и насколько хорошо она может повысить ценность бизнеса или любой другой организации.
- Сбор данных: после определения постановки задачи следующим очевидным шагом является поиск данных, которые могут потребоваться для вашей модели. Вы должны провести хорошее исследование, найти все, что вам нужно. Данные могут быть в любой форме, т. е. неструктурированными или структурированными. Это может быть в различных формах, таких как видео, электронные таблицы, закодированные формы и т. Д. Вы должны собрать все эти виды источников.
- Очистка данных: поскольку вы сформулировали свой мотив, а также собрали свои данные, следующим шагом будет очистка. Очистка данных — самое любимое занятие для специалистов по обработке и анализу данных. Очистка данных заключается в удалении отсутствующих, избыточных, ненужных и дублирующихся данных из вашей коллекции. Существуют различные инструменты для этого с помощью программирования на R или Python. Выбор одного из них полностью зависит от вас. У разных ученых есть свое мнение, на что выбрать. Когда дело доходит до статистической части, R предпочтительнее Python, поскольку он имеет привилегию более 12 000 пакетов. В то время как python используется, так как он быстрый,

легкодоступный, и мы можем выполнять то же самое, что и в R, с помощью различных пакетов.

- Анализ и исследование данных: это одна из главных вещей в науке о данных, которую нужно сделать, и время, чтобы вывести внутреннего Холмса наружу. Речь идет об анализе структуры данных, нахождении в них скрытых закономерностей, изучении поведения, визуализации влияния одной переменной на другие и последующем заключении. Мы можем исследовать данные с помощью различных графиков, сформированных с помощью библиотек с использованием любого языка программирования. В R GGplot является одной из самых известных моделей, в то время как Matplotlib в Python.
- Моделирование данных: после того, как вы закончите свое исследование, которое вы сформировали на основе визуализации данных, вы должны начать строить модель гипотезы, чтобы она могла дать вам хороший прогноз в будущем. Здесь вы должны выбрать хороший алгоритм, который лучше всего подходит для вашей модели. Существуют различные виды алгоритмов от регрессии до классификации, SVM (машины опорных векторов), кластеризации и т. Д. Ваша модель может быть алгоритмом машинного обучения. Вы обучаете модель с помощью данных обучения, а затем тестируете ее с помощью тестовых данных. Для этого существуют различные способы. Одним из них является метод K-fold, при котором вы разбиваете все данные на две части: одна - Train, а другая - тестовые данные. На этих базах вы обучаете свою модель.
- Оптимизация и развертывание: Вы выполнили каждый шаг и, следовательно, построили модель, которая, по вашему мнению, лучше всего подходит. Но как вы можете решить, насколько хорошо работает ваша модель? Вот тут-то и приходит оптимизация. Вы проверяете свои данные и обнаруживаете, насколько хорошо они работают, проверяя их точность. Короче говоря, вы проверяете эффективность модели данных и, таким образом, пытаетесь оптимизировать ее для более точного прогнозирования. Развертывание связано с запуском вашей модели и позволяет людям за ее пределами извлечь из этого выгоду. Вы также можете получить обратную связь от организаций и людей, чтобы узнать их потребности, а затем больше работать над своей моделью.

### 2.3 Управление версиями с помощью Git и GitHub

Git — это популярная система контроля версий, которая используется для отслеживания изменений в проектах разработки программного обеспечения, позволяя нескольким разработчикам одновременно работать над одной и той же кодовой базой. GitHub — это веб-служба хостинга для репозитория Git, которая предоставляет дополнительные функции, такие как инструменты для совместной работы, отслеживание проблем и непрерывная интеграция. Ниже приведен обзор того, как использовать Git и GitHub для управления версиями:

- Настройка Git: чтобы использовать Git, сначала необходимо установить его на локальный компьютер. Скачать последнюю версию Git можно с официального сайта. После установки Git вы можете настроить конфигурацию Git с помощью следующих команд:

```
$ git config --global user.name "Your Name"
$ git config --global user.email you@example.com
```

- Создание репозитория Git: Репозиторий Git — это каталог, содержащий все файлы и папки для вашего проекта. Вы можете создать новый репозиторий Git с помощью следующих команд:

```
$ mkdir myproject
```

```
$ cd myproject  
$ git init
```

- Добавление файлов в репозиторий: После того, как вы создали репозиторий Git, вы можете добавить в него файлы с помощью следующей команды:

```
$ git add myfile.txt
```

Эта команда добавляет myfile.txt в промежуточную область, где подготавливаются изменения перед фиксацией в репозитории.

- Фиксация изменений: после добавления файлов в промежуточную область их можно зафиксировать в репозитории с помощью следующей команды:

```
$git commit -m "Added myfile.txt"
```

Эта команда создает новую фиксацию с сообщением, описывающим внесенные изменения.

- Ветвление и слияние: Git позволяет создавать ветви, которые являются копиями кодовой базы, с которыми можно работать независимо. После внесения изменений в ветвь их можно объединить обратно в основную ветвь с помощью следующей команды:

```
$ git checkout главная  
$ git merge mybranch
```

Эта команда объединяет изменения из mybranch в основную ветвь.

- Использование GitHub: GitHub предоставляет веб-интерфейс для управления репозиториями Git и совместной работы с другими разработчиками. Чтобы использовать GitHub, сначала необходимо создать учетную запись на веб-сайте GitHub. После создания учетной записи вы можете создать новый репозиторий на GitHub и отправить в него локальный репозиторий с помощью следующих команд:

```
$ git remote add origin https://github.com/username/myproject.git  
$ git push -u origin main
```

Эта команда добавляет удаленный репозиторий с именем "origin" в локальный репозиторий и отправляет изменения в удаленный репозиторий.

- Сотрудничество с другими разработчиками: GitHub предоставляет ряд инструментов для совместной работы, которые позволяют нескольким разработчикам одновременно работать над одним и тем же проектом. Эти инструменты включают запросы на вытягивание, которые позволяют разработчикам предлагать изменения в кодовой базе и запрашивать их слияние с основной ветвью.

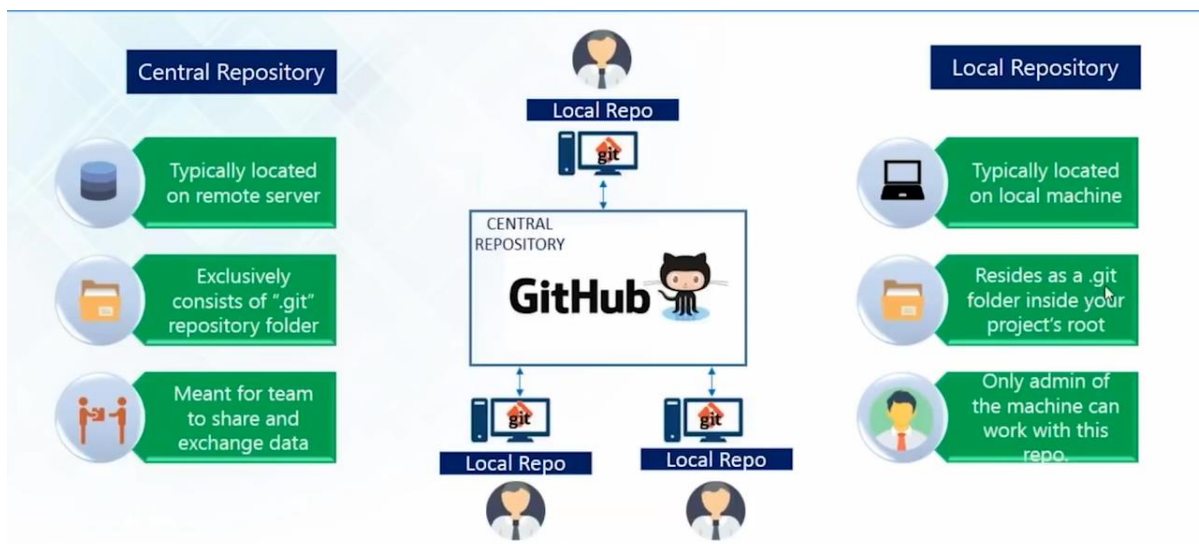


Рисунок 3 – Комбинация Git и Github

Подводя итог, можно сказать, что Git и GitHub являются мощными инструментами для управления версиями и совместной работы в проектах разработки программного обеспечения. С помощью Git вы можете отслеживать изменения в своей кодовой базе, создавать ветви для независимой работы над функциями и объединять изменения обратно в основную ветвь. С помощью GitHub вы можете размещать свои репозитории Git в Интернете, сотрудничать с другими разработчиками и использовать ряд инструментов для отслеживания проблем, проверки кода и непрерывной интеграции. Эффективно используя Git и GitHub, разработчики могут более эффективно работать вместе, снижать риск конфликтов и ошибок, а также поддерживать четкую историю изменений в своей кодовой базе.

### 3. Основы аналитики и визуализации с помощью Power BI

#### 3.1 Общие сведения о Power BI

Power BI — это служба бизнес-аналитики, разработанная Microsoft, которая позволяет пользователям анализировать данные и создавать визуализации. Это облачный сервис, который предоставляет ряд инструментов для моделирования данных, анализа данных и визуализации данных. Power BI может подключаться к различным источникам данных, включая электронные таблицы Excel, базы данных SQL, облачные сервисы, такие как Azure и Google Analytics, и неструктурированные файлы, такие как CSV или JSON. После подключения пользователи могут создавать модели данных, импортируя данные из разных источников и объединяя их в единую модель данных.

Power BI предоставляет ряд параметров визуализации, включая диаграммы, таблицы, матрицы, карты и датчики. Пользователи могут настраивать визуализации, изменяя цвета, шрифты и другие параметры форматирования. Они также могут создавать пользовательские вычисления и меры с помощью выражений анализа данных (DAX), языка формул, используемого в Power BI.

Power BI позволяет пользователям создавать интерактивные панели мониторинга и отчеты, которые предоставляют аналитические сведения об их данных. Панели мониторинга — это интерактивные представления, которые отображают ключевые метрики и визуализации в одном месте. Пользователи могут создавать настраиваемые панели мониторинга для отслеживания показателей производительности, отслеживания тенденций и выявления возможностей для улучшения. Отчеты — это интерактивные документы, которые позволяют пользователям более подробно изучать данные. Отчеты могут включать в себя несколько визуализаций и страниц, а пользователи могут фильтровать данные, детализировать детали и экспортировать данные в другие форматы. Решение Power BI можно опубликовать на веб-сайте Power BI. На веб-сайте Power BI обновление источника данных можно запланировать (зависит от источника и от того, поддерживает ли он обновление данных по расписанию или нет). Для отчета можно создавать панели мониторинга, и к нему можно предоставить общий доступ другим пользователям. Веб-сайт Power BI даже дает вам возможность нарезать и нарезать данные в Интернете, не требуя никаких других инструментов, а только простого веб-браузера. Вы также можете создавать отчеты и визуализации непосредственно на сайте Power BI. На рисунке ниже показано представление сайта Power BI и созданных на нем панелей мониторинга.

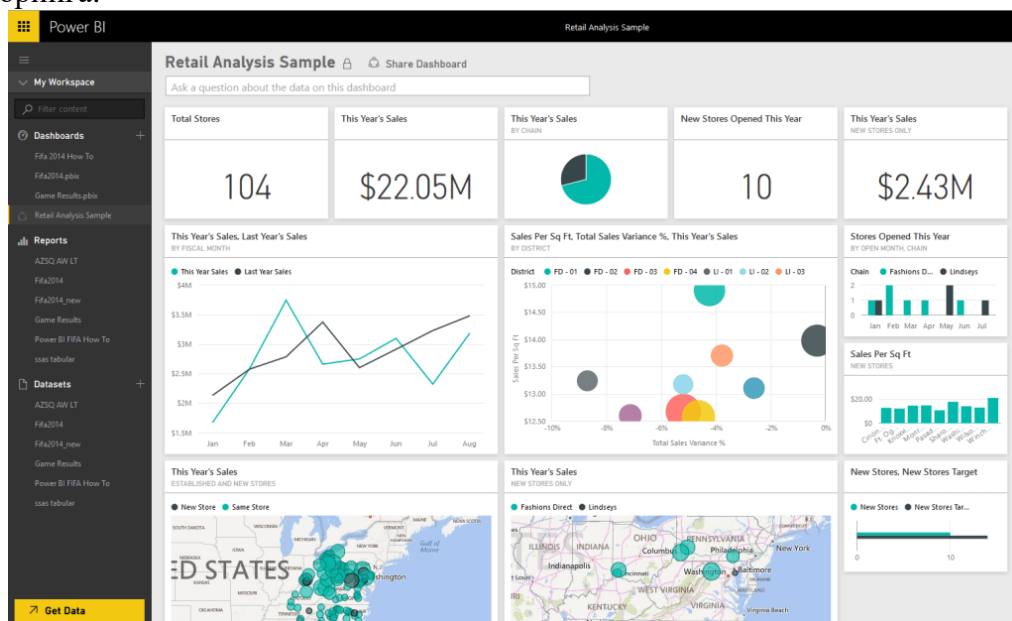


Рисунок 4 – Панели мониторинга веб-приложения Power BI



Power BI предоставляет ряд вариантов общего доступа и совместной работы. Пользователи могут предоставлять доступ к панелям мониторинга и отчетам другим пользователям как внутри своей организации, так и за ее пределами. Они также могут совместно работать над отчетами и панелями мониторинга, что позволяет нескольким пользователям вносить свой вклад в один и тот же проект. Power BI также предоставляет мобильные приложения для устройств iOS, Android и Windows, позволяя пользователям получать доступ к своим данным и взаимодействовать с ними на ходу.

Общий алгоритм действий в Power BI выглядит следующим образом:

- Перенесите данные в Power BI Desktop и создайте отчет.
- Публикация в службе Power BI, где можно создавать новые визуализации или создавать панели мониторинга.
- Делитесь информационными панелями с другими, особенно с людьми, которые находятся в пути.
- Просматривайте общие панели мониторинга и отчеты и взаимодействуйте с ними в приложениях Power BI Mobile.

В целом, Power BI — это мощный инструмент для предприятий и организаций, которые хотят получить аналитические сведения из своих данных. Его набор инструментов моделирования данных, анализа данных и визуализации данных в сочетании с возможностями совместного использования и совместной работы делают его популярным выбором для предприятий любого размера.

### 3.2 Моделирование и преобразования данных (DAX)

Моделирование и преобразование данных с помощью DAX (выражений анализа данных) являются основными функциями Power BI. DAX — это язык формул, используемый для создания пользовательских вычислений и мер на основе данных из нескольких таблиц. Он используется для создания сложных вычислений и манипулирования данными перед визуализацией.

Моделирование данных — это процесс определения связей между таблицами, создания иерархий и определения вычисляемых столбцов и мер. Перед созданием визуализаций важно создать модель данных. В Power BI процесс моделирования данных начинается с импорта данных из нескольких источников, а затем создания связей между таблицами. Power BI предоставляет визуальный интерфейс для определения этих связей, которые могут быть "один ко многим", "многие ко многим" или "один к одному".

Вычисляемые столбцы и меры являются другими важными компонентами моделирования данных. Вычисляемые столбцы создаются с помощью формул DAX для определения нового столбца на основе данных в существующем столбце. Меры используются для расчета данных на основе конкретного расчета или формулы. Меры можно использовать для расчета агрегатов, таких как сумма, среднее, минимальное, максимальное и количество.

Формулы DAX используются для создания сложных вычислений на основе данных из нескольких таблиц. Формулы DAX можно использовать для управления данными, выполнения математических операций и фильтрации данных. Формулы DAX можно использовать для создания новых столбцов, мер и таблиц.

Power BI также предоставляет средство преобразования данных под названием Power Query. Power Query позволяет пользователям очищать и преобразовывать данные из нескольких источников перед их импортом в Power BI. Power Query предоставляет визуальный интерфейс для преобразования данных, включая фильтрацию, сортировку, слияние и группировку данных. Это также позволяет пользователям удалять дубликаты, разделять столбцы и заменять значения. Power Query — это мощное средство, которое может помочь пользователям подготовить данные для моделирования и визуализации.

Таким образом, моделирование и преобразование данных с помощью DAX являются основными функциями Power BI. Моделирование данных включает в себя создание связей

между таблицами, определение иерархий и создание вычисляемых столбцов и мер. Формулы DAX используются для создания сложных вычислений на основе данных из нескольких таблиц. Power Query — это средство преобразования данных, которое позволяет пользователям очищать и преобразовывать данные перед их импортом в Power BI. С помощью этих средств пользователи могут создать надежную модель данных, которая предоставляет аналитические сведения о данных и создает привлекательные визуализации с помощью Power BI.

### 3.3 Визуализация данных и отчеты Power BI

Ниже приведен обзор использования Power BI для визуализации данных и отчетов.

- Импорт данных: чтобы использовать Power BI, сначала необходимо импортировать данные в приложение Power BI Desktop. Вы можете импортировать данные из различных источников, включая файлы Excel, текстовые файлы, базы данных и облачные сервисы, такие как Salesforce и Google Analytics.
- Создание визуализаций: Power BI предоставляет ряд инструментов для создания визуализаций, таких как линейчатые диаграммы, линейные диаграммы, точечные диаграммы, карты, датчики и таблицы. Чтобы создать визуализацию, можно перетащить поля из данных на холст визуализации, а затем настроить визуализацию с помощью параметров форматирования, предоставляемых Power BI.
- Создание отчетов: после создания визуализаций их можно упорядочить в отчеты, которые обеспечивают более полное представление данных. Отчеты в Power BI могут содержать несколько страниц и могут быть настроены с помощью ряда параметров форматирования, таких как цвета, шрифты и фон.
- Создание панелей мониторинга: панели мониторинга в Power BI предоставляют обзор ключевых метрик и ключевых показателей эффективности для вашего бизнеса или организации. Панели мониторинга могут включать визуализации из нескольких отчетов и могут быть настроены с помощью ряда параметров форматирования, таких как плитки, фильтры и срезы.
- Общий доступ к отчетам и панелям мониторинга: создав отчеты и панели мониторинга в Power BI, вы можете поделиться ими с другими пользователями в вашей организации. Power BI предоставляет ряд вариантов общего доступа, таких как общий доступ по электронной почте, внедрение на веб-сайт или сайт SharePoint или публикация в службе Power BI. Вы также можете управлять доступом к отчетам и панелям мониторинга с помощью безопасности на основе ролей, которая позволяет ограничить доступ определенным пользователям или группам.
- Создание интерактивных отчетов: Power BI позволяет создавать интерактивные отчеты, которые позволяют пользователям изучать и анализировать данные в режиме реального времени. Интерактивные отчеты в Power BI могут включать такие функции, как детализация, детализация и перекрестная фильтрация, которые позволяют пользователям более подробно изучать данные и получать аналитические сведения о ключевых тенденциях и закономерностях.
- Создание пользовательских визуализаций: Power BI также предоставляет ряд средств для создания пользовательских визуализаций с помощью пакета SDK для Power BI. С помощью пакета SDK для Power BI можно создавать собственные пользовательские визуальные элементы с помощью HTML, CSS и JavaScript, а затем публиковать их в Power BI Marketplace для скачивания и использования другими пользователями.



Рисунок 5 – Типы диаграмм

### 3.4 Обмен данными и совместная работа

Power BI предоставляет ряд функций для совместного использования данных и совместной работы, которые позволяют нескольким пользователям совместно работать над одними и теми же отчетами и панелями мониторинга. Ниже приведен обзор использования Power BI для совместного использования данных и совместной работы.

- **Общий доступ к отчетам и панелям мониторинга:** Power BI позволяет предоставлять общий доступ к отчетам и панелям мониторинга другим пользователям как внутри организации, так и за ее пределами. Чтобы предоставить общий доступ к отчету или панели мониторинга, можно использовать кнопку "Поделиться" на портале Power BI или в приложении Power BI Desktop. Вы можете предоставить доступ к отчету или панели мониторинга определенным пользователям или группам, а также управлять их уровнем доступа с помощью безопасности на основе ролей.
- **Совместная работа с рабочими областями Power BI:** рабочие области Power BI предоставляют среду совместной работы команд над одними и теми же отчетами и панелями мониторинга. Рабочие области позволяют нескольким пользователям получать доступ к одному и тому же содержимому и редактировать его, а также предоставляют такие функции, как управление версиями, совместное редактирование и комментирование. Чтобы создать рабочую область, можно использовать портал Power BI или приложение Power BI Desktop.
- **Совместная работа в режиме реального времени с помощью Power BI Live Connection:** Power BI Live Connection позволяет нескольким пользователям совместно работать над одним и тем же отчетом или панелью мониторинга в режиме реального времени. С помощью Live Connection изменения, внесенные одним пользователем, сразу же видны всем остальным пользователям, что позволяет сотрудничать и принимать решения в режиме реального времени. Для динамического подключения требуется подключение к серверу служб Analysis Services или набору данных Power BI.
- **Общий доступ к данным с помощью потоков данных Power BI:** потоки данных Power BI позволяют обмениваться данными с другими пользователями структурированным и управляемым способом. Потоки данных позволяют извлекать, преобразовывать и загружать данные из различных источников, а затем предоставлять доступ к этим данным другим пользователям в отчетах и

панелях мониторинга Power BI. Потоки данных могут совместно использоваться определенными пользователями или группами и управляться с помощью безопасности на основе ролей.

- Совместная работа с Power BI Embedded: Power BI Embedded позволяет разработчикам внедрять отчеты и панели мониторинга Power BI в пользовательские приложения и веб-сайты. С помощью Power BI Embedded несколько пользователей могут совместно работать над одними и теми же отчетами и панелями мониторинга в контексте пользовательского приложения или веб-сайта. Для Power BI Embedded требуется лицензия Power BI Pro или Premium.
- Общий доступ к внешним пользователям: Power BI позволяет предоставлять общий доступ к отчетам и панелям мониторинга внешним пользователям, таким как клиенты или партнеры. Для внешнего общего доступа требуется лицензия Power BI Pro или Premium, и им можно управлять с помощью безопасности на основе ролей.
- Совместная работа с Microsoft Teams: Power BI интегрирован с Microsoft Teams, что позволяет пользователям совместно работать над отчетами и панелями мониторинга в контексте канала Teams. Благодаря интеграции с Teams пользователи могут обмениваться отчетами и панелями мониторинга, добавлять комментарии и заметки, а также получать уведомления об изменениях и обновлениях.

## 5 Аналитика данных с помощью Python

### 5.1 Python (структура данных, функции, списки, словари и т. д.)

Python — популярный язык программирования, который широко используется в науке о данных, машинном обучении, веб-разработке и других областях. Ниже приведен обзор некоторых ключевых понятий Python, включая структуры данных, функции, списки и словари:

1. Python предоставляет ряд структур данных для хранения данных и управления ими, включая списки, кортежи, наборы и словари. Списки представляют собой упорядоченные наборы элементов и могут содержать данные любого типа. Кортежи похожи на списки, но являются неизменяемыми (т. е. не могут быть изменены после создания). Наборы представляют собой неупорядоченные коллекции уникальных элементов и полезны для выполнения операций с множествами, таких как объединение, пересечение и различие. Словари представляют собой пары "ключ-значение" и полезны для хранения данных в структурированном виде.
2. Функции: Функции — это многократно используемые блоки кода, которые выполняют определенную задачу. Функции в Python могут принимать входные аргументы, возвращать выходные значения и могут вызываться из других частей кода с помощью имени функции. Ниже приведен пример простой функции, которая принимает два входных аргумента и возвращает их сумму:

```
def add_numbers(x, y):  
    return x + y
```

3. Списки: Списки являются одной из наиболее часто используемых структур данных в Python. Списки определяются квадратными скобками и могут содержать данные любого типа. Вот пример простого списка:  

```
my_list = [1, 2, 3, "привет", "мир"]
```

Вы можете получить доступ к отдельным элементам списка с помощью индексирования, которое начинается с 0. Ниже приведен пример доступа к первому элементу списка:

```
first_item = my_list[0]
```

Вы также можете изменить элементы в списке с помощью индексирования. Вот пример изменения второго пункта списка:

```
my_list[1] = "Python"
```

4. Словари: Словари — еще одна часто используемая структура данных в Python. Словари определяются с помощью фигурных скобок и состоят из пар ключ-значение. Вот пример простого словаря:

```
my_dict = {"name": "John", "age": 30, "city": "New York"}
```

Вы можете получить доступ к отдельным значениям словаря с помощью ключа. Ниже приведен пример доступа к значению, связанному с ключом name:

```
name_value = my_dict["имя"]
```

Можно также изменить значение, связанное с ключом в словаре. Ниже приведен пример изменения значения, связанного с ключом «возраст»:

```
my_dict["возраст"] = 40
```

5. **Функции:** Функции — это блоки многократно используемого кода, которые выполняют определенную задачу. Функции в Python могут принимать входные аргументы, возвращать выходные значения и могут вызываться из других частей кода с помощью имени функции. Ниже приведен пример простой функции, которая принимает два входных аргумента и возвращает их сумму:

```
def add_numbers(x, y):  
    return x + y
```

Эта функция принимает два входных аргумента, *x* и *y*, и возвращает их сумму.

6. **Циклы:** Циклы используются для перебора последовательности элементов, таких как список или словарь. Python предоставляет два типа циклов: циклы *for* и *while*. Ниже приведен пример простого цикла *for*, который выполняет итерацию по списку:

```
my_list = [1, 2, 3, 4, 5]
```

```
Для товара в my_list:  
    print(элемент)
```

Этот цикл перебирает элементы в *my\_list* и выводит каждый элемент на консоль.

7. **Условные операторы:** Условные операторы используются для выполнения кода на основе условия. Python предоставляет два типа условных операторов: операторы *if* и операторы *elif*. Ниже приведен пример простого оператора *if*:

```
x = 10  
  
if x > 5:  
    print("x больше 5")
```

Этот оператор *if* проверяет, больше ли значение *x* 5, и если это так, он выводит сообщение на консоль.

## 5.2 Манипулирование данными и их анализ с помощью Pandas

Pandas — популярная библиотека Python для обработки и анализа данных. Pandas предоставляет ряд структур данных и инструментов для работы со структурированными данными, такими как табличные данные в электронных таблицах или базах данных. Вот обзор того, как использовать Pandas для обработки и анализа данных:

8. **Импорт данных:** чтобы использовать Pandas, вам сначала нужно импортировать данные в датафрейм Pandas. Вы можете импортировать данные из различных источников, включая CSV-файлы, файлы Excel, базы данных SQL и веб-API. Ниже приведен пример импорта CSV-файла в кадр данных Pandas:

```
Импорт панд как ПД  
df = pd.read_csv('my_data.csv')
```

9. Очистка данных: после того, как вы импортировали данные в DataFrame, вам может потребоваться очистить и предварительно обработать данные перед их анализом. Pandas предоставляет ряд инструментов для очистки данных, таких как удаление дубликатов, заполнение отсутствующих значений и преобразование типов данных. Ниже приведен пример удаления дубликатов из DataFrame:

```
df = df.drop_duplicates()
```

10. Манипулирование данными: Pandas предоставляет ряд инструментов для управления данными, таких как фильтрация, сортировка, группировка, слияние и изменение формы. Ниже приведен пример фильтрации DataFrame на основе условия:

```
filtered_df = df[df['column'] > 10]
```

Этот код фильтрует DataFrame, чтобы включить только строки, значение которых в столбце "столбец" больше 10.

11. Анализ данных: после того, как вы очистили свои данные и манипулировали ими, вы можете использовать Pandas для выполнения анализа данных и получения информации о ваших данных. Pandas предоставляет ряд инструментов для анализа данных, таких как вычисление сводной статистики, вычисление корреляций и визуализация данных. Ниже приведен пример вычисления среднего значения столбца в DataFrame:

```
mean_value = df['column'].mean()
```

12. Визуализация данных: Pandas предоставляет встроенные инструменты визуализации, которые позволяют создавать диаграммы и графики для визуализации данных. Инструменты визуализации Pandas основаны на популярной библиотеке Matplotlib и предоставляют ряд типов диаграмм, таких как линейные диаграммы, гистограммы, точечные диаграммы и гистограммы. Ниже приведен пример создания линейной диаграммы столбца в DataFrame:

```
import matplotlib.pyplot as plt
```

```
df.plot(x='столбец1', y='столбец2')  
plt.show()
```

Этот код создает линейную диаграмму значений в столбце "column2" DataFrame, при этом значения в столбце "column1" используются в качестве оси x.

### 5.3 Предварительная обработка данных

Предварительная обработка данных является важным этапом в конвейере анализа данных, который включает очистку, преобразование и подготовку данных к анализу. Целью предварительной обработки данных является обеспечение того, чтобы данные были в формате, подходящем для анализа, а также выявление и исправление любых ошибок или несоответствий в данных. Вот некоторые распространенные методы, используемые при предварительной обработке данных:

1. Очистка данных: Очистка данных включает в себя выявление и исправление ошибок и несоответствий в данных, таких как отсутствующие значения, повторяющиеся записи и выбросы. Распространенные методы очистки данных включают условное исчисление (заполнение отсутствующих значений), удаление дубликатов и удаление выбросов.

2. **Преобразование данных:** Преобразование данных включает преобразование данных из одного формата в другой, например, преобразование категориальных данных в числовые данные или масштабирование данных в общий диапазон. Общие методы преобразования данных включают одноразовое кодирование (преобразование категориальных данных в числовые данные), нормализацию (масштабирование данных до общего диапазона) и выбор признаков (выбор наиболее релевантных признаков для анализа).
3. **Интеграция данных:** Интеграция данных включает в себя объединение данных из нескольких источников для создания единого набора данных для анализа. Это может включать в себя слияние наборов данных с общим идентификатором или агрегирование данных в нескольких наборах данных.
4. **Сокращение данных:** Сокращение данных включает в себя уменьшение размера набора данных при сохранении наиболее важной информации. Общие методы сокращения данных включают анализ главных компонент (РСА), который уменьшает размерность данных за счет определения наиболее важных признаков, и извлечение признаков, которое создает новые признаки из существующих.
5. **Дискретизация данных:** Дискретизация данных включает преобразование непрерывных данных в дискретные категории или ячейки. Это может быть полезно для анализа данных, которые имеют большой диапазон значений, таких как возраст или доход. Общие методы дискретизации данных включают биннинг равной ширины (разделение данных на ячейки одинаковой ширины) и биннинг равной частоты (разделение данных на ячейки с равным количеством наблюдений).
6. **Нормализация данных:** нормализация данных включает масштабирование данных до общего диапазона. Это может быть полезно для сравнения данных, которые имеют разные единицы измерения или диапазоны, такие как температура и количество осадков. Общие методы нормализации данных включают минимальное и максимальное масштабирование (масштабирование данных в диапазоне от 0 до 1) и нормализацию z-оценки (масштабирование данных для получения среднего значения 0 и стандартного отклонения 1).
7. **Обработка отсутствующих данных:** Отсутствующие данные являются распространенной проблемой при анализе данных, и существует несколько методов ее обработки. К распространенным методам относятся условное исчисление (заполнение недостающих значений на основе значений других наблюдений), исключение (удаление наблюдений с отсутствующими значениями) и оценка (оценка отсутствующих значений с использованием статистических моделей).

## 5.4 Визуализация данных с помощью Matplotlib и Seaborn

Matplotlib и Seaborn — две популярные библиотеки Python для визуализации данных. Matplotlib — это низкоуровневая библиотека построения графиков, которая предоставляет широкий спектр инструментов для создания статических, интерактивных и анимированных визуализаций на Python. Seaborn — это библиотека более высокого уровня, построенная на основе Matplotlib и предоставляющая более оптимизированный интерфейс для создания статистических визуализаций. Ниже приведен обзор того, как использовать эти библиотеки для визуализации данных.

1. **Импорт библиотек:** чтобы использовать Matplotlib и Seaborn, вам сначала нужно импортировать их в среду Python. Это можно сделать с помощью следующего кода:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

2. **Создание фигуры:** Фигура — это контейнер верхнего уровня для всех элементов графика в Matplotlib. Вы можете создать новый рисунок, используя следующий код:



```
fig = plt.figure()
```

3. Создание подсюжета: Подсюжет — это контейнер внутри фигуры, который может содержать один или несколько графиков. Вы можете создать новый подсюжет с помощью следующего кода:

```
топор = fig.add_subplot(1, 1, 1)
```

Это создаст подграфик с одной строкой, одним столбцом и индексом 1.

4. Создание сюжетов: После того, как вы создали подсюжет, вы можете создавать различные типы графиков, используя Matplotlib и Seaborn. Вот пример создания точечной диаграммы с помощью Seaborn:

```
import seaborn as sns
# Загрузить демонстрационные данные из библиотеки Seaborn
tips = sns.load_dataset("tips")
# Создайте точечную диаграмму с помощью Seaborn
sns.scatterplot(data=tips, x="total_bill", y="tip")
```

5. Настройка графиков: Matplotlib и Seaborn предоставляют широкий спектр возможностей настройки графиков, таких как изменение цветов, добавление меток и настройка границ осей. Ниже приведен пример настройки точечной диаграммы Seaborn:

```
import seaborn as sns
# Загрузить демонстрационные данные из библиотеки Seaborn
tips = sns.load_dataset("tips")
# Создайте точечную диаграмму с помощью Seaborn
sns.scatterplot(data=tips, x="total_bill", y="tip", hue="sex", size="size",
palette="Set1")
# Добавить метки и заголовок
plt.xlabel("Общий счет")
plt.ylabel("Tip")
plt.title("Общий счет против чаевых по полу и размеру партии")
```

В этом примере мы настроили точечную диаграмму, добавив цветовое кодирование в зависимости от пола, настроив размер маркеров в зависимости от размера группы и используя пользовательскую цветовую палитру.

## 5.5 Статистический анализ с помощью NumPy и SciPy

NumPy и SciPy — две мощные библиотеки в Python для выполнения статистического анализа и научных вычислений. NumPy обеспечивает поддержку многомерных массивов и матричных операций, в то время как SciPy обеспечивает поддержку научных вычислений, включая оптимизацию, интеграцию, интерполяцию и статистический анализ. Ниже приведен обзор того, как использовать эти библиотеки для статистического анализа:

1. Импорт библиотек: Чтобы использовать NumPy и SciPy, вам сначала нужно импортировать их в среду Python. Это можно сделать с помощью следующего кода:

```
import numpy as np
import scipy.stats as stats
```

2. Создание массивов: NumPy обеспечивает поддержку многомерных массивов и матричных операций. Вы можете создать новый массив с помощью следующего кода:

```
a = np.array([1, 2, 3, 4])
```

3. Описательная статистика: NumPy предоставляет широкий спектр функций для расчета описательной статистики, таких как среднее значение, медиана, дисперсия и стандартное отклонение. Ниже приведен пример вычисления среднего значения и стандартного отклонения массива:

```
import numpy as np

# Создать массив случайных чисел
a = np.random.normal(loc=0, scale=1, size=100)

# Рассчитайте среднее значение и стандартное отклонение
среднее = np.mean(a)
std = np.std(a)

print("Mean:", mean)
print("Стандартное отклонение:", std)
```

4. Проверка гипотез: SciPy обеспечивает поддержку проверки гипотез, включая t-тесты, ANOVA и тесты хи-квадрат. Вот пример выполнения t-теста с использованием SciPy:

```
import numpy as np
import scipy.stats as stats

# Сгенерируйте две случайные выборки
sample1 = np.random.normal(loc=0, scale=1, size=100)
sample2 = np.random.normal(loc=1, scale=1, size=100)

# Выполните t-критерий с двумя выборками
t_stat, p_value = stats.ttest_ind(выборка1, выборка2)

print("Т-статистика:", t_stat)
print("P-значение:", p_value)
```

5. Корреляционный анализ: NumPy обеспечивает поддержку расчета коэффициентов корреляции между переменными. Вот пример расчета коэффициента корреляции Пирсона между двумя переменными:

```
import numpy as np
import scipy.stats as stats

# Сгенерируйте две случайные выборки
x = np.random.normal(loc=0, scale=1, size=100)
y = np.random.normal(loc=0, scale=1, size=100)

# Рассчитать коэффициент корреляции Пирсона
corr_coef, p_value = stats.pearsonr(x, y)

print("Коэффициент корреляции:", corr_coef)
print("P-значение:", p_value)
```

6. Линейная регрессия: SciPy обеспечивает поддержку линейного регрессионного анализа, включая подгонку модели линейной регрессии и вычисление коэффициентов

детерминации (значение R-квадрата). Ниже приведен пример подгонки линейной регрессионной модели и вычисления значения R-квадрата:

```
import numpy as np
from scipy.stats import linregress

# Сгенерируйте две случайные выборки
x = np.random.normal(loc=0, scale=1, size=100)
y = 2 * x + np.random.normal(loc=0, scale=0.5, size=100)

# Подгонка линейной регрессионной модели
наклон, пересечение, r_value, p_value, std_err = linregress(x, y)

# Вычислить значение R-квадрата
r_squared = r_value** 2

print("Наклон:", наклон)
print("Перехват:", перехват)
print("R-квадрат:", r_squared)
```

Таким образом, NumPy и SciPy являются мощными библиотеками для статистического анализа и научных вычислений в Python. Они предоставляют широкий спектр инструментов для выполнения описательной статистики, проверки гипотез, корреляционного анализа и линейного регрессионного анализа. С помощью этих библиотек вы можете изучать и анализировать свои данные строгим и систематическим образом.

# Практическое занятие 1. Установка программной среды PYTHON, ПАКЕТОВ NUMPY, PANDAS, IPYTHON

## I. Установка Python

Для установки Python необходимо произвести следующие шаги:

- перейдите на сайт <http://www.python.org/>
- на вкладке «Downloads» выберите свою операционную систему
- выберите и скачайте интересующую вас версию(Python 3.4.+).
- запустите скачанный установочный файл и следуйте инструкциям по установке

После того как все будет установлено, можно начинать работу с Python. Есть возможность работы в двух режимах — в командной строке и через IDLE (Shell-оболочка для Python). Выбор режима работы показан на рис. 1.

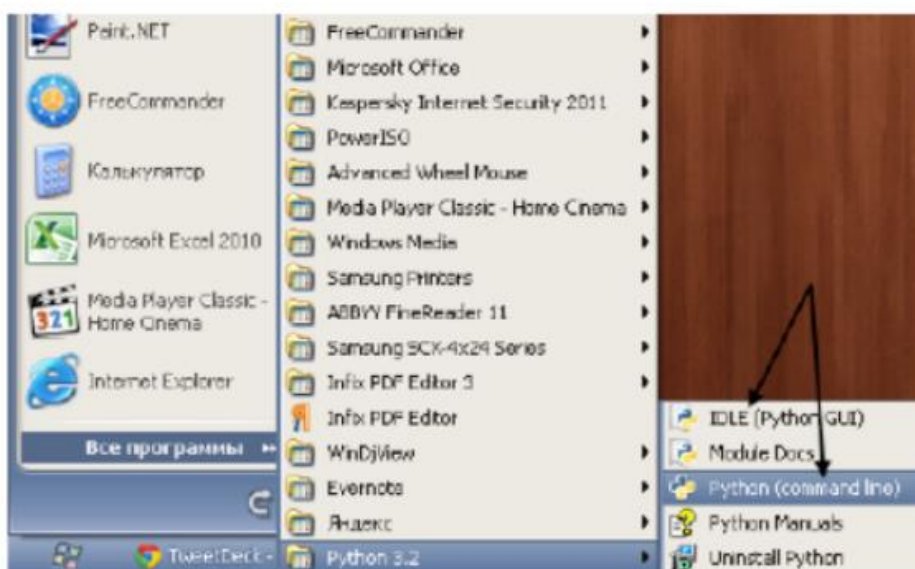
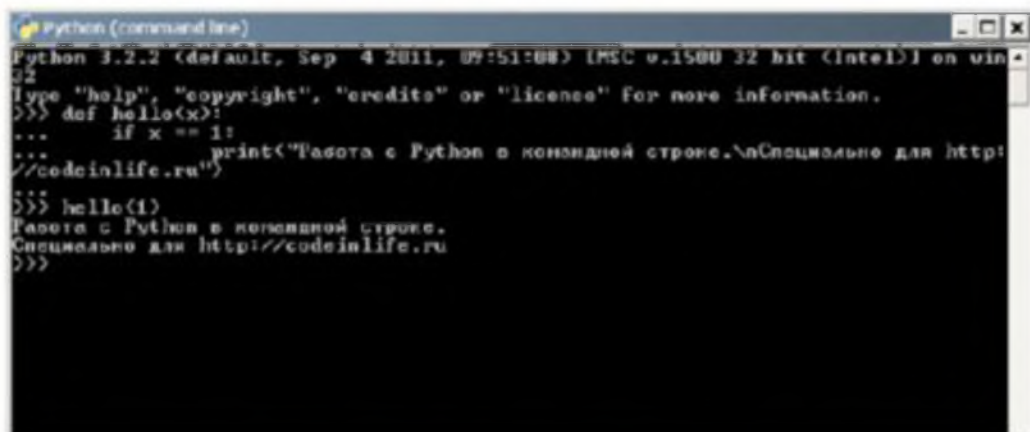


Рис 1. Выбор режима работы

Пример работы в командной строке приведен на рис. 2.



```
Python (command line)
Python 3.2.2 (default, Sep  4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> def hello(x):
...     if x == 1:
...         print("Работа с Python в командной строке.\nСпециально для http://codeinlife.ru")
...
>>> hello(1)
Работа с Python в командной строке.
Специально для http://codeinlife.ru
>>>
```

Рис. 2 Пример работы в командной строке

Пример работы через IDLE (рис. 3).



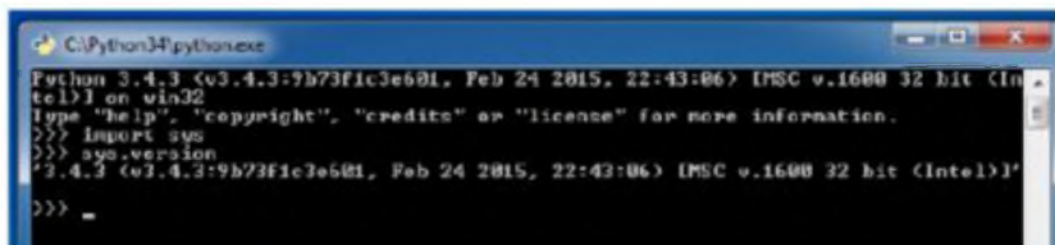
```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.2 (default, Sep  4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> def hello(x):
...     if x == 1:
...         print("Работа с Python в Shell-оболочке.\nСпециально для http://codeinlife.ru")
...
>>> hello(1)
Работа с Python в Shell-оболочке.
Специально для http://codeinlife.ru
>>>
```

Рис. 3 Пример работы в оболочке IDLE

Чтобы узнать версию установленного пакета введите следующие команды:

```
import sys
sys.version
```

Данная функция возвращает версию установленного пакета Python (рис. 4).



```
C:\Python34\python.exe
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import sys
>>> sys.version
'3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)]'
>>>
```

Рис. 4. Версия установленного пакета Python

## II. Установка сборщика пакетов pip

Для начала установки пакетов нужно установить установщик пакетов "pip"

- Скачайте `get-pip.py`, который находится по ссылке <https://bootstrap.pypa.io/get-pip.py>
- Нажмите правой кнопкой мыши на тексте. Выберите опцию "Сохранить как...". Сохраните файл с тем же именем с расширением ".py"
- Выполните скачанный файл, в среде Python, при этом будет установлен Pip (данный инструмент необходим для легкой установки модулей)
- Откройте командную строку (рис. 2.5) и перейдите в папку %путь до каталога Python%\Scripts\ (командой «cd»).
- Из данной строки можно запускать установку пакетов для Python с помощью команды:

```
pip install %name_of_package"
```

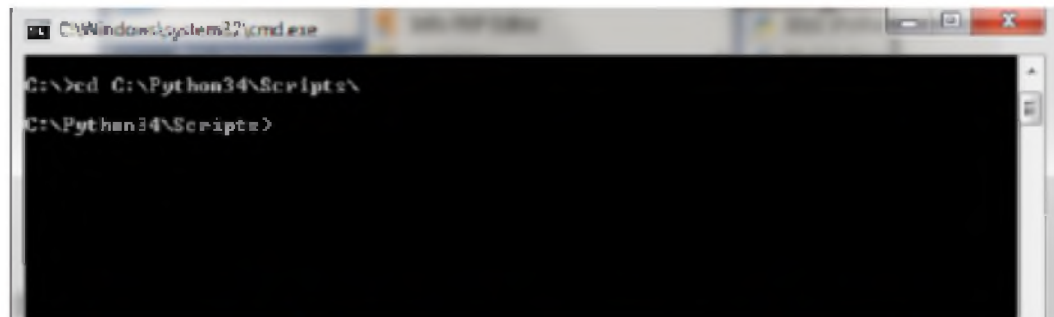


Рис 5. Переход в командной строке Windows в папку со скриптами Python

## III. Установка пакетов Pandas, Numpy, IPython

Для установки пакетов Python, необходимых для дальнейших лабораторных работ:

- Запустите командную строку Windows от имени Администратора, перейдите в папку с скриптами Python (C:\Python34\Scripts)
- Введите команды для установки соответствующих пакетов:

```
pip install pandas  
pip install numpy  
pip install ipython
```

## IV. Работа с системой Google Colaboratory

- Зайти на сайт <https://colab.research.google.com>
- зарегистрироваться на нем, ознакомиться с работой системы Google Colaboratory для возможности программировать на Python online.



Лабораторные работы будут осуществляться в лаборатории Гугл (Google Colaboratory), в нее можно перейти по следующей ссылке: <https://colab.research.google.com/notebooks/welcome> (рис.6).

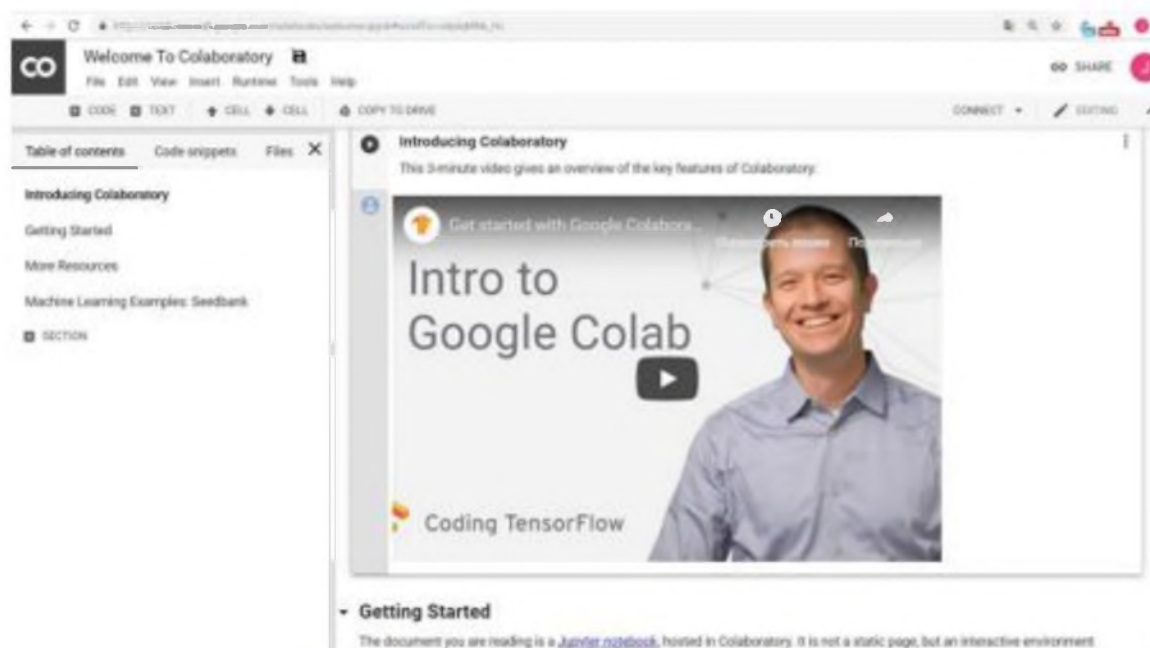


Рис.6. Начальная страница лаборатории Гугл

Работа будет вестись в блокноте Jupyter (<https://jupyter.org/>), размещенным в лаборатории. Блокнот Jupyter представляет собой не статическую страницу, а интерактивную среду, которая позволяет писать и выполнять код на Python и других языках.

Отличительной особенностью написания кода в блокноте Jupyter является то, что код разбивается на ячейки (cell), каждая из которых может быть выполнена отдельно.

Файлы, создаваемые и редактируемые в блокноте Jupyter имеют расширение .ipynb.

Для начала работы можно запустить код, представленный на первой странице лаборатории (рис.10). Для этого нужно нажать на стрелку в соответствующей ячейке или выделить ячейку и нажать **ctrl+enter**. Для выполнения кода последовательно во всех ячейках можно воспользоваться сочетанием клавиш **ctrl+F9**.



Рис. 10. Пример кода на языке Python в блокноте Jupyter

В данной программе производится подсчет количества секунд в сутках. Переменной `seconds_in_a_day` присваивается значение, равное  $24(\text{часа}) \cdot 60(\text{минут}) \cdot 60(\text{секунд})$ . Вторая строка предназначена для вывода значения, хранящегося в переменной `seconds_in_a_day`.

Для создания нового файла вашей программы нужно перейти в меню File → New Python 3 notebook.

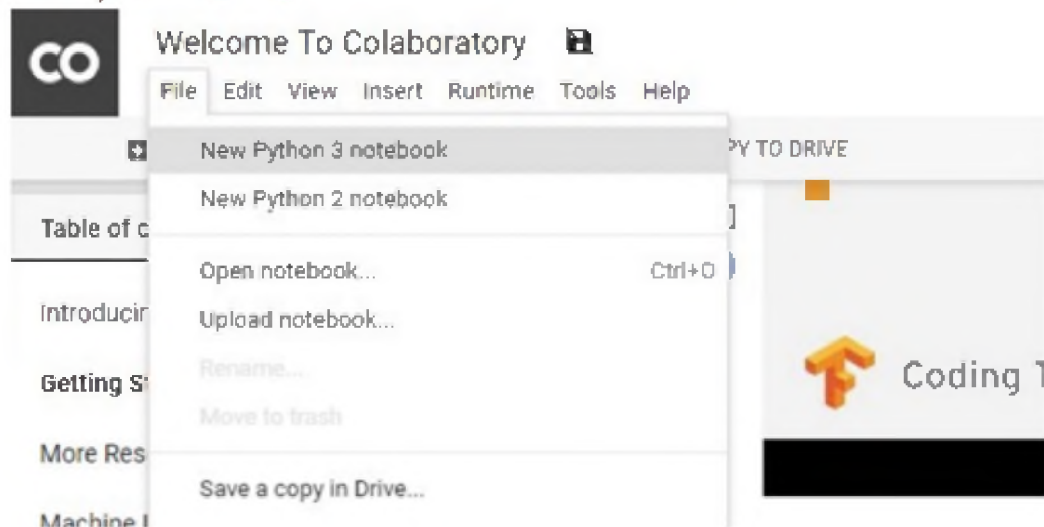


Рис. 11. Создание нового программного файла .ipynb

В новой вкладке откроется файл `Untitled2.ipynb` (рис. 12), с которым вы будете работать далее.

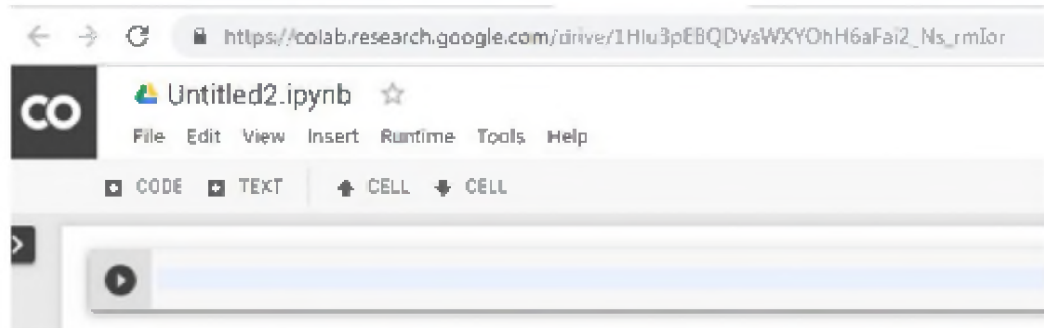
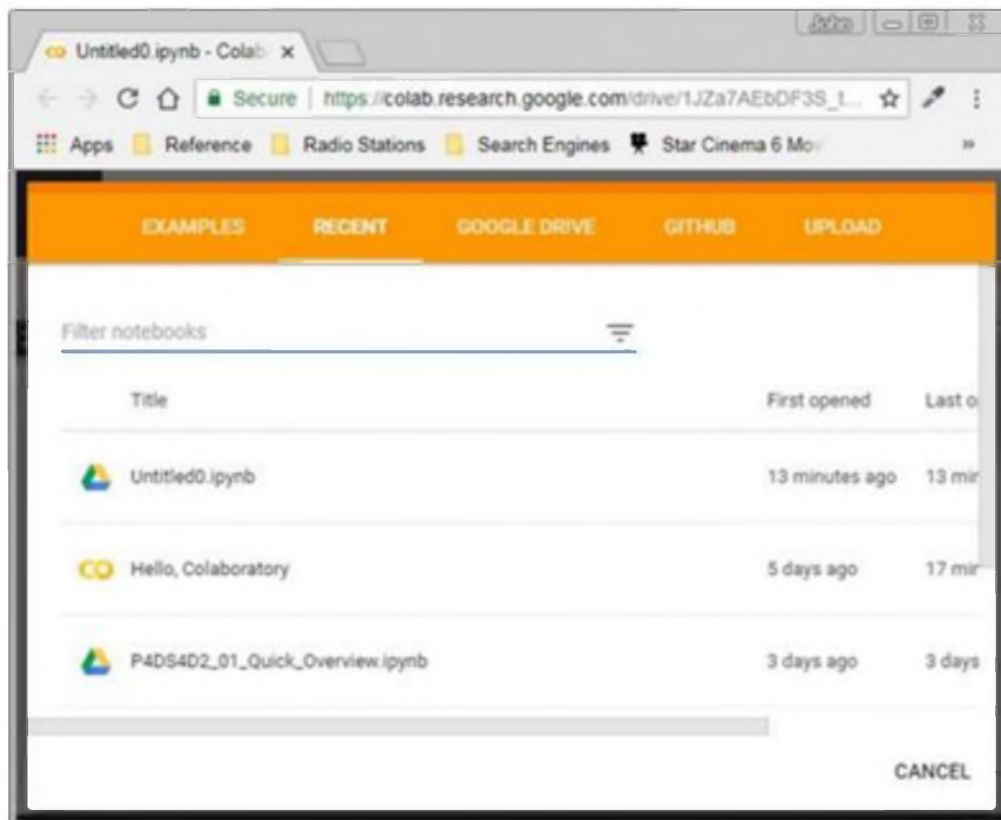


Рис. 12. Пример нового программного файла Переименовать файл можно выбрав в меню File → Rename.

Открытие ранее созданных проектов осуществляется выбором File → Open Notebook. Появится диалоговое окно, отображающее недавно открытые файлы.





Для того чтобы сохранить файл с новым именем на жесткий диск необходимо в меню выбрать File->download .ipynb. После выполнения лабораторной работы нужно будет сохранять файлы на диске.

Для добавления новой ячейки выберите Insert->Code cell (рис.13).

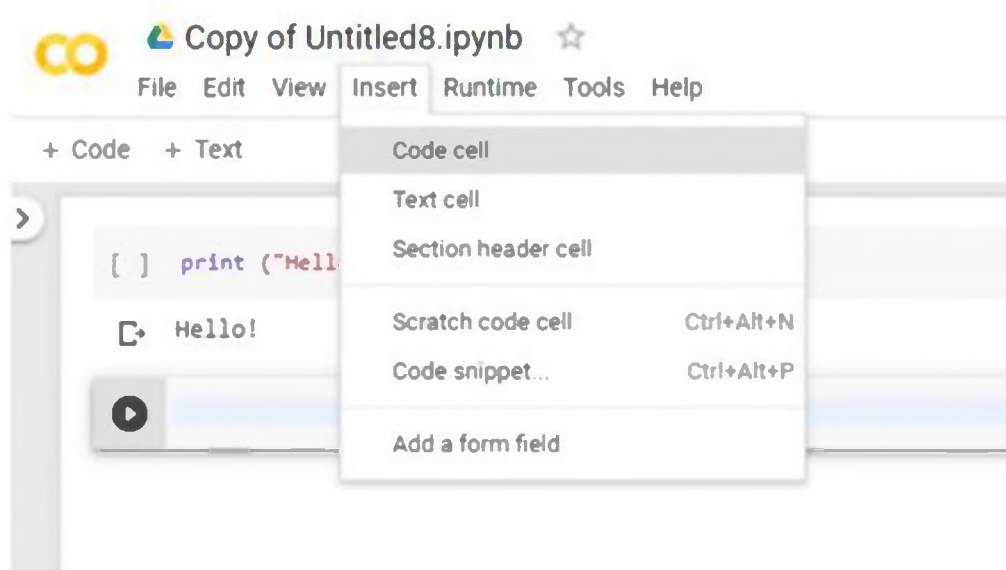


Рис. 13. Добавление новой ячейки кода

При наведении курсора мыши на ячейку кода, справа сверху появляется всплывающее меню (рис. 14). Нажатие на значок «сообщение» позволяет добавить комментарий к ячейке. Нажатие на корзину – удалить ячейку. Нажатие на «шестиренку» даст возможность редактирования настроек данной ячейки.



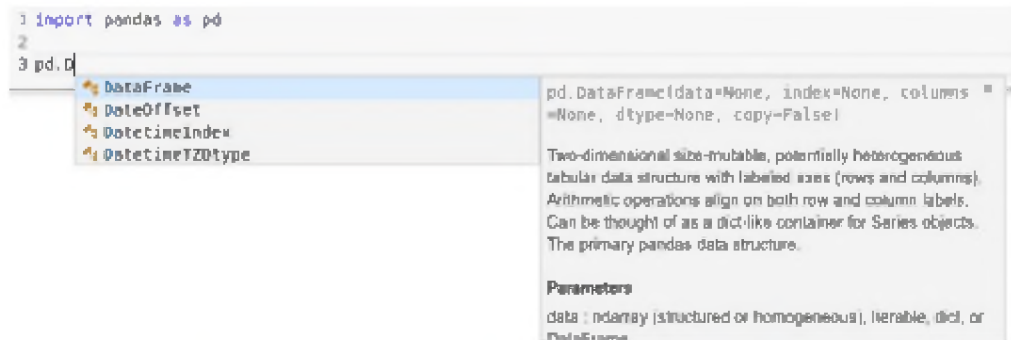
Рис. 14. Всплывающее окно ячейки кода

### Автоматическая подсказка синтаксиса кода

Дополнения кода и подсказки из документации происходят автоматически при вводе. Для этого используйте следующие сочетания клавиш:

**Ctrl-пробел**, чтобы заново открыть завершение кода.

**Ctrl-shift-пробел**, чтобы заново открыть подсказки параметров.



Для ознакомления с функциями языка Python можно воспользоваться документацией по языку Python: <https://docs.python.org/3/tutorial/>

#### Задание на лабораторную работу:

1. Установите Python 3.4.+
2. Установите сборщик пакетов pip
3. Установите пакеты Pandas, Numpy, Ipython
4. Знакомитесь с Google colab

#### Отчет должен содержать:

- название и цель работы;
- краткие теоретические сведения;
- лабораторное задание;
- результаты выполненной работы (снимки экрана с указанными версиями каждого из пакетов);
- вывод о проделанной работе.

#### Контрольные вопросы

1. Что представляет собой python. Назовите особенности языка программирования Python.
2. Для чего предназначен установщик пакетов pip?
3. Назовите специализацию пакетов Pandas, NumPy, Ipython.
4. Назовите режимы работы с программной средой Python.

## Практическое занятие 2. Предварительная обработка данных в Pandas

Заказчик — кредитный отдел банка. Нужно разобраться, влияет ли семейное положение и количество детей клиента на факт погашения кредита в срок. Входные данные от банка — статистика о платёжеспособности клиентов.

Результаты исследования будут учтены при построении модели **кредитного скоринга** — специальной системы, которая оценивает способность потенциального заёмщика вернуть кредит банку.

- 1 Знакомство с данными
- 2 Предобработка данных
  - 2.1 Обработка пропусков
  - 2.2 Замена типа данных
  - 2.3 Обработка дубликатов
  - 2.4 Категоризация данных
- 3 Ответы на вопросы
- 4 Общий вывод

### Описание данных

- *children* — количество детей в семье
- *days\_employed* — общий трудовой стаж в днях
- *dob\_years* — возраст клиента в годах
- *education* — уровень образования клиента
- *education\_id* — идентификатор уровня образования
- *family\_status* — семейное положение
- *family\_status\_id* — идентификатор семейного положения
- *gender* — пол клиента
- *income\_type* — тип занятости
- *debt* — имел ли задолженность по возврату кредитов
- *total\_income* — ежемесячный доход
- *purpose* — цель получения кредита

### Знакомство с данными

Откроем файл, предоставленный кредитным отделом банка и изучим информацию в нем.

```
# открытие и запись в переменную data таблицы
import pandas as pd
```

```
data = pd.read_csv('data.csv')
data.head(10)
```

```
Out[1]:
```

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income
0	1	-8437.673028	42	высшее	0	женат / замужем	0	F	сотрудник	0	253875.639
1	1	-4024.803754	36	среднее	1	женат / замужем	0	F	сотрудник	0	112080.014
2	0	-5623.422610	33	Среднее	1	женат / замужем	0	M	сотрудник	0	145885.952

### Вывод

На первый взгляд выгрузка содержит мало ошибок, но в таблице почти 22 тысячи строк. Столбцы таблицы названы корректно, переименовывать их не придётся. Если присмотреться к данным повнимательнее, то возникает ряд вопросов.

- Столбец 'days\_employed' — это общий трудовой стаж в днях. Почти все числа в нём отрицательные (возможно, это строки и минус — это тире, или количество дней считали, вычитая начала стажа текущий момент, а не наоборот?), а те, что не отрицательные - очень большие. Например, если человек работал 340266 дней все 365 дней в году, значит его стаж - почти 932 года! Это маловероятно, а точнее невероятно. Возможно, это указан стаж в часах? Тогда его стаж составит примерно 38 лет. Это уже ближе к вероятному, но для 53-летнего человека всё равно многовато. Также видно, что большие значения указаны там, где income\_type это 'пенсионер'
- Столбец 'total\_income' — это ежемесячный доход. В каких единицах измерения? В рублях? В йенах? В цветочках? Допустим, в рублях.

## Предобработка данных

### Обработка пропусков

Сначала посмотрим общую информацию о таблице.

```
# просмотр общей информации
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   children              21525 non-null  int64
1   days_employed         19351 non-null  float64
2   dob_years             21525 non-null  int64
3   education              21525 non-null  object
4   education_id          21525 non-null  int64
5   family_status         21525 non-null  object
6   family_status_id      21525 non-null  int64
7   gender                21525 non-null  object
8   income_type           21525 non-null  object
9   debt                  21525 non-null  int64
10  total_income          19351 non-null  float64
11  purpose               21525 non-null  object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

Видно, что пропуски есть в столбцах days\_employed и total\_income. Но нельзя исключать, что пропуски отсутствуют и в других столбцах, ведь они могут быть записаны, как строки или числовые значения. Начнем сначала. Столбец с количеством детей содержит целочисленный тип, поэтому строк там точно нет. Проверим уникальные значения.

```
# уникальные значения количества детей в таблице
data['children'].unique()
Out[3]: array([ 1,  0,  3,  2, -1,  4, 20,  5], dtype=int64)
```

Видим значение минус 1. Возможно, клиенты, у которых нет детей, просто пропустили этот вопрос и по умолчанию значение стоит -1.

```
# количество клиентов со значением 'children' равным -1
data[data['children'] < 0]['children'].count()
```

Так как одной из основных задач является нахождение влияния наличия детей на факт погашения кредита, проверим столбец 'debt' у этих клиентов.

```
# сколько среди таких клиентов должников
data[data['children'] < 0]['debt'].value_counts()
```

Почти все, за исключением одного пользователя, не имели задолженностей по кредиту, поэтому если мы самостоятельно установим значения 1 или 0 в таких строках, то мы искусственно повлияем на результат. Это достаточно маленький процент записей из всей таблицы, поэтому мы можем просто исключить эти строки из исследования.

```
# удаляем строки и проверяем результат
data = data.loc[data['children'] >= 0, :]
data.reset_index(drop=True, inplace=True)
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21478 entries, 0 to 21477
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   children              21478 non-null  int64
1   days_employed         19307 non-null  float64
2   dob_years             21478 non-null  int64
3   education             21478 non-null  object
4   education_id          21478 non-null  int64
5   family_status         21478 non-null  object
6   family_status_id      21478 non-null  int64
7   gender                21478 non-null  object
8   income_type           21478 non-null  object
9   debt                 21478 non-null  int64
10  total_income          19307 non-null  float64
11  purpose               21478 non-null  object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

Строки удалились, проверяем дальше. Посмотрим, все ли клиенты в этой выгрузке имеют возраст.

```
# сортируем количество людей определённого возраста по его возрасти
data['dob_years'].value_counts().sort_index().head()

Out[7]: 0      101
        19      14
        20      51
        21     111
        22     183
        Name: dob_years, dtype: int64
```

101 человек с нулевым возрастом. Посмотрим, какие у них другие значения в таблице.

```
# выводим информацию о клиентах с нулевым возрастом
data[data['dob_years'] == 0].head()
```

Даже по стажу можно сказать, что пропущенный возраст может быть абсолютно разным, поэтому просто заменить на средний возраст тут не получится. Посмотрим, как они выплачивают кредиты.

```
# сколько среди таких клиентов должников
data[data['dob_years'] == 0]['debt'].value_counts()
```

Только 8% не выплачивают кредит вовремя. И 101 человек - это тоже достаточно маленький процент от всей выгрузки, поэтому исключение этих строк не очень повлияет на результат.

```
# проверяем, удалились ли строки
```

```

data = data.loc[data['dob_years'] > 0, :]
data.reset_index(drop=True, inplace=True)
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21377 entries, 0 to 21376
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   children              21377 non-null  int64
1   days_employed         19216 non-null  float64
2   dob_years             21377 non-null  int64
3   education             21377 non-null  object
4   education_id          21377 non-null  int64
5   family_status         21377 non-null  object
6   family_status_id      21377 non-null  int64
7   gender                21377 non-null  object
8   income_type           21377 non-null  object
9   debt                 21377 non-null  int64
10  total_income          19216 non-null  float64
11  purpose               21377 non-null  object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB

```

Пропуски успешно удалились. Видно, что пропуски остались в столбцах `days_employed` и `total_income`, причем их одинаковое количество. Возможно, они даже содержатся в одних и тех же записях. Проверим это.

```

# проверка количества пропусков в столбцах
if data['days_employed'].isna().sum() == data['total_income'].isna().sum():
    print('Количество пропусков в столбцах days_employed и total_income совпадает')

```

Количество пропусков в столбцах `days_employed` и `total_income` совпадает

```

# вывод на экран таблицы с пропусками
data_nan = data[data['days_employed'].isna()]
data_nan.head()
# проверка пропусков
if data_nan['days_employed'].isna().sum() == data_nan['total_income'].isna().sum():
    print('Пропуски в столбцах days_employed и total_income содержатся в одних и тех же записях.')

```

Пропуски в столбцах `days_employed` и `total_income` содержатся в одних и тех же записях.

Значения зарплаты отсутствуют тогда и только тогда, когда отсутствуют сведения о стаже. Возможно, в таблице указывался стаж по трудовой книжке, поэтому значения могут быть пропущены. Соответственно, и зарплата указана только там, где стаж есть.

Проверим, может быть, таких случаев немного и мы сможем их просто исключить из исследования.

```

# расчёт процента строк с пропусками
print('Процент строк, содержащих пропуски: {:.0%}'.format(data_nan['children'].count() / data['children'].count()))

```

Процент строк, содержащих пропуски: 10%

10% строк с пропусками — это довольно большой процент, поэтому просто удалить эти строки нельзя, ведь мы потеряем существенное количество важной информации. Основной задачей исследования является обнаружение влияния семейного положения и количества детей на факт погашения клиентом кредита. Эта информация представлена в таблице полностью. Даже по первым строкам таблицы с пропусками видно, что отсутствие

информации о стаже и заработной плате клиента не влияет на наличие значения в столбце о наличии у него задолженностей по кредиту.

Значения в столбцах с пропусками являются количественными характеристиками, поэтому мы можем заменить эти значения нулями, средним арифметическим или медианой.

Но прежде, чем мы это сделаем, разберемся с отрицательными и огромными значениям в столбце 'days\_employed'.

```
# поиск минимального и максимального положительного числа в столбце
max_positive = data['days_employed'].max()
print('Максимальное положительное значение:', max_positive)
min_positive = data['days_employed'][data['days_employed'] >= 0].min()
print('Максимальное положительное значение:', min_positive)
```

Максимальное положительное значение: 401755.40047533

Максимальное положительное значение: 328728.72060451825

Максимальное и минимальное положительное значения довольно велики, значит все положительные значения указаны некорректно. Проверим гипотезу о том, что такие значения установлены только у пенсионеров.

```
# уникальные значения в столбце 'income_type', где зарплата больше
минимального положительного значения
max_money_income_type_unique = data[data['days_employed'] >=
min_positive]['income_type'].unique()
max_money_income_type_unique
```

Гипотеза не подтвердилась, огромный стаж установлен и для безработных.

Проверим, установлен ли этот стаж в часах. Разделим каждое такое значение на количество часов в году и сравним с возрастом клиента.

```
# проверка корректности данных:
# из возраста вычитаем стаж, рассчитанный в годах для тех, у кого стаж указан
положительный
years_not_employed = data[data['days_employed'] >=
min_positive]['dob_years'] - data[data['days_employed'] >=
min_positive]['days_employed'] / 365 / 24
years_not_employed[years_not_employed <= 0].sort_values().head()
```

Посмотрим подробную информацию на подобных клиентах.

```
# проверка конкретного пользователя
data.loc[19299][:]
Out[18]: children 0
days_employed 389397.167577
dob_years 26
education высшее
education_id 0
family_status женат / замужем
family_status_id 0
gender F
income_type пенсионер
debt 0
total_income 214963.301941
purpose покупка недвижимости
Name: 19299, dtype: object
```

Странно - пенсионер в возрасте 26 лет. Возраст здесь явно указан неверно. Посмотрим, сколько у нас таких пенсионеров.

```
# сортировка количества пенсионеров определённого возраста по его
возрастанию
```



```

data[data['income_type']
'пенсионер']['dob_years'].value_counts().sort_index().head()

# приведение аномальных значений стажа к более реальному:
# делим существующие значения на количество часов в дне
data.loc[data['days_employed'] >= min_positive, 'days_employed'] =
data[data['days_employed'] >= min_positive]['days_employed'] / 24
print('Максимальное положительное значение:', data['days_employed'].max())

# замена всех отрицательных значений стажа на положительные
data['days_employed'] = abs(data['days_employed'])
print('Минимальное значение стажа - {:.2f}, максимальное -
{:.2f}'.format(data['days_employed'].min(), data['days_employed'].m

# группируем столбцы 'days_employed' и 'total_income' по значениям
'income_type'
# ищем медиану сгруппированных значений
# добавляем результат вместо соответствующих пропущенных значений
data['days_employed'] =
data['days_employed'].fillna(data.groupby('income_type')['days_employed'].transf
orm('median'))
data['total_income'] =
data['total_income'].fillna(data.groupby('income_type')['total_income'].transfor
m('median'))

```

Проверим, успешно ли заполнены пропуски в таблице.

```

data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21377 entries, 0 to 21376
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   children               21377 non-null  int64
1   days_employed          21377 non-null  float64
2   dob_years              21377 non-null  int64
3   education              21377 non-null  object
4   education_id           21377 non-null  int64
5   family_status          21377 non-null  object
6   family_status_id       21377 non-null  int64
7   gender                 21377 non-null  object
8   income_type            21377 non-null  object
9   debt                   21377 non-null  int64
10  total_income           21377 non-null  float64
11  purpose                21377 non-null  object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB

```

## Вывод

На этапе обработки пропусков были приняты следующие решения:

- Данные с нулевым возрастом клиента и наличием -1 ребенка были удалены из датасета: количество таких строк не велико (около 1%), поэтому это не отразится на результатах значительно.
- Данные стажа были обработаны в соответствии со здравым смыслом.
- Пропуски, обнаруженные в столбцах стажа и зарплаты, оказались в одних и тех же строчках, но таких строчек около 10%. Столько строчек удалять из выгрузки не стоит, к тому же в других ячейках они содержат важную информацию, которую терять в таком

количестве нельзя. Поэтому было решено заполнить эти пропуски медианными значениями в разрезе типа занятости.

Проведённая предобработка данных может минимально исказить результаты исследования, и не стоит об этом забывать.

## Замена типа данных

Взглянем ещё раз на информацию о таблице и обратим внимание на типы данных.

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21377 entries, 0 to 21376
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   children              21377 non-null  int64  
 1   days_employed         21377 non-null  float64
 2   dob_years             21377 non-null  int64  
 3   education              21377 non-null  object  
 4   education_id          21377 non-null  int64  
 5   family_status         21377 non-null  object  
 6   family_status_id      21377 non-null  int64  
 7   gender                21377 non-null  object  
 8   income_type           21377 non-null  object  
 9   debt                  21377 non-null  int64  
10   total_income          21377 non-null  float64
11   purpose               21377 non-null  object  
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

На самом деле с типами данных тут всё в порядке, обязательной замены тут не требуется. Но можно изменить столбцы 'days\_employed' и 'total\_income' на целочисленный тип, потому что вряд ли нам сильно понадобится дробная часть дней стажа или рублей.

```
# замена типа столбцов на целочисленный
data['days_employed'] = data['days_employed'].astype('int')
data['total_income'] = data['total_income'].astype('int')

# просмотр таблицы
data.head()
```

## Вывод

Вещественный тип данных в двух столбцах таблицы был изменен на целочисленный тип, но это не было обязательным действием, поскольку вещественный тип никак не помешал бы расчётам.

## Обработка дубликатов

Прежде чем искать дубликаты, посмотрим на уникальные значения в столбцах 'education', 'family\_status' и 'income\_type'. Возможно, значения записаны в разных регистрах. Со значениями в столбце 'purpose' разберёмся позже, когда будем лемматизировать.

Начнём с образования.

```
# количество различных значений в столбце образования
data['education'].value_counts()
```

```
Out[27]: среднее          13660
         высшее          4678
         СРЕДНЕЕ         766
         Среднее         706
         неоконченное высшее 665
         ВЫСШЕЕ         272
         Высшее         266
         начальное       250
         Неоконченное высшее 47
         НЕОКОНЧЕННОЕ ВЫСШЕЕ 29
         НАЧАЛЬНОЕ       17
         Начальное       15
         ученая степень   4
         Ученая степень   1
         УЧЕНАЯ СТЕПЕНЬ  1
         Name: education, dtype: int64
```

Почти все типы образования записаны в различных регистрах. Исправим это переводом всех значений в нижний регистр.

```
# понижение регистра значений в столбце образования
data['education'] = data['education'].str.lower()

# количество различных значений в столбце образования
data['education'].value_counts()
```

```
Out[29]: среднее          15132
         высшее          5216
         неоконченное высшее 741
         начальное       282
         ученая степень   6
         Name: education, dtype: int64
```

Все 5 типов образования записаны в едином виде.

Рассмотрим значения в столбце о семейном положении.

```
# количество различных значений в столбце о семейном образовании
data['family_status'].value_counts()
# количество различных значений в столбце о семейном образовании
data['income_type'].value_counts()
```

Исключим вероятность полного совпадения записей разных людей и будем считать, что соответствующие дубликаты содержат информацию об одном и том же человеке.

```
# количество строчек, у которых есть дубликат
data[data.duplicated()] ['days_employed'].count()
```

71 строчка с дубликатами. Появление дубликатов можно объяснить тем, что человек мог оформить заявку на кредит через сайт банка, а потом прийти в банк, где в базу внесут те же самые данные.

```
# удаляем дубликаты из таблицы и обновляем индексацию
data = data.drop_duplicates().reset_index(drop=True)
# проверка на оставшиеся дубликаты
data[data.duplicated()] ['days_employed'].count()
```

**Вывод**

Все явные дубликаты удалены, но мы не проанализировали значения в последнем столбце, ведь строки в нем различные, но написанные по разному цели получения кредита на деле могут значить одно и тоже.

## Категоризация данных

Имеет смысл разделить на категории доход. Выделим три категории дохода:

- низкий - до 100000
- средний - от 100000 до 200000
- высокий - от 200000

```
# функция определения категории дохода
def income_category(income):
    if income < 100000:
        return 'низкий'
    elif income < 200000:
        return 'средний'
    else:
        return 'высокий'

# создаем новый столбец и заполняем его результатом вызова функции к каждому
значению столбца 'total_income'
data['income_category'] = data['total_income'].apply(income_category)

# распределение категорий
data['income_category'].value_counts()
# посмотрим, что получилось
data.head()
```

Теперь цели кредита заработная плата категоризированы, так будет удобнее анализировать влияние цели взятия кредита на факт его погашения.

## Вывод

Категоризация зарплаты поможет проанализировать влияние именно уровня дохода на факт выплаты кредита.

## Общий вывод

Чтобы ответить на следующие вопросы, воспользуемся сводными таблицами. Они позволят объединить и обработать данные и представить результат в удобном для восприятия виде.

```
# функция для составления сводной таблицы
def pivot_on_groups(df, feature):
    return df.groupby(feature)['debt'].agg(['count', 'sum', lambda x:
'{:.2%}'.format(x.mean())])
```

- Есть ли зависимость между наличием детей и возвратом кредита в срок?

```
# сводная таблица в разрезе количества детей
pivot_on_groups(data, 'children')
```

```
# сводная таблица по бинарной классификации
pivot_on_groups(data, data['children'].apply(lambda x: 'Есть' if x>0 else
'Нет'))
```

## Вывод

С ростом числа детей в семье процент людей, имеющих долги по кредиту, растет. Так же падает количество клиентов. Значит количество детей и правда влияет на факт погашения кредита в срок. Гипотеза подтвердилась. К тому же клиенты, у которых нет детей, чаще погашают кредит вовремя.

- Есть ли зависимость между семейным положением и возвратом кредита в срок?

```
# создание сводной таблицы в разрезе семейного положения
pivot_on_groups(data, 'family_status')
```

## Вывод

Клиенты, никогда не состоящие в официальном браке, чаще остальных имеют задолженность по кредиту. Значит семейное положение влияет на погашение кредита в срок.

- Есть ли зависимость между уровнем дохода и возвратом кредита в срок?

```
# создание сводной таблицы относительно уровня дохода
pivot_on_groups(data, 'income_category')
```

## Вывод

Нельзя точно сказать, что чем выше у клиента зарплата, тем вероятнее он закроет кредит в срок. Возможно, если бы мы категоризировали доходы клиентов по другим интервалам, то результат был бы иной. В нашем случае с ростом дохода вероятность долгов по кредиту изменяется не монотонно. У среднего класса долги возникают чаще. Люди с низким доходом понимают, что не потянут еще и процент за просрочку, поэтому отказывают себе в чем-то, чтобы отложить деньги на выплату кредита. Богатые люди, как правило, более финансово грамотные и кредиты обычно берут на бизнес. А вот средний класс может позволить себе задержку, ведь *все равно через неделю зарплата*.

## Общий вывод

Начальная гипотеза о том, что количество детей и семейное положение влияет на факт погашения кредита в срок подтвердилась: чем больше у клиента детей, тем вероятнее он будет иметь долг по кредиту, а клиенты, состоящие когда-либо в официальном браке, вероятнее всего выплатят кредит в срок.

Предобработка данных включала в себя обработку артефактов и пропущенных или некорректных значений, что могло минимально повлиять на получившийся результат. Также были удалены дубликаты, а для категоризации целей кредита было выделено пять основных категорий целей кредита.

Подводя итог, можно выделить самые надёжные и ненадёжные категории граждан в каждой группе.

Факторы, показывающие, что клиент с большей вероятностью закроет кредит в срок (с указанием процента невозврата):

- отсутствие детей - 7.55%
- семейный статус "в разводе" - 7.19%, или "вдова/вдовец" - 6.52%
- высокий уровень заработной платы - 7.09%

Факторы, влияющие на возможное наличие долгов по кредиту:

- многодетные семьи (например, 4 ребёнка - 9.76%)
- семейный статус не женат/не замужем - 9.78%, или гражданский брак 9.35%
- средний уровень заработной платы - 8.63%

## 1. Цели и задачи

Цель работы: изучение программных средств для организации рабочего места специалиста по анализу данных и машинному обучению.

Основные задачи:

- получение программного доступа к данным, содержащимся в источниках различного типа;
- выполнение предварительного анализа данных и получение обобщенных характеристик наборов данных;
- исследование простых методов визуализации данных;
- изучение основных библиотек Python для работы с данными.

Перед выполнением работы необходимо ознакомиться с базовыми принципами языка Python, используя следующие источники: [1-5]. Особое внимание необходимо уделить репозитарию [5] с исходными кодами.

## Методика и порядок выполнения работы

Необходимо организовать подготовку данных для построения модели (допустим модели классификации). В качестве данных выбран набор данных об ирисах Фишера. Это, пожалуй, самый известный набор данных, с которого многие начинают исследование алгоритмов машинного обучения.

Данный набор данных предназначен для построения модели классификации. Данные о 150 экземплярах ириса (рис. 1), по 50 экземпляров из трёх видов – Ирис щетинистый (*Iris setosa*), Ирис виргинский (*Iris virginica*) и Ирис разноцветный (*Iris versicolor*). Для каждого экземпляра измерялись четыре характеристики (в сантиметрах):

- 1) длина наружной доли околоцветника (sepal length);
- 2) ширина наружной доли околоцветника (sepal width);
- 3) длина внутренней доли околоцветника (petal length);
- 4) ширина внутренней доли околоцветника (petal width).

На основании этого набора данных требуется построить правило классификации, определяющее вид растения по данным измерений. Это задача многоклассовой классификации, так как имеется три класса – три вида ириса.



а)



б)



в)

Рисунок 1 – Внешний вид классифицируемых ирисов:  
а) Iris setosa; б) Iris virginica; в) Iris versicolor

1. Необходимо скачать набор данных из репозитория Center for Machine Learning and Intelligent Systems (необходим только один текстовый файл с данными измерений): <http://archive.ics.uci.edu/ml/datasets/Iris>.

Файл Iris.data при просмотре выглядит следующим образом (рис. 2):

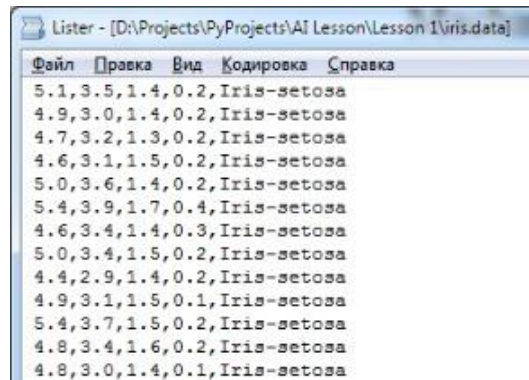


Рисунок 2 – Внешний вид данных файла Iris.data

2. Использовать текстовые редакторы для просмотра и анализа данных из определенных наборов – нерациональный вариант. Поэтому запустим Jupyter Notebook и начнем работать с загруженным набором с использованием среды Python. Используем метод `genfromtxt()` из пакета `scipy`.

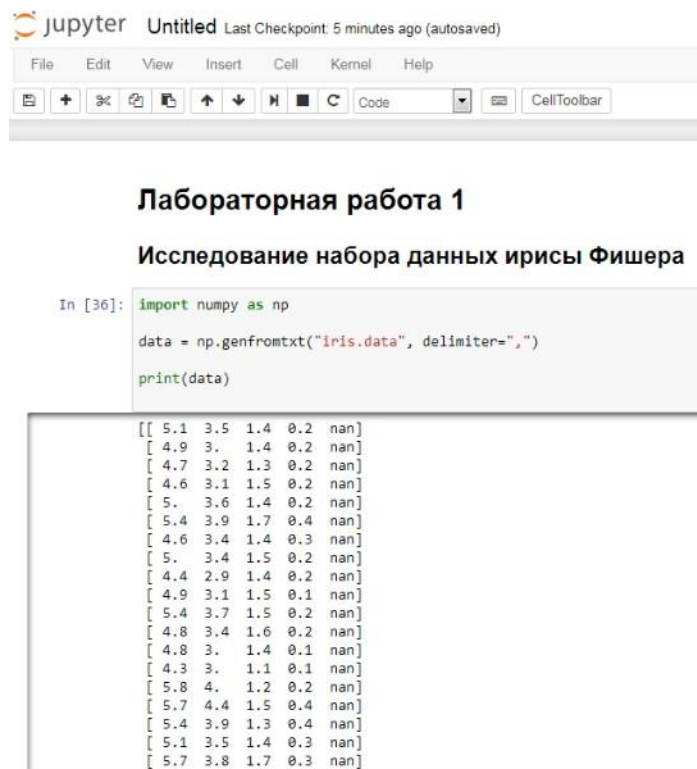


Рисунок 3 – Загрузка данных файла Iris.data

Метод `genfromtxt()` возвращает массив `numpy` (тип `numpy.ndarray`). Следует обратить внимание, что пятый столбец содержит неопределенные значения `numpy.NaN` (объясните – почему?).

3. Производить вывод всего источника данных – нерациональный путь. В реальных задачах данных может оказаться слишком много, поэтому



чаще всего используют подвыборку данных для поверхностного обзора исследуемой обучающей выборки (рис. 4).

```
In [37]: print ( "Data type : ", type(data) )
print ( "Data shape : ", data.shape )
print ( data[:10] )

Data type : <class 'numpy.ndarray'>
Data shape : (150, 5)
[[ 5.1  3.5  1.4  0.2  nan]
 [ 4.9  3.   1.4  0.2  nan]
 [ 4.7  3.2  1.3  0.2  nan]
 [ 4.6  3.1  1.5  0.2  nan]
 [ 5.   3.6  1.4  0.2  nan]
 [ 5.4  3.9  1.7  0.4  nan]
 [ 4.6  3.4  1.4  0.3  nan]
 [ 5.   3.4  1.5  0.2  nan]
 [ 4.4  2.9  1.4  0.2  nan]
 [ 4.9  3.1  1.5  0.1  nan]]
```

Рисунок 4 – Начальное исследование Iris.data

Из представленного фрагмента видно, что data – это двумерный массив размером 150x5, или можно сказать, что это одномерный массив, каждый элемент которого также одномерный массив размером 5 элементов.

4. В рамках данной задачи необходимо все-таки получить значения пятого столбца. Для этого желательно использовать другой подход (рис. 5):

```
In [69]: data1 = np.genfromtxt("iris.data", delimiter=";", dtype=None)
print(data1.shape)
print(type(data1))
print(type(data1[0]))
print(type(data1[0][4]))
print(data1[:10])

(150,)
<class 'numpy.ndarray'>
<class 'numpy.void'>
<class 'numpy.bytes_'>
[(5.1, 3.5, 1.4, 0.2, b'Iris-setosa') (4.9, 3.0, 1.4, 0.2, b'Iris-setosa')
 (4.7, 3.2, 1.3, 0.2, b'Iris-setosa') (4.6, 3.1, 1.5, 0.2, b'Iris-setosa')
 (5.0, 3.6, 1.4, 0.2, b'Iris-setosa') (5.4, 3.9, 1.7, 0.4, b'Iris-setosa')
 (4.6, 3.4, 1.4, 0.3, b'Iris-setosa') (5.0, 3.4, 1.5, 0.2, b'Iris-setosa')
 (4.4, 2.9, 1.4, 0.2, b'Iris-setosa') (4.9, 3.1, 1.5, 0.1, b'Iris-setosa')]
```

Рисунок 5 – Загрузка данных разного типа в массив

Сразу же желательно (на первоначальных этапах изучения Python) проводить анализ типов различных значений. На рис. 6 представлен еще один вариант загрузки данных в массив numpy.ndarray.

```
In [70]: dt = np.dtype("f8, f8, f8, f8, U30")
data2 = np.genfromtxt("iris.data", delimiter=";", dtype=dt)
print(data2.shape)
print(type(data2))
print(type(data2[0]))
print(type(data2[0][4]))
print(data2[:10])

(150,)
<class 'numpy.ndarray'>
<class 'numpy.void'>
<class 'numpy.str_'>
[(5.1, 3.5, 1.4, 0.2, 'Iris-setosa') (4.9, 3.0, 1.4, 0.2, 'Iris-setosa')
 (4.7, 3.2, 1.3, 0.2, 'Iris-setosa') (4.6, 3.1, 1.5, 0.2, 'Iris-setosa')
 (5.0, 3.6, 1.4, 0.2, 'Iris-setosa') (5.4, 3.9, 1.7, 0.4, 'Iris-setosa')
 (4.6, 3.4, 1.4, 0.3, 'Iris-setosa') (5.0, 3.4, 1.5, 0.2, 'Iris-setosa')
 (4.4, 2.9, 1.4, 0.2, 'Iris-setosa') (4.9, 3.1, 1.5, 0.1, 'Iris-setosa')]
```

Рисунок 6 – Загрузка данных в массив с типом, определяемым пользователем

Поясните различие в структурах данных, получаемых с использованием

представленных листингов.

5. Было загружено 150 элементов данных, но даже при такой маленькой выборке невозможно что-либо сказать о наборе данных. Для получения дополнительной информации необходимо визуализировать загруженные данные. В нашем случае каждый элемент данных представлен вещественными признаками – это существенно упрощает визуализацию (рис. 1.7). Но сложность заключается в том, что приходится работать с элементами 4-мерного пространства, поэтому строится не графическое представление распределения, а отдельные проекции.

6. Уже из графического распределения на рис. 7 видно, что тип ирисов Setosa хорошо отделяется. На данном графике представлено отображение в плоскости признаков ('Sepal Width', 'Sepal Length') Но исследователь имеет возможность построить столько графиков, сколько необходимо для глубокого анализа данных. Изменим ячейку In[72] в соответствии с листингом, представленным на рис. 8.

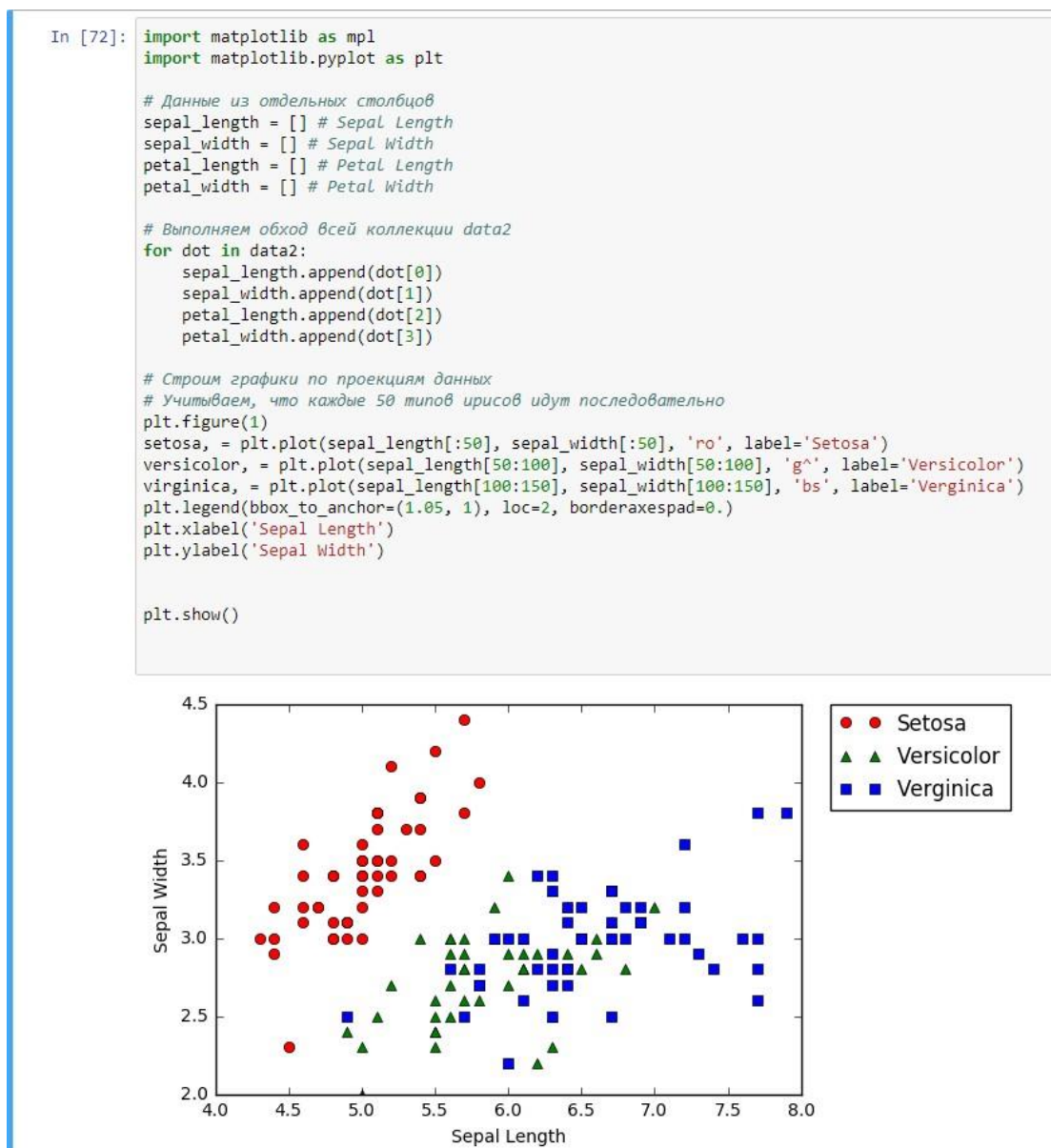


Рисунок 7 – Простейший анализ данных по графическому представлению

```

In [74]: import matplotlib as mpl
import matplotlib.pyplot as plt

# Данные из отдельных столбцов
sepal_length = [] # Sepal Length
sepal_width = [] # Sepal Width
petal_length = [] # Petal Length
petal_width = [] # Petal Width

# Выполняем обход всей коллекции data2
for dot in data2:
    sepal_length.append(dot[0])
    sepal_width.append(dot[1])
    petal_length.append(dot[2])
    petal_width.append(dot[3])

# Строим графики по проекциям данных
# Учитываем, что каждые 50 типов ирисов идут последовательно
plt.figure(1)
setosa, = plt.plot(sepal_length[:50], sepal_width[:50], 'ro', label='Setosa')
versicolor, = plt.plot(sepal_length[50:100], sepal_width[50:100], 'g^', label='Versicolor')
virginica, = plt.plot(sepal_length[100:150], sepal_width[100:150], 'bs', label='Verginica')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')

plt.figure(2)
setosa, = plt.plot(sepal_length[:50], petal_length[:50], 'ro', label='Setosa')
versicolor, = plt.plot(sepal_length[50:100], petal_length[50:100], 'g^', label='Versicolor')
virginica, = plt.plot(sepal_length[100:150], petal_length[100:150], 'bs', label='Verginica')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Sepal Length')
plt.ylabel('Petal Length')

plt.figure(3)
setosa, = plt.plot(sepal_length[:50], petal_width[:50], 'ro', label='Setosa')
versicolor, = plt.plot(sepal_length[50:100], petal_width[50:100], 'g^', label='Versicolor')
virginica, = plt.plot(sepal_length[100:150], petal_width[100:150], 'bs', label='Verginica')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Sepal Length')
plt.ylabel('Petal Width')

plt.show()

```

Рисунок 8 – Построение простейшего графика для отображения различных проекций данных

Вывод данного кода представлен на рис. 9.

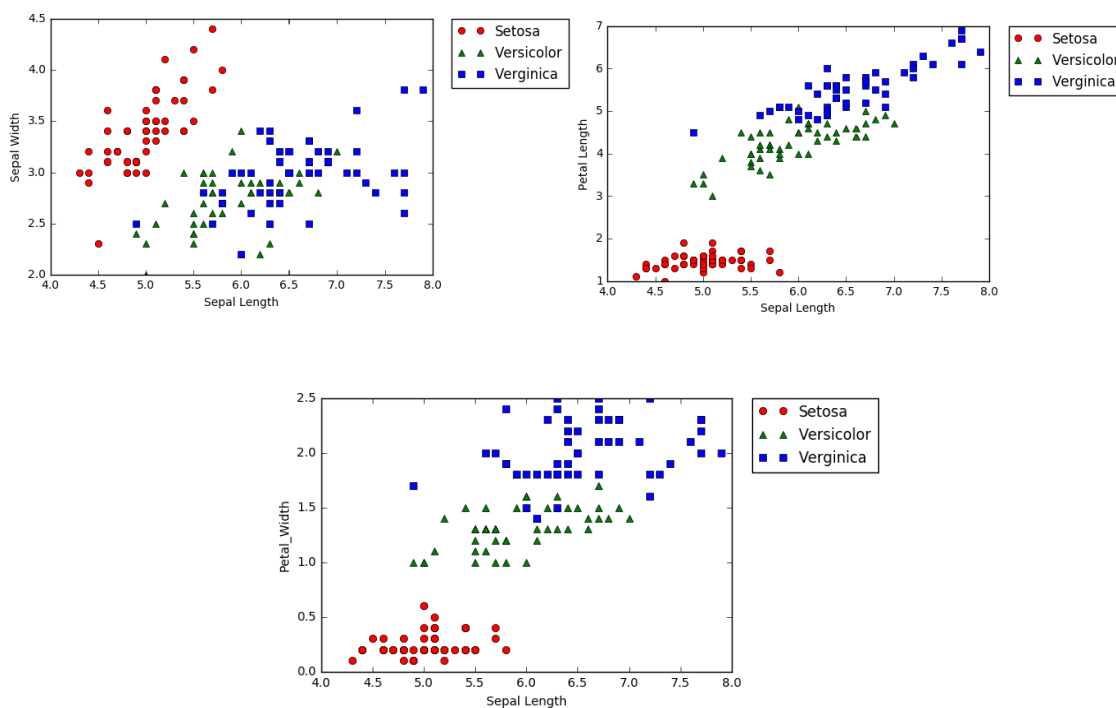


Рисунок 9 – Вывод графика для отображения различных проекций данных

Из графиков 9 уже хорошо видно, что множество Setosa хорошо отделимо, а множества Versicolor и Virginica представляют собой множества, разделение которых является непростой задачей.

Постройте другие проекции исходных данных. Сколько всего различных проекций можно построить для данного набора данных?

#### Важные замечания

1. Несмотря на кажущуюся простоту и «понятность» данных в результате визуализации, исследователь не должен делать поспешных выводов (например, было бы ошибочно делать вывод по рис. 1.9 о том, что ирисы Setosa те, у которых petal width менее 0,75). Следует помнить, что цель первичного исследования данных – получение представления о структуре и природе данных, а не построение модели предсказания, классификации и т.п.

1. В качестве среды разработки используйте языки программирования Python, Java или C#. По согласованию с преподавателем студент может самостоятельно выбрать язык программирования и среду разработки (при этом студенту необходимо критически обосновать свой выбор).

2. При выборе набора данных (data set) на ресурсах [5] необходимо согласовать свой выбор с другими студентами группы и преподавателем, так как работа над одинаковыми наборами данных недопустима.

3. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.

#### 4. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.
2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.
3. Ответы на контрольные вопросы.
4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

#### Контрольные вопросы

1. Какие инструментальные средства используются для организации рабочего места специалиста Data Science?

2. Какие библиотеки Python используются для работы в области машинного обучения? Дайте краткую характеристику каждой библиотеке.

3. Почему при реализации систем машинного обучения широкое распространение получили библиотеки Python?

Для работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1-7].

#### Список литературы

1. Уэс, Маккинли. Python и анализ данных Электронный ресурс / Маккинли Уэс ; пер. А. А. Слинкин. - Python и анализ данных, 2022-04-19. - Саратов : Профобразование,

2017. - 482 с.

2. Сузи, Р.А. Язык программирования Python Электронный ресурс : учебное пособие / Р.А. Сузи. - Язык программирования Python, 2020-07-28. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 350 с.

3. Стенли, Липпман. Язык программирования C++ Электронный ресурс : Полное руководство / Липпман Стенли, Лажойе Жози ; пер. А. Слинкин. - Язык программирования C++, 2022-04-19. - Саратов : Профобразование, 2017. - 1104 с.

4. <https://github.com/enikolaev/MMO> – Репозиторий с примерами кода из лабораторных работ.

5. <https://archive.ics.uci.edu/ml/index.html> – Репозиторий наборов данных для машинного обучения (Центр машинного обучения и интеллектуальных систем).

6. <https://www.kaggle.com> – Портал и система проведения соревнований по проблемам анализа данных.

7. <https://www.mockaroo.com> – Сайт для генерации наборов данных.

## 1. Цели и задачи

Цель работы: изучение программных средств для визуализации наборов данных.  
Основные задачи:

- установка и настройка matplotlib, seaborn;
- изучение основных типов графиков библиотеки matplotlib;
- изучение основных типов графиков библиотеки seaborn;
- получение навыков анализа данных по визуальным представлениям данных.

Перед выполнением работы необходимо ознакомиться с базовыми принципами языка Python, используя следующие источники [1-5]. Особое внимание необходимо уделить репозиторию [5] с исходными кодами.

### Методика и порядок выполнения работы

Выполним анализ набора данных «Предсказание ухода клиента». Данный набор данных используется в качестве учебного набора при изучении методов прогнозирования. Набор представляет собой данные об активности клиента о телекоммуникационной компании (количество часов разговоров, видеозвонков, ночные и дневные разговоры и прочие). Набор данных подходит для обучения моделей логистической регрессии, моделей классификации (CNN, kNN, Logic tree). Набор данных можно получить в репозитории [5] или на портале Kaggle [4].

Рассмотрим основные признаки, представленный в наборе. Загрузим набор данных с использованием pandas и выведем признаки набора данных (рисунок 1).

1	import numpy as np
2	import pandas as pd
3	from matplotlib import pyplot as plt
4	import seaborn as sns
5	%matplotlib inline

1	data_path = "../datasets/telecom_churn/telecom_churn.csv"
2	data = pd.read_csv(data_path)
3	data.head()

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	To d char
0	KS	128	415	No	Yes	25	265.1	110	45.
1	OH	107	415	No	Yes	26	161.6	123	27.
2	NJ	137	415	No	No	0	243.4	114	41.

Рисунок 1 – Загрузка данных и получение и первичный анализ признаков

Набор данных telecom\_churn.csv содержит большое количество признаков.

Для детального изучения воспользуемся методом info() класса DataFrame (рисунок 2).



```

1 data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
State                3333 non-null object
Account length       3333 non-null int64
Area code            3333 non-null int64
International plan    3333 non-null object
Voice mail plan       3333 non-null object
Number vmail messages 3333 non-null int64
Total day minutes     3333 non-null float64
Total day calls       3333 non-null int64
Total day charge      3333 non-null float64
Total eve minutes     3333 non-null float64
Total eve calls       3333 non-null int64
Total eve charge      3333 non-null float64
Total night minutes   3333 non-null float64
Total night calls     3333 non-null int64
Total night charge    3333 non-null float64
Total intl minutes    3333 non-null float64
Total intl calls      3333 non-null int64
Total intl charge     3333 non-null float64
Customer service calls 3333 non-null int64
Churn                 3333 non-null bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 498.1+ KB

```

Рисунок 2 – Информация о признаках набора данных

Графики, используемые при анализе данных, делят не по библиотекам, с использованием которых они строятся, а по типам признаков, для анализа которых предназначены графики.

### Визуализация количественных признаков

Для представления распределения простого количественного признака подходит обычная гистограмма, содержащаяся во всех библиотеках (рисунок 3).

```

3 data['Total day minutes'].hist();

```

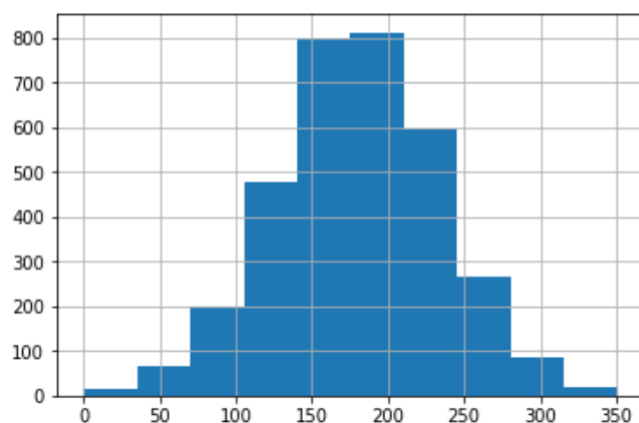


Рисунок 3 – Информация о признаках набора данных

Для построения гистограммы вызывается метод **hist()** класса **DataFrame**. На самом деле используется метод из библиотеки **matplotlib**. Метод **hist()** можно использовать для построения гистограмм по нескольким признакам (рисунок 4). При этом неколичественные признаки игнорируются.

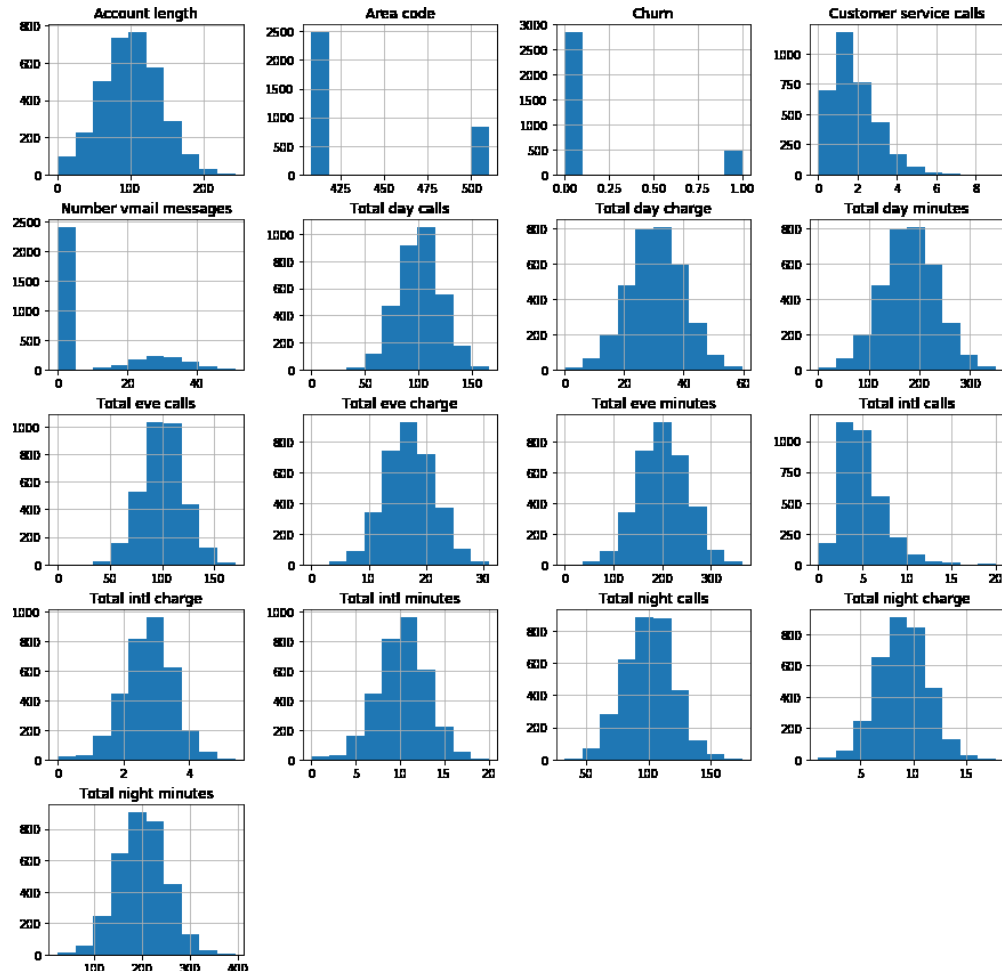


Рисунок 4 – Применение метода **hist()** для визуализации распределения нескольких признаков

Аналогичный тип графика можно получить с использованием **matplotlib** (рисунок 5). Если необходимо построить график распределения, аналогичный представленному на рисунке 2.3, то нужно выполнить дополнительные расчеты (рисунок 6).

```
1 plt.bar(data.index, data['Total day minutes'])
2 plt.show()
```

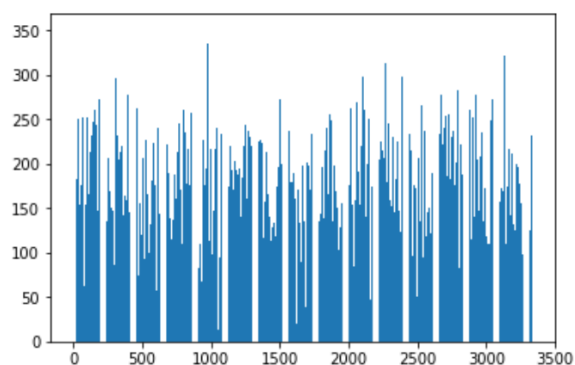


Рисунок 5 – Построение гистограммы с использованием **matplotlib**



```
1 hist = data['Total day minutes'].value_counts()
2 plt.bar(hist.index, hist);
```

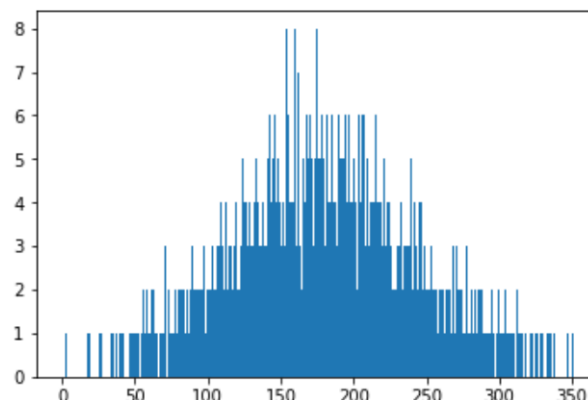


Рисунок 6 – Использование matplotlib для представления распределения значений признака

Один из эффективных типов графиков для анализа количественных признаков – это «ящик с усами» (boxplot). На рисунке 7 показан код и реализованный график. Для анализа нескольких признаков графики boxplot также эффективны. На рисунке 8 представлен код и результат построения графиков для анализа пяти штатов с максимальным объемом дневных звонков.

```
5 sns.boxplot(data['Total day minutes']);
```

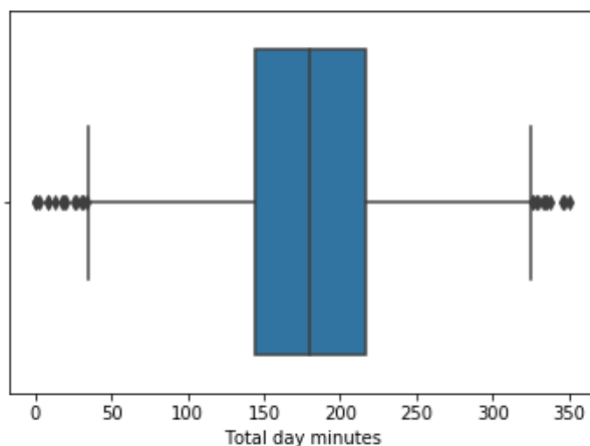


Рисунок 7 – График «ящик с усами» для отдельного признака

```

1 top_data = data[['State', 'Total day minutes']]
2 top_data = top_data.groupby('State').sum()
3 top_data = top_data.sort_values('Total day minutes', ascending=False)
4 top_data = top_data[:5].index.values
5 sns.boxplot(y='State',
6             x='Total day minutes',
7             data=data[data.State.isin(top_data)], palette='Set3');

```

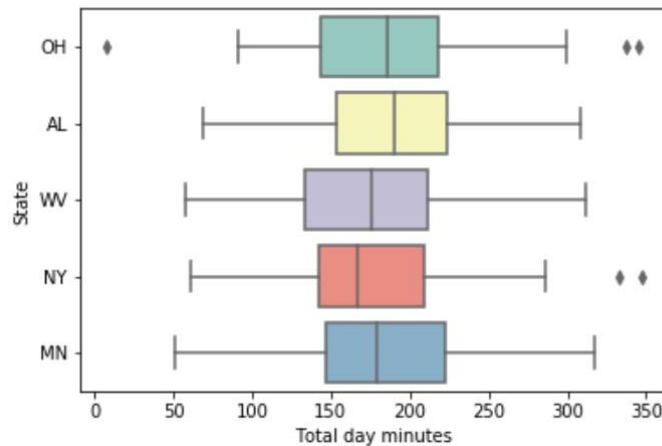


Рисунок 8 – Использование boxplot для анализа признака для пяти штатов

График boxplot состоит из коробки, усов и точек. Коробка показывает интерквартильный размах распределения, то есть соответственно 25% (первая квартиль,  $Q1$ ) и 75% ( $Q3$ ) перцентили. Черта внутри коробки обозначает медиану распределения (можно получить с использованием метода `median()` в `pandas` и `numpy`). Усы отображают весь разброс точек кроме выбросов, то есть минимальные и максимальные значения, которые попадают в промежуток  $(Q1 - 1,5 \cdot IQR, Q3 + 1,5 \cdot IQR)$ , где  $IQR = Q3 - Q1$  – интерквартильный размах.

Точками на графике обозначаются выбросы (outliers), то есть те значения, которые не вписываются в промежуток значений, заданный усами графика (рисунок 9).

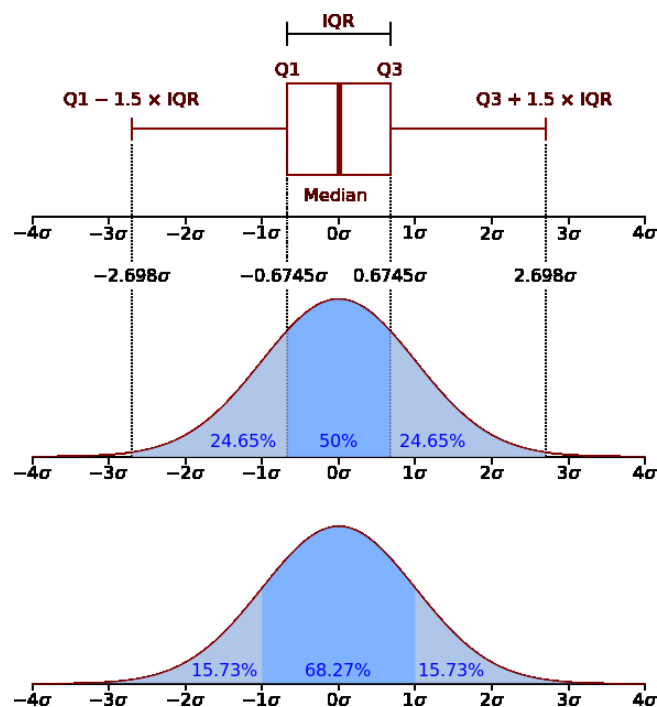
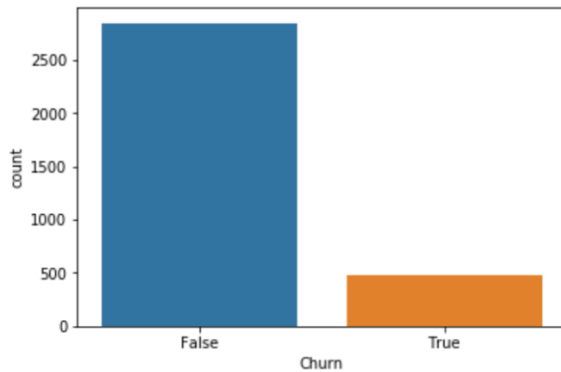


Рисунок 9 – Структура графика типа «ящик с усами»

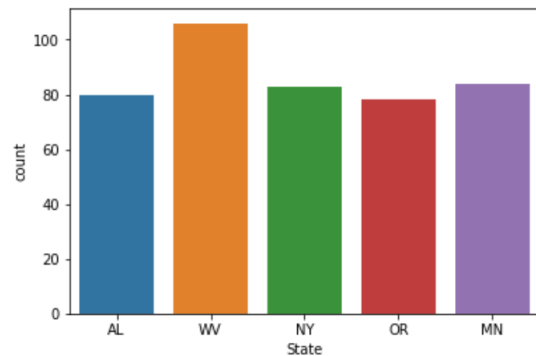
## Категориальные признаки

Типичным категориальным признаком в анализируемом наборе данных является «Штат» (State). Под категориальный признак подходит также «Отказ» (Churn) (хотя он является логическим). На рисунке 10 представлены графики типа `countplot()` из библиотеки `seaborn`, которые строят гистограммы, но не по сырым данным, а по рассчитанному количеству разных значений признака.

```
1 sns.countplot(data['Churn']);
```



```
1 # гистограмма "популярных" штатов
2 sns.countplot(data[data['State'].isin(data['State'].value_counts().head(5).index)]['State']);
```



а)

б)

Рисунок 10 – График `countplot`: а) визуализация распределения признака `Churn`; б) визуализация пяти популярных штатов

## Визуализация соотношения количественных признаков

Одним из вариантов визуализации соотношения количественных признаков является диаграмма по нескольким признакам (рисунки 4, 8). Рассмотрим пример демонстрирующий сравнение распределений показателей, связанных с финансовыми затратами клиентов. Упрощенно, можно сказать, что это все показатели, содержащие подстроку «charge» в имени показателя. На рисунке 11 представлен код для отбора требуемых показателей.

```

1 # Отбор числовых признаков, содержащих слово 'charge'
2 feats = [f for f in data.columns if 'charge' in f]
3 feats

['Total day charge',
 'Total eve charge',
 'Total night charge',
 'Total intl charge']

```

Рисунок 11 – Отбор показателей, связанных с затратами клиентов

После отбора интересующих показателей можно построить диаграммы для сравнения (рисунок 12).

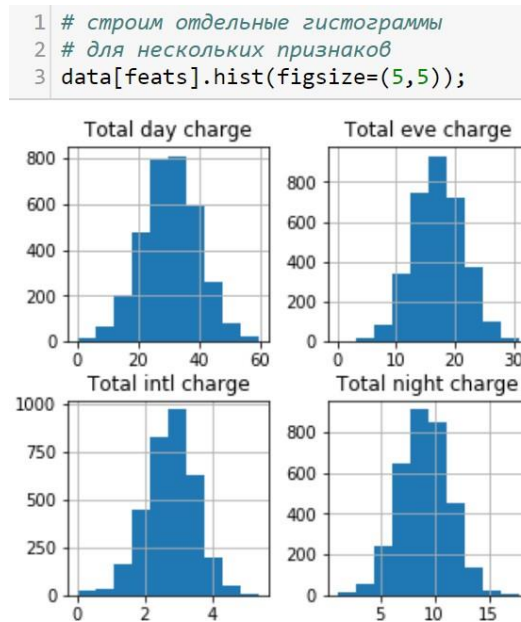


Рисунок 12 – Диаграммы для сравнения распределения числовых показателей

Часто используют попарное сравнение признаков для обеспечения широкого взгляда на набор данных (рисунок 13). На диагональных графиках рисунка 13 представлены гистограммы распределения отдельного признака, на недиагональных позициях – попарные распределения.

```

1 # Попарное распределение признаков
2 # Применение Seaborn
3 sns.pairplot(data[feats]);

```

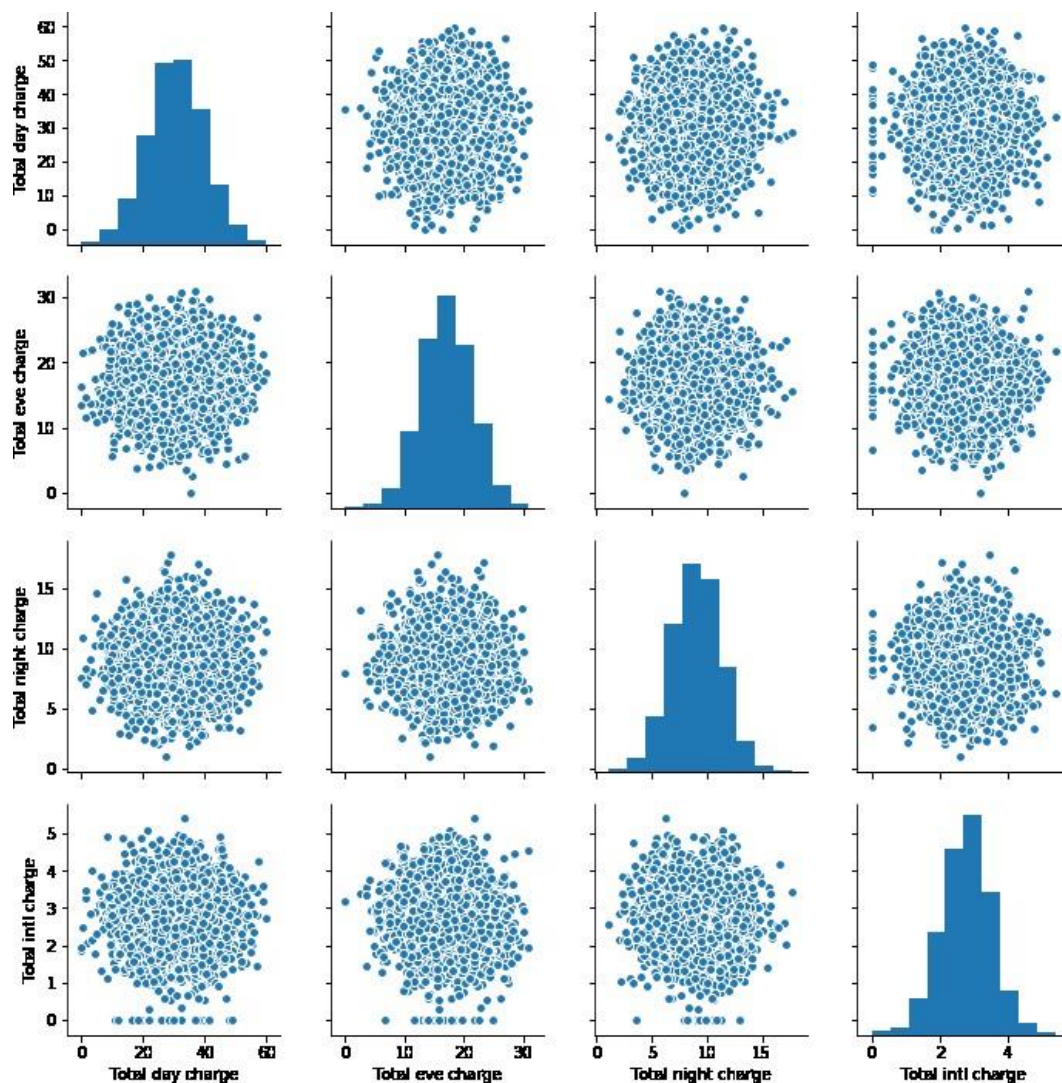


Рисунок 13 – Попарное распределение признаков

Можно реализовать более сложные графики. Например, если требуется добавить к существующим признакам, целевой признак Churn (количество отказов) и раскрасить разные типы элементов, то можно воспользоваться попарными распределениями, но с отображением подмножеств отказов (рисунок 14).

До сих пор использовались возможности библиотеки seaborn, а также методы pandas (которые производят визуализацию, обращаясь к библиотеке matplotlib). Библиотека matplotlib наиболее известная и широко применяемая при анализе данных в рамках стека технологий python.

```
1 sns.pairplot(data[feats + ['Churn']], hue='Churn');
```



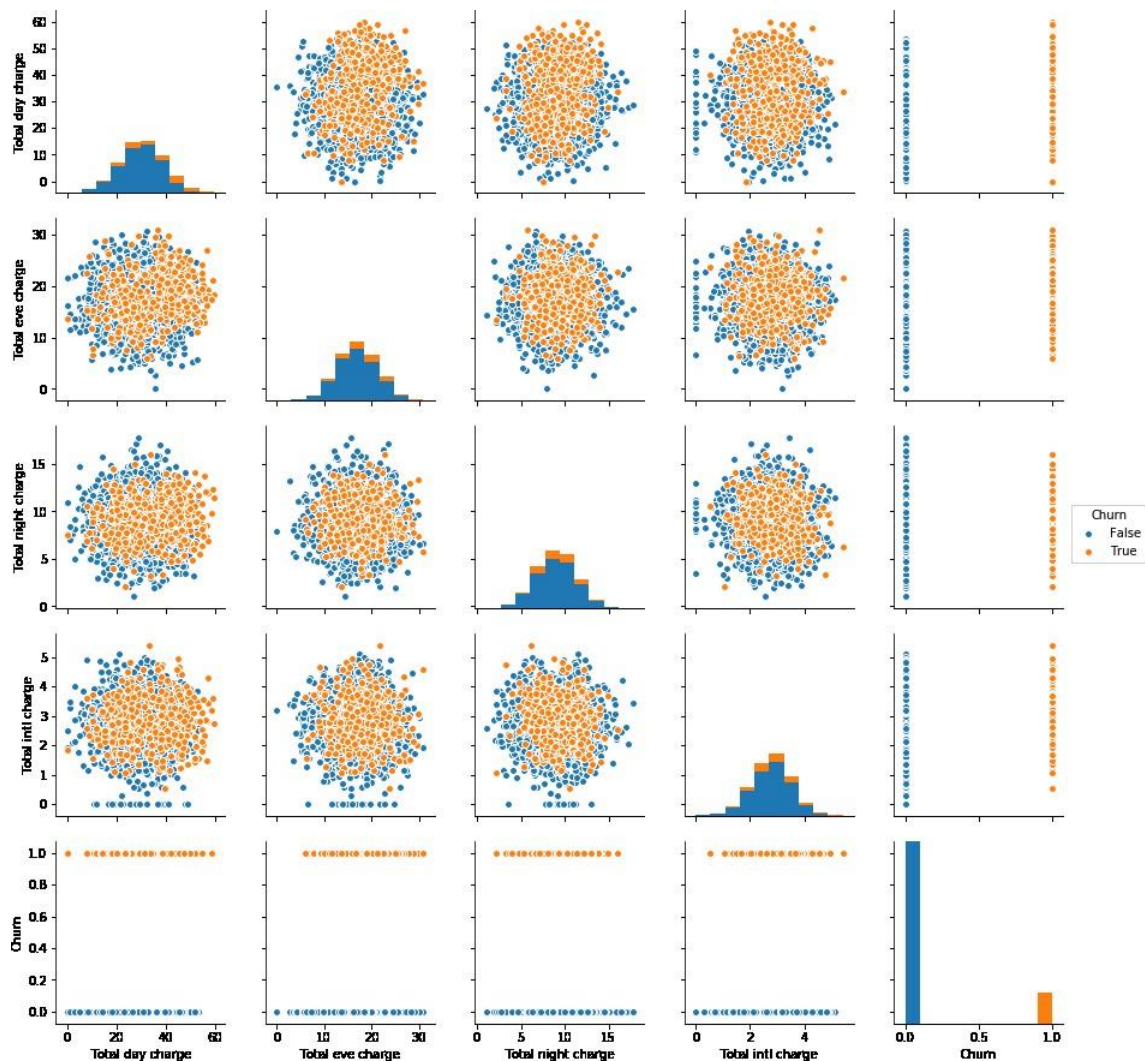


Рисунок 14 – Попарное распределение признаков с визуализацией отзовов

На рисунке 15 показан пример использования графика scatter библиотеки matplotlib, предназначенного для вывода множества точек.

```
1 plt.scatter(data['Total day charge'],
2             data['Total intl charge'],
3             color='lightblue', edgecolors='blue')
4 plt.xlabel('Дневные начисление')
5 plt.ylabel('Международн. начисление')
6 plt.title('Распределение по 2 признакам');
```

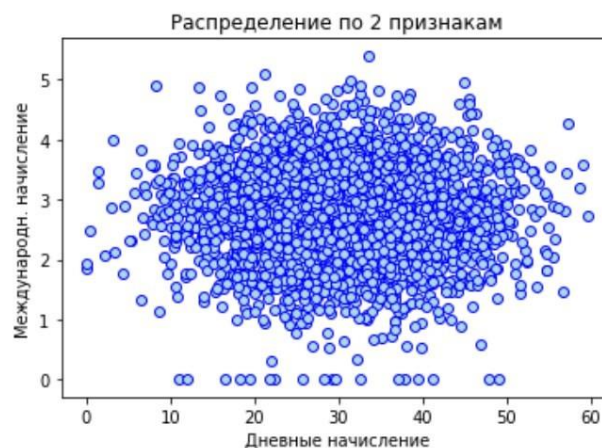


Рисунок 15 – График scatter библиотеки matplotlib

На рисунке 16 показан пример более тонкой настройки параметров графика.

```

1 # Раскрашивание данных
2 # Цвет в зависимости от ухода клиента
3 c = data['Churn'].map({False: 'lightblue', True: 'orange'})
4 edge_c = data['Churn'].map({False: 'blue', True: 'red'})
5 # Настройка графика
6 plt.scatter(data['Total day charge'], data['Total intl charge'],
7             color=c, edgecolors=edge_c
8             )
9 plt.xlabel('Дневные начисление')
10 plt.ylabel('Международн. начисление');

```

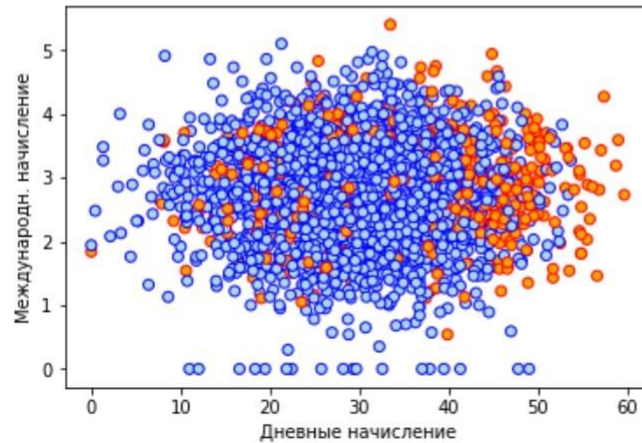


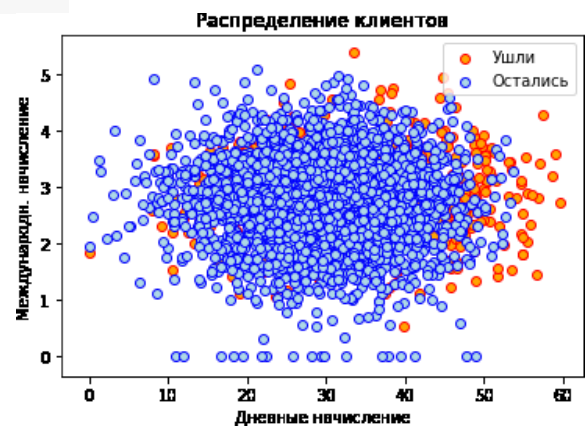
Рисунок 16 – Настройка графика: цвет точки зависит от целевого значения признака

График на рисунке 2.16 можно построить различными способами, например, можно добавлять множества точек отдельными подмножествами, указывая параметры визуализации для каждого подмножества (рисунок 17).

```

4 # Ушедшие клиенты
5 data_churn = data[data['Churn']]
6 # Оставшиеся клиенты
7 data_loyal = data[~data['Churn']]
8
9 plt.scatter(data_churn['Total day charge'],
10            data_churn['Total intl charge'],
11            color='orange',
12            edgecolors='red',
13            label='Ушли')
14
15 plt.scatter(data_loyal['Total day charge'],
16            data_loyal['Total intl charge'],
17            color='lightblue',
18            edgecolors='blue',
19            label='Остались')
20
21 plt.xlabel('Дневные начисление')
22 plt.ylabel('Международн. начисление')
23 plt.title('Распределение клиентов')
24 plt.legend();

```



а)

б)

Рисунок 17 – Построение отдельных подмножеств с легендой;  
а)исходный код; б) полученный график

В реальных задачах машинного обучения при первичном анализе данных необходимо выявить корреляции признаков обучающей выборки. В пакете Pandas имеется встроенный инструмент для этого – метод **corr()** класса DataFrame. На рисунке 18 показан фрагмент вывода этой функции.

```

1 # Применяется функция corr() из Pandas
2 data.corr()

```

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Tota
Account length	1.000000	-0.012463	-0.004628	0.006216	0.038470	0.006214	-0.006757	0.01
Area code	-0.012463	1.000000	-0.001994	-0.008264	-0.009646	-0.008264	0.003580	-0.01
Number vmail messages	-0.004628	-0.001994	1.000000	0.000778	-0.009548	0.000776	0.017562	-0.00
Total day minutes	0.006216	-0.008264	0.000778	1.000000	0.006750	1.000000	0.007043	0.01

Рисунок 18 – Определение коррелирующих признаков набора данных

Полученная матрица имеет размер  $17 \times 17$ . Это незначительный размер (в реальных задачах машинного обучения размеры матриц корреляции имеют порядки  $10^6 - 10^{10}$  и более),



но даже для матрицы рассматриваемого набора данных проанализировать корреляцию признаков вручную – трудоемкая задача. Например, можно использовать скрипты, для выделения больших коэффициентов корреляции. Но лучше использовать специальный тип графика – heatmap (рисунок 19).

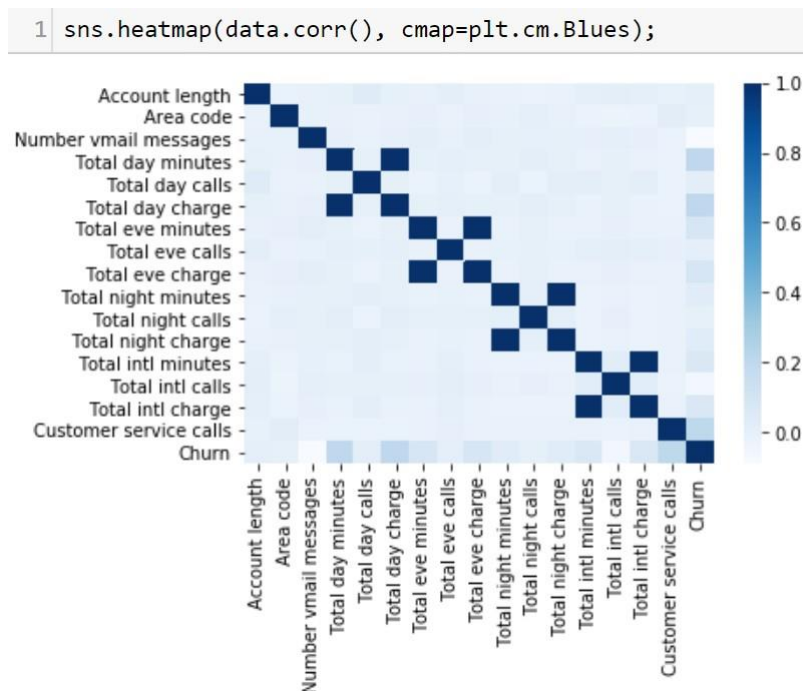


Рисунок 19 – Визуализация матрица корреляции с использованием графика типа heatmap

Коррелирующие признаки обычно удаляются и не рассматриваются в процессе обучения.

## Важные замечания

1. Статья о типах графиков при первичном анализе данных: <https://medium.com/open-machine-learning-course/open-machine-learning-course-topic-2-visual-data-analysis-in-python-846b989675cd>

2. В качестве среды разработки используйте языки программирования Python, Java или C#. По согласованию с преподавателем студент может самостоятельно выбрать язык программирования и среду разработки (при этом студенту необходимо критически обосновать свой выбор).

2. При выборе набора данных (data set) на ресурсах [3, 4] необходимо согласовать свой выбор с другими студентами группы и преподавателем, так как работа над одинаковыми наборами данных недопустима.

3. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.

## Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.
2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.
3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

### Контрольные вопросы

1. Какие инструментальные средства используются для организации рабочего места специалиста Data Science?
  2. Какие библиотеки Python используются для работы в области машинного обучения? Дайте краткую характеристику каждой библиотеке.
  3. Почему при реализации систем машинного обучения широкое распространение получили библиотеки Python?
  4. Перечислите функции Python, которые были изучены в рамках данной лабораторной работы и которые используются для визуализации данных.
  5. Какая библиотека python предназначена для управления наборами данных: numpy, pandas, sklearn, opencv, matplotlib?
  6. Какая стратегия является нежелательной при обработке пропусков в данных?
    - а) замена пропущенных значений в столбце медианным значением по данному столбцу;
    - б) удаление строк, содержащих пропуски в данных;
    - в) замена пропущенных значений в столбце средним арифметическим значением по данному столбцу;
    - г) замена пропущенных значений в столбце наиболее часто встречающимся значением по данному столбцу;
  7. Обоснуйте ответ на следующую проблему предварительной обработки данных: имеется независимая категориальная переменная  $y$ , которая представляет собой категориальный признак, определенный на домене {C#, Java, Python, R}. Нужно ли применять к данному целевому признаку OneHotEncoder?
  8. Поясните принцип разбиения набора данных на обучающую и тестовую выборку. Какое соотношение «тестовая:обучающая» наиболее оптимально: 20:80, 50:50, 25:75, 5:95, 40:30?
  9. Какой код лучше использовать при загрузке данных из csv-файла?
    - а) `dataset = read_csv("data.csv")`
    - б) `dataset = import("data.csv")`
    - в) `dataset = read.csv("data.csv")`
    - г) `dataset = import.csv("data.csv")`
    - д) `dataset = read_xls("data.csv")`
- Для выполнения работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1-4]

### Список литературы

1. Уэс, Маккинли. Python и анализ данных Электронный ресурс / Маккинли Уэс ; пер. А. А. Слинкин. - Python и анализ данных, 2022-04-19. - Саратов : Профобразование, 2017. - 482 с.
2. Сузи, Р.А. Язык программирования Python Электронный ресурс : учебное пособие / Р.А. Сузи. - Язык программирования Python, 2020-07-28. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 350 с.
3. Стенли, Липпман. Язык программирования C++ Электронный ресурс : Полное руководство / Липпман Стенли, Лажойе Жози ; пер. А. Слинкин. - Язык программирования C++, 2022-04-19. - Саратов : Профобразование, 2017. - 1104 с.

4. <https://github.com/enikolaev/MMO> – Репозиторий с примерами кода из лабораторных работ.
5. <https://archive.ics.uci.edu/ml/index.html> – Репозиторий наборов данных для машинного обучения (Центр машинного обучения и интеллектуальных систем).
6. <https://www.kaggle.com> – Портал и система проведения соревнований по проблемам анализа данных.
7. <https://www.mockaroo.com> – Сайт для генерации наборов данных.

## Практическое занятие 5. Диаграммы в SEABORN

Seaborn - интерфейс к Matplotlib для создания и оформления статистических диаграмм из наборов данных Pandas.

Для примера воспользуемся набором данных об автомобилях от Kaggle под лицензией Open database. Код ниже импортирует необходимые библиотеки, устанавливает стиль и загружает набор данных.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')
sns.set(font_scale=1.3)
cars = pd.read_csv('edited_cars.csv')
```

Элементы Seaborn делятся на две группы: оси и диаграммы.

- Оси имитируют Matplotlib и с помощью параметра `ax` могут объединяться в поддиаграммы. Они возвращают объект `axes` и используют обычные функции стилизации Matplotlib.
- Диаграммы управляют диаграммой, создать можно только диаграммы с какими-то значениями и связанные поддиаграммы. Они *не* поддерживают параметр `ax` и возвращают объекты FacetGrid, PairGrid или JointGrid. Используют разные стили и кастомизацию входных данных.

Изучение отношений между числовыми столбцами

Значения числовых функций — это непрерывные данные или числа. Первые визуализации будут матричными. Чтобы отобразить все попарные распределения на одной диаграмме, в них передаётся весь фрейм данных.

### 1. Парные диаграммы

Pairplot для сравнения распределения пар числовых переменных создаёт сетку точечных диаграмм. Он также содержит гистограмму для каждой функции в диагональных прямоугольниках.

Описание в документации

```
# Функции:
sns.pairplot() # диаграмма.
```

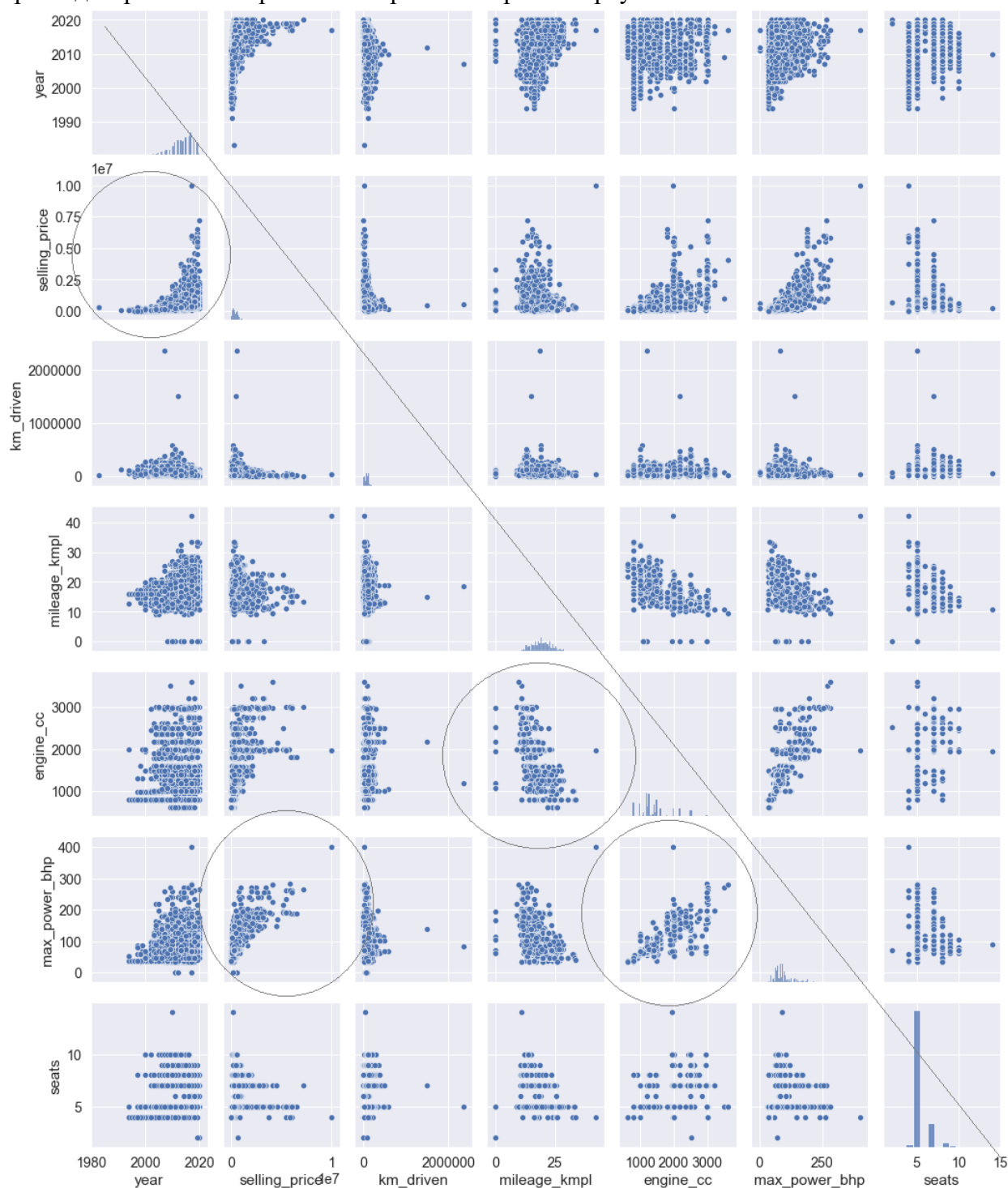
Код примера:

```
sns.pairplot(cars);
```

Стоит обратить внимание на:

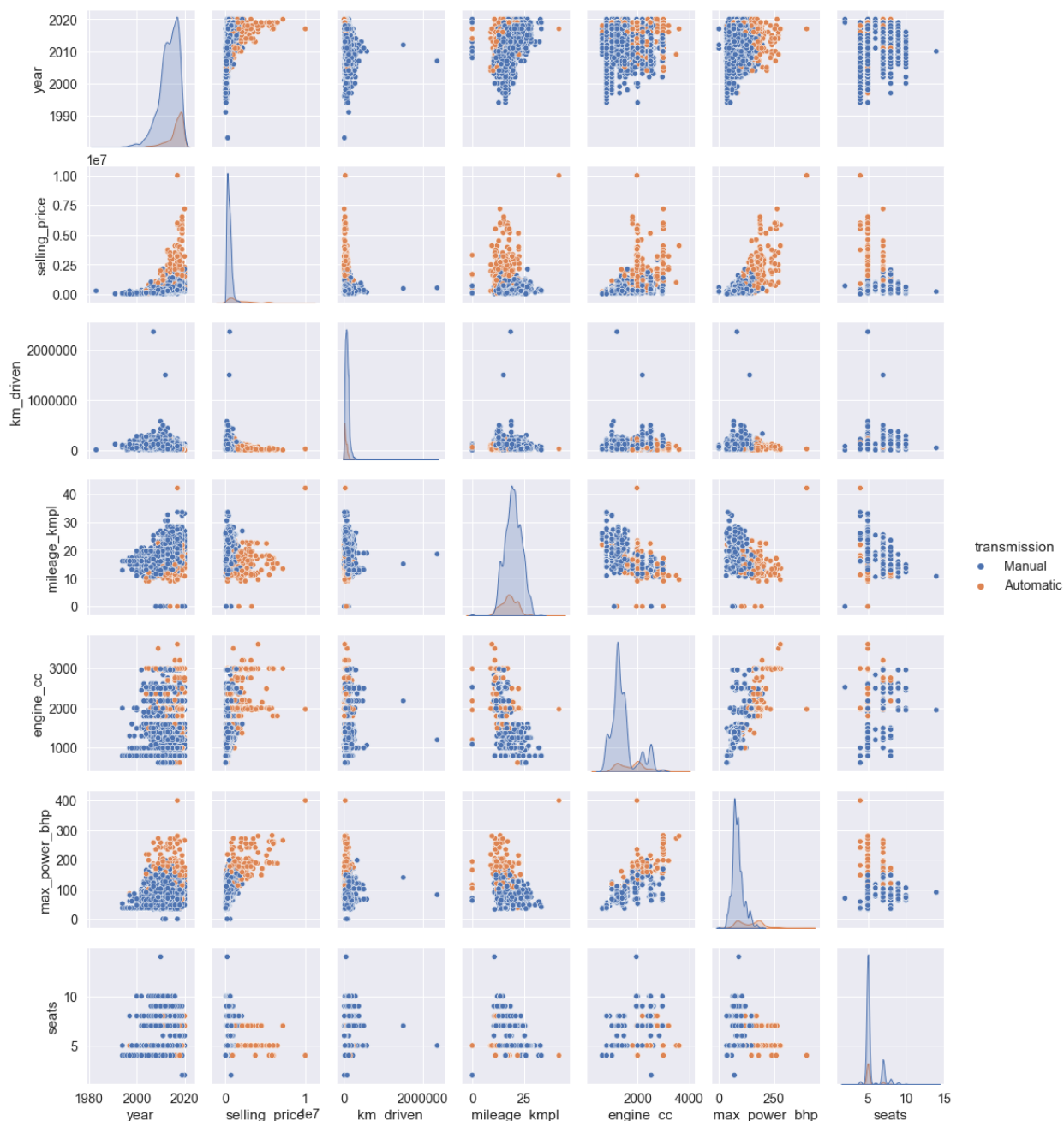
- Диаграммы рассеяния, показывающие положительные линейные отношения (когда  $x$  увеличивается, увеличивается  $y$ ), либо отрицательные (когда  $x$  увеличивается,  $y$  уменьшается).
- Гистограммы в диагональных прямоугольниках, показывающие распределение конкретных признаков.

На парной диаграмме обведённые диаграммы показывают очевидную линейную зависимость. Диагональ указывает на гистограммы каждого признака; левый треугольник парной диаграммы — зеркальное отражение правого треугольника.



Пример:

```
sns.pairplot(
    data=cars,
    aspect=.85,
    hue='transmission'); # hue='cat_col' подсвечивает указанную категорию другим цветом.
```



Стоит обратить внимание на кластеры разных цветов на точечных диаграммах

## 2. Тепловая карта

Тепловая карта — графическое представление значений в сетке цветовой кодировкой. Нанесённые значения — это коэффициенты корреляции пар, отражающие меру линейных отношений, поэтому диаграмма идеальна для пары. Тепловая карта отображает значения с помощью цвета, тогда как парная диаграмма показывает интуитивно понятные тенденции данных.

Описание в документации

```
# Функции:
sns.heatmap() # оси.
```

Пример:

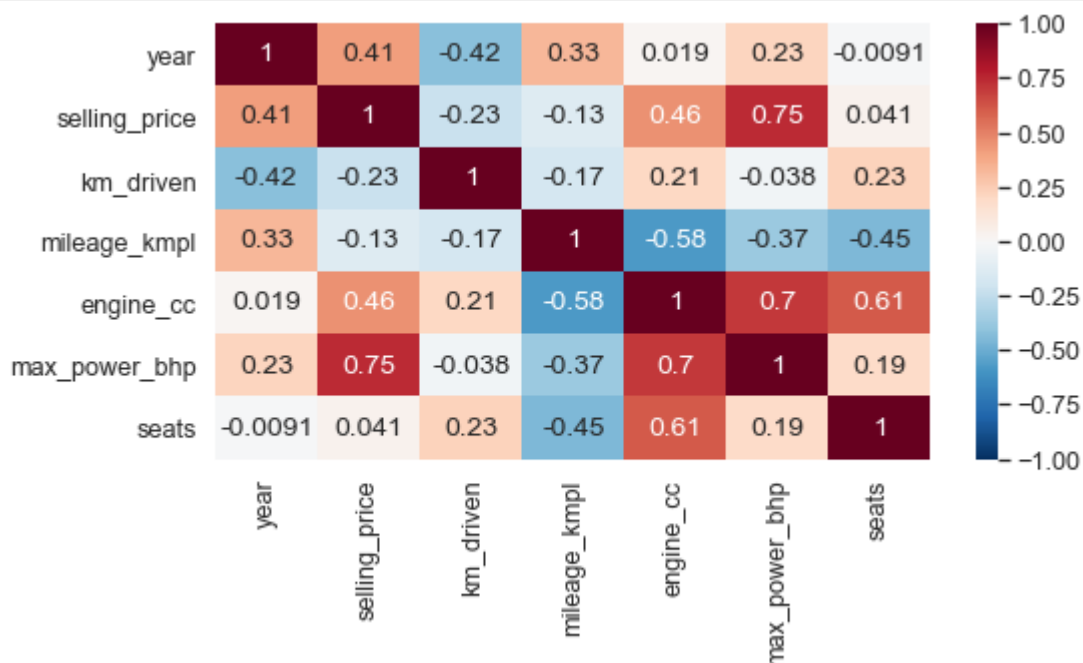
```
# Сначала запускаем df.corr(), чтобы получить таблицу коэффициентов корреляции:
```

```
cars.corr()
```

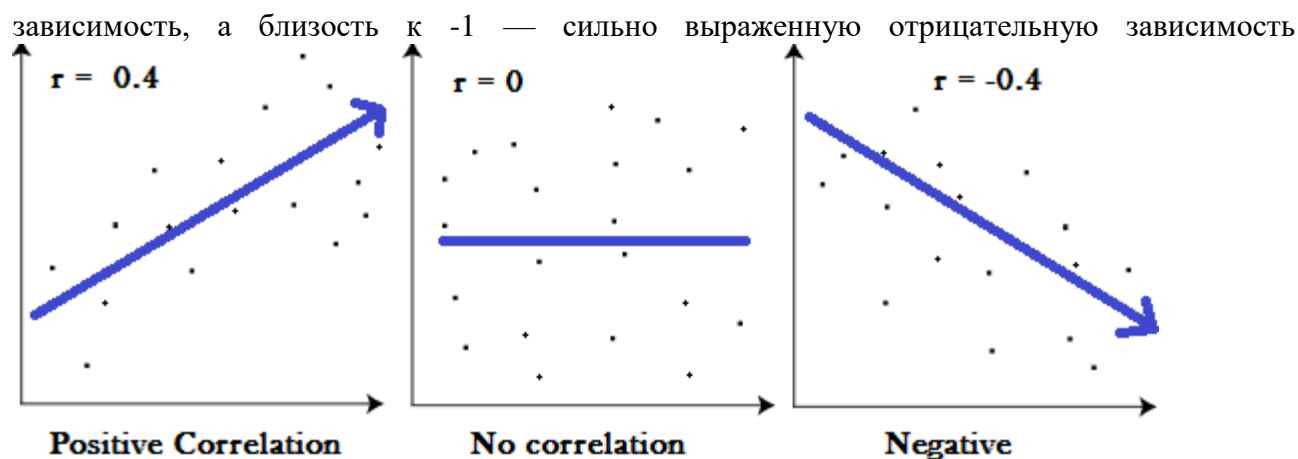
	year	selling_price	km_driven	mileage_kmpl	engine_cc	max_power_bhp	seats
year	1.000000	0.414092	-0.418006	0.329145	0.018848	0.226320	-0.009144
selling_price	0.414092	1.000000	-0.225534	-0.126054	0.455734	0.748489	0.041358
km_driven	-0.418006	-0.225534	1.000000	-0.173073	0.205914	-0.038075	0.227336
mileage_kmpl	0.329145	-0.126054	-0.173073	1.000000	-0.575831	-0.374621	-0.452085
engine_cc	0.018848	0.455734	0.205914	-0.575831	1.000000	0.703975	0.610309
max_power_bhp	0.226320	0.748489	-0.038075	-0.374621	0.703975	1.000000	0.191999
seats	-0.009144	0.041358	0.227336	-0.452085	0.610309	0.191999	1.000000

Эту таблицу называют матрицей корреляций. Эта таблица не очень понятна, поэтому с помощью `sns.heatmap()` отрисуем тепловую карту:

```
sns.set(font_scale=1.15)
plt.figure(figsize=(8,4))
sns.heatmap(
    cars.corr(),
    cmap='RdBu_r', # задаёт цветовую схему
    annot=True, # рисует значения внутри ячеек
    vmin=-1, vmax=1); # указывает начало цветовых кодов от -1 до 1.
```



Стоит обратить внимание на признаки с высокой корреляцией — тёмно-красные и тёмно-синие клетки. Близость к единице означает сильно выраженную положительную линейную



### 3. Диаграмма рассеяния

Диаграмма рассеяния показывает взаимосвязь между двумя числовыми признаками с помощью точек, показывающих движение этих переменных вместе.

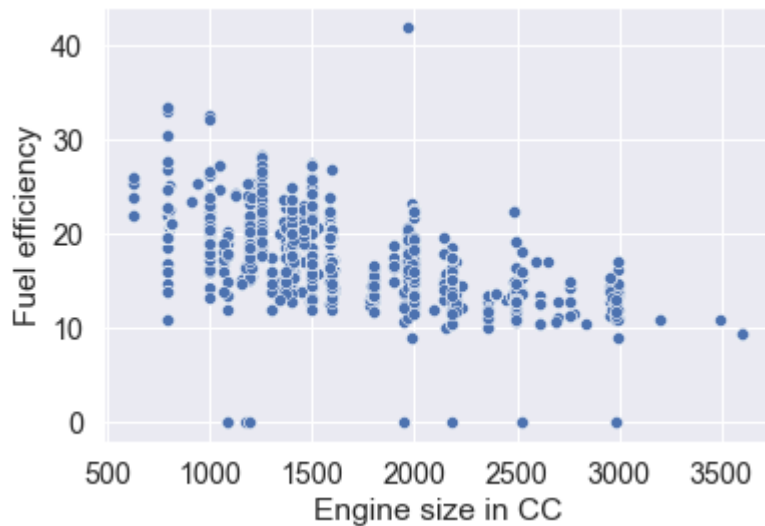
Описание в документации.

```
# Функции:
sns.scatterplot() # оси.
sns.relplot(kind='line') # диаграмма.
# Функции с линией регрессии:
sns.regplot() # оси.
sns.lmplot() # диаграмма.
```

Пример:

```
# Покажем объём двигателя и пробег автомобиля с помощью
sns.scatterplot(x='num_col1', y='num_col2', data=df)
sns.set(font_scale=1.3)
sns.scatterplot(
    x='engine_cc',
    y='mileage_kmpl',
    data=cars)
plt.xlabel(
    'Engine size in CC')
plt.ylabel(
    'Fuel efficiency')
```



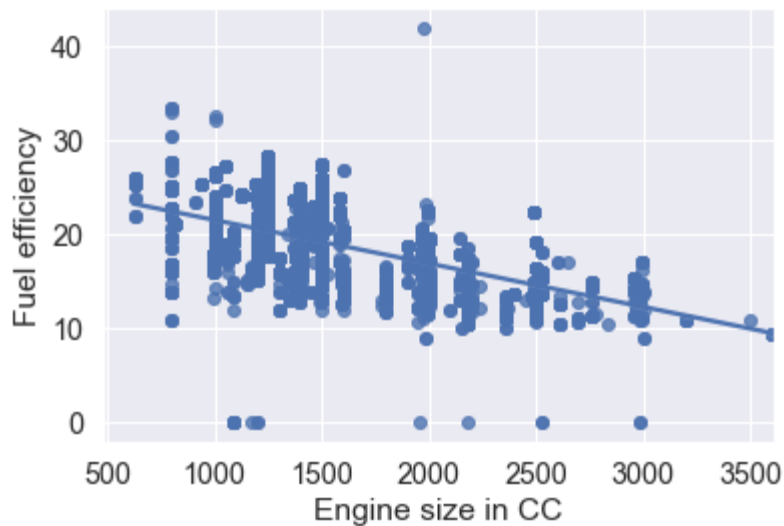


Описание в документации.

Функция `regplot` рисует диаграмму рассеяния с линией регрессии, показывающей тенденцию в данных.

Код:

```
sns.regplot(
    x='engine_cc',
    y='mileage_kmpl',
    data=cars)
plt.xlabel(
    'Engine size in CC')
plt.ylabel(
    'Fuel efficiency')
```

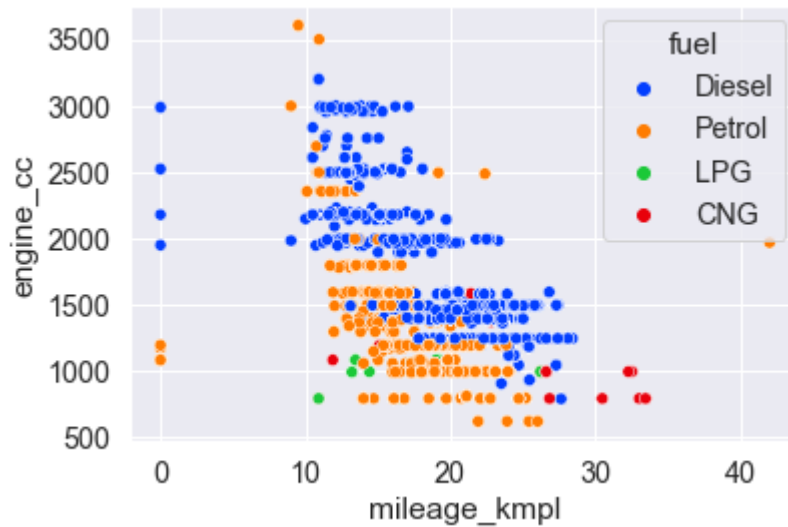


Описание в документации.

Код:

```
sns.scatterplot(
    x='mileage_kmpl',
    y='engine_cc',
    data=cars,
```

```
palette='bright',  
hue='fuel'); # подсвечивает указанную категорию другим цветом
```

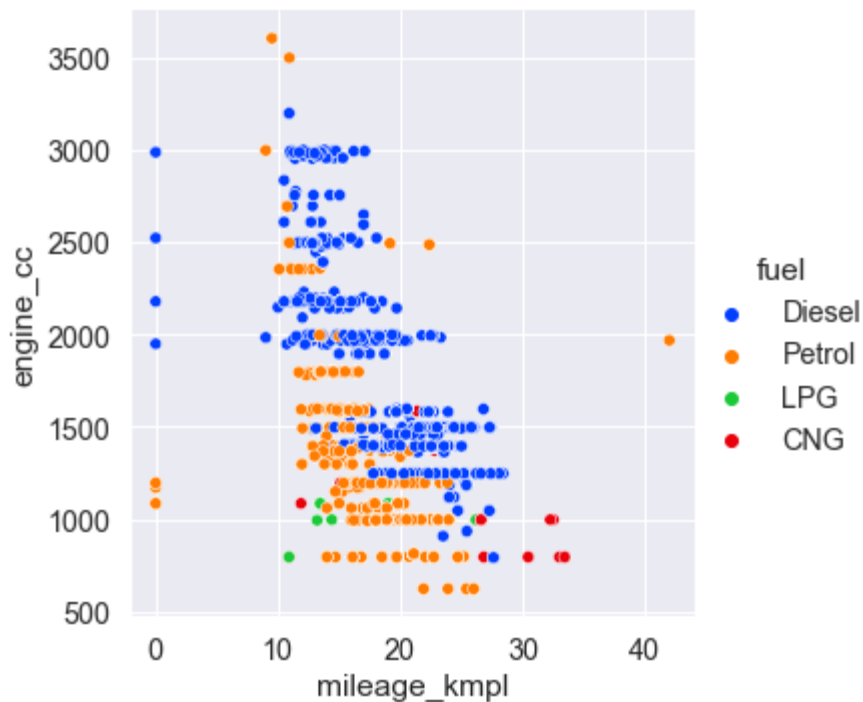


Описание в документации.

Диаграмма отношений relplot используется для создания диаграммы рассеяния с помощью `kind='scatter'` (установленного по умолчанию), или линейной диаграммы (`kind='line'`). Для разделения по цвету в `kind='scatter'` используется `hue='cat_col'`.

Код примера:

```
sns.relplot(  
    x='mileage_kmpl',  
    y='engine_cc',  
    data=cars,  
    palette='bright',  
    kind='scatter',  
    hue='fuel');
```

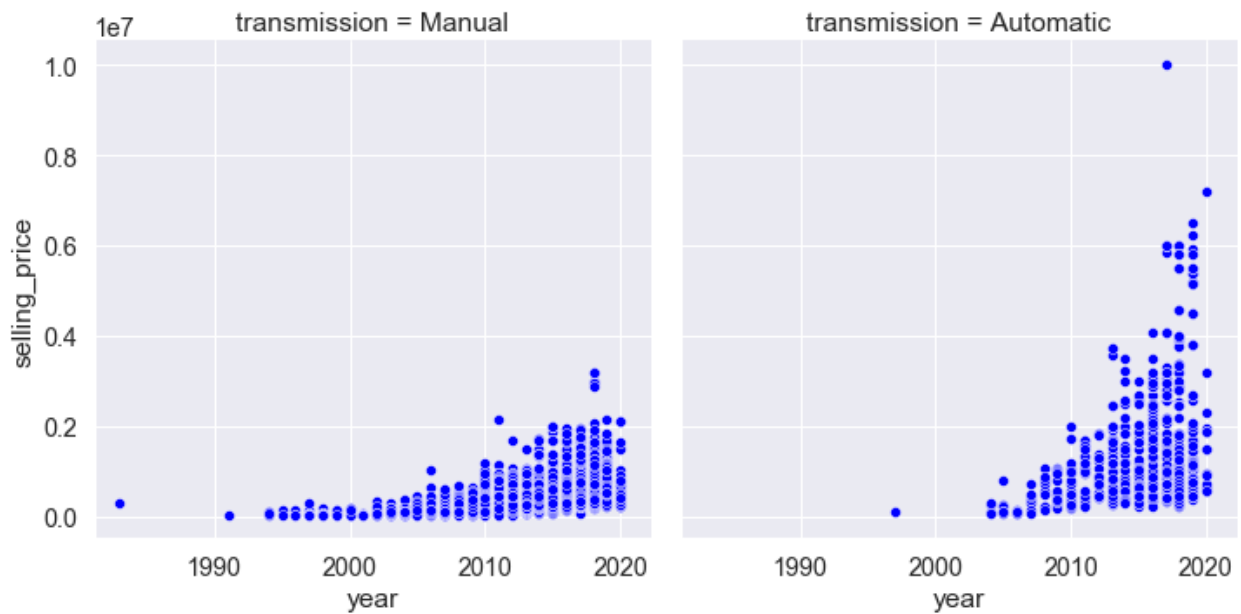


Обратите внимание, что две диаграммы выше аналогичны.

```
sns.relplot(x, y, data, kind='scatter', col='cat_col') # можно создавать поддиаграммы
сегментов по столбцам, используя col='cat_col' и/или по строкам с помощью row='cat_col'.
```

Пример:

```
sns.relplot(
    x='year',
    y='selling_price',
    data=cars,
    kind='scatter',
    col='transmission'); # данные разбиты на разные диаграммы по типу передачи
автомобиля
```

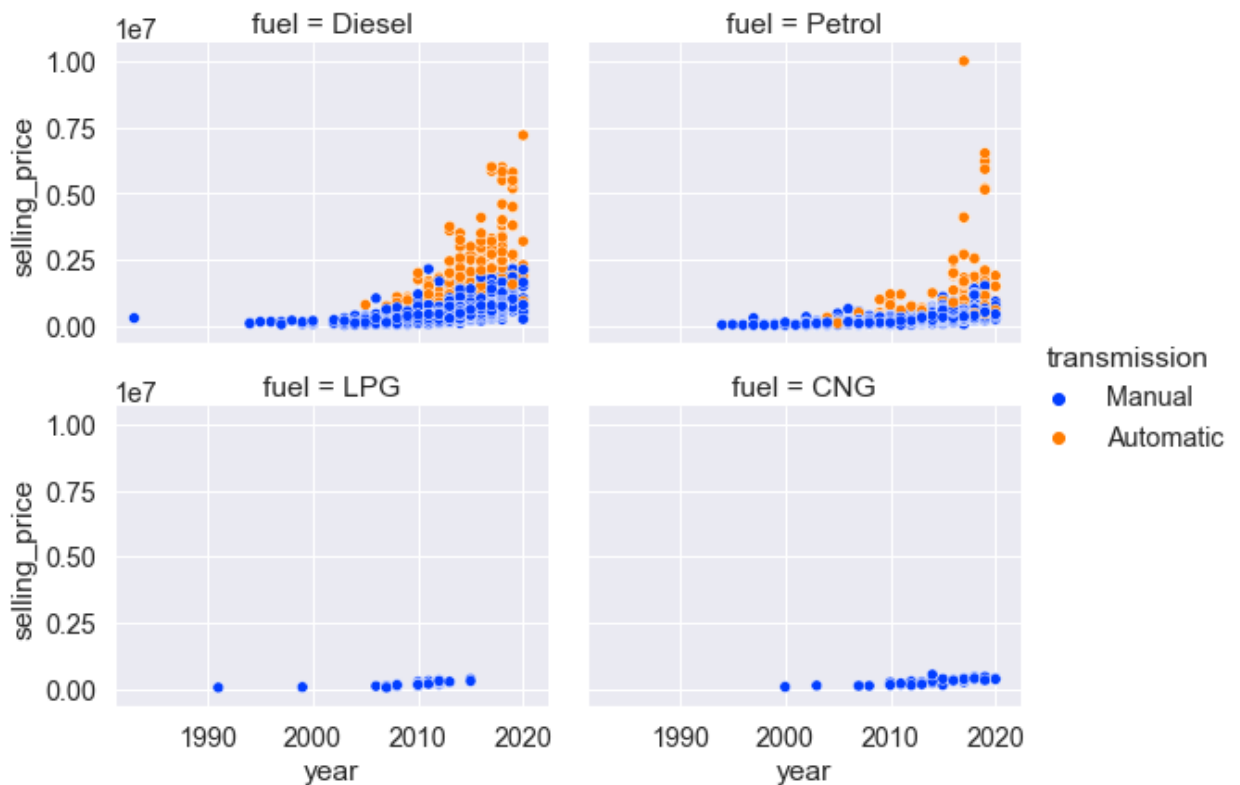


Описание в документации.

```
sns.relplot(x,y,data, hue='cat_coll1', col='cat_col2') #
```

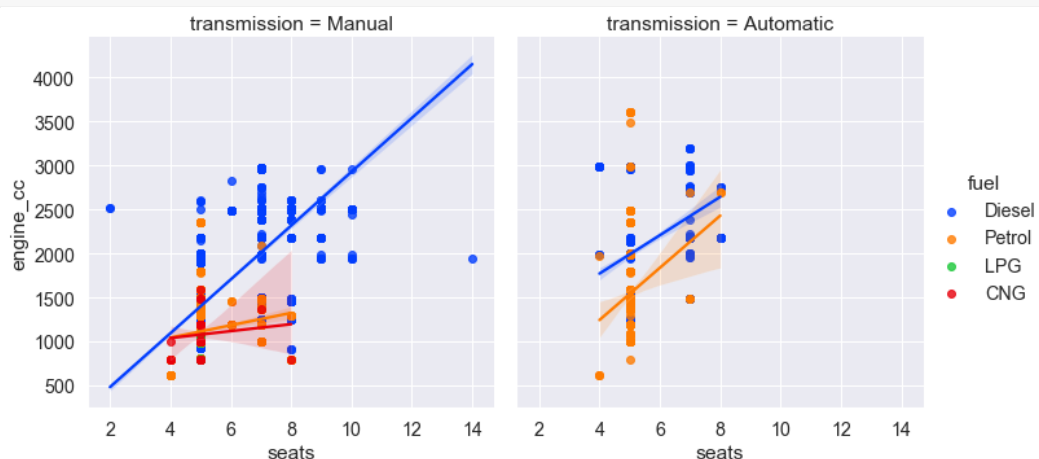
Код примера:

```
sns.relplot(
    x='year',
    y='selling_price',
    data=cars,
    palette='bright',
    height=3, aspect=1.3,
    kind='scatter',
    hue='transmission',
    col='fuel',
    col_wrap=2); # указывает количество столбцов для измерений в одной визуализации
```



Описание в документации. `lplot` — разновидность `regplot` на уровне диаграммы: она рисует диаграмму рассеяния с линией регрессии на `FacetGrid`. Параметра `kind` у `lplot` нет. Код примера:

```
sns.lplot(
    x="seats",
    y="engine_cc",
    data=cars,
    palette='bright',
    col="transmission",
    hue="fuel");
```



#### 4. Линейный график

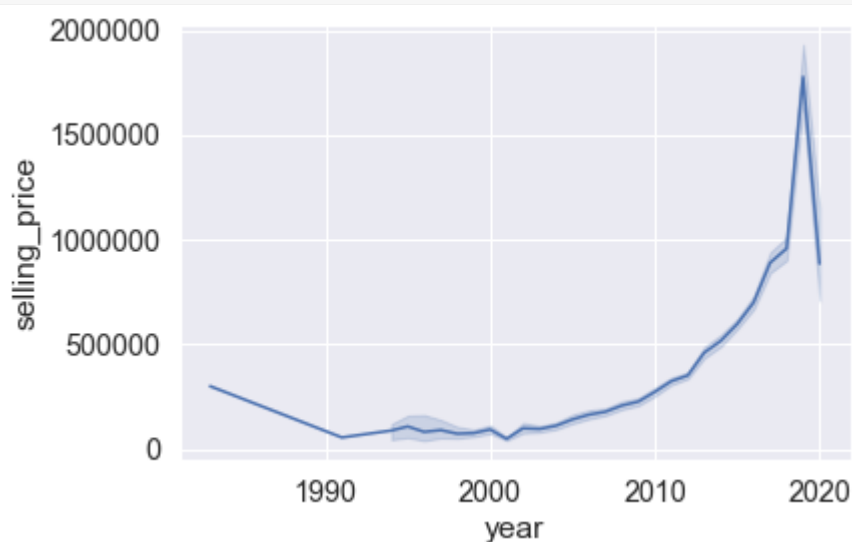
Линейный график состоит из точек, соединённых линией, которая показывает связь между переменными x и y. Ось x обычно содержит временные интервалы, ось y — числовую переменную, изменения которой во времени нужно отследить.

Описание в документации.

```
# Функции:  
sns.lineplot() # оси.  
sns.relplot(kind='line') # диаграмма.
```

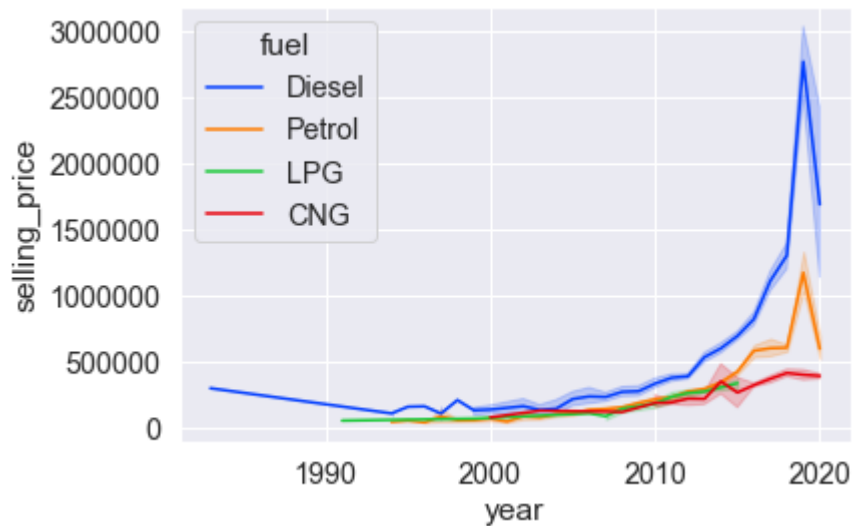
Код примера:

```
sns.lineplot(  
    x="year",  
    y="selling_price",  
    data=cars)
```



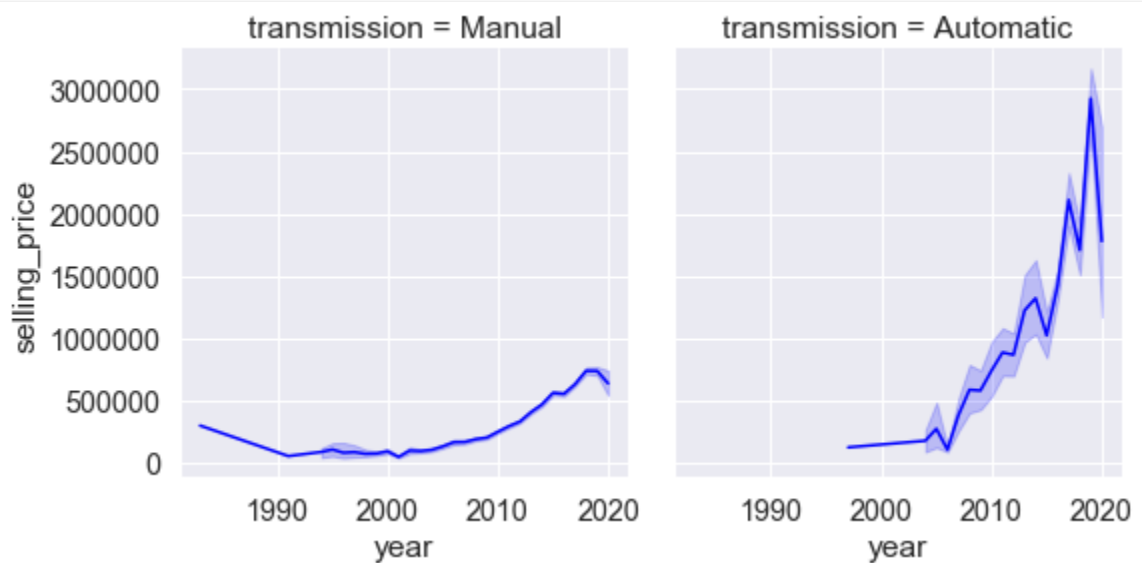
Пример:

```
sns.lineplot(  
    x="year",  
    y="selling_price",  
    data=cars,  
    palette='bright',  
    hue='fuel');
```



Пример:

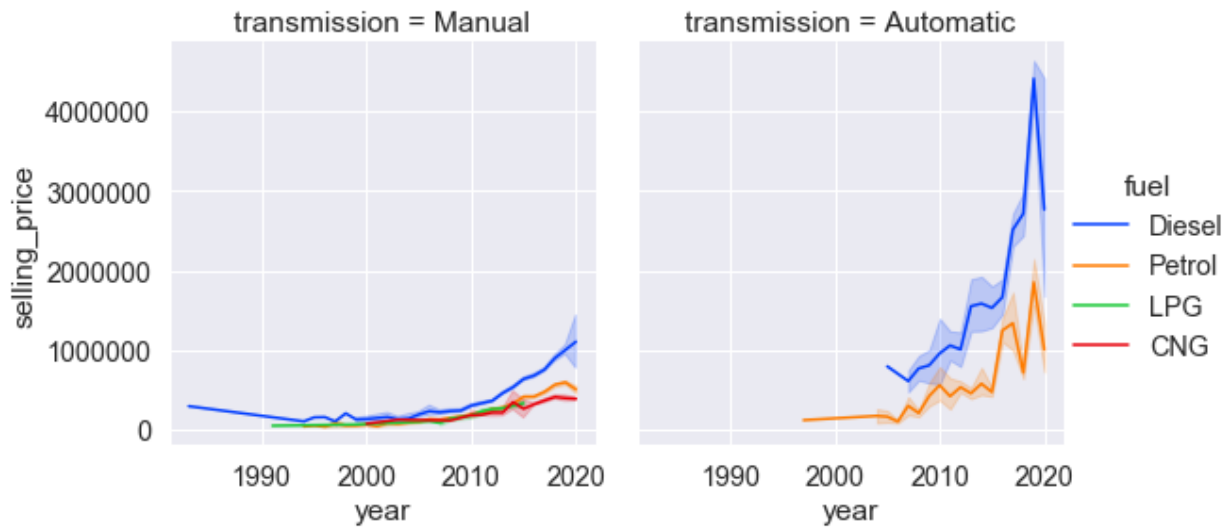
```
sns.relplot(
    x="year",
    y="selling_price",
    data=cars,
    color='blue', height=4
    kind='line', # строит линейный график
    col='transmission'); # рисует график отношений двух классов 'transmission'.
```



# Подобные диаграммы можно получить, используя kind='line' и hue:

```
sns.relplot(
    x="year",
    y="selling_price",
    data=cars,
    palette='bright',
    height=4,
    kind='line',
    col='transmission',
```

```
hue="fuel");
```



### 5. Сводная диаграмма

Состоит из трёх диаграмм в одной. Центр содержит бивариантную зависимость между переменными  $x$  и  $y$ . На диаграммах сверху и справа показано одномерное распределение переменных по осям  $x$  и  $y$  соответственно.

Описание в документации.

# Функции:

`sns.jointplot()` # диаграмма.

`sns.jointplot(x='num_col1', y='num_col2', data=df)`

# по умолчанию центральная диаграмма — это диаграмма рассеяния (`kind='scatter'`)

# Боковые диаграммы — это гистограммы.

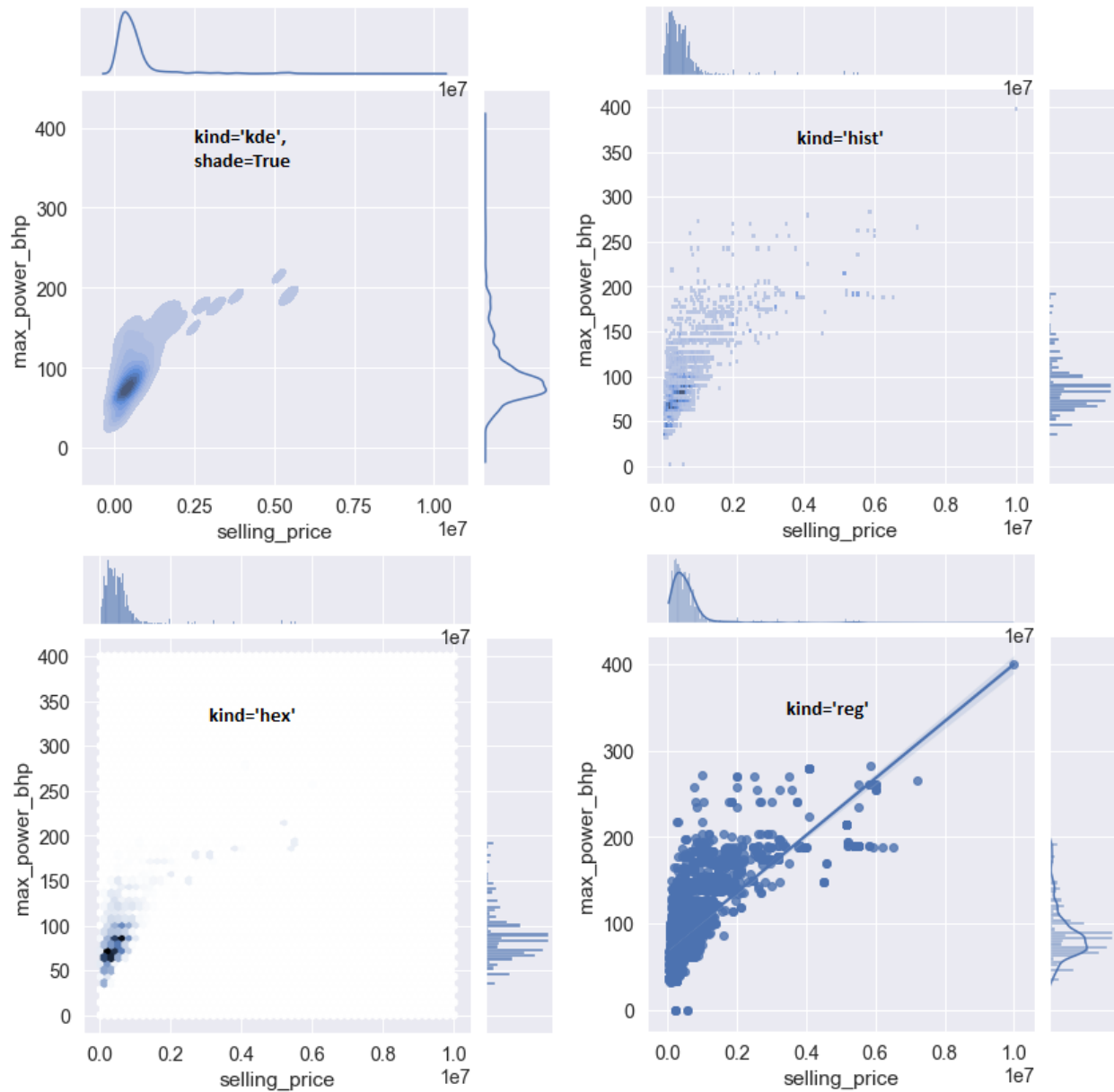
Код примера:

```
sns.jointplot(  
    x='max_power_bhp',  
    y='selling_price',  
    data=cars);
```



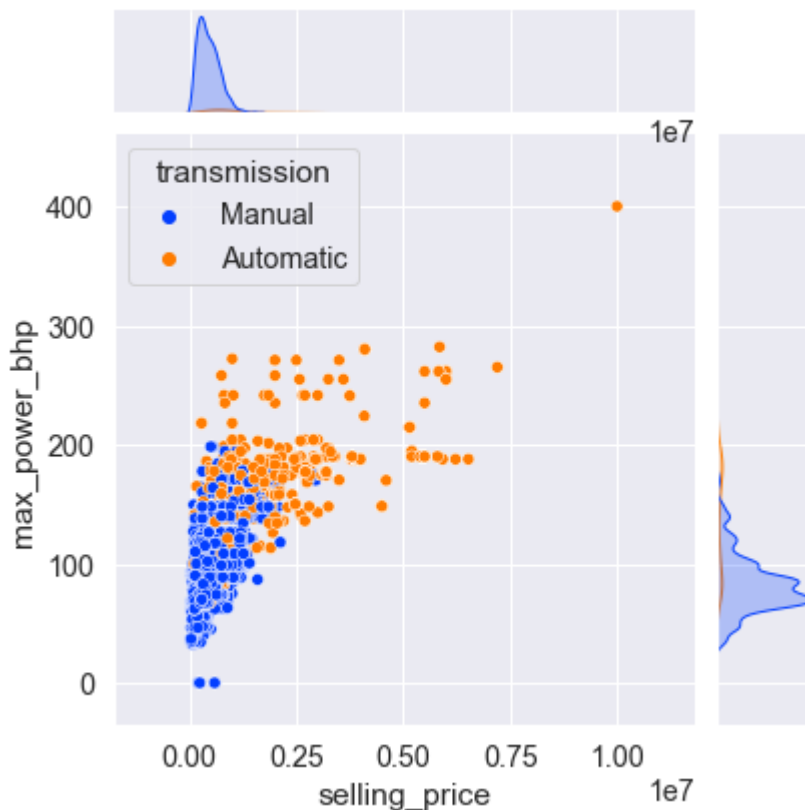






Пример:

```
sns.jointplot(
    x='selling_price',
    y='max_power_bhp',
    data=cars,
    palette='bright',
    hue='transmission');
```



Изучение отношений между категориальными и числовыми данными

На следующих диаграммах ось x будет содержать категориальную переменную, а ось Y — числовую.

## 6. Гистограмма

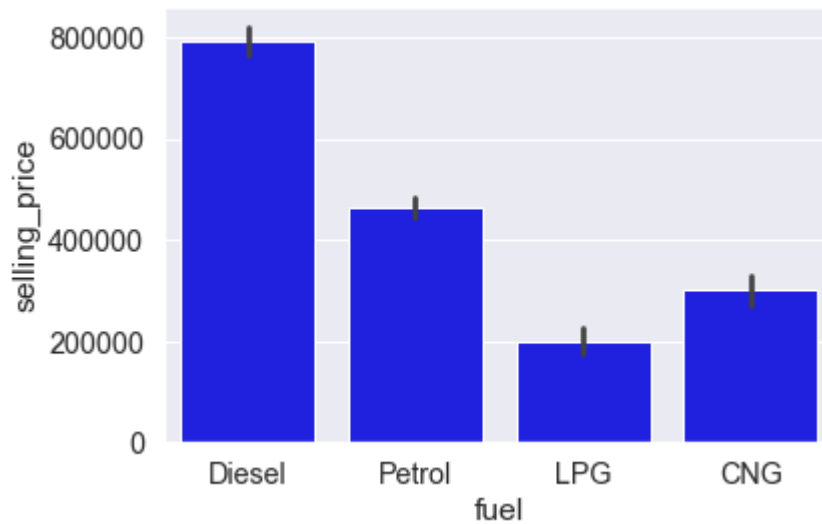
Гистограмма использует столбцы разной высоты для сравнения распределения числовой переменной между группами категориальной переменной. По умолчанию высота столбца оценивается с помощью «среднего»: параметр `estimator` изменяет эту функцию агрегирования с помощью встроенных функций Python, таких как `estimator=max` или `len`, или функций NumPy наподобие `np.max` и `np.median`.

Описание в документации.

```
# Функции:
sns.barplot() # оси.
sns.catplot(kind='bar') # диаграмма.
```

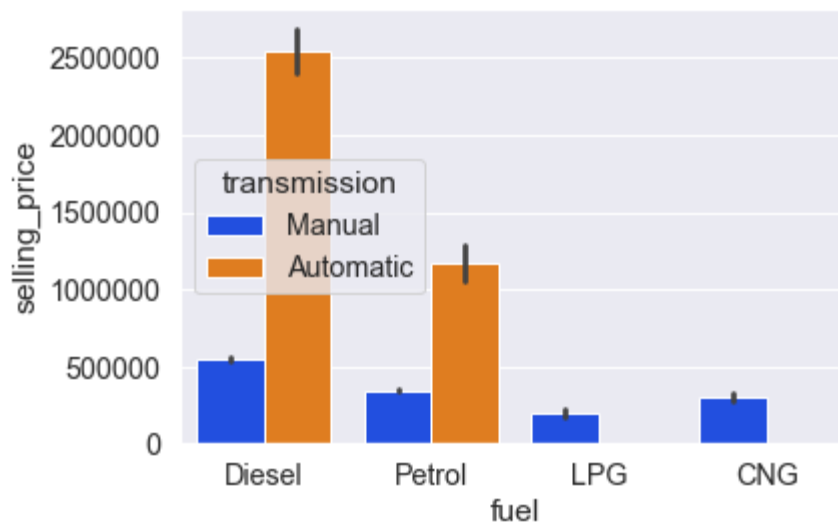
Пример:

```
sns.barplot(
    x='fuel',
    y='selling_price',
    data=cars,
    color='blue',
    # estimator=sum,
    # estimator=np.median);
```



Пример:

```
sns.barplot(
    x='fuel',
    y='selling_price',
    data=cars,
    palette='bright'
    hue='transmission');
```



Описание в документации.

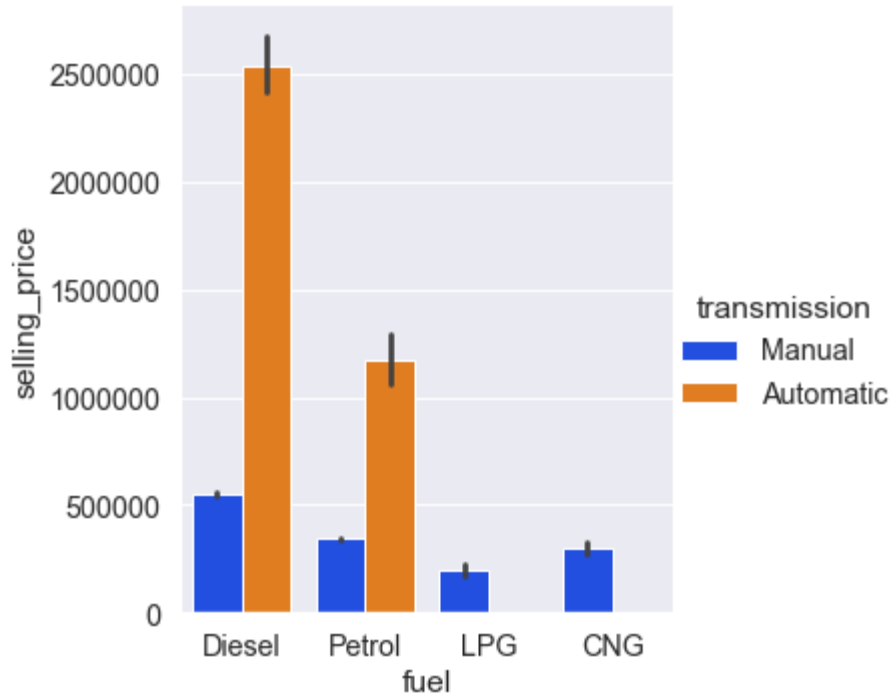
```
sns.catplot(x, y, data, kind='bar', hue='cat_col')
```

Тип категориальной диаграммы указывается параметром `kind`, по умолчанию он равен `'strip'`. Допустимы значения `'swarm'`, `'box'`, `'violin'`, `'boxen'`, `'point'` и `'bar'`. Воспользуемся `catplot` для создания диаграммы, похожей на предыдущую.

Код примера:

```
sns.catplot(
    x='fuel',
    y='selling_price',
```

```
data=cars,
palette='bright',
kind='bar',
hue='transmission');
```



Пример гистограммы через catplot:

```
g = sns.catplot(
    x='fuel',
    y='selling_price',
    data=cars,
    palette='bright',
    height=3, aspect=1.3,
    kind='bar',
    hue='transmission',
    col='seller_type',
    col_wrap=2) # указывает количество столбцов для измерений, отображаемых в
одной визуализации
g.set_titles(
    'Seller: {col_name}');
```



#### 7. Точечная диаграмма

Вместо столбцов точечная диаграмма отображает точки, представляющие среднее значение (или другую оценку) каждой группы категорий. Точки соединяются линией, что упрощает сравнение изменений центральной тенденции переменной у для групп.

Описание в документации.

```
# Функции:
sns.pointplot() # оси.
sns.catplot(kind='point') # диаграмма.
```

Код примера:

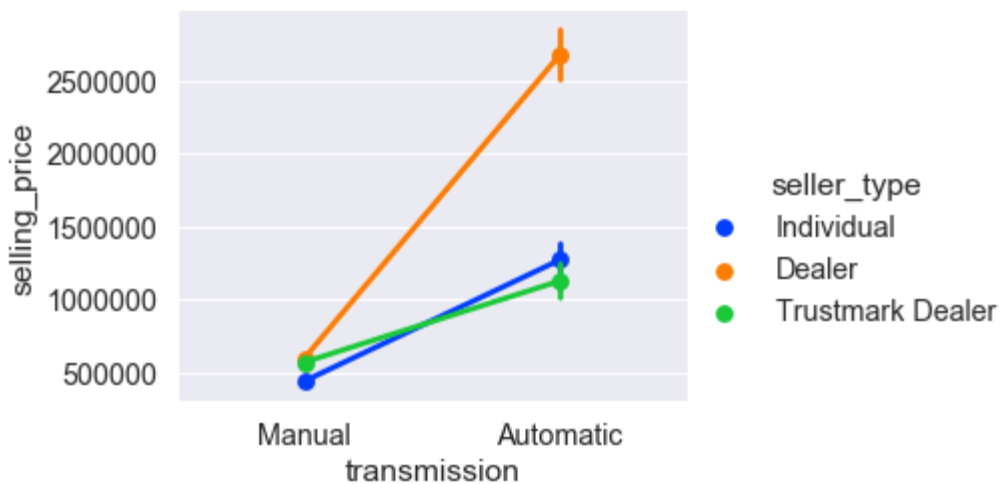
```
sns.pointplot(
    x='seller_type',
    y='mileage_kmpl',
    data=cars);
```



Когда вы добавляете третью категорию с помощью hue, точечная диаграмма становится информативнее гистограммы, потому что через каждый переданный hue класс проводится линия, которая упрощает сравнение изменений по группе переменной x.

Точечная диаграмма с catplot:

```
sns.catplot(
    x='transmission',
    y='selling_price',
    data=cars,
    palette='bright',
    kind='point', # точечная диаграмма
    hue='seller_type');
# Ту же диаграмму можно получить, используя sns.paitplot и параметр hue.
```

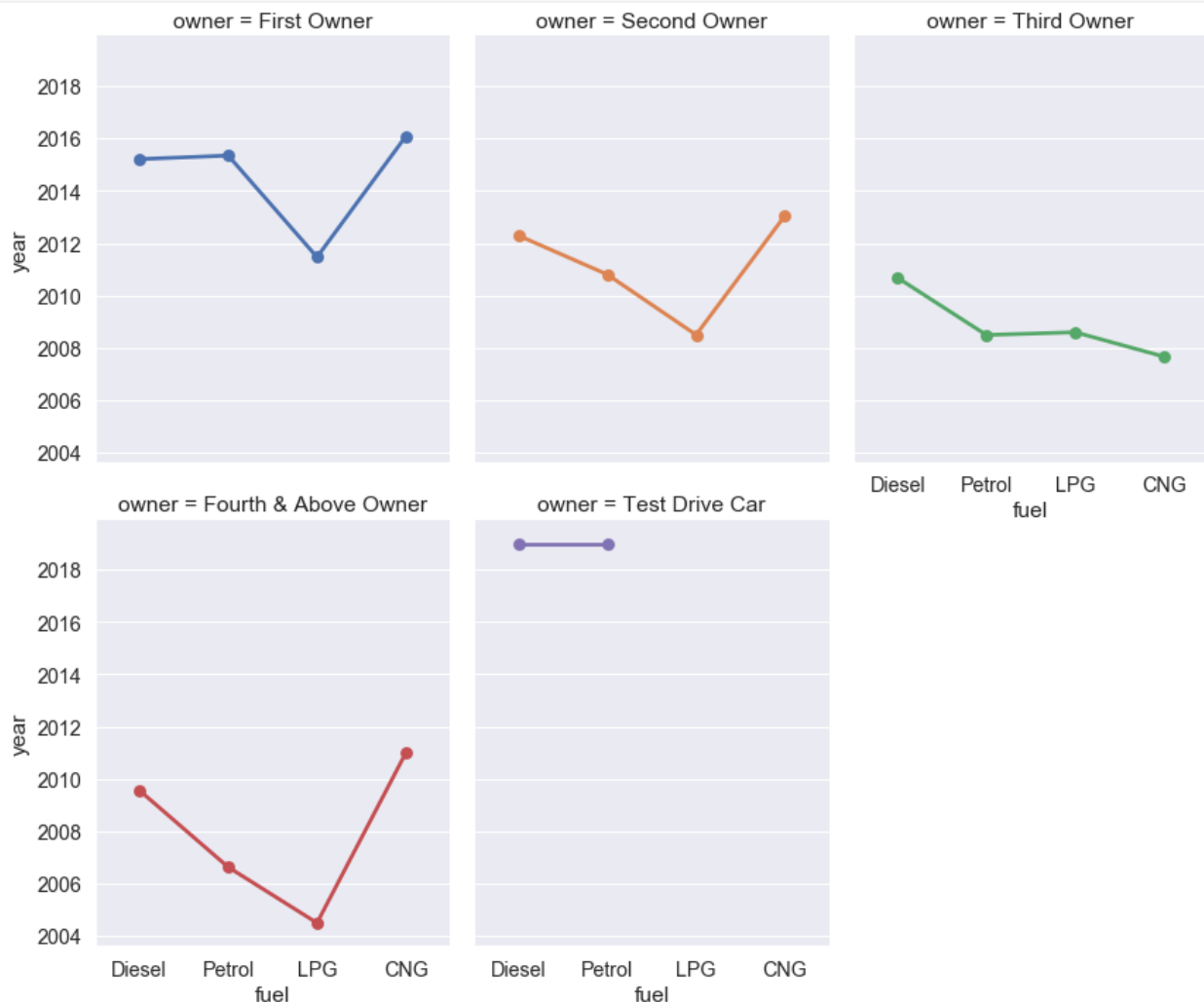


Пример:

```
sns.catplot(
    x='fuel',
    y='year',
    data=cars,
    ci=None,
    height=5, # по умолчанию
    aspect=.8,
    kind='point',
```

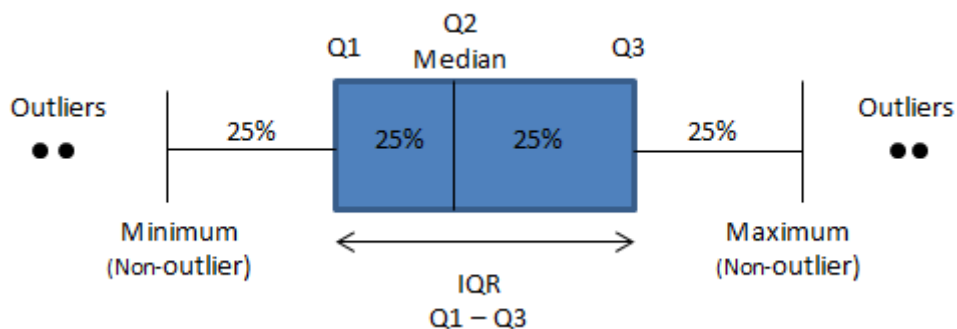


```
# В hue и col указана одна и та же категория
hue='owner',
col='owner',
col_wrap=3);
```



## 8. Ящик с усами

Ящик с усами визуализирует распределение между числовыми и категориальными переменными, отображая информацию о квантилях.

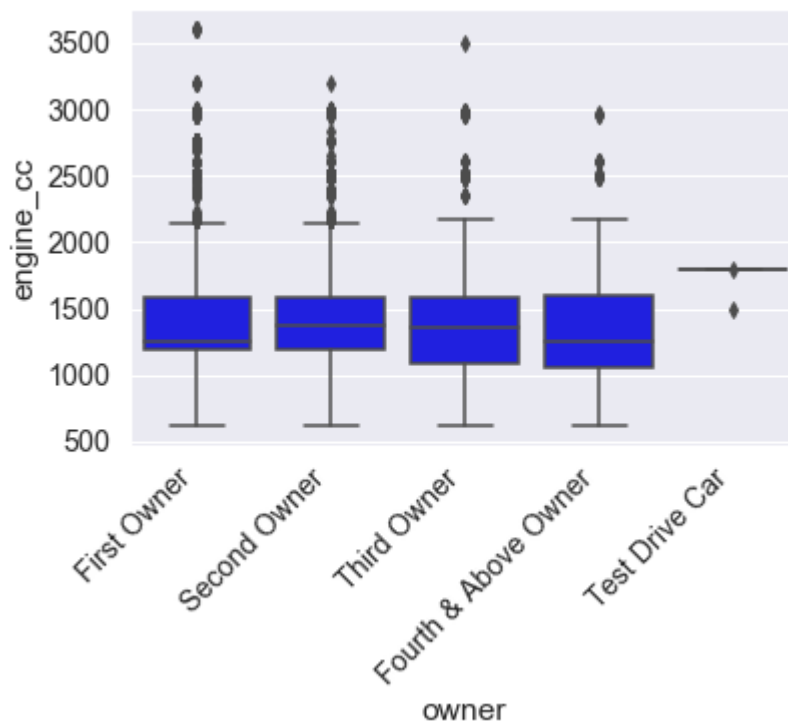


На диаграммах видно наименьшее значение, медиану, наибольшее значение и выбросы для каждого класса категорий.

```
# Функции:  
sns.boxplot() # оси.  
sns.catplot(kind='box') # диаграмма.
```

Код примера:

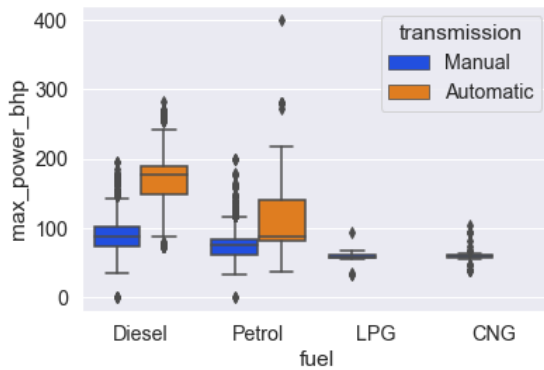
```
sns.boxplot(x='cat_col', y='num_col', data=df)  
sns.boxplot(  
    x='owner',  
    y='engine_cc',  
    data=cars,  
    color='blue')  
plt.xticks(rotation=45,  
            ha='right');
```



Описание в документации.

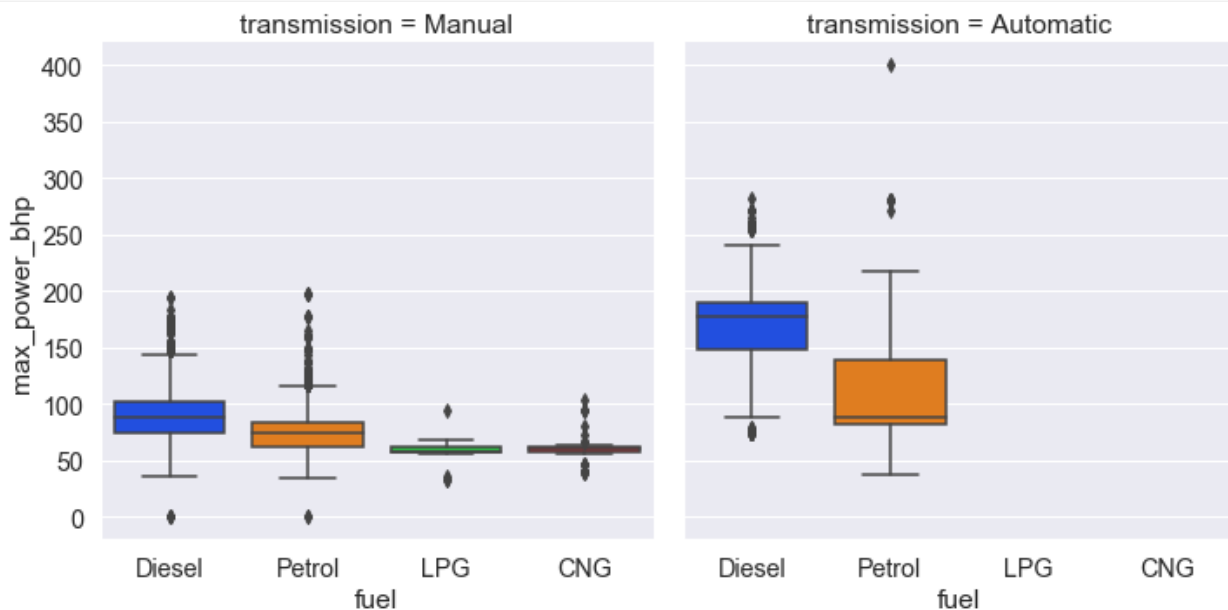
Ещё один пример:

```
sns.boxplot(  
    x='fuel',  
    y='max_power_bhp',  
    data=cars,  
    palette='bright',  
    hue='transmission');
```



То же через catplot:

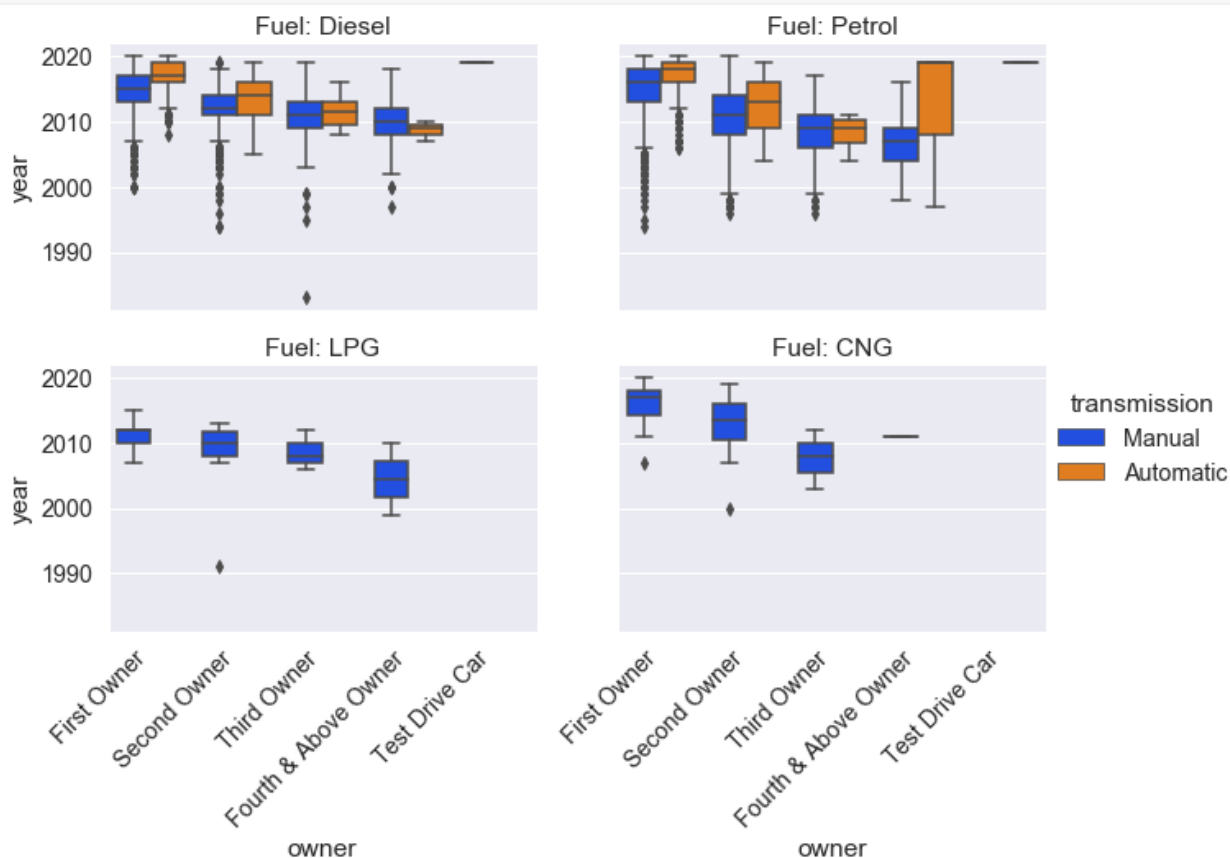
```
sns.catplot(
    x='fuel',
    y='max_power_bhp',
    data=cars,
    palette='bright',
    kind='box', # используйте catplot с kind='box'
    col='transmission'); # для создания поддиаграмм укажите параметр col
```



Пример с catplot сложнее:

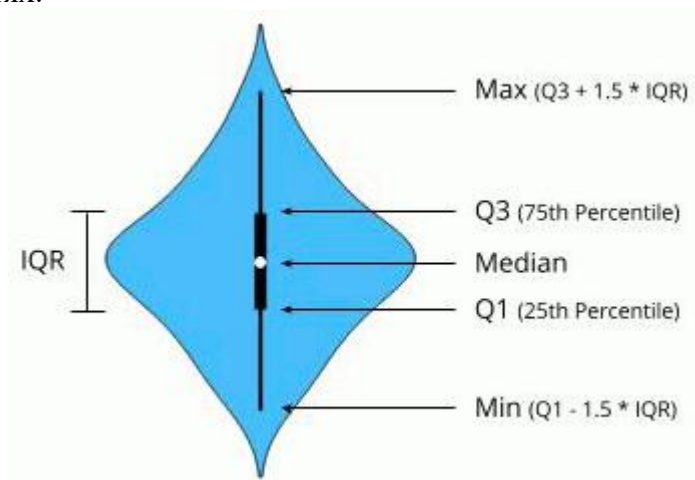
```
g = sns.catplot(
    x='owner',
    y='year',
    data=cars,
    palette='bright',
    height=3, aspect=1.5,
    kind='box',
    hue='transmission',
    col='fuel',
    col_wrap=2)
g.set_titles(
```

```
'Fuel: {col_name}');
g.set_xticklabels(
    rotation=45, ha='right')
```



## 9. Скрипичная диаграмма

В дополнение к квартилям ящика с усами, скрипичная диаграмма рисует кривую ядерной оценки плотности, отражающую вероятности наблюдений в различных областях.



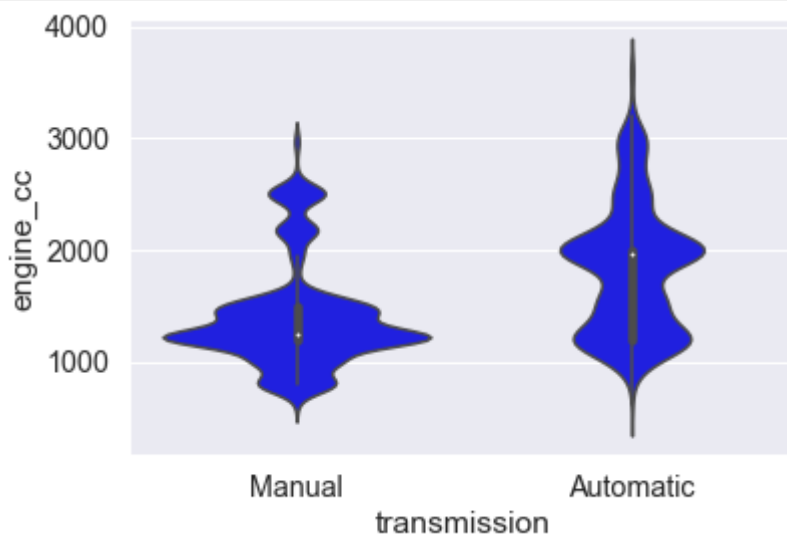
Описание в документации.

# Функции:  
sns.violinplot() # оси.

```
sns.catplot(kind='violin') # диаграмма.
```

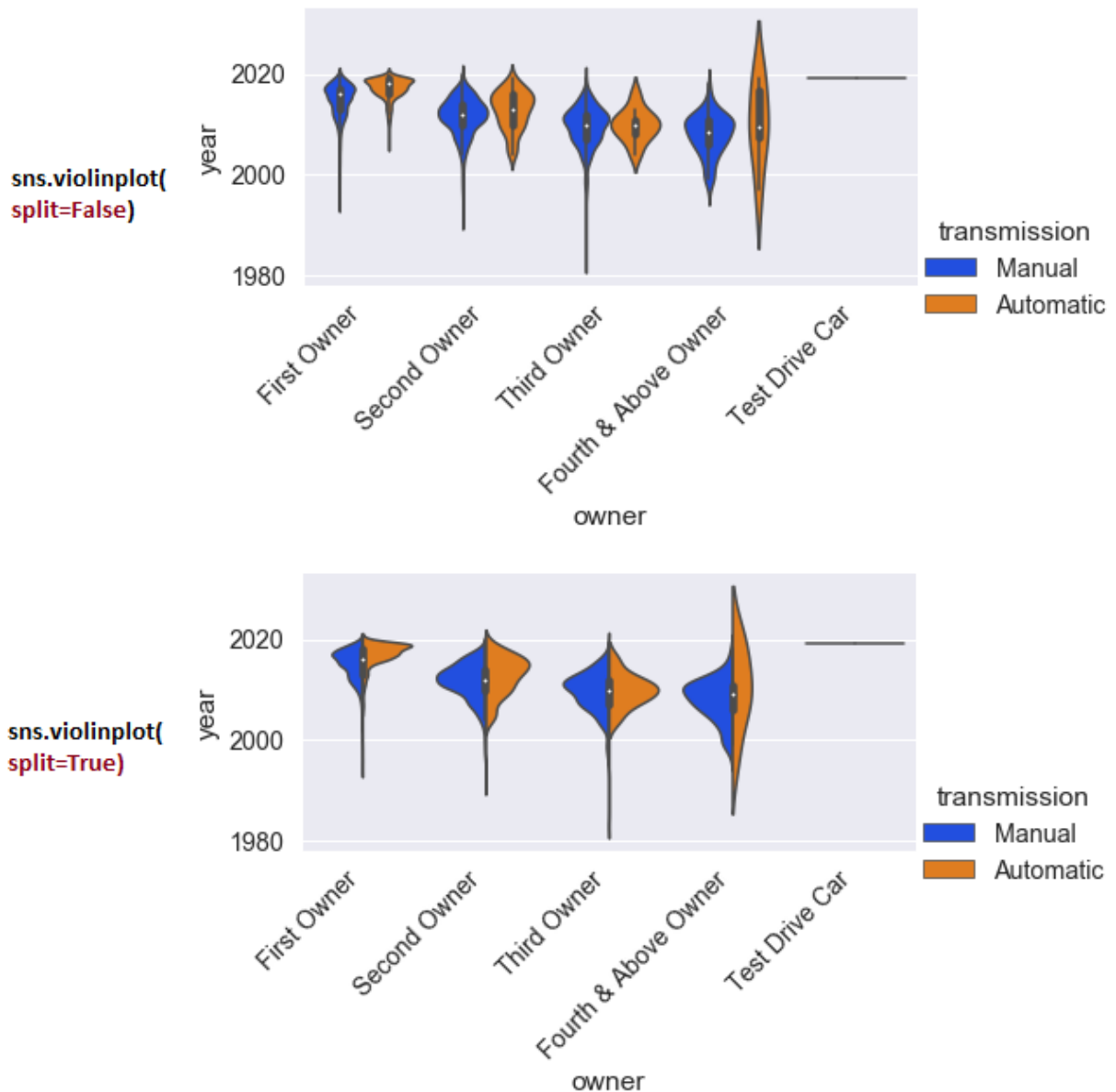
Код примера:

```
sns.violinplot(x='cat_col', y='num_col', data=df)
sns.violinplot(
    x='transmission',
    y='engine_cc',
    data=cars,
    color='blue');
```



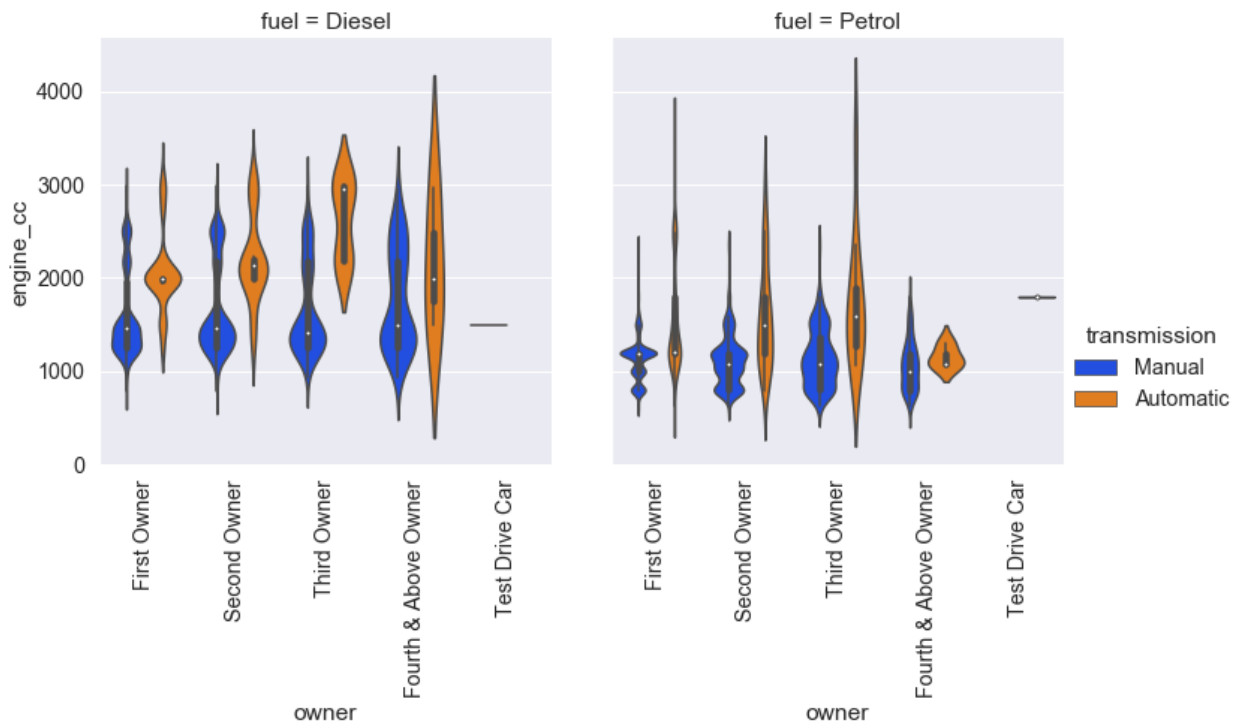
Пример с catplot:

```
g = sns.catplot(
    x='owner',
    y='year',
    data=cars,
    palette='bright',
    height=3,
    aspect=2
    split=False,
    # split=True рисует половину диаграммы для каждого категориального класса.
    # Это работает, когда переменная hue имеет только два класса.
    # Смотрите ниже
    kind='violin',
    hue='transmission')
g.set_xticklabels(
    rotation=45,
    ha='right')
# Тот же результат можно получить через sns.violinplot с параметром hue
```



Фильтрация по двум классам через `catplot`:

```
# Здесь рассматриваем данные только для 'diesel' и 'petrol':
my_df = cars[cars['fuel'].isin(['Diesel','Petrol'])]
g = sns.catplot(
    x="owner",
    y="engine_cc",
    data=my_df,
    palette='bright',
    kind = 'violin',
    hue="transmission",
    col = 'fuel')
g.set_xticklabels(
    rotation=90);
```



#### 10. Одномерная диаграмма рассеяния

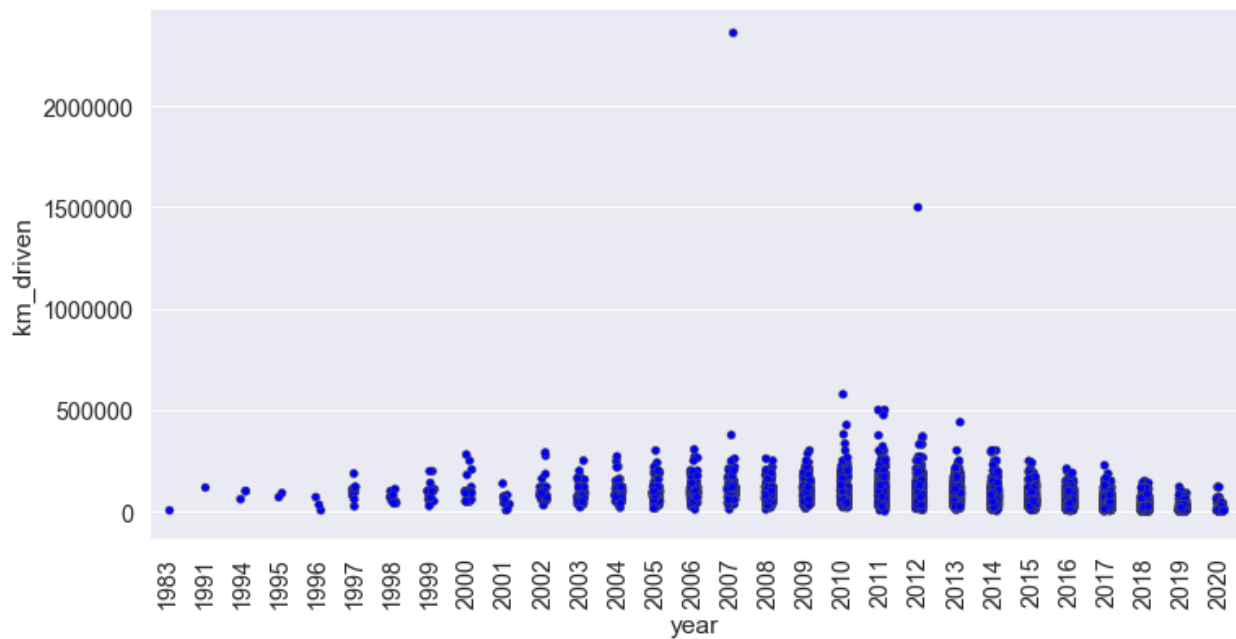
Точки этой диаграммы показывают, как числовая переменная распределяется между классами категориальной переменной. Её можно представить как точечную диаграмму, где одна из осей отражает признак в данных.

Описание в документации.

```
# Функции:
sns.stripplot() # оси.
sns.catplot(kind='strip') # диаграмма.
```

Код примера:

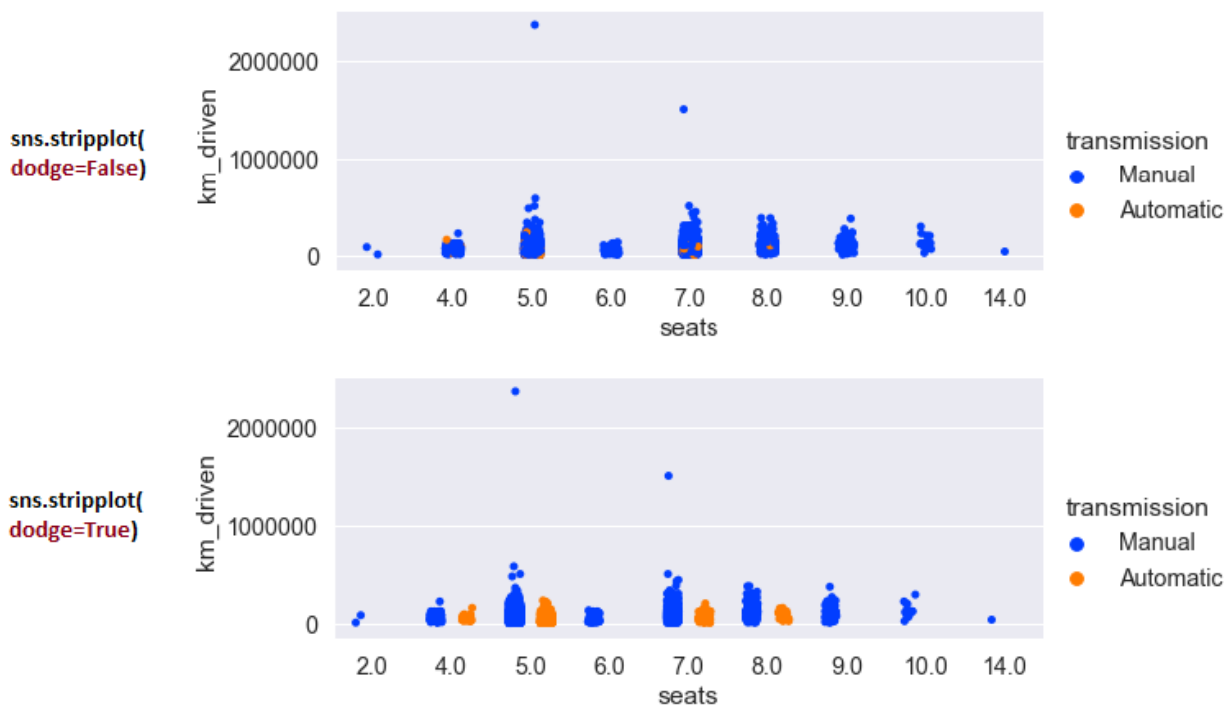
```
sns.stripplot(x='cat_col', y='num_col', data=df)
plt.figure(
    figsize=(12, 6))
sns.stripplot(
    x='year',
    y='km_driven',
    data=cars,
    linewidth=.5,
    color='blue')
plt.xticks(rotation=90);
```



Пример через catplot:

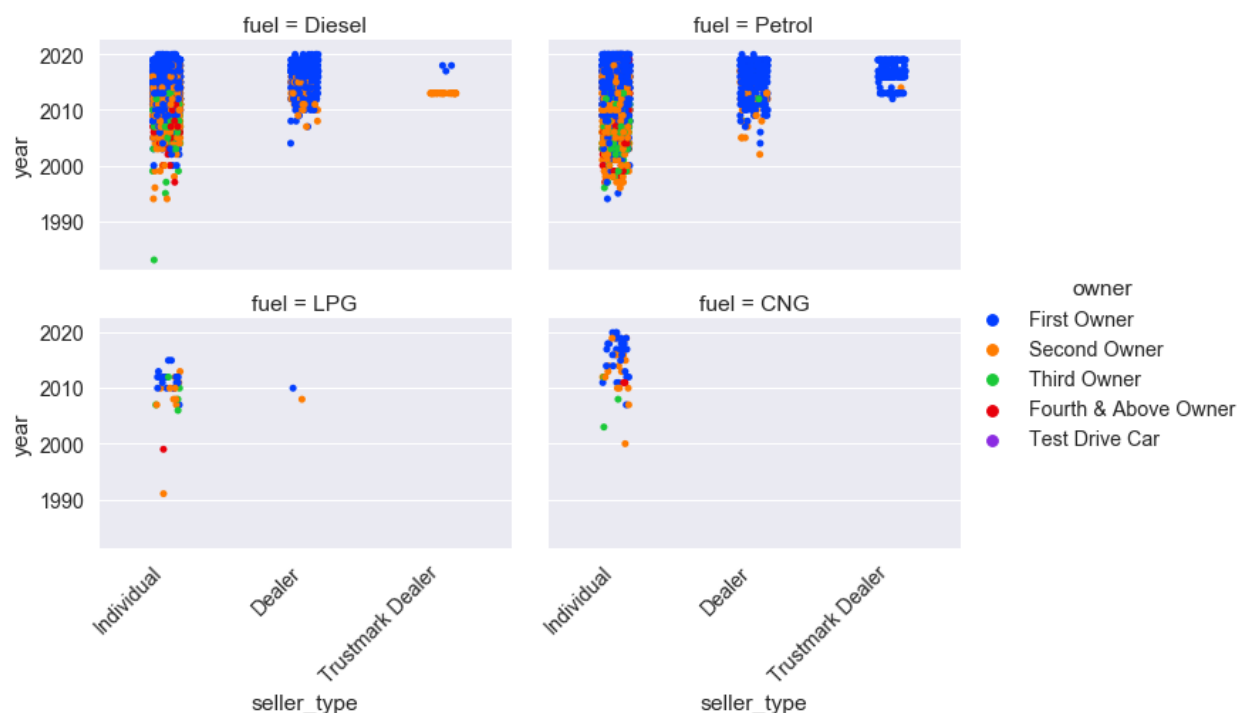
```
sns.catplot(
    # kind='strip' по умолчанию
    x='seats',
    y='km_driven',
    data=cars,
    palette='bright',
    height=3,
    aspect=2.5,
    # Аргумент dodge=True (по умолчанию dodge=False) делит вертикальную линию
    # точек по цвету
    kind='strip',
    hue='transmission');
```





Пример с `catplot`:

```
g = sns.catplot(
    x="seller_type",
    y="year",
    data=cars,
    palette='bright',
    height=3, aspect=1.6,
    kind='strip',
    hue='owner',
    col='fuel',
    col_wrap=2)
g.set_xticklabels(
    rotation=45,
    ha='right');
```

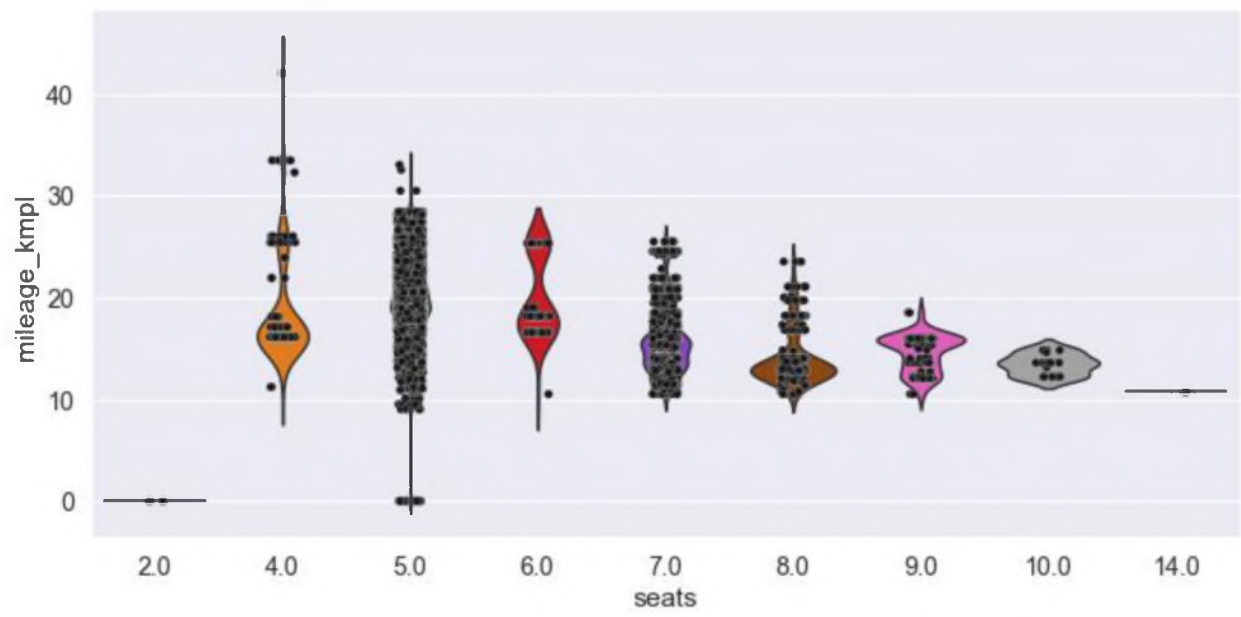


Комбинация одномерной диаграммы рассеяния и скрипичной диаграммы

Одномерную диаграмму рассеяния можно комбинировать со скрипичной диаграммой или ящиком с усами, чтобы показать положение пропусков или выбросов в данных.

Пример с catplot:

```
g = sns.catplot(
    x='seats',
    y='mileage_kmpl',
    data=cars,
    palette='bright',
    aspect=2,
    inner=None,
    kind='violin')
sns.stripplot(
    x='seats',
    y='mileage_kmpl',
    data=cars,
    color='k',
    linewidth=0.2,
    edgecolor='white',
    ax=g.ax);
```



## Практическое занятие 6. Парсинг и анализ данных из интернета

**Цель работы:** получение навыков работы по сбору информации с сайта в исходном виде, переработка полученной информации и генерация результатов проведенного анализа. Изучение функций, посредством которых осуществляется парсинг и анализ данных.

В результате выполнения лабораторных работ студенты должны знать:

- что такое парсинг и в каких целях он применяется;
- уметь анализировать полученную с сайта информацию;
- осуществлять генерацию результатов;
- знать схему действия основных функций.

Парсинг — это синтаксический анализ информации. Под парсингом HTML, как правило, подразумевают выборочное извлечение большого количества информации с других сайтов и ее последующее использование, т.е. нахождение на странице необходимого участка кода, в котором заключена нужная информация. Если части кода повторяются в пределах одной страницы и на других страницах, имеющих аналогичную структуру, можно выделить и импортировать повторяющиеся данные автоматически, существенно сэкономив время и предупредив возможные ошибки копирования этой информации вручную.

Для работы с данными мы будем использовать библиотеки matplotlib – это библиотека графических построений для языка программирования Python, и его расширения вычислительной математики NumPy. Библиотека Matplotlib обеспечивает объектно-ориентированный интерфейс для встраивания графиков в приложения, используя инструменты GUI общего назначения, такие как WxPython, Qt, или GTK+. Существует также процедурный rpylab-интерфейс, напоминающий MATLAB. Библиотека научных расчетов SciPy также использует matplotlib для работы с графиками.

Для чтения текстовых файлов используются две функции `read_csv()` и `read_table()`. Они обе используют похожий код разбора данных для преобразования табличных данных в объект `DataFrame`.

Под функцией "groupby" будем понимать процесс, включающий один или более шагов:

- Разделение данных на группы по каким-либо критериям;
- Применение функции к каждой группе независимо друг от друга;
- Объединение результатов в единую структуру данных.

В большинстве ситуаций вы можете разделить набор данных в группы и сделать что-то с этими группами самостоятельно. На данном этапе можно выполнить одно из следующих действий:

- Агрегация: вычисления сводной статистики (или статистики) о каждой группе.

Некоторые примеры:

- Вычисление суммы или среднего
- Размеры группы

- Трансформация: выполнение некоторых конкретных вычислений над группой, с сохранением ее размера.

Некоторые примеры:

- Стандартизация данных (zscore) в пределах группы
- Заполнение неизвестных значений (NA) в группах со значением полученного из каждой группы

- Фильтрация: исключение групп, не удовлетворяющих поставленному условию. Некоторые примеры
- Отбросив данные, относящиеся к группам с несколькими членами
- Фильтрация данных на основе групповой суммы или среднее значение

Некоторые сочетания GroupBy будут рассмотрены как результаты применения шагов и попытаются вернуть объединение результата, если он не вписывается ни в одну из вышеперечисленных категорий. Функция GroupBy должна быть хорошо знакома тем, кто использовал SQL, в котором вы можете написать следующий код:

```
SELECT Column1, Column2, mean(Column3), sum(Column4)FROMSomeTable
GROUPBY Column1, Column2
```

Рассмотрим задачу анализа показателей температуры и осадков на велодорожках, для выделения наиболее благоприятных для поездок дней. Необходимую информацию о погоде мы возьмем с канадского сайта погоды и получим данные за март 2012 года. Для получения данных о погоде в Монреале сохраним ссылку на сайт в переменную url\_template:

```
url_template =
"http://climate.weather.gc.ca/climateData/bulkdata_e.html?format=csv&stationID=5415&Year
={year}&Month={month}&timeframe=1&submit=Download+Data"
```

Чтобы получить данные на март 2013 года, мы должны подставить соответствующие значения в строку шаблона:

```
url = url_template.format(month=3, year=2012)
```

Теперь можно загрузить данные по указанной ссылке:

```
weather_mar2012 = pd.read_csv(url, skiprows=16, index_col='Date/Time', parse_dates=True,
encoding='latin1')
```

Таким образом, мы передаем ссылку на данные в функцию read\_csv. Параметр skiprows указывает pandas на необходимость пропустить 16 первых строк данных, в которых содержатся малозначительные для нас метаданные. Мы также указали pandas на необходимость разбора строк с датами и установили поле 'Date / Time' индексом нашего набора. В результате получится следующий набор данных:

```

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 744 entries, 2012-03-01 00:00:00 to
2012-03-31 23:00:00
Data columns (total 24 columns):
Year                744 non-null values
Month              744 non-null values
Day               744 non-null values
Time              744 non-null values
Data Quality       744 non-null values
Temp (°C)          744 non-null values
Temp Flag          0 non-null values
Dew Point Temp (°C) 744 non-null values
Dew Point Temp Flag 0 non-null values
Rel Hum (%)        744 non-null values
Rel Hum Flag       0 non-null values
Wind Dir (10s deg) 715 non-null values
Wind Dir Flag      0 non-null values
Wind Spd (km/h)    744 non-null values
Wind Spd Flag      3 non-null values
Visibility (km)     744 non-null values
Visibility Flag     0 non-null values
Stn Press (kPa)     744 non-null values

Stn Press Flag      0 non-null values
Hmdx12 non-null values
Hmdx Flag           0 non-null values
Wind Chill          242 non-null values
Wind Chill Flag     1 non-null values
Weather             744 non-null values
dtypes: float64(14), int64(5), object(5)

```

Изобразим данные в виде графика, используя библиотеку matplotlib (рис. 1).

```
weather_mar2012[u"Temp (°C)"].plot(figsize=(15,5))
```

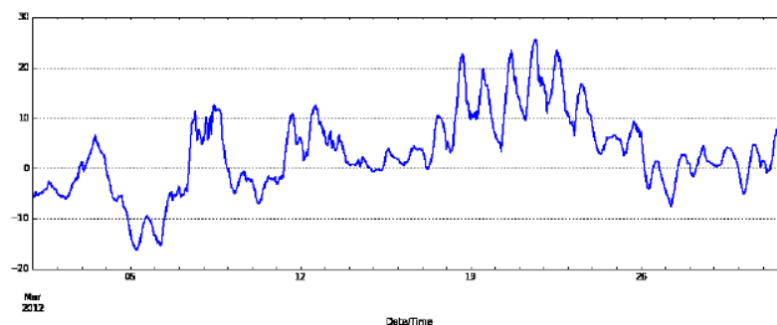


Рисунок 1 - График изменения температуры

Здесь мы использовали неверно прочитанный символ  $\hat{A}$  для того, чтобы извлечь данные из поля с названием - Temp (°C). Чтобы удалить этот символ, вместе с символом ° из нашего набора, можно воспользоваться функцией `replace` над названием поля.

```
weather_mar2012.columns = [s.replace("\hat{A}°", "") for s in
weather_mar2012.columns]
```

Как вы могли заметить, есть несколько полей, которые либо полностью пустые, либо частично заполнены. От них мы можем избавиться с помощью функции `dropna`, которая удаляет строки или столбцы. Аргумент `axis` указывает, что нужно производить удаление по полям или строкам (1 и 0 соответственно), аргумент `how` указывает условие удаления при наличии пустых значений (`any` - при любом пустом значении, `all` - при всех пустых). Следующий код удалит те поля, в которых есть хотя бы одно пустое значение.

```
weather_mar2012 = weather_mar2012.dropna(axis=1,how='any')weather_mar2012[:5]
```

	Year	Month	Day	Time	Data Quality	Temp (C)	Dew Point Temp (C)	Rel Hum (%)	Wind Spd (km/h)	Visibility (km)	Stn Press (kPa)	Weather
Date/Time												
2012-03-01 00:00:00	2012	3	1	00:00		-5.6	-9.7	72	24	4.0	100.97	Snow
2012-03-01 01:00:00	2012	3	1	01:00		-5.7	-8.7	79	26	2.4	100.87	Snow
2012-03-01 02:00:00	2012	3	1	02:00		-5.4	-8.3	80	28	4.8	100.80	Snow
2012-03-01 03:00:00	2012	3	1	03:00		-4.7	-7.7	79	28	4.0	100.69	Snow
2012-03-01 04:00:00	2012	3	1	04:00		-5.4	-7.8	83	35	1.6	100.62	Snow

5 rows x 12 columns

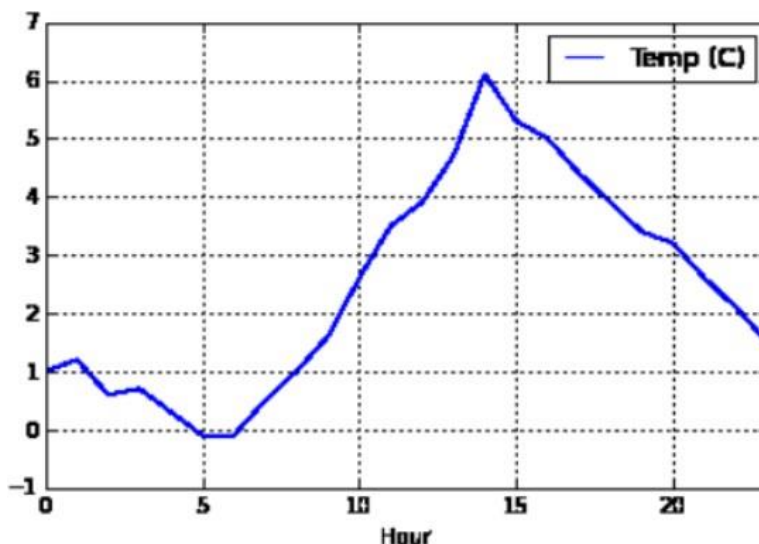
Данную таблицу можно оптимизировать, объединив избыточные поля `Year/Month/Day/Time` и столбец `DataQuality`, так как он не содержит интересующей нас информации. Давайте избавляться от них.

```
weather_mar2012 = weather_mar2012.drop(['Year','Month','Day','Time','Data Quality'], axis=1)
weather_mar2012[:5]
```

	Temp (C)	DewPoint Temp (C)	RelHum (%)	WindSpd (km/h)	Visibility (km)	StnPress (kPa)	Weather
Date/Time							
2012-03-01 00:00:00	-5.5	-9.7	72	24	4.0	100.97	Snow
2012-03-01 01:00:00	-5.7	-8.7	79	26	2.4	100.87	Snow
2012-03-01 02:00:00	-5.4	-8.3	80	28	4.8	100.80	Snow
2012-03-01 03:00:00	-4.7	-7.7	79	28	4.0	100.69	Snow
2012-03-01 04:00:00	-5.4	-7.8	83	35	1.6	100.62	Snow

Перейдем к построению температуры по времени суток. Для этого воспользуемся функциями GroupBy и aggregate.

```
temperatures= weather_mar2012[['Temp (C)']] temperatures['Hour'] =
weather_mar2012.index.hour
temperatures.groupby('Hour').aggregate(np.median).plot()
```



Перейдем к следующему шагу - получение данных за целый год. Для начала добавим все сделанное нами ранее в отдельную функцию, которая будет возвращать погоду для данного месяца. Также, учтем ошибку, когда при выполнении запроса данных за январь, выдаются данные за предыдущий год.

```
def download_weather_month(year, month):
    if month == 1:
        year += 1
    url = url_template.format(year=year, month=month)
    weather_data = pd.read_csv(url, skiprows=16,
                                index_col='Date/Time', parse_dates=True, encoding="latin1")
    weather_data = weather_data.dropna(axis=1)
    weather_data.columns = [col.replace('Â°', '') for col in weather_data.columns]
    weather_data = weather_data.drop(['Year', 'Day', 'Month', 'Time', 'Data Quality'], axis=1)
    return weather_data
```

Теперь мы сможем получить сразу данные по всем месяцам следующим образом:

```
data_by_month= [download_weather_month(2012, i) for i in range(1, 13)]
```



Теперь мы можем легко объединить все наборы данных вместе в один набор с использованием `pd.concat`. Получим данные за целый год.

```
weather_2012 = pd.concat(data_by_month)
weather_2012.info()
```

Сохраним в формате CSV.  
`weather_2012.to_csv('weather_2012.csv')`

**Задание :**

1. Загрузить файл csv по ссылке <http://www.ed.ac.uk/schools-departments/geosciences/weather-station/download-weather-data>, где представлены данные о погоде.
2. Сделать задание согласно своему варианту

№ варианта	Название файла
1	JCMB_2014_Dec.csv
2	JCMB_2014_Nov.csv
3	JCMB_2014_Oct.csv
4	JCMB_2014_Sep.csv
5	JCMB_2014_Aug.csv
6	JCMB_2014_Jul.csv
7	JCMB_2014_Jun.csv
8	JCMB_2014_May.csv
9	JCMB_2014_Apr.csv
10	JCMB_2014_Mar.csv

3. Провести анализ погодных показателей (температура, осадки, атмосферное давление и т. д.), полученных с сайта.
4. Представить информацию из набора в виде таблиц и графиков. Исключить из таблицы пустые поля.
5. Сделать выводы о проделанной работе.

Отчет должен содержать:

- название и цель работы;
- краткие теоретические сведения;
- лабораторное задание и этапы его выполнения;
- результаты работы, выполненной в автоматизированном режиме и в ручном;
- выводы о проделанной работе.

### Контрольные вопросы

1. Что такое парсинг? В каких случаях его удобно использовать и сферы применения.
2. Парсинг с помощью Python.
3. Опишите применение библиотеки `matplotlib`? Каким образом ее использование необходимо при анализе данных?
4. Опишите применение и работу функций `read_csv()` и `read_table()`.

5. Зачем при анализе данных из интернета используются функции чтения текстовых файлов
6. Опишите функцию GroupBy.
7. Каким образом осуществляется разделение объекта на группы?
8. Опишите методы над сгруппированными данными

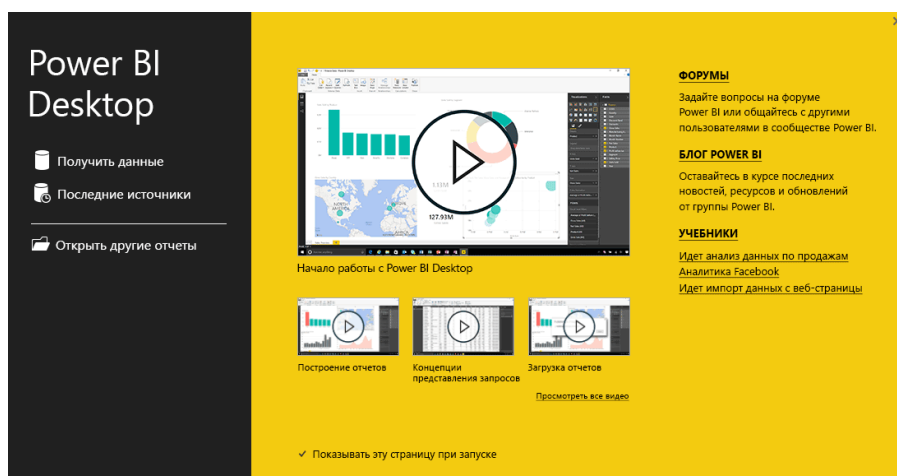
## Практическое занятие 7. Работа с данными в Power BI Desktop

**Цель работы:** освоить приемы подключения и преобразования данных, полученных из разных источников.

**Задачи работы:** изучить интерфейс Power BI Desktop, подключить, преобразовать и объединить данные на основе примера, выполнить задание для самостоятельной работы.

### Интерфейс Power BI Desktop

При запуске Power BI Desktop отображается экран приветствия:



На экране приветствия можно **Получить данные**, просмотреть **Последние источники** или **Открыть другие отчеты** непосредственно с экрана приветствия (по ссылкам в области слева). Если закрыть этот экран (нажать **x** в правом верхнем углу), открывается окно Power BI Desktop в представлении **Отчет**.

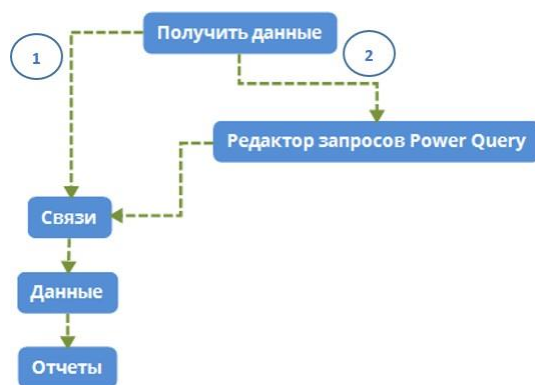
Power BI Desktop предлагает зарегистрироваться. Для этого нужно иметь какой-либо корпоративный e-mail. Почтовый ящик на общедоступных почтовых серверах (mail.ru, rambler.ru и пр.) не подходит. Но можно работать без регистрации. Регистрация нужна позже, если потребуется продолжить работу в **Power BI Service**.

**Задание 1.** Закройте экран приветствия.

### Схема работы в Power BI Desktop

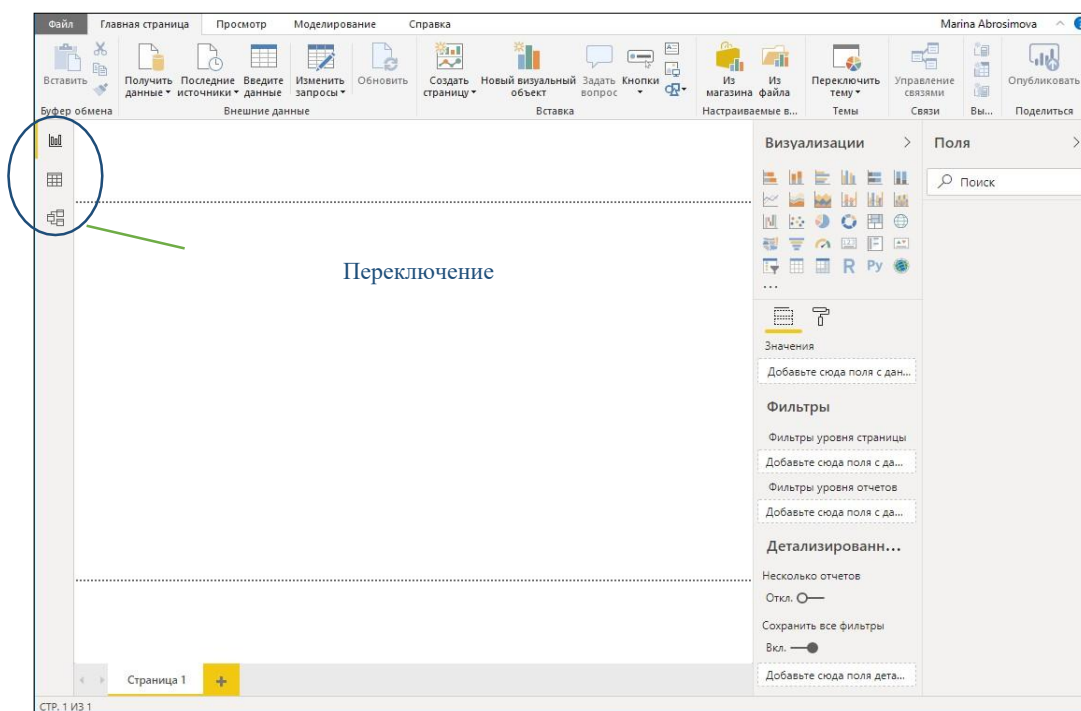
Работа в Power BI Desktop начинается с команды **Получить данные**, наданных потом и строятся все остальные процессы.

1. Если полученные данные изначально были подготовлены, то на их основе, минуя их обработку, сразу же можно приступить к созданию связей, дополнительных данных или к построению отчетов. За что, соответственно, и отвечают представления (каждое со своим рабочим окном) в Power BI Desktop (Отчет, Данные, Связи).
2. Если же данные не подготовлены и их нужно для начала обработать или источников данных несколько, то после получения данных, прежде чем устанавливать связи или строить отчеты, нужно вызвать редактор запросов Power Query и в нем подготовить все данные: обработать их, очистить, возможно, объединить или, наоборот, разъединить. И только потом приступить к работе в других окнах программы (Отчет, Данные, Связи).



Работа в Power BI Desktop заканчивается после формирования отчетов в виде серии графиков и диаграмм.

### Окно Power BI Desktop



Отображаемое в настоящее время представление обозначено желтой полосой слева. Для смены представления нужно выбрать любой из трех значков.

**Задание 2.** Выполните переход между представлениями **Отчет**, **Данные** и **Связи**. Останьтесь на представлении **Данные**.

### **Подключение к данным**

После установки Power BI Desktop Вы готовы подключаться к непрерывно расширяющейся среде данных. Подключение к данным возможно из любого представления Power BI Desktop.

В настоящей лабораторной работе Вы будете работать в представлении **Данные**

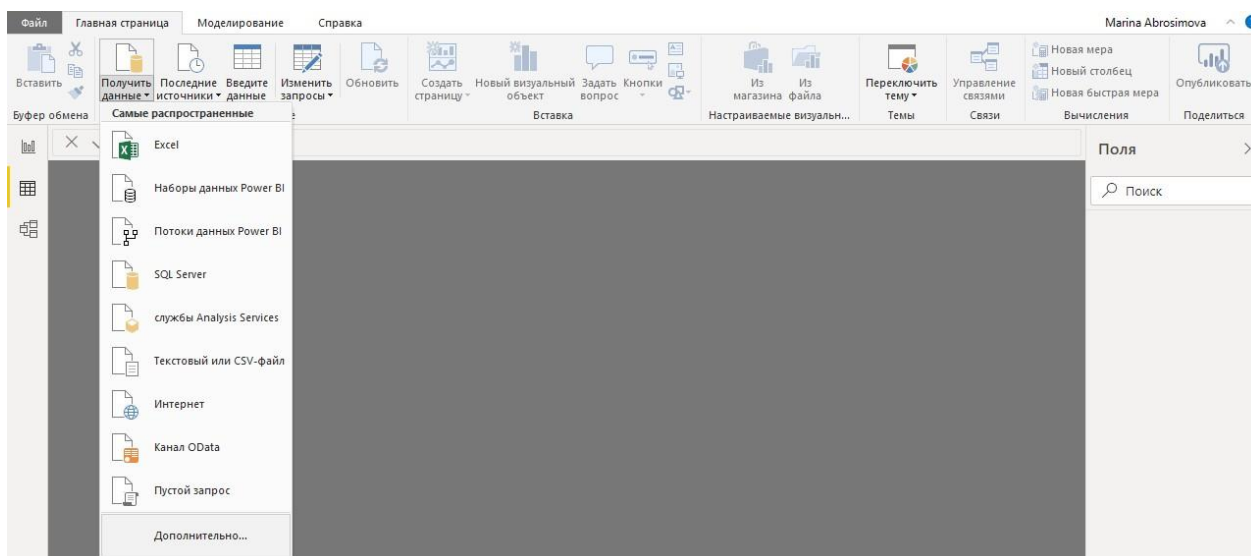
Представление **Данные** существует для просмотра подключенных данных и создания дополнительных данных (меры, столбцы, таблицы) из уже имеющихся. Именно в этом представлении можно дополнять свои модели данных недостающими расчетами (расчеты производятся в строке формул на основе языка DAX - специально разработанном Microsoft языке формул и функций, используемый в Excel и Power BI Desktop). Представлением **Данные** в большинстве случаев нужно пользоваться только после того как были установлены связи во всех таблицах в представлении **Связи**. В данном представлении в окне справа появляется дополнительная панель **Поля**. В этой панели отображаются таблицы и связанные с этими таблицами столбцы, а также созданные Вами меры. При выборе мышкой в панели **Поля** конкретной таблицы или столбца, по центру PowerBI Desktop в главной рабочей области отобразится соответствующая таблица со всеми внутренними данными. Если таблица, столбец или меры были созданы Вами при помощи формул DAX, то при выборе этого объекта в панели **Поля**, кроме того, что на рабочей области отобразится таблица, в строке формул выше таблицы, также, отобразится формула, по которой был создан этот объект.

**Задание 3.** Для дальнейшей работы:

1. создайте свою папку на диске D: с именем «Работа с наборами данных\_Ваша фамилия»
2. сохраните файл в своей папке под именем «Начало работы с Power BI Desktop».

Для подключения к данным нужно выполнить:

1. На вкладке **Главная страница** выбрать **Получить данные**;
2. Выбрать нужный источник.



В лабораторной работе мы будем подключаться к нескольким разным источникам данных в Интернет, рассматривая пример, приведенный в документации Microsoft: Вы аналитик данных и должны помочь клиенту выбрать новое место жительства: он хочет жить там, где много солнечного света, мягкие налоги и хорошее здравоохранение. Интересующие вас данные имеются на веб-ресурсе:

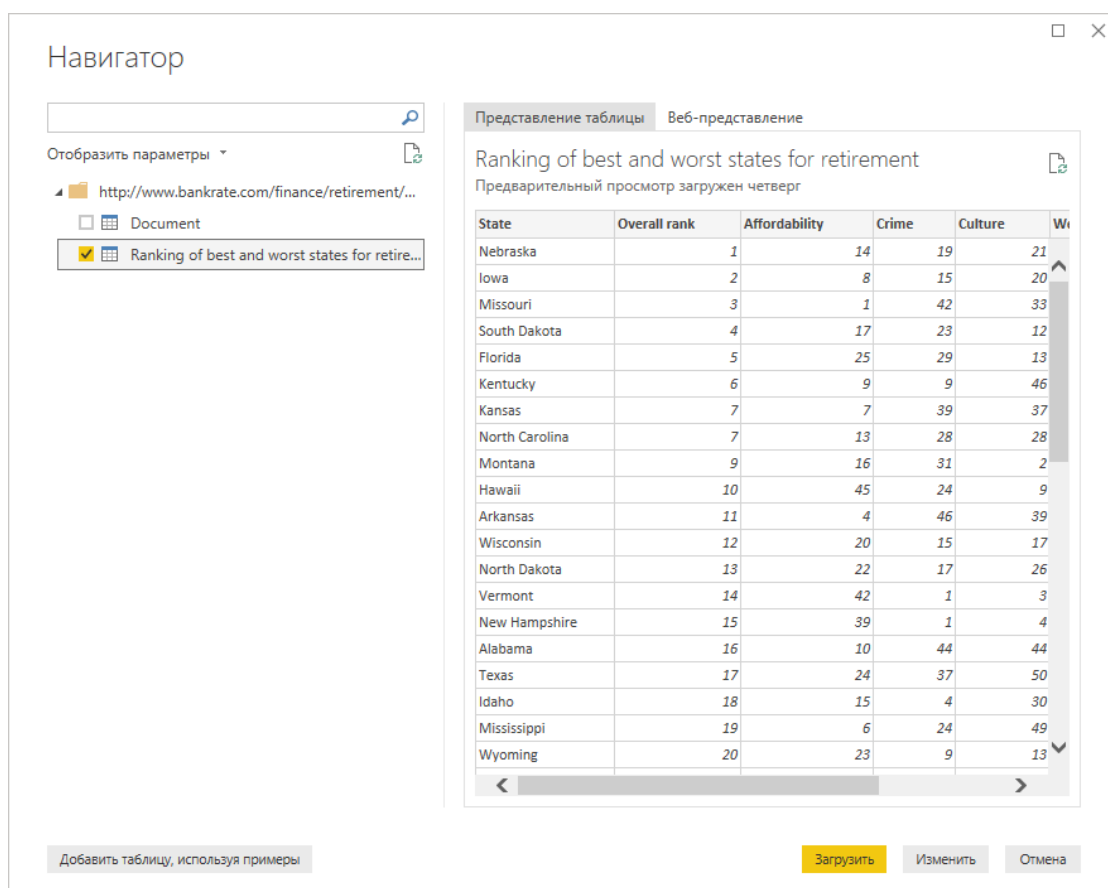
<http://www.bankrate.com/finance/retirement/best-places-retire-how-state-ranks.aspx>

Вам нужно:

1. На вкладке **Главная страница** выбрать **Получить данные**;
2. Выбрать **Интернет**.
3. Вставить адрес веб-ресурса:



Когда вы нажмете кнопку **ОК**, начнет работу функциональность «Запрос Power BI Desktop». Запрос обращается к веб-ресурсу, и в окне **Навигатор** отображается найденное на веб-странице. В данном случае найдены общий интернет-документ и таблица «Ranking of best and worst states for retirement». Нас интересует эта таблица, поэтому мы выбираем ее из списка. В окне **Навигатор** доступен предварительный просмотр содержимого.



На этом этапе можно изменить запрос перед загрузкой таблицы, нажав кнопку **Изменить** внизу окна, или загрузить эту таблицу. Сейчас мы этого неделаем.

**Задание 4.** Загрузите таблицу «Ranking of best and worst states for re- tirement» без изменений.

## Просмотр данных

Для просмотра данных Вы должны находиться в представлении **Данные**. В полученной таблице 50 строк

В Вашей таблице данных приведены рейтинги штатов по нескольким показателям. В области **Поля** можно просмотреть список всех полей. Числовые данные помечены значком  $\Sigma$ .

В следующем разделе мы настроим данные так, чтобы они соответствовали нашим потребностям. Процесс настройки подключенных данных называется *формированием* данных.

Power BI Desktop interface showing a table of state rankings for retirement. The table has columns for State, Overall rank, Affordability, Crime, Culture, Weather, and Wellness. The 'Главная страница' (Home) ribbon is active, and the 'Поля' (Fields) pane on the right shows a list of fields including 'Ranking of best and...'.

State	Overall rank	Affordability	Crime	Culture	Weather	Wellness
Nebraska	1	14	19	21	30	8
Iowa	2	8	15	20	34	12
Missouri	3	1	42	33	19	27
South Dakota	4	17	23	12	39	10
Florida	5	25	29	13	2	31
Kentucky	6	9	9	46	15	24
Kansas	7	7	39	37	20	21
North Carolina	7	13	28	28	12	33
Montana	9	16	31	2	45	20
Hawaii	10	45	24	9	1	9
Arkansas	11	4	46	39	9	34
Wisconsin	12	20	15	17	43	7
North Dakota	13	22	17	26	49	2
Vermont	14	42	1	3	44	1
New Hampshire	15	39	1	4	41	3

ТАБЛИЦА: Ranking of best and worst states for retirement (строк: 50)

**Задание 5.** Просмотрите данные.

## Формирование данных

Теперь, когда мы подключились к источнику данных, необходимо настроить данные для наших потребностей.

Иногда настройка означает преобразование данных, например, переименование столбцов или таблиц, изменение текста на числа, удаление строк, установку первой строки в качестве заголовков и т. д.

Для преобразования данных можно использовать вкладку **Главная страница** (контекстное меню), вкладку **Моделирование** или **Редактор запросов** в Power BI Desktop.

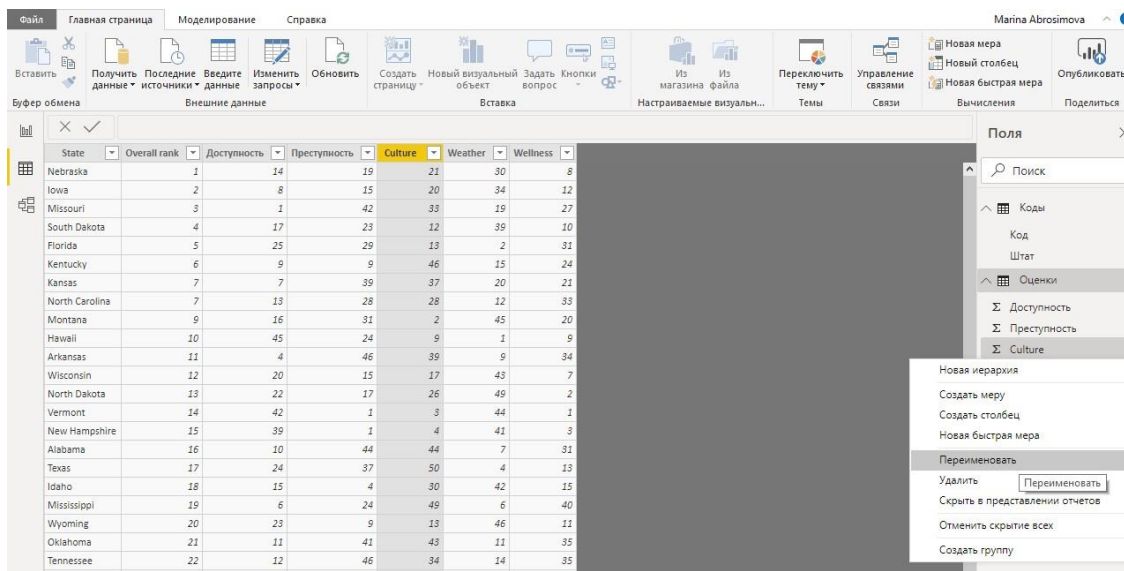
## Использование вкладки Главная страница

Сначала воспользуемся возможностями вкладки **Главная страница** и контекстного меню. Здесь можно удалять строки и столбцы, переименовывать поля, изменять тип данных и пр.

### *Переименование столбцов в таблице*

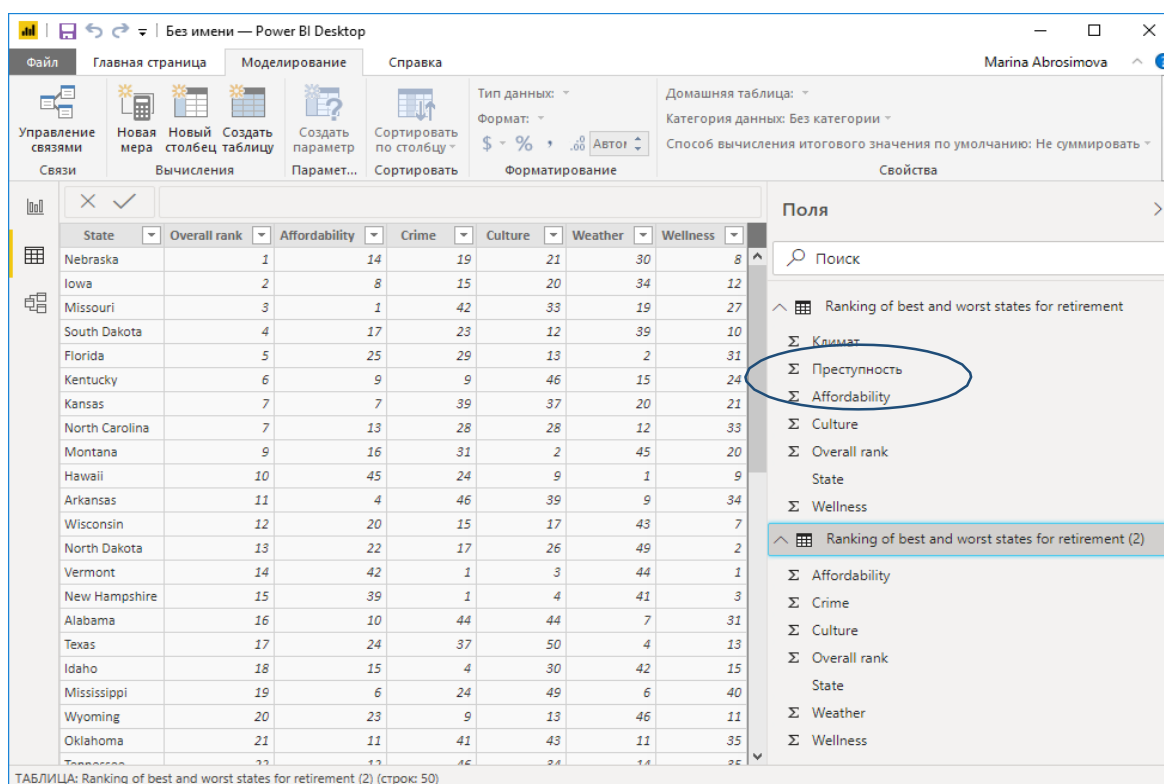
Например, для дальнейшей работы переименуем поля, дав им имена на русском языке:





**Задание 6.** Измените имя поля «Overall rank» на «Общий рейтинг» и т.д.

Обратите внимание, что в области **Поля** отображаются сделанные Вами изменения – формируется конкретное представление данных. Исходный источник данных (выделено затемнением) остается без изменений. В этом Вы можете убедиться, щелкнув на имени таблицы Ranking of best and worst states for retirement (2):

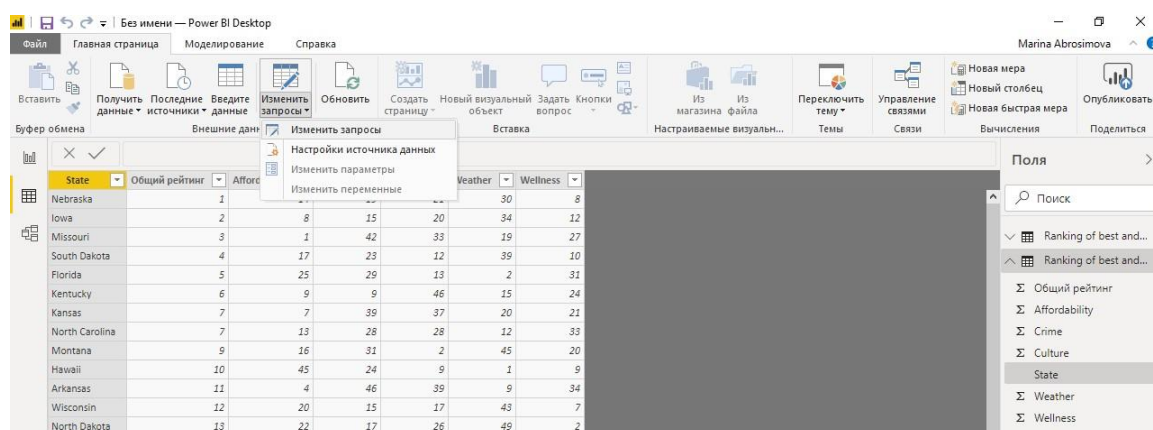


**Задание 7.** Отсортируйте данные по возрастанию значений в поле «Штат».

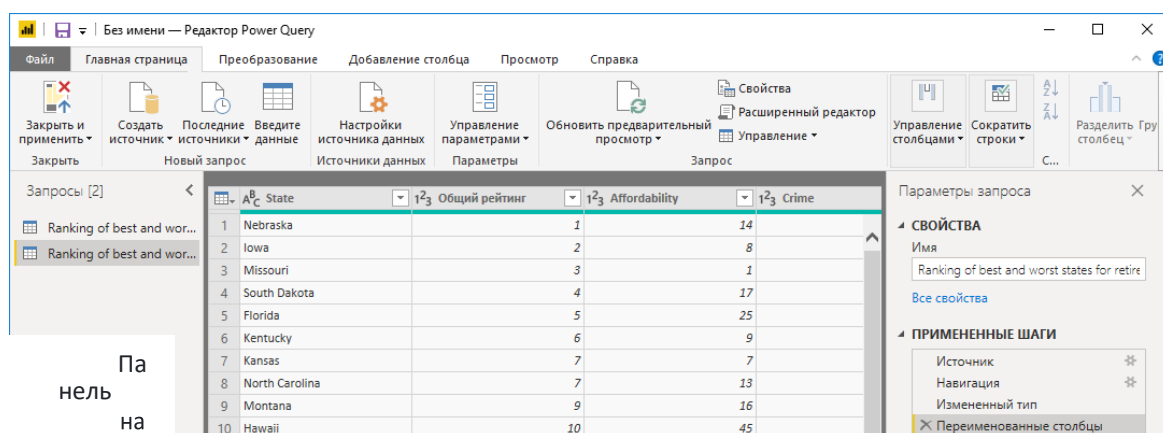
## Использование редактора запросов

При формировании данных в Редакторе запросов вы создаете пошаговые инструкции (которые автоматически выполняются в этом Редакторе запросов) для настройки данных по мере их загрузки и отображения в редакторе. Это также, как и в случае с редактированием на вкладке **Главная страница** или **Моделирование** не влияет на исходный источник данных; корректируется или формируется только это конкретное представление данных.

Для использования Редактора запросов нужно на вкладке **Главная страница** выбрать **Изменить запросы**:



Откроется Редактор запросов (Редактор Power Query):



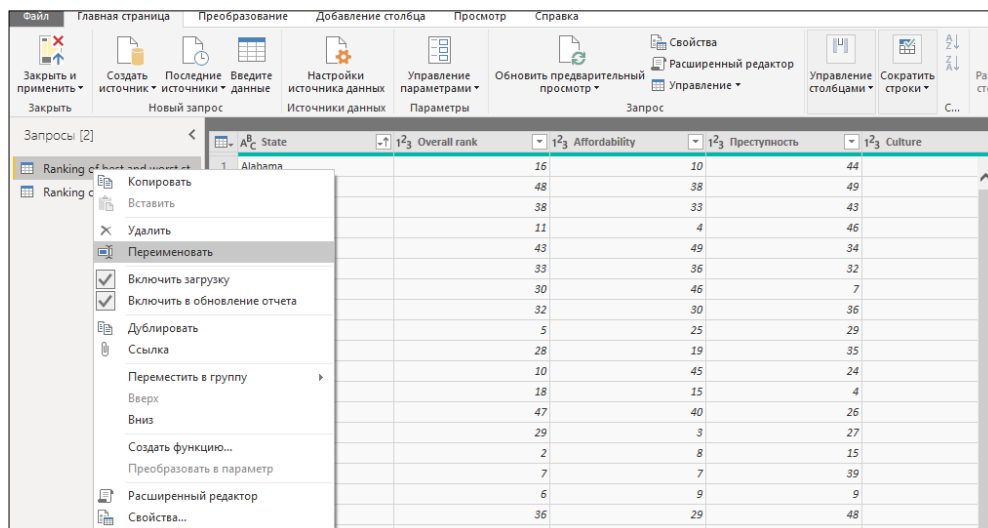
Использовать Редактор запросов для изменения данных удобнее для многих действий с данными. Например, если Вы хотите быстро удалить несколько столбцов, можно их предварительно выделить, что Редактор запросов позволяет сделать.

Выполняемые действия с данными (переименования полей или таблицы, преобразование типа данных, удаление столбцов и пр.) записываются Редактором запросов, и каждый раз, когда этот запрос подключается к источнику данных, эти действия выполняются, чтобы данные всегда были сформированы указанным образом. Это происходит всякий раз, когда вы используете этот запрос в Power BI Desktop или, когда кто-либо другой использует ваш открытый для общего доступа запрос, например, в **службе Power BI**. Эти действия последовательно записываются в области **Параметры запроса** в разделе **Примененные шаги**. Например, в Вашем случае, записано, что было переименование столбцов (полей).

Сейчас Вы можете сделать дополнительные изменения, чтобы получить еще более удобные данные. Например, удалить столбец, если он не нужен, изменить имя таблицы и пр.

### *Переименование таблицы*

Для того, чтобы переименовать таблицу, выделите ее имя в навигаторе запросов слева и выберите из контекстного меню **Переименовать**.

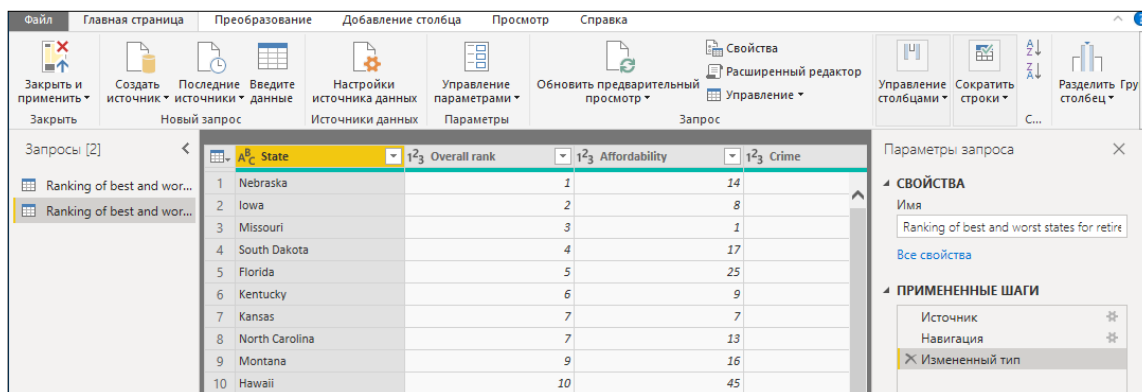


**Задание 8.** Переименуйте таблицу. Дайте ей название на русском языке «Рейтинг лучших и худших штатов для выхода на пенсию».

### *Редактирование параметров запроса*

В области **Параметры запроса** в разделе **примененных шагов** будут отражаться все сделанные изменения. По мере внесения в данные дополнительных изменений редактор запросов будет записывать эти изменения в раздел **примененных шагов**, где вы можете настраивать шаги, повторно переходить в шаги, изменять их порядок или удалять их при необходимости.

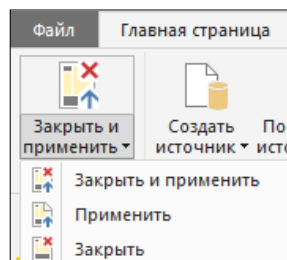
В Редакторе запросов важна последовательность примененных шагов, так как она может влиять на способ формирования данных. Один шаг может повлиять на последующий шаг; если удалить шаг из раздела **Примененные действия**, то последующие шаги могут работать не так, как предполагалось, из-за влияния последовательности шагов запроса.



**Задание 9.** Переименуйте столбец Culture, дав русское имя «Культура». Удалите шаг «Переименованные столбцы».

*Сохранение изменений в редакторе запросов*

Закончив, нужно выбрать **Заккрыть и применить** на вкладке Главная страница, и Power BI Desktop применит все изменения и закроет редактор запросов.



Если Вы не хотите сохранять результаты изменений, то выберите **Зарыть**.  
**Задание 10.**

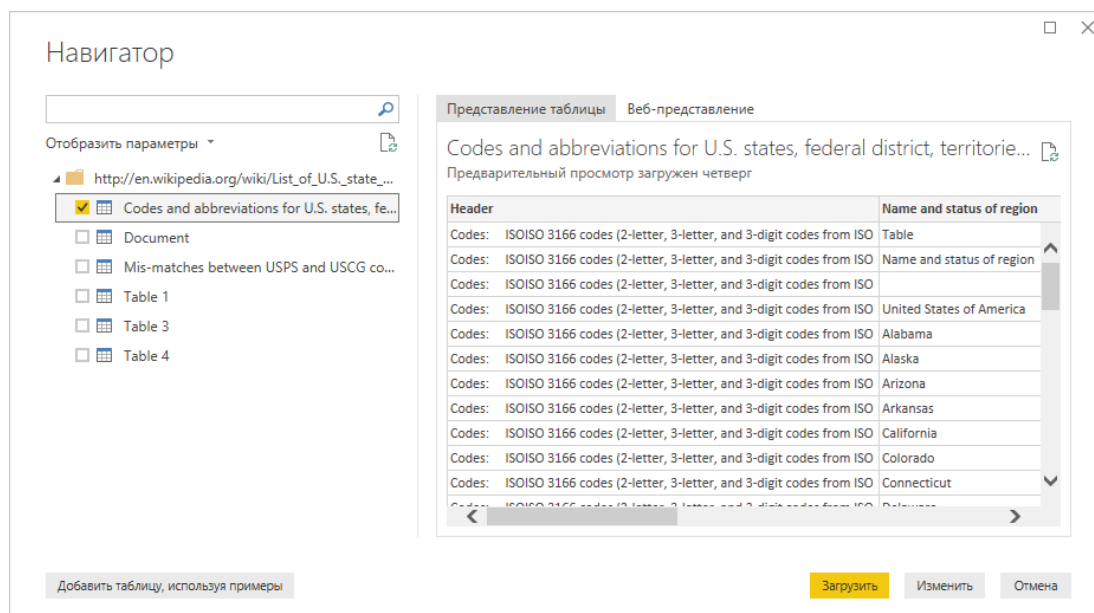
1. Переименуйте все поля таблицы, дав им имена на русском языке.
2. Сохраните изменения и выйдите из редактора запроса.
3. Просмотрите таблицу «Рейтинг лучших и худших штатов для выхода на пенсию» и область **Поля**.

*Удаление строк в таблице*

Сформированные данные о различных штатах представляют интерес и будут использоваться для создания дополнительных аналитических исследований и запросов. Для удобства нужно каким-либо способом связать названия штатов с их кодами. Для этого можно использовать другой общедоступный источник данных:

[http://en.wikipedia.org/wiki/List\\_of\\_U.S.\\_state\\_abbreviations](http://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations)

На вкладке **Главная** страница в представлении **Данные** выберите **Получить данные**, далее **Интернет**, введите адрес, нажмите кнопку **ОК**. В окне **Навигатор** отобразится содержимое этой веб-странице.



### Задание 11.

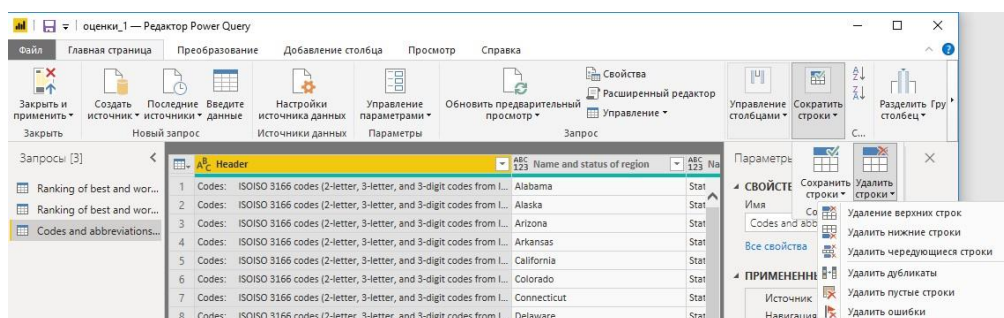
1. Откройте окно Навигатора для источника данных [http://en.wikipedia.org/wiki/List\\_of\\_U.S.\\_state\\_abbreviations](http://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations).
2. Просмотрите в окне предварительного просмотра справа все данные.

Нужные данные о кодах находятся в таблице «Codes and abbreviations for US states, federal district, territories, and other areas», однако для Вас нужны только два столбца: «Name and status of region» и «ANSI». Поэтому уже в процессе открытия данных можно сделать выборку данных, запустив Редактор запроса. Для этого нужно в Навигаторе нажать кнопку **Изменить**.

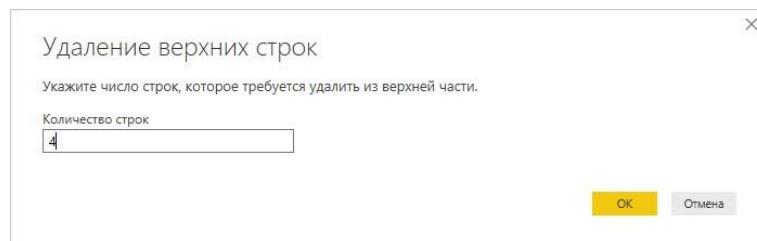
### Задание 12. В Навигаторе войдите в Редактор запроса.

В таблице сверху присутствуют ненужные строки, получившиеся в результате формирования с веб-страницы. Они не нужны. Для их удаления нужно в Редакторе запроса, в котором Вы сейчас находитесь:

- на вкладке **Главная страница** выбрать **Сократить строки**;
- далее **Удалить верхние строки**:



- далее указать количество удаляемых строк:



### Задание 13.

1. Удалите первые четыре строки.
2. Удалим последние 26 строк: это все территории, которые включать ненужно. Поступаем аналогично.

В итоге в таблице осталась 51 строка.

#### *Удаление столбцов в таблице*

Вам требуется только сопоставление штата с его официальным двухбуквенным кодом, поэтому мы можем удалить другие столбцы. Для их удаления нужно в **Редакторе запроса**, в котором Вы сейчас находитесь:

- Выделить столбцы, которые нужно удалить, используя клавишу Shift и щелчки мышью;
- на вкладке **Главная страница** выбрать **Управление столбцами**;  
далее **Удалить столбцы**.

### Задание 14.

1. Удалите все столбцы, оставив только столбцы «Name and status of region» и «ANSI».
2. Измените имена столбцов на «Штат», «Код».
3. Измените имя таблицы на «Коды штатов».
4. Сохраните результаты запроса, выполнив **Заккрыть и применить** на вкладке **Главная страница**.

### **Объединение данных**

После того как данные сформированы – появились две новые таблицы (или два запроса, поскольку таблицы, которые у нас получились, представляют собой результат запросов к данным и их часто называются *запросами*) -

«Рейтинг лучших и худших штатов для выхода на пенсию» и «Коды штатов», можно эти таблицы (или запросы) объединить в одну.

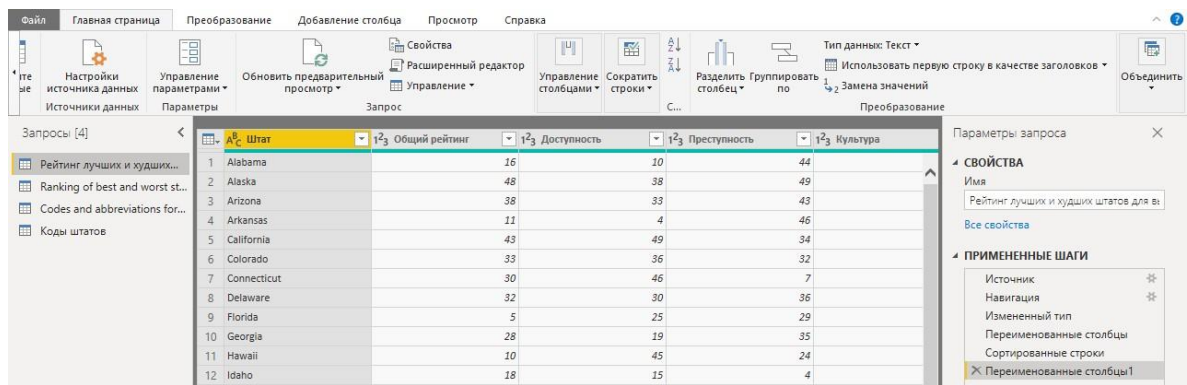
Существует два основных способа объединения запросов: *слияние* и *дополнение*.

Если имеется один или несколько столбцов, которые требуется добавить в другой запрос, нужно выполнить **слияние** запросов. При наличии дополнительных строк данных, которые нужно добавить в существующий запрос, выполняется **дополнение** запроса.

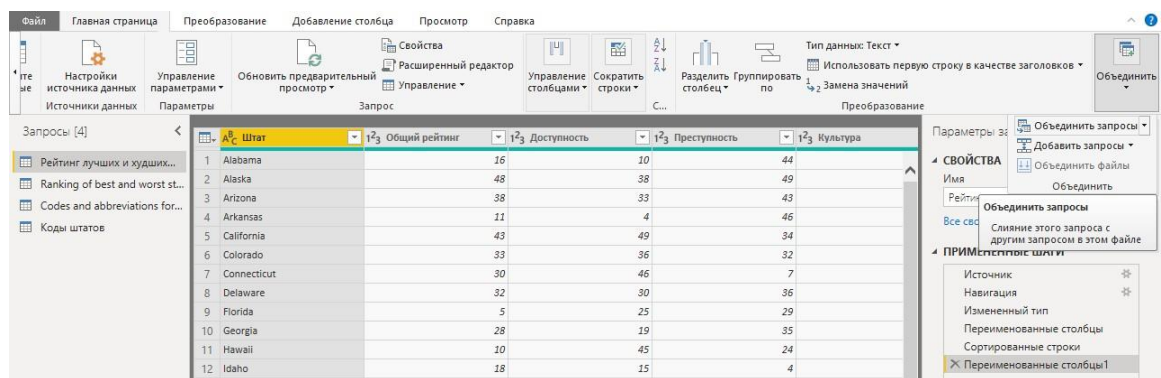
В данном случае мы хотим слить запросы. Для этого:

- войти в режим запроса («Изменить запросы»);
- выбрать запрос, с которым хотим выполнить слияние другого запроса. Для этого на правой панели выберите нужный запрос. У Вас это таблица (запрос) «Рейтинг лучших и худших штатов для выхода на пенсию»





- выбрать **Объединить запросы** на вкладке **Главная страница** (нужно перемотать ленту **Главная страница** вправо).




Появится окно **Слияние**, предлагающее выбрать таблицу для слияния выбранной и соответствующие столбцы для слияния.

Слияние

✕

Выберите таблицу и совпадающие столбцы для создания объединенной таблицы.

Рейтинг лучших и худших штатов для выхода на пенсию 

Штат	Общий рейтинг	Доступность	Преступность	Культура	Климат	Здоровье
Alabama	16	10	44	44	7	31
Alaska	48	38	49	24	50	26
Arizona	38	33	43	39	10	29
Arkansas	11	4	46	39	9	34
California	43	49	34	17	13	19

Предварительный просмотр недоступен

Тип соединения

Внешнее соединение слева (все из первой таблици... ▾)

☐ Использовать нечеткие соответствия для слияния

> Параметры нечеткого слияния

OK

Отмена

Выберите таблицу и совпадающие столбцы для создания объединенной таблицы.

## Рейтинг лучших и худших штатов для выхода на пенсию

Штат	Общий рейтинг	Доступность	Преступность	Культура	Климат	Здоровье
Alabama	16	10	44	44	7	31
Alaska	48	38	49	24	50	26
Arizona	38	33	43	39	10	29
Arkansas	11	4	46	39	9	34
California	43	49	34	17	13	19

Предварительный просмотр недоступен

### Тип соединения

Внешнее соединение слева (все из первой таблиц... ▾

☐ Использовать нечеткие соответствия для слияния

### ▷ Параметры нечеткого слияния

OK

Отмена

- Определить таблицу для слияния с выбранной - «Коды штатов» и соответствующие столбцы для слияния «Штат»:

## Слияние

Выберите таблицу и совпадающие столбцы для создания объединенной таблицы.

Рейтинг лучших и худших штатов для выхода на пенсию

Штат	Общий рейтинг	Доступность	Преступность	Культура	Климат	Здоровье
Alabama	16	10	44	44	7	31
Alaska	48	38	49	24	50	26
Arizona	38	33	43	39	10	29
Arkansas	11	4	46	39	9	34
California	43	49	34	17	13	19

Коды штатов

Штат	Код
Alabama	AL
Alaska	AK
Arizona	AZ
Arkansas	AR
California	CA

Тип соединения

Внешнее соединение слева (все из первой таблиц...)

☐ Использовать нечеткие соответствия для слияния

Параметры нечеткого слияния

Выделенный фрагмент соответствует строкам из первой таблицы (50...

OK

Отмена

Выберите таблицу и совпадающие столбцы для создания объединенной таблицы.

## Рейтинг лучших и худших штатов для выхода на пенсию

Штат	Общий рейтинг	Доступность	Преступность	Культура	Климат	Здоровье
Alabama	16	10	44	44	7	31
Alaska	48	38	49	24	50	26
Arizona	38	33	43	39	10	29
Arkansas	11	4	46	39	9	34
California	43	49	34	17	13	19

Коды штатов

Штат	Код
Alabama	AL
Alaska	AK
Arizona	AZ
Arkansas	AR
California	CA

Тип соединения

Внешнее соединение слева (все из первой таблиц... ▾

☐ Использовать нечеткие соответствия для слияния

### ▷ Параметры нечеткого слияния

✓ Выделенный фрагмент соответствует строкам из первой таблицы (50...

Ok

Отмена

- кнопка **ОК**.



**Задание 15.** Выполните слияние таблиц «Рейтинг лучших и худших штатов для выхода на пенсию» и «Коды штатов»


### Просмотр объединенных данных

В результате запроса в таблице «Рейтинг лучших и худших штатов для выхода на пенсию» создается новый столбец «Коды штатов», содержащий данные из таблицы (запроса), которая была объединена с существующим запросом.

	1 <sup>2</sup> <sub>3</sub> Преступность	1 <sup>2</sup> <sub>3</sub> Культура	1 <sup>2</sup> <sub>3</sub> Климат	1 <sup>2</sup> <sub>3</sub> Здоровье	Коды штатов
1	44	44	7	31	Table
2	49	24	50	26	Table
3	43	39	10	29	Table
4	46	39	9	34	Table
5	34	17	13	19	Table
6	32	22	37	6	Table
7	7	8	29	5	Table
8	36	9	16	41	Table
9	29	13	2	31	Table

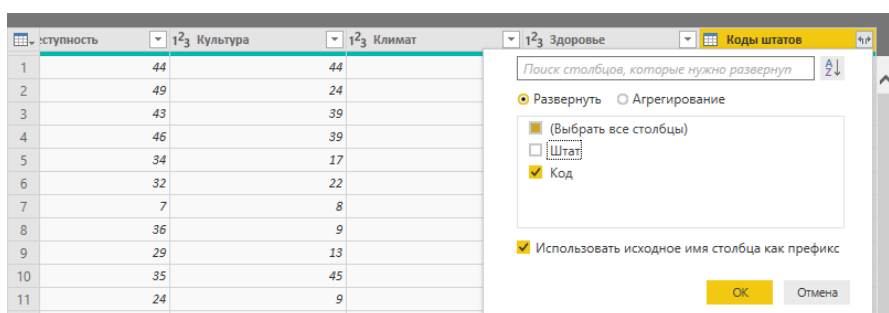
Обратите внимание, что в результате запроса сформировано 50 строк(см. строку состояния внизу экрана).

Все столбцы из присоединенного запроса включены в новый столбец. Можно их развернуть. Для этого:

- щелкните значок развертывания (  ).

1 <sup>2</sup> <sub>3</sub> Здоровье	Коды штатов
31	Table
26	Table
29	Table
34	Table

- Появится окно **Развернуть**. Нам нужен только столбец «Код штата»,поэтому мы выберем только этот столбец, а затем нажмем кнопку **ОК**.



В итоге получаем объединенные данные:

Скриншот интерфейса Power BI Desktop. В центре экрана отображается таблица с 5 столбцами: t2: Культура, t2: Климат, t2: Здоровье, t2: Коды штатов.Код. Таблица содержит 17 строк данных. В правой панели открыт раздел 'Параметры запроса', который разделен на 'СВОЙСТВА' (с полем 'Имя' и значением 'Рейтинг лучших и худших штатов') и 'ПРИМЕНЕННЫЕ ШАГИ' (список действий: Источник, Навигация, Измененный тип, Переименованы..., Сортированные..., Переименованы..., Объединенные за..., Объединенные за..., Удаленные столб..., Развернутый эле...). В нижней части экрана отображается информация: 'СТОЛБЦОВ: 5, СТРОК: 50' и 'ПРЕДВАРИТЕЛЬНЫЙ ПРОСМОТР ЗАГРУЖЕН В 15:45'.

	t2: Культура	t2: Климат	t2: Здоровье	t2: Коды штатов.Код
1	44	44	7	31 AL
2	49	24	50	26 AK
3	43	39	10	29 AZ
4	46	39	9	34 AR
5	34	17	13	19 CA
6	32	22	37	6 CO
7	7	8	29	5 CT
8	36	9	16	41 DE
9	29	13	2	31 FL
10	35	45	5	44 GA
11	24	9	1	9 HI
12	4	30	42	15 ID
13	26	32	23	49 IL
14	27	41	25	46 IN
15	15	20	34	12 IA
16	39	37	20	21 KS
17	^	^	^	^

***Примечание.** Можно опробовать разные способы выведения таблицы на экран. Вы можете немного поэкспериментировать, и, если вас не устроит результат, просто удалите этот шаг из списка **примененных действий** в области **параметров запроса**; ваш запрос вернется в состояние до применения шага **Развернуть**. Это как бесплатная попытка, которую вы можете повторять сколько угодно, пока процесс развертывания не будет происходить так, как нужно.*

**Задание 16.** Сохраните изменения в запросе после объединения данных.

Теперь у нас есть один запрос (таблица), объединяющий два источника данных, каждый из которых сформирован так, как нам нужно. Этот запрос может служить основой для большого количества дополнительных подключений к представляющим интерес данным, таким как стоимость содержания, демографические данные или вакансии в любом штате.

**Задание 17.** Сохраните файл «Начало работы с Power BI Desktop» и закройте его.

### Задание для самостоятельного выполнения

Вам нужно подготовить данные для анализа тенденций развития телекоммуникаций в России.

**Задание 0.** Создать в Power BI Desktop в своей папке новый файл «Собственный проект».

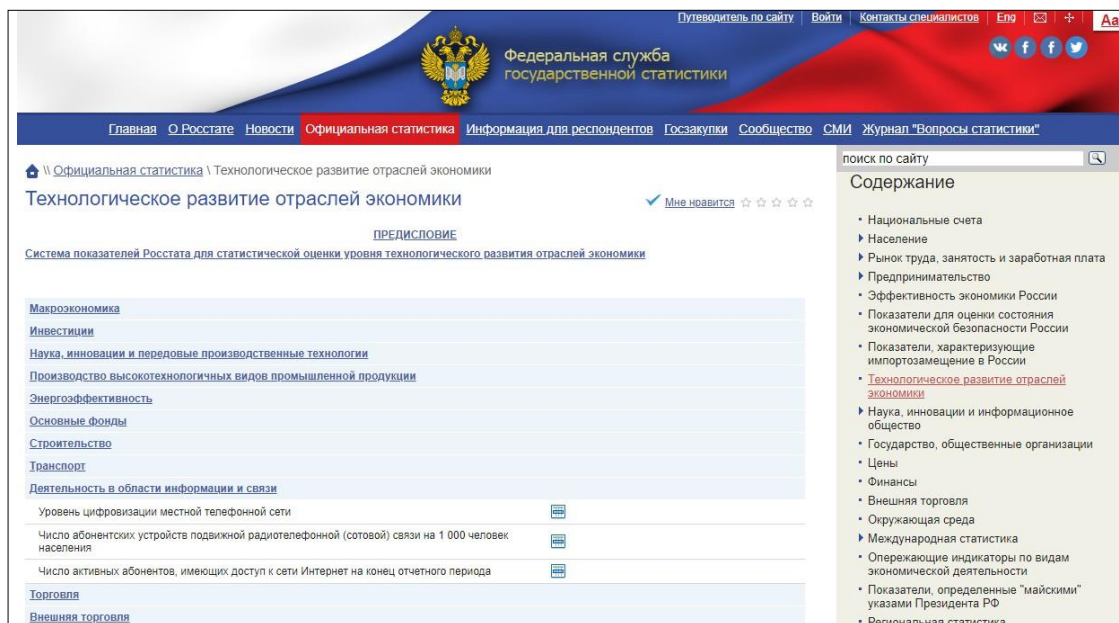
Источниками данных могут являться государственные и негосударственные поставщики информационных ресурсов. Ваши источники:

- Первый источник данных: Федеральная служба государственной статистики (Росреестр) [www.gks.ru](http://www.gks.ru);
- Второй источник данных: Институт демографии Национального исследовательского университета "Высшая школа экономики" <http://www.demoscope.ru> ;
- Третий источник данных: универсальная интернет-энциклопедия «Википедия» <https://ru.wikipedia.org/wiki>.

## Задание 1. Получить и преобразовать данные первого источника

1. Получите данные из первого источника. Для этого на портале Федеральной службы государственной статистики (Росреестр) [www.gks.ru](http://www.gks.ru). Далее:

- Зайти в раздел «Официальная статистика», далее «Технологическое развитие отраслей экономики».



- Загрузить на свой компьютер Excel-файл «Число активных абонентов, имеющих доступ к сети Интернет на конец отчетного периода»:

	A	D	E	F	G	H	I	J	K	L	M	N	O
1		абонентов фиксированного широкополосного доступа к сети Интернет											
2		на конец отчетного периода, (единиц)											
3													
4		2013	2014				2015				2016		
5			I квартал	II квартал	III квартал	IV квартал	I квартал	II квартал	III квартал	IV квартал	I квартал	II квартал	III квартал
6	Российская Федерация	23 745 346	24 278 952	24 211 170	24 348 174	24 825 202	25 225 065	25 326 097	25 568 870	26 755 612	27 328 700	26 954 207	27 117 461
7	Центральный федеральный округ	7 270 945	7 336 987	7 371 885	7 465 436	7 687 619	7 536 309	7 542 581	7 636 822	8 011 643	8 147 905	8 108 667	8 118 320
8	Белгородская область	175 181	178 136	177 918	182 240	200 684	278 494	279 253	274 686	272 859	278 629	284 221	286 387
9	Брянская область	158 018	157 621	159 003	159 630	163 072	160 611	155 086	157 888	163 202	164 839	165 366	169 110
10	Владимирская область	215 349	218 727	220 338	194 821	196 652	187 448	181 686	184 262	193 427	235 328	235 882	238 854
11	Воронежская область	406 407	411 042	413 964	418 784	429 766	457 268	466 110	458 205	463 544	467 455	465 544	468 550
12	Ивановская область	119 356	123 280	123 649	126 177	129 577	180 424	182 279	184 874	187 744	190 290	186 828	185 306
13	Калужская область	173 972	175 286	175 944	180 905	186 923	199 500	200 436	207 836	216 614	211 764	210 648	211 581
14	Костромская область	118 019	119 407	119 506	120 274	123 148	122 065	121 543	123 508	124 759	125 777	122 996	122 085
15	Курская область	132 694	136 440	139 846	146 069	166 310	234 809	247 517	266 935	252 369	256 656	258 954	265 636
16	Липецкая область	148 262	149 194	145 460	143 842	147 938	178 318	181 757	175 948	172 458	179 967	180 015	184 091
17	Московская область	622 350	646 511	653 030	669 754	682 180	691 337	666 907	688 084	771 125	818 791	834 205	873 327
18	Орловская область	93 033	94 871	96 659	99 356	102 781	152 663	152 498	151 227	148 722	152 340	151 283	151 433
19	Рязанская область	144 946	147 378	149 990	151 952	154 451	180 696	181 087	185 809	191 094	195 211	197 532	201 811
20	Смоленская область	145 712	146 013	149 974	152 131	160 933	175 815	174 057	175 995	181 039	183 470	180 745	184 299
21	Тамбовская область	149 605	153 562	156 785	160 357	163 590	160 729	158 095	163 395	165 409	170 344	167 356	165 484
22	Тверская область	156 820	157 091	157 107	162 604	164 765	160 804	158 655	162 232	165 326	169 249	168 376	170 021
23	Тульская область	280 220	283 987	284 264	290 684	266 545	303 097	305 425	309 050	316 833	307 646	309 318	313 073
24	Ярославская область	234 872	236 070	231 376	234 225	240 448	254 536	255 260	258 750	271 117	275 055	275 647	278 134
25	г. Москва	3 796 129	3 802 371	3 817 072	3 871 631	4 007 856	3 457 695	3 474 930	3 508 138	3 754 362	3 765 094	3 713 851	3 669 138
26	Северо-Западный федеральный округ	2 771 064	2 822 064	2 809 305	2 810 565	2 827 516	2 994 045	3 004 896	3 022 738	3 083 849	3 122 853	3 108 359	3 129 745
27	Республика Карелия	162 450	165 877	167 748	168 760	174 221	176 276	177 146	176 892	182 207	183 600	183 900	183 708
28	Республика Коми	147 758	149 470	152 991	149 842	151 879	222 699	222 699	222 651	228 841	234 493	236 802	242 050
29	Архангельская область	192 972	194 559	190 504	192 991	198 519	200 834	200 999	200 718	205 794	207 598	203 427	202 277
30	Вологодская область	210 928	216 131	217 619	220 280	225 607	229 430	227 628	231 857	237 584	240 099	241 088	240 108
31	Калининградская область	209 100	211 887	215 756	189 738	180 555	210 912	209 259	209 467	212 290	214 545	215 769	215 595
32	Ленинградская область	175 533	168 834	167 336	168 289	172 537	173 459	173 493	175 567	179 471	178 102	178 168	178 690
33	Мурманская область	142 003	154 986	156 322	158 541	149 470	168 308	170 911	173 600	178 713	183 838	185 411	190 375
34	Новгородская область	94 336	92 728	93 135	92 350	94 937	94 304	91 070	92 134	96 282	95 801	95 744	96 454
35	Псковская область	54 450	55 048	54 712	55 046	55 530	55 301	54 819	51 759	57 560	57 650	57 652	58 026
36	г. Санкт-Петербург	1 381 534	1 412 544	1 392 982	1 414 728	1 424 261	1 462 967	1 476 872	1 488 093	1 505 107	1 527 127	1 510 398	1 522 462
37	Южный федеральный округ	1 811 135	1 873 285	1 893 648	1 910 630	1 962 680	1 943 099	1 985 389	2 025 206	2 200 748	2 234 862	2 198 323	2 294 209
38	Республика Адыгея	25 200	26 486	26 935	27 317	27 803	28 311	28 600	29 077	31 722	42 266	42 800	43 576
39	Республика Калмыкия	21 301	22 273	23 165	23 584	23 953	23 414	24 263	23 663	26 761	26 622	25 860	25 590
40	Республика Крым	-	...	...	...	9 468	10 325	19 468	32 068	47 423	65 322	71 278	78 549
41	Краснодарский край	712 703	736 865	749 775	777 574	810 859	829 308	847 705	867 479	943 285	949 842	933 854	935 378

- выбрать нужный лист, например ШПД – широкополосный доступ;
- скопировать лист в новый файл и сохранить Excel-файл «Число активных абонентов, имеющих доступ к сети Интернет на конец отчетного периода» в своей папке;
- использовать Excel-файл «Число активных абонентов, имеющих доступ к сети Интернет на конец отчетного периода» как источник данных в Power BI Desktop. Выберите лист Excel-файла «ШПД» – широкополосный доступ.

## 2. Преобразуйте данные:

- Удалите ненужные строки сверху;
- Удалите ненужные строки снизу. Если их очень много, то просмотрите, до какой строки Вам нужны данные и в режиме Редактора запросов выполните «Сократить строки», далее «Сохранить верхние строки»

Имя столбца	Просмотр	Справка
Управление параметрами	Обновить предварительный просмотр	Свойства
Параметры	Запрос	Расширенный редактор
		Управление
		Управление столбцами
		Сократить строки
		Разделить столбцы
		Группировать
		Свойства
Аб. Регион	123 2014	123 2015
1 Алтайский край	2390638	2384
2 Амурская область	811274	809
3 Республика Адыгея	446406	4491
4 Архангельская область	1191785	11835
5 Республика Алтай	211645	2135
6 Архангельская область (без АО)	1148760	11401
7 Республика Башкортостан	4069698	40716

- в диалоге «Сократить строки» укажите, что нужно сохранить именно нужное Вам количество строк:

×

### Сохранить верхние строки

Укажите, сколько строк сохранить.

Количество строк

94

ОК
Отмена

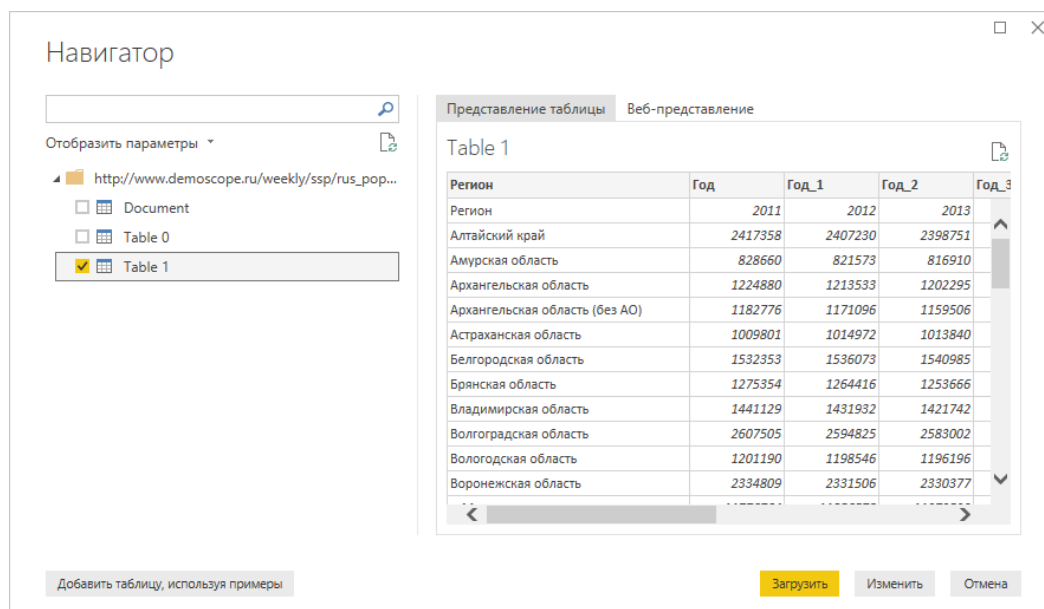
- Удалите столбцы данных по 2011, 2012, 2013 гг.;
- По оставшимся годам оставьте данные только по 1 кварталу;
- Переименуйте столбцы по примеру: ШПД 2014, ШПД 2015 и т.п.
- Измените имя таблицы на «Число активных абонентов, имеющих до-ступ к Интернет».

**Задание 2.** Получить и преобразовать данные второго источника.

1. Получите данные из второго источника - сайта Института демографии Национального исследовательского университета "Высшая школа экономики", Таблица «Население субъектов Российской Федерации на 1 января 2011 - 2019 гг.»:

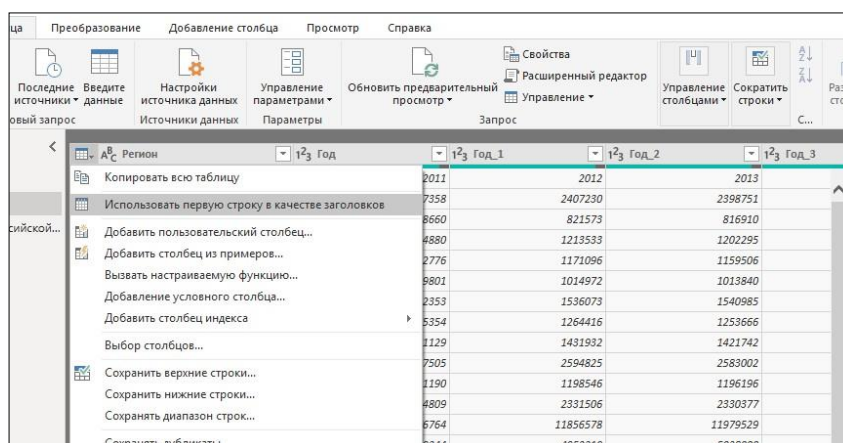
Население субъектов Российской Федерации на 1 января 2011 - 2019 гг.									
Регион	Год								
	2011	2012	2013	2014	2015	2016	2017	2018	2019
Алтайский край	2417358	2407230	2398751	2390638	2384708	2378774	2365680	2350080	2332813
Амурская область	828880	821573	816910	811274	809814	805689	801752	798424	793194
Архангельская область	1224880	1213533	1202295	1191785	1183501	1174078	1165750	1155028	1144119
Архангельская область (без АО)	1182776	1171096	1159506	1148760	1140109	1130240	1121813	1111031	1100290
Астраханская область	1009801	1014972	1013840	1018516	1021942	1018826	1018886	1017514	1014085
Белгородская область	1532353	1536073	1540985	1544108	1547845	1550137	1552885	1549876	1547418
Брянская область	1275354	1264416	1253666	1242599	1232885	1225741	1220530	1210882	1200187
Владимирская область	1441129	1431932	1421742	1413321	1405741	1397168	1389599	1378337	1365805
Волгоградская область	2607505	2594825	2583002	2569126	2557689	2545937	2535202	2521276	2507509
Вологодская область	1201190	1198546	1196196	1193371	1191009	1187685	1183880	1179689	1167713
Воронежская область	2334809	2331506	2330377	2328959	2331511	2333477	2335408	2333768	2327821
г. Москва	11776764	11856578	11979529	12108257	12184015	12330126	12380664	12508468	12615279
г. Санкт-Петербург	4999344	4953219	5028000	5131942	5197114	5225690	5281579	5351935	5383890
г. Севастополь					400865	416263	428753	436670	443212
Дальневосточный федеральный округ	6284932	6265833	6251496	6226640	6211384	6194999	6182679	6165284	6188823
Еврейская автономная область	176304	174412	172871	170377	168408	166120	164217	162014	159913
Забайкальский край	1106155	1099396	1095169	1090344	1087479	1083012	1078983	1072806	1068785
Ивановская область	1080109	1054040	1048961	1043130	1037079	1029838	1023170	1014646	1004180
Иркутская область	2427954	2424355	2422026	2418348	2415695	2412800	2408901	2404195	2397763
Кабардино-Балкарская Республика	859792	859063	858948	858397	860808	862254	864454	865828	866219
Калининградская область	941823	946796	954773	963128	968256	976439	986261	994599	1002187
Калужская область	1009191	1008229	1005585	1004544	1009709	1009772	1014570	1012156	1009380

- ☐ Получить данные из источника данных в интернет с адреса:  
[http://www.demoscope.ru/weekly/ssp/rus\\_pop\\_reg\\_2011.php](http://www.demoscope.ru/weekly/ssp/rus_pop_reg_2011.php)
- ☐ Загрузить Table 1.



## 2. Преобразуйте данные:

- ☐ Создайте запрос для того, чтобы использовать первую строку в качестве заголовка. Для этого создайте запрос и в Редакторе запроса откройте меню рабочей области выберите **Использовать первую строку в качестве заголовка**:



- ☐ Удалите столбцы 2011, 2012, 2013, 2019.
- ☐ Переименуйте столбцы по примеру: Население 2014, Население 2015 ит.д.
- ☐ Переименуйте таблицу: с Table 1 на «Население субъектов РФ на 1 ян-варя 2014 - 2018»

## Задание 3. Получить и преобразовать данные третьего источника.

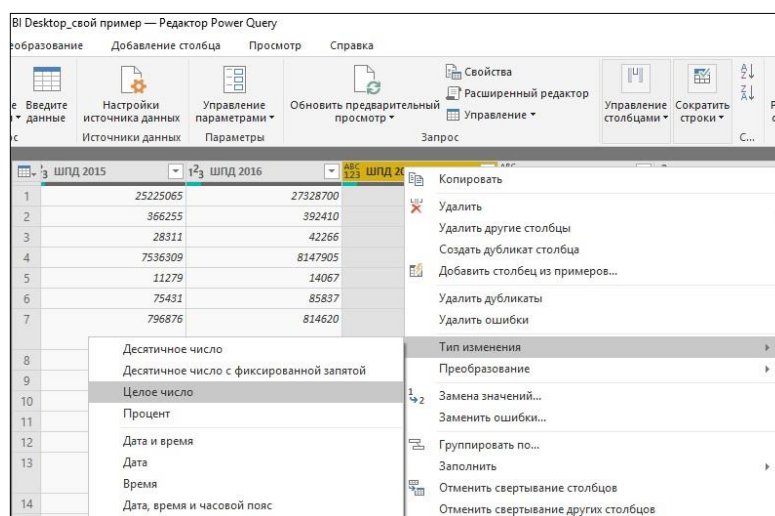
1. Получите данные из третьего источника: универсальной интернет-энциклопедии «Википедия», страница «Действующие главы субъектов Российской Федерации»



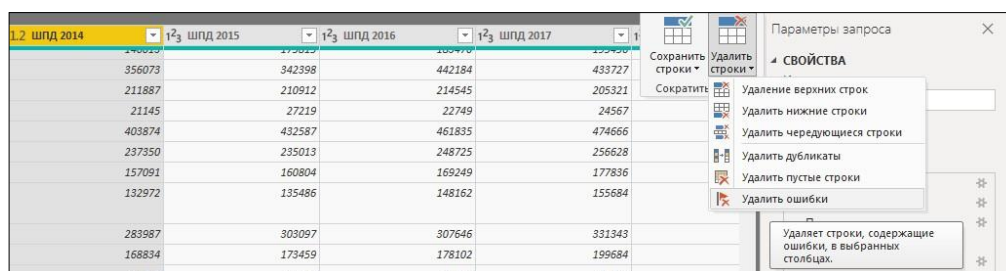
[illegible]

#### Задание 4. Объедините данные из источников:

берите Тип изменения, далее выберите нужный Вам тип, например, «Целое число»:



- Если в процессе преобразования данных объединенной таблицы возникнут ошибки, то Power BI Desktop укажет Вам, словом, error в соответствующей ячейке таблицы. В этом случае выполните в Редакторе запроса «Удалить строки», далее «Удалить ошибки».



**Задание 5.** Сохраните файл «Собственный проект».

## Контрольные вопросы

1. Для чего предназначен Power BI Desktop и для чего Power BI Service?
2. Опишите кратко функциональные возможности Power BI Desktop.
3. Для чего используется представление **Данные** в Power BI Desktop?
4. Что понимается под настройкой данных, и какие инструменты Power BIdesktop позволяют ее выполнить?
5. В чем состоят преимущества использования **Редактор запросов** при настройке данных по сравнению с инструментами вкладки **Главная**?
6. Какую роль выполняет раздел **Примененные шаги** окна **Редактора запросов**?
7. В чем отличие слияния от дополнения при объединении данных, полученных из разных источников?



## Практическое занятие 8. Визуализация данных в Power BI Desktop

**Цель работы:** освоить приемы визуализации данных.

**Задачи работы:** создать и отредактировать визуализации данных, наложить фильтры и произвести вычисления в отчетах по примеру, выполнить задание для самостоятельной работы.

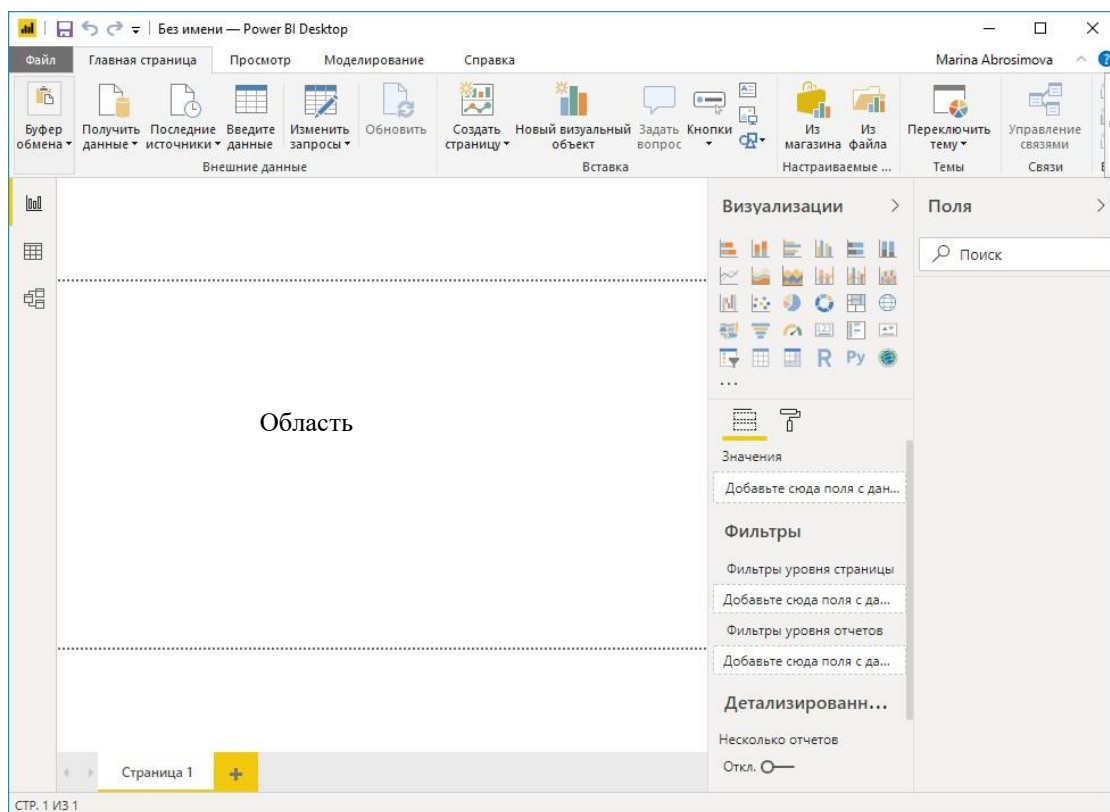
Визуализация данных в Power BI Desktop доступна в представлении **Отчет**. В отчетах Power BI Desktop могут использоваться:

- данные более чем 70-ти источникам (txt, xlsx, csv файлы, SQL БД, Интернет-подключение, специально преднастроенные службы, прямое подключение к любым сервисам через API и так далее).
- создаваться новые данные на основе моделирования новых таблиц, фильтров, вычислений, формул и дополнительных метрик;
- интерактивные визуализации графиков, гистограмм, диаграмм, таблиц, карточек, матриц, срезов, карт и прочих визуализаций.

Преимущество Power BI Desktop состоит в том, что при нажатии кнопки **Обновить** данные в модели файла и отчете обновляются измененными данными из исходного источника данных.

Отчеты Power BI Desktop могут быть опубликованы в **Power BI Service** - онлайн-аналоге, который специализируется на мониторинге и анализе готовых отчетов. В этом случае Ваши отчеты могут быть использованы в групповой работе профессионального сообщества.

**Задание.** Запустите Power BI Desktop. Перейдите в представление **Отчет** щелчком на значке **Отчет** на правой панели.



Представление **Отчет** имеет пять основных областей:

1. **Лента** меню с вкладками, на которой отображаются стандартные задачи, связанные с отчетами и визуализациями;
2. Представление **Отчет**, где создаются и оформляются визуализации;
3. область вкладок **Страницы** внизу, где можно выбрать и добавить страницу отчета;
4. область **Визуализации**, где можно выбрать тот или иной инструмент визуализации, изменить визуализации, настроить цвета и оси, применить фильтры, перетащить поля и многое другое;
5. область **Поля**, где элементы запросов и фильтры можно перетащить в представление **Отчет** или область **Фильтры** зоны Визуализации.

Области **Визуализации** и **Поля** можно свернуть, щелкнув маленькую стрелку сбоку, чтобы освободить больше места в представлении **Отчет** для построения впечатляющих визуализаций. При изменении представлений вы также увидите эти указывающие вверх или вниз стрелки, означающее, что можно развернуть или свернуть этот раздел соответственно.

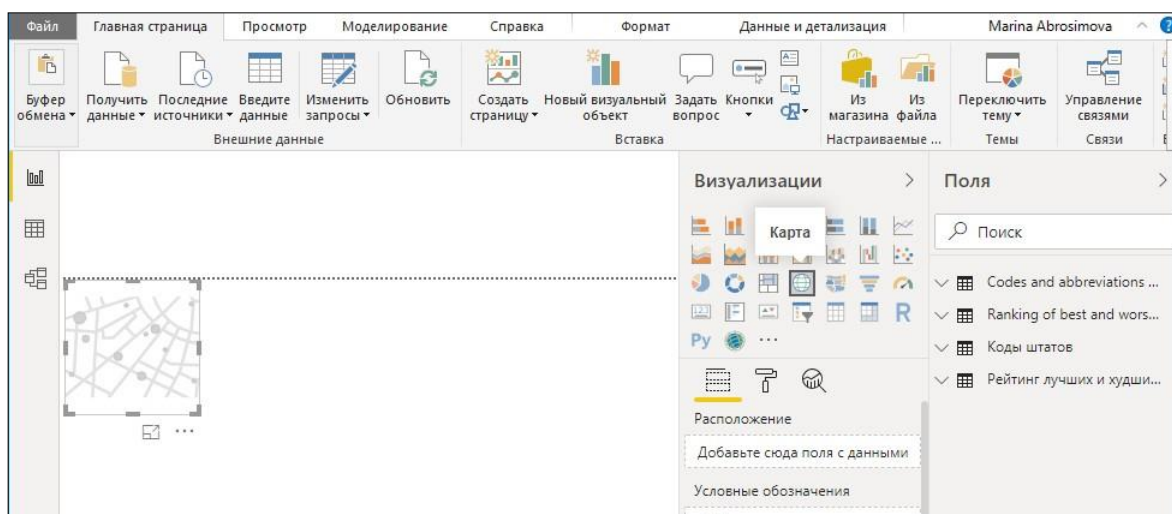
### Создание визуализации

Для создания отчетов нужны данные. В настоящей лабораторной работе Вы будете использовать данные, подготовленные на прошлой лабораторной работе.

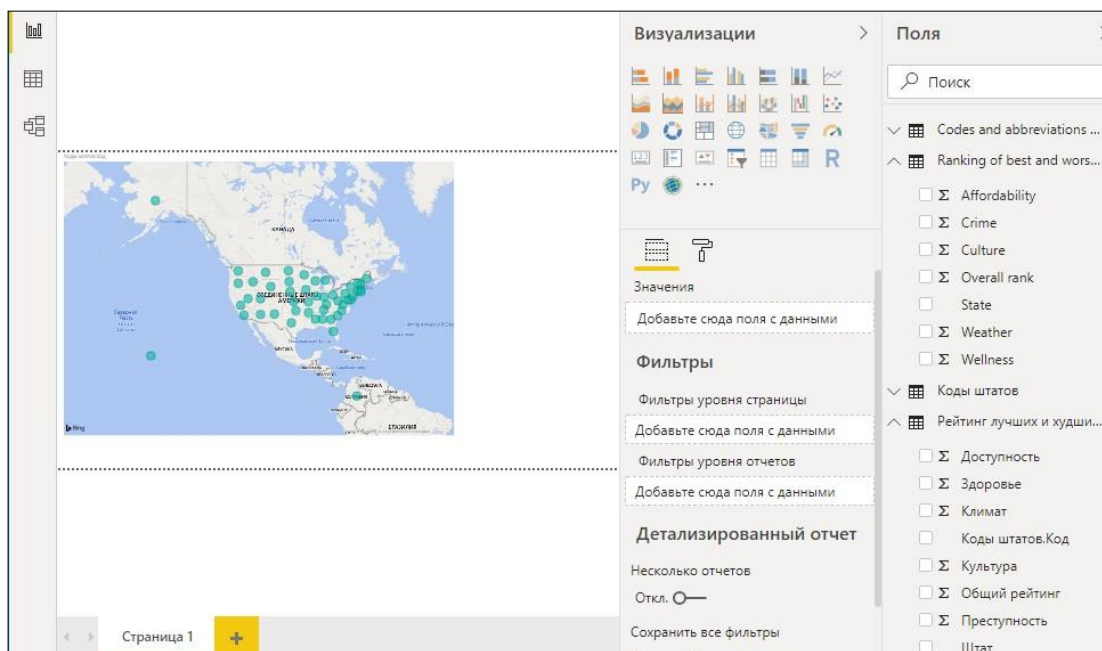
**Задание 1.** Откройте файл «Собственный проект». Перейдите в представление **Отчет**.

Для создания визуализации в отчете Вы должны:

1. Выбрать щелчком мыши инструмент Визуализации. Макет Вашей визуализации появится в области представления **Отчета**, а область **Визуализация** настроится на параметры данного инструмента. Например, выберите инструмент **Карта**:



2. Перетащить мышью в область **Визуализации** столбцы данных выбранных Вами таблиц из области **Поля**. Например, перетащите поле «Коды штатов» таблицы «Рейтинг лучших и худших штатов для выхода на пенсию» в свободное поле «Расположение».



**Задание 2.** Создайте визуализацию с использованием инструмента «Карта» на основе таблицы «Рейтинг лучших и худших штатов для выхода на пенсию».

## Редактирование визуализации

Для редактирования визуализации нужно:

- щелчком мыши выделить визуализацию в области представления

### Отчет;

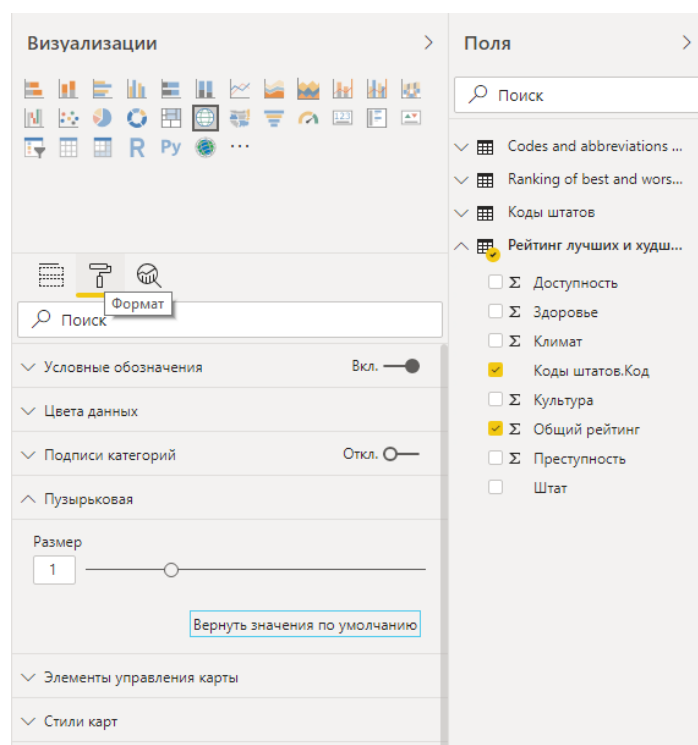
- изменять параметры визуализации, перетаскивая столбцы данных из области **Поля** в свободные поля области **Визуализация**.

Например,

1. перетащите столбец с данными «Общий рейтинг» из области **Поля** в свободное поле «Обозначение» области **Визуализация**;
2. перетащите столбец с данными «Общий рейтинг» из области **Поля** в свободное поле «условные обозначения» области **Визуализация**:



3. Уменьшите размеры визуальных элементов на карте. Для этого откройте в области Визуализация панель Формат и выберите параметр «Пузырьковая», далее используйте ползунок:



**Задание 3.** Отредактируйте визуализацию.

### Просмотр визуализации

Для того, чтобы посмотреть, как визуализация будет выглядеть, перейдите на вкладку **Просмотр**, далее **Просмотр страницы**, далее **Фактический размер**.

**Задание 4.** Просмотрите визуализацию.

Как правило, каждая страница отчета визуализирует определенные элементы данных. На каждой странице отчета может располагаться несколько визуализаций.

Для добавления на страницу новой визуализации нужно

- Снять выделение с уже имеющейся визуализации;
- Повторить шаги по созданию визуализации, описанные выше.

Например, на первой странице можно отобразить все штаты на основе общего рейтинга. В дополнение к уже имеющейся визуализации на основе инструмента **Карты** построим две новых визуализации на основе инструмента «График и гистограмма с накоплением»:

Первая визуализация имеет следующую структуру:

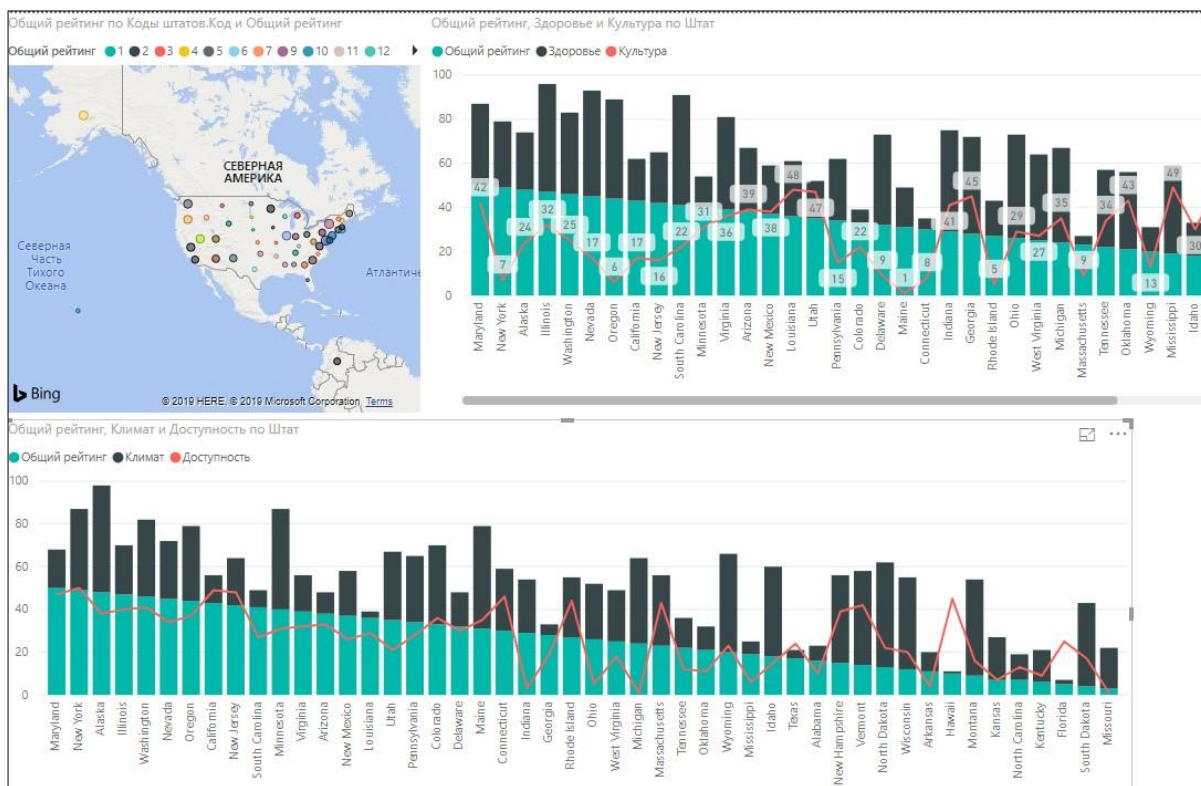
- **Общая ось** использует данные «Штат»;
- **Значения столбцов** использует данные «Общий рейтинг» и «Здоровье»;
- **Значения строк** использует данные «Культура»;
- Выведены метки данных.

Вторая визуализация имеет следующую структуру:

- **Общая ось** использует данные «Штат»;

- Значения столбцов использует данные «Общий рейтинг» и «Климат»;
- Значения строк использует данные «Доступность».

Изменяя размеры визуализаций, размещаем их на странице. У Вас должно получиться примерно так:



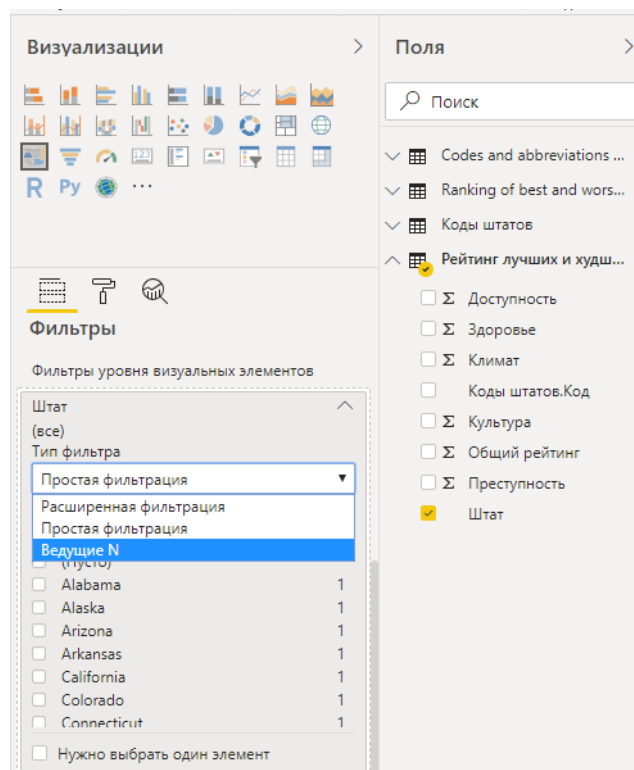
**Задание 5.** На первой странице отчета отобразите все штаты на основе общего рейтинга. Используйте инструмент «График и гистограмма с накоплением».

### Фильтрация в отчетах

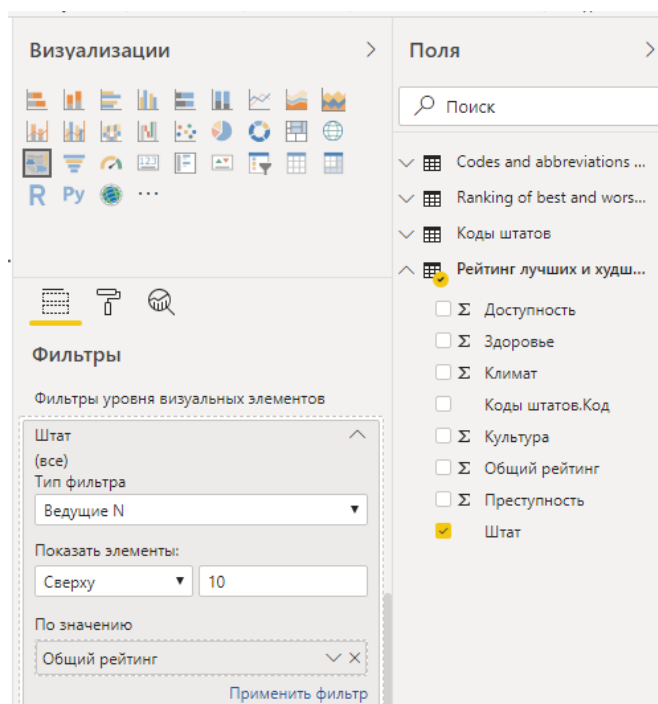
На второй странице отчета отобразим сведения о десяти лучших штатах исходя из общего рейтинга. Для этого построим те же визуализации, но в каждой из них нужно будет установить фильтр на вывод штатов, входящих в первую десятку по общему рейтингу.

Для того, чтобы фильтр установить Вы должны в области **Визуализация** в группе **Фильтры**

- Выбрать «Фильтры уровня визуальных элементов»;
- Открыть список способов фильтрации и выбрать «Ведущие N»;



- Далее «Показать элементы»: Сверху, первые 10;
- В свободное поле «По значению» перетащить столбец данных. В нашем примере – «Общий рейтинг»



- Щелкнуть «Применить фильтр».

Указанный алгоритм примените ко всем трем визуализациям. Для разнообразия при



построении первой визуализации выберите инструмент «Заполненная карта» и добавьте в третью визуализацию метки данных. В итоге у Вас должно получиться примерно так:



**Задание 6.** На второй странице отчета отобразите все штаты на основе общего рейтинга. Используйте инструмент «График и гистограмма с накоплением» и фильтрацию.

### Вычисления в отчетах

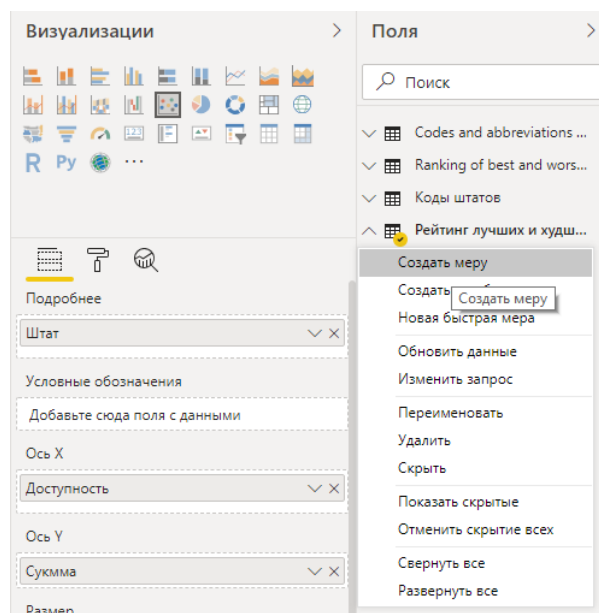
Часто при анализе данных необходимо сделать вычисления. Для этого используется DAX специально разработанный компанией Microsoft язык формул и функций для использования в Excel и Power BI Desktop. DAX помогает создавать новую информацию из данных, существующих в Вашей модели.

Допустим, Вас особенно интересуют три фактора по штатам: Культура, Здоровье, Климат. Для того, чтобы визуализировать сравнительные данные по штатам в этом разрезе создадим в таблице с исходными данными новый столбец «Комфортность среды», значения в котором будут рассчитаны как среднее значение полей Культура, Здоровье, Климат.

Для применения DAX нужно в представлении Отчет:

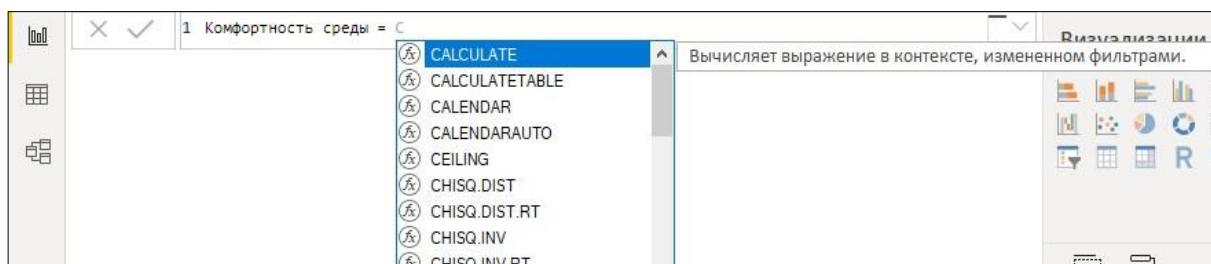
1. в списке **Поля** щелкните правой кнопкой мыши таблицу «Рейтинг лучших и худших штатов для выхода на пенсию», а затем выберите пункт **Создать меру**.



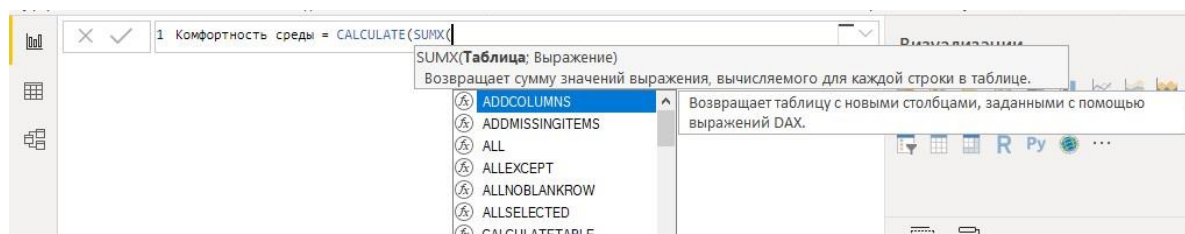


В результате в списке полей этой таблицы появился новый столбец Мера.

2. В строке формул замените «Мера», введя новое имя меры: «Комфортность среды»;
3. В этой формуле нужно использовать функцию CALCULATE. Поэтому после знака равенства введите первые несколько букв CAL и дважды щелкните функцию, которую нужно использовать:

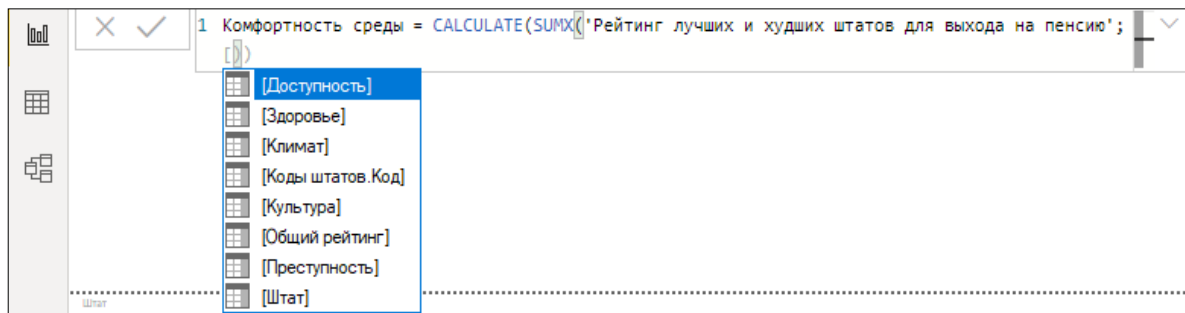


4. Пользуясь аналогичной подсказкой, введите имя функции, которая вычислит среднее значение оценок по трем полям: Культура, Здоровье. Климат. Такой функцией является функция SUMX



5. Далее введите имя таблицы, в которой создан столбец «Комфортность среды» путем выбора из раскрывающегося списка (смотрите в самый низ);
6. Далее вводим выражение:

$$([Культура] + [Здоровье] + [Климат])/3$$



Таким образом, значение данных в поле «Комфортность среды» вычисляется по формуле:

$$\text{Комфортность среды} = \text{CALCULATE}(\text{SUMX}(\text{'Рейтинг лучших и худших штатов для выхода на пенсию';} ([Здоровье] + [Климат] + [Культура]) / 3))$$

- Далее для ввода формулы в модель данных нажимаем кнопку с галочкой в строке формул или клавишу Enter.

**Задание 7.** Введите формулу для расчета значений в столбце «Комфортность среды». Перейдите в модель и убедитесь, что в таблице появилось поле «Комфортность среды».

Теперь новые данные можно использовать в отчете. На третьей странице отчета построим визуализации, отражающие 10 лучших штатов по доступности с использованием только что полученных новых данных. Она может выглядеть так:



Первая визуализация использует инструмент «Заполненная карта». Структура визуализации:

- Расположение** использует данные «Штат»;

- **Стиль карты:** вид с воздуха.
- **Фильтр** на вывод 10 лучших штатов по доступности.

Вторая визуализация использует инструмент «Линейчатая диаграмма с группировкой». Структура визуализации:

- **Ось** использует данные «Штат»;
- **Значения** использует данные «Преступность»;
- **Фильтр** на вывод 10 лучших штатов по доступности.

Третья визуализация использует инструмент «Диаграмма ленты». Структура визуализации:

- **Ось** использует данные «Штат»;
- **Значения** использует данные «Доступность», «Преступность»;
- Добавлены **метки значений**;
- **Фильтр** на вывод 10 лучших штатов по доступности.

Четвертая визуализация использует инструмент «Точечная диаграмма».

Структура визуализации:

- **Ось X** использует данные «Доступность»;
- **Ось Y** использует данные «Комфортность среды»;
- **Фильтр** на вывод 10 лучших штатов по доступности;
- Включены **подписи категорий**;
- **Размер фигур** – 77.

**Задание 8.** На третьей странице постройте четыре визуализации, отражающие 10 лучших штатов по доступности, используя новые данные по полю «Комфортность среды».

Ваш отчет состоит из трех **страниц**, каждая из которых визуализирует определенные элементы данных.

### **Задание для самостоятельного выполнения**

В файле «Собственный проект» создайте отчет из трех страниц аналогично вышеописанным отчетам.

**Задание 1.** Создать первую страницу отчета из двух визуализаций.

Первая визуализация использует инструмент «Линейчатая диаграмма с накоплением». Структура визуализации:

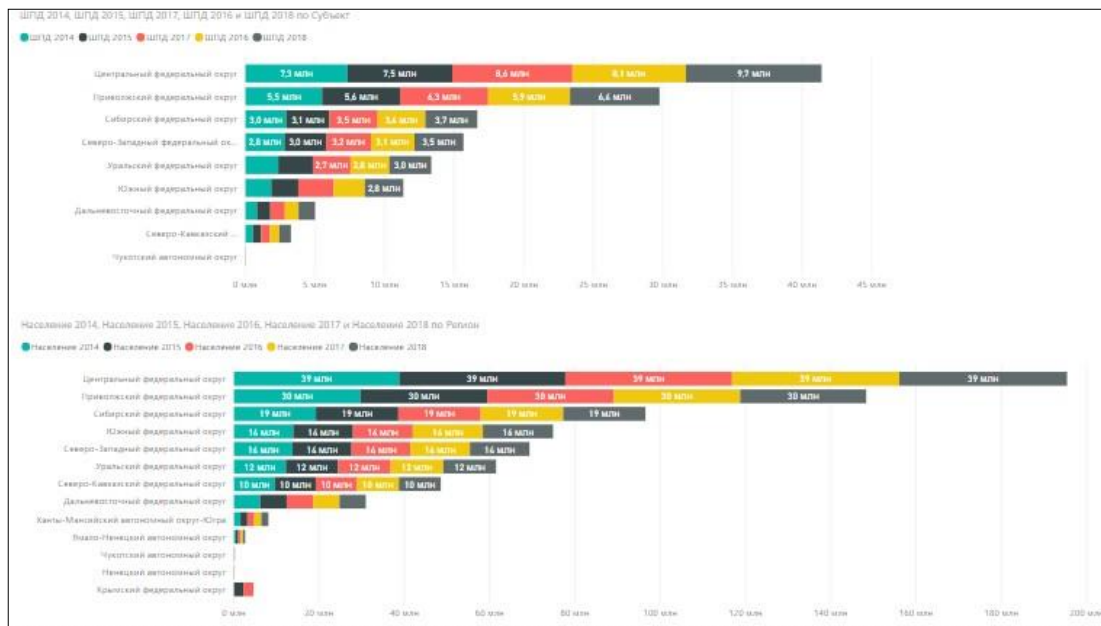
- **Ось** использует данные «Регион»;
- **Значения** использует данные «2014», «2015», «2016», «2017», «2018»;
- **Фильтр** уровня визуальных элементов: Расширенная фильтрация, «Регион» содержит слово «округ»;
- Выведены **метки** данных.

Вторая визуализация использует инструмент «Линейчатая диаграмма с накоплением». Структура визуализации:

- **Ось** использует данные «Регион»;
- **Значения** использует данные «Население 2014», «Население 2015», «Население 2016», «Население 2017», «Население 2018»;
- **Фильтр** уровня визуальных элементов: Расширенная фильтрация, «Регион»

содержит слово «округ»;

- Выведены метки данных.



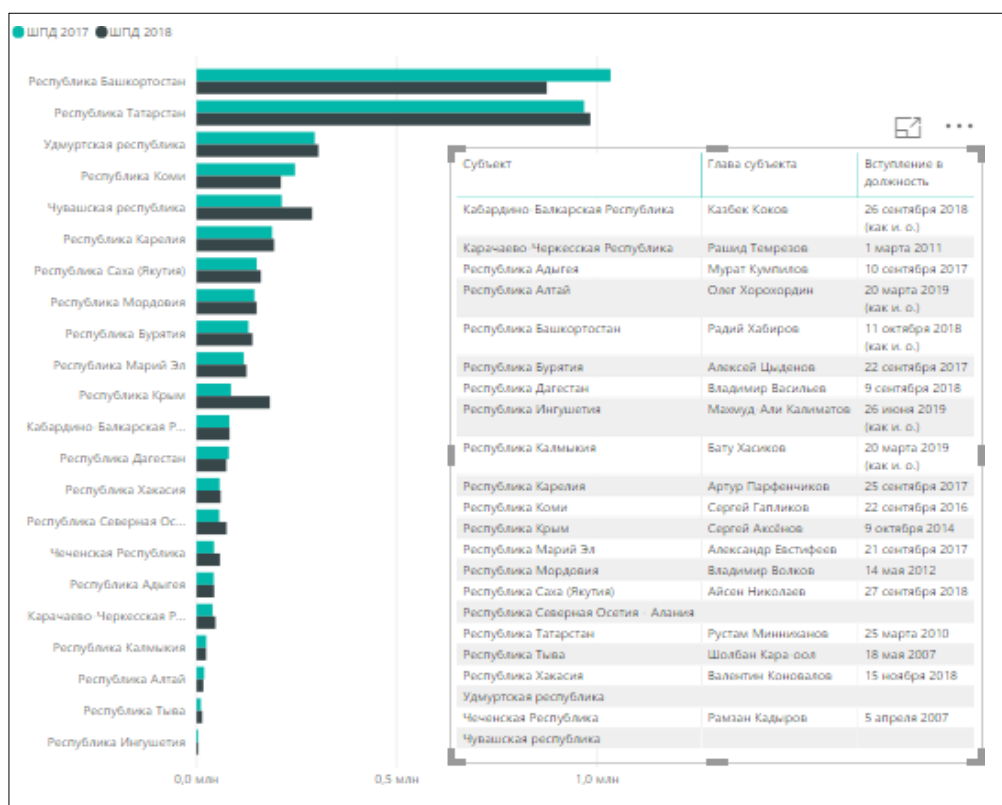
**Задание 2.** Создать вторую страницу отчета из двух визуализаций.

Первая визуализация использует инструмент «Линейчатая диаграмма с группировкой». Структура визуализации:

- **Ось** использует данные «Регион»;
- **Значения** использует данные «ШПД 2017», «ШПД 2018»;
- **Фильтр** уровня визуальных элементов: Расширенная фильтрация, «Регион» содержит слово «республика»;

Вторая визуализация использует инструмент «Таблица». Структура визуализации:

- **Значения** использует данные «Регион», «Глава субъекта», «Вступление в должность».
- **Фильтр** уровня визуальных элементов: Расширенная фильтрация, «Регион» содержит слово «республика».



**Задание 3.** Создать третью страницу отчета из трех визуализаций.

Первая визуализация использует инструмент «**Диаграмма ленты**».  
Структура визуализации:

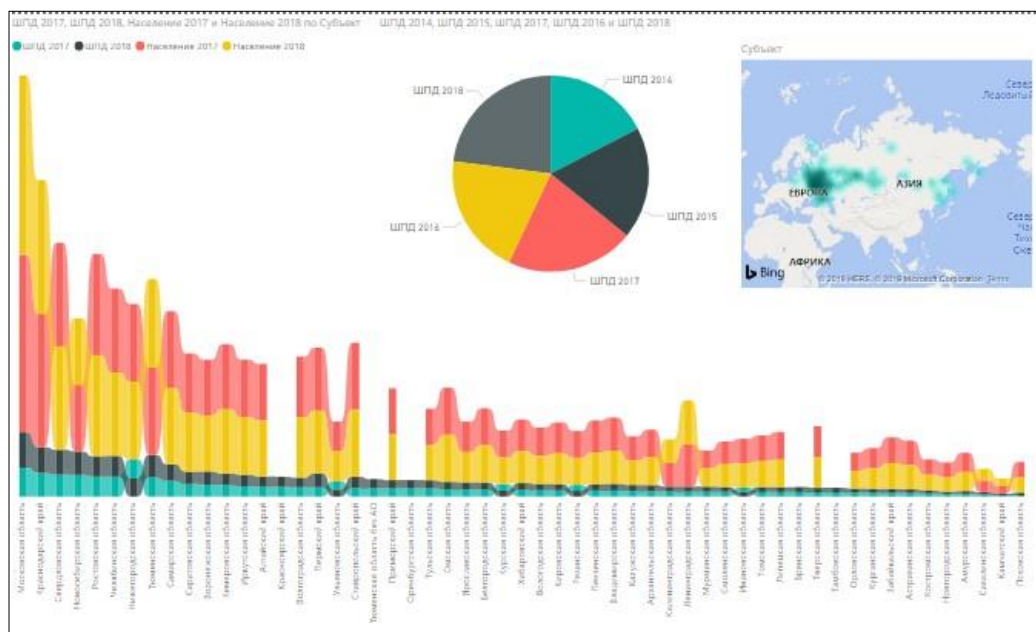
- **Ось** использует данные «Регион»;
- **Значения** использует данные «ШПД 2017», «ШПД 2018», «Население 2017»; «Население 2018»;
- **Фильтр** уровня визуальных элементов: Расширенная фильтрация, «Регион» содержит слова «область» или «край».

Вторая визуализация использует инструмент «**Круговая диаграмма**».  
Структура визуализации:

- **Значения** использует данные «ШПД 2014», «ШПД 2015», «ШПД 2016», «ШПД 2017», «ШПД 2018»;
- **Фильтр** уровня визуальных элементов: Расширенная фильтрация, «Регион» содержит слова «область» или «край».

Третья визуализация использует инструмент «**Карта**». Структура визуализации:

- **Расположение** использует данные «Регион»;
- **Фильтр** уровня визуальных элементов: Расширенная фильтрация, «Регион» содержит слова «область» или «край».



## Контрольные вопросы

1. В каком представлении доступна визуализация данных?
2. Из каких областей состоит представление **Отчет**?
3. Сколько визуализаций может размещаться на странице отчета?
4. Опишите инструментарий для редактирования визуализаций.
5. Что такое мера в Power BI Desktop?
6. Для чего используется DAX?
7. Опишите структуру визуализации.

## Контрольная работа

### Примерный перечень тем

1. Методы обработки и анализа данных
2. Методы и инструменты визуализации данных

### Примерные задания

1. Для чего используется визуализация данных?
  1. Изучение заданного набора данных.
  2. Обмен объективным представлением данных.
  3. Поддержка рекомендаций для различных заинтересованных сторон.
  4. Все вышеперечисленное.
2. Что из перечисленного ниже является слоем в архитектуре Matplotlib?
  1. Рисунок Слой
  2. FigureCanvas Слой
  3. Backend\_Bases Слой
  4. Серверный уровень
3. Какой слой позволяет полностью контролировать и тонко настраивать Matplotlib — контейнер верхнего уровня для всех элементов графика?
  1. Сценарий уровня
  2. Серверный уровень
  3. Уровень событий
  4. Слой художника
4. Какой из приведенных ниже кодов создаст график с накопленной областью данных в датафрейм pandas, area\_df, со значением прозрачности 0,75?
  1. 

```
import matplotlib.pyplot as plt
transparency = 1 - 0.75
area_df.plot(kind='area', alpha=transparency, stacked=False, figsize=(20, 10))
plt.title('Plot Title')
plt.ylabel('Метка вертикальной оси')
plt.xlabel('Метка горизонтальной оси')
plt.show()
```
  2. 

```
прозрачность = 0,75
ax = area_df.plot(kind='area', alpha=прозрачность, stacked=False, figsize=(20, 10))
ax.set_title('Название графика')
ax.set_ylabel('Метка вертикальной оси')
ax.set_xlabel('Метка горизонтальной оси')
```
  3. 

```
import matplotlib.pyplot as plt
прозрачность = 0.35
area_df.plot(kind='area', alpha=прозрачность, figsize=(20, 10))
plt.title('Заголовок графика')
plt.ylabel('Метка вертикальной оси')
plt.xlabel('Метка горизонтальной оси')
plt.show()
```
  4. 

```
import matplotlib.pyplot as plt
transparency = 0.75
area_df.plot(kind='area', alpha=transparency, figsize=(20, 10))
plt.title('Plot Title')
plt.ylabel('Vertical Axis Label')
```

```
plt.xlabel('Horizontal Axis Label')  
plt.show()
```

5. Каков способ статистического представления распределения данных по пяти основным измерениям?

1. Гистограмма
2. Коробка
3. Точечная диаграмма
4. Линейный график

6. Что является разновидностью точечной диаграммы, отображающей три измерения данных?

1. Ничего из вышеперечисленного
2. Пузырьковый сюжет
3. Точечная карта
4. Тепловая карта

7. Что из перечисленного НЕ ВЕРНО в отношении облака слов?

1. Облако слов может быть сгенерировано на Python с помощью пакета `word_cloud`, разработанного Андреасом Мюллером.

2. Облако слов — это изображение частоты различных слов в некоторых текстовых данных.

3. Ничего из вышеперечисленного.

4. Облако слов — это изображение частоты стоп-слов.

9. Что из перечисленного НЕ является стилем плиток карт Folium?

1. Тычиночная местность
2. Тычиночный тонер
3. Река Прибрежная
4. OpenStreetMap

10. Что из перечисленного НЕ соответствует стилю плитки Stamen Terrain для карт Folium?

1. Идеально подходит для гибридных приложений данных и исследования речных меандров и прибрежных зон

2. Демонстрирует расширенную маркировку и обобщение линий дорог с двусторонним движением

3. Особенности естественных цветов растительности

4. Особенности затенения холмов

11. Визуализации в виде графика не могут быть отображены каким из следующих способов:

1. Отображается в записной книжке Jupyter

2. Сохранено в HTML-файлы

3. Служит в качестве чистого python-приложения для сборки с использованием

Dash

4. Ничего из вышеперечисленного

12. Площадные графики по умолчанию не складываются

1. Правда

2. Неверно

13. Следующий код создаст гистограмму ряда панд, `series_data`, и выровняет края ячейки с горизонтальными галочками.

```
count, bin_edges = np.histogram(series_data)
```

```
series_data.plot(kind='hist', xticks = count, bin_edges)
```

1. Правда.

2. Неверно.

14. Что создаст горизонтальную гистограмму рассматриваемых данных?

1. `question.plot(type='bar', rot=90)`



2. `question.plot(kind='bar', orientation='horizontal')`
3. `question.plot(kind='barh')`
4. `question.plot(kind='bar')`
5. `question.plot(kind='bar', type='horizontal')`

### Визуализация данных с помощью Python

Задание 1. Был проведен опрос для оценки интереса аудитории к различным темам науки о данных, а именно: Большие данные (Spark/Hadoop) Анализ данных / Статистика / Визуализация данных / Глубокое обучение / Машинное обучение/ У участников было три варианта оценки по каждой теме: «Очень интересно», «В некоторой степени заинтересовано» и «Не заинтересовано». В опросе приняли участие 2 233 респондента. Результаты опроса были сохранены в CSV-файле, и к ним можно получить доступ по этой ссылке: [https://cocl.us/datascience\\_survey\\_data](https://cocl.us/datascience_survey_data). Если вы изучите CSV-файл, вы обнаружите, что в первом столбце представлены разделы для обработки и анализа данных, а в первой строке представлены варианты для каждого раздела. Используйте метод `pandas read_csv` для чтения CSV-файла в кадр данных `pandas`. Чтобы считывать данные во фрейм данных, одним из способов сделать это является использование параметра `index_col` для загрузки первого столбца в качестве индекса кадра данных. Вот документация по методу `read_csv` панд: [https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read\\_csv.html](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html) После того, как вам удалось создать вышеуказанный кадр данных, загрузите снимок экрана вашего кадра данных с фактическими числами.

Задание 2. Используйте слой художника `Matplotlib`, чтобы воспроизвести приведенную ниже гистограмму, чтобы визуализировать процент интереса респондентов к различным опрошенным темам науки о данных. Чтобы создать эту гистограмму, вы можете выполнить следующие шаги: Отсортируйте кадр данных в порядке убывания - Очень интересно. Переведите цифры в проценты от общего числа респондентов. Напомним, что в опросе приняли участие 2 233 респондента. Округлите проценты до 2 знаков после запятой. Что касается диаграммы: используйте размер рисунка (20, 8), ширину полосы 0,8, используйте цветной `#5cb85c` для очень интересующих столбцов, цветной `#5bc0de` для несколько интересующих столбцов и цветной `#d9534f` для неинтересующих столбцов, используйте размер шрифта 14 для подписей столбцов, процентов и легенды, используйте размер шрифта 16 для заголовка, и отобразите проценты над полосами, как показано выше, и удалите левую, верхнюю и правую границы. Как только вы будете удовлетворены своей диаграммой, загрузите скриншот вашего графика.

Задание 3. В этом вопросе вам необходимо создать карту `Choropleth` для визуализации преступности в Сан-Франциско. Прежде чем вы будете готовы приступить к построению карты, давайте реструктурируем данные так, чтобы они были в правильном формате для карты `Choropleth`. По сути, вам нужно будет создать кадр данных, в котором будет указан каждый район Сан-Франциско вместе с соответствующим общим количеством преступлений. Основываясь на наборе данных о преступности в Сан-Франциско, вы обнаружите, что Сан-Франциско состоит из 10 основных районов, а именно: Центральный, Южный, Бэйвью, Миссия, Парк, Ричмонд, Инглсайд, Таравал, Северный и Тендерлоин. Преобразуйте набор данных Сан-Франциско, который вы также можете найти здесь, [https://cocl.us/sanfran\\_crime\\_dataset](https://cocl.us/sanfran_crime_dataset), в кадр данных `pandas`, который представляет общее количество преступлений в каждом районе. Как только вы будете довольны своим фреймом данных, загрузите скриншот фрейма данных `pandas`.

Задание 4. Теперь вы должны быть готовы приступить к созданию карты `Choropleth`. Как вы узнали в лаборатории карт `Choropleth`, вам понадобится файл `GeoJSON`, который отмечает границы различных районов Сан-Франциско. Нужный файла доступен по этой ссылке: [https://cocl.us/sanfran\\_geojson](https://cocl.us/sanfran_geojson). Для карты убедитесь, что: она сосредоточена вокруг Сан-Франциско, вы используете уровень масштабирования 12, вы используете `fill_color = 'YlOrRd'`, вы определяете `fill_opacity = 0.7`, вы определяете `line_opacity = 0.2`, а также вы

определяете легенду и используете пороговую шкалу по умолчанию. Когда вы будете готовы отправить свою карту, загрузите скриншот вашей карты Choropleth.

## Домашняя работа

### Примерный перечень тем

#### Первичный анализ данных

Исследование данных сервиса “Яндекс.Музыка” — сравнение пользователей двух городов

Исследование надёжности заёмщиков — анализ банковских данных

Продажа квартир в Санкт-Петербурге — анализ рынка недвижимости

Определение выгодного тарифа для телека компании

Изучение закономерностей, определяющих успешность игр

Изучение базы данных интернет-сервиса для чтения книг

Анализ убытков приложения ProcrastinatePRO+

Проверка гипотез по увеличению выручки в интернет-магазине

Рынок заведений общественного питания Москвы

Анализ пользовательского поведения в мобильном приложении

Создание дашборда по пользовательским событиям для агрегатора новостей

Прогнозирование вероятности оттока пользователей для фитнес-центров

### Примерные задания

1. На реальных данных Яндекс.Музыки с помощью библиотеки Pandas и её возможностей проверить данные и сравнить поведение и предпочтения пользователей двух столиц — Москвы и Санкт-Петербурга.

2. На основе статистики о платёжеспособности клиентов исследовать, влияет ли семейное положение и количество детей клиента на факт возврата кредита в срок.

3. Используя данные сервиса Яндекс.Недвижимость, определить рыночную стоимость объектов недвижимости и типичные параметры квартир.

4. На основе данных клиентов оператора сотовой связи проанализировать поведение клиентов и поиск оптимального тарифа.

5. Используя исторические данные о продажах компьютерных игр, оценки пользователей и экспертов, жанры и платформы, выявить закономерности, определяющие успешность игры.

6. Проект по анализу базы данных сервиса для чтения книг по подписке.

7. Задача для маркетингового аналитика развлекательного приложения Procrastinate Pro+. Несмотря на огромные вложения в рекламу, последние несколько месяцев компания терпит убытки. Задача — разобраться в причинах и помочь компании выйти в плюс.

8. Используя данные интернет-магазина приоритезировать гипотезы, произвести оценку результатов А/В-тестирования различными методами.

9. Исследование рынка общественного питания на основе открытых данных, подготовка презентации для инвесторов.

10. На основе данных использования мобильного приложения для продажи продуктов питания проанализировать воронку продаж, а также оценить результаты А/А/В-тестирования.

11. Используя данные Яндекс.Дзена построить дашборд с метриками взаимодействия пользователей с карточками статей.

12. На основе данных о посетителях сети фитнес-центров спрогнозировать вероятность оттока для каждого клиента в следующем месяце, сформировать с помощью кластеризации портреты пользователей.

13. Первичный анализ данных

1. Подберите набор данных и согласуйте свой выбор с преподавателем. Студент может предложить синтезированный набор данных.

2. Проведите первичный анализ данных. В результате анализа данных студент должен предоставить следующую информацию о наборе данных:

2.1. Описание набора данных, пояснения, позволяющие лучше понять природу данных. Назначение набора данных и возможные модели, которые можно построить на основе данного набора данных (практические задачи, решаемые с использованием данного обучающего набора данных). Описание каждого признака и его тип.

2.2. Форма набора данных: количество элементов набора, количество признаков, количество пропущенных значений, среднее значение отдельных признаков, максимальные и минимальные значения отдельных признаков и прочие показатели. Предположения, которые можно сделать, проведя первичный анализ.

2.3. Графические представления, позволяющие судить о неоднородности исследуемого набора данных. Построение графиков желательно произвести по нескольким проекциям.

## Зачет

### Примерные вопросы к зачету

1. Какие инструментальные средства используются для организации рабочего места специалиста Data Science?
2. Какие библиотеки Python используются для работы в области машинного обучения? Дайте краткую характеристику каждой библиотеке.
3. Почему при реализации систем машинного обучения широкое распространение получили библиотеки Python?
4. Перечислите функции Python для визуализации данных.
5. Какая библиотека python предназначена для управления наборами данных: numpy, pandas, sklearn, opencv, matplotlib?
6. Какая стратегия является нежелательной при обработке пропусков в данных?
  - а) замена пропущенных значений в столбце медианным значением по данному столбцу;
  - б) удаление строк, содержащих пропуски в данных;
  - в) замена пропущенных значений в столбце средним арифметическим значением по данному столбцу;
  - г) замена пропущенных значений в столбце наиболее часто встречающимся значением по данному столбцу;
7. Обоснуйте ответ на следующую проблему предварительной обработки данных: имеется независимая категориальная переменная  $y$ , которая представляет собой категориальный признак, определенный на домене {C#, Java, Python, R}. Нужно ли применять к данному целевому признаку OneHotEncoder?
8. Поясните принцип разбиения набора данных на обучающую и тестовую выборку. Какое соотношение «тестовая:обучающая» наиболее оптимально: 20:80, 50:50, 25:75, 5:95, 40:30?
9. Какой код лучше использовать при загрузке данных из csv-файла?
  - а) `dataset = read_csv("data.csv")`
  - б) `dataset = import("data.csv")`
  - в) `dataset = read.csv("data.csv")`
  - г) `dataset = import.csv("data.csv")`
  - д) `dataset = read_xls("data.csv")`
10. Для чего предназначен Power BI Desktop и для чего Power BI Service?
11. Опишите кратко функциональные возможности Power BI Desktop.
12. Для чего используется представление **Данные** в Power BI Desktop?
13. Что понимается под настройкой данных, и какие инструменты Power BI Desktop позволяют ее выполнить?
14. В чем состоят преимущества использования **Редактор запросов** при настройке данных по сравнению с инструментами вкладки **Главная страница**?
15. Какую роль выполняет раздел **Примененные шаги** окна **Редактора запросов**?
16. В чем отличие слияния от дополнения при объединении данных, полученных из разных источников?

## Практическая часть

### Выполнение проектов на темы:

#### 1. Определение неэффективных операторов

Необходимо помочь провайдеру виртуальной телефонии «Нупозвони» найти самых неэффективных операторов.

#### 2. Оценка результатов A/B-теста

Проверка данных на соответствие данным требованиям технического задания. Проведение исследовательского анализа данных, оценка результатов A/B-тестирования.