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

_____ С.Т. Князев

«_7_» _сентября_ 2023 г.

Анализ данных и искусственный интеллект

Учебно-методические материалы по направлению подготовки
09.03.03 Прикладная информатика
Образовательная программа «Прикладной искусственный интеллект»

Екатеринбург

2023

РАЗРАБОТЧИКИ УЧЕБНО-МЕТОДИЧЕСКИХ МАТЕРИАЛОВ

Доцент, канд.техн.наук _____ Денисов Дмитрий Вадимович

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

ВВЕДЕНИЕ

Во введении мы рассмотрим процесс установки необходимого программного обеспечения и напишем простейшую программу на языке Python, которая выводит сообщение Hello World. В качестве основного программного обеспечения, который будет использован для работы с инструментами анализа данных мы будем использовать Anaconda. Это дистрибутив языков программирования Python и R, включающий набор популярных свободных библиотек, объединенных проблематиками науки о данных и машинного обучения. Скачать дистрибутив можно с сайта anaconda.com.

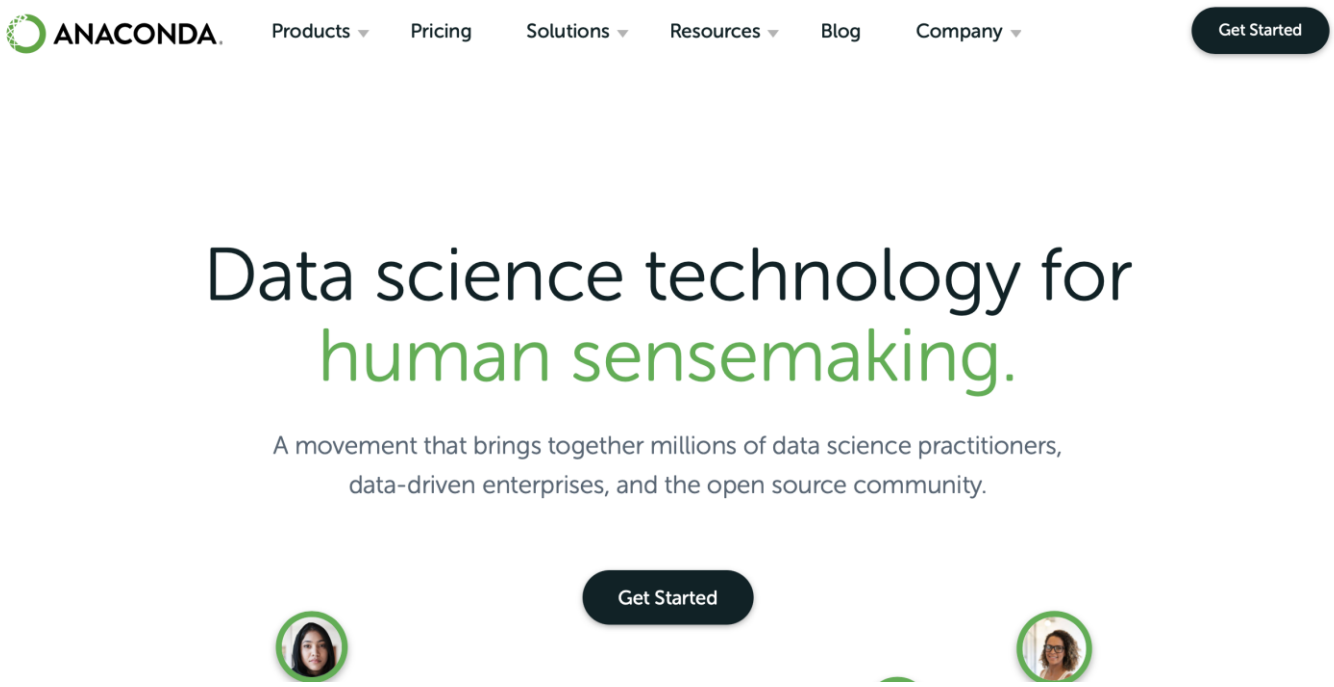


Рисунок В.1

Для скачивания переходим в верхнем меню сайта во вкладку Products – Individual Edition - Download:

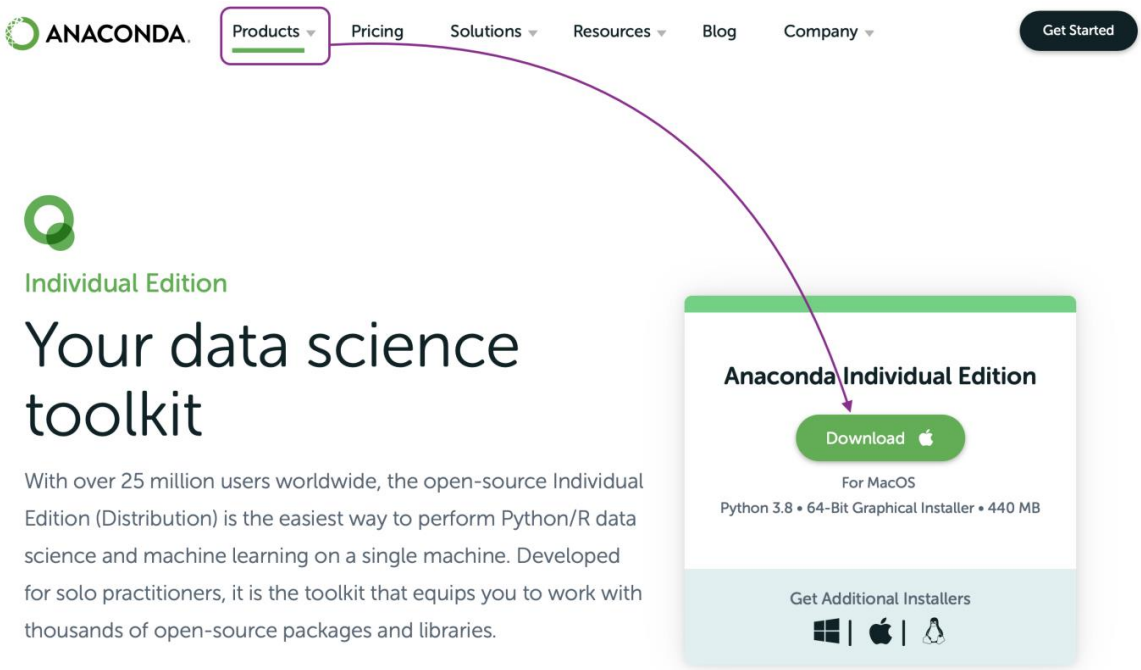


Рисунок В.2

Запускаем скачанный дистрибутив:

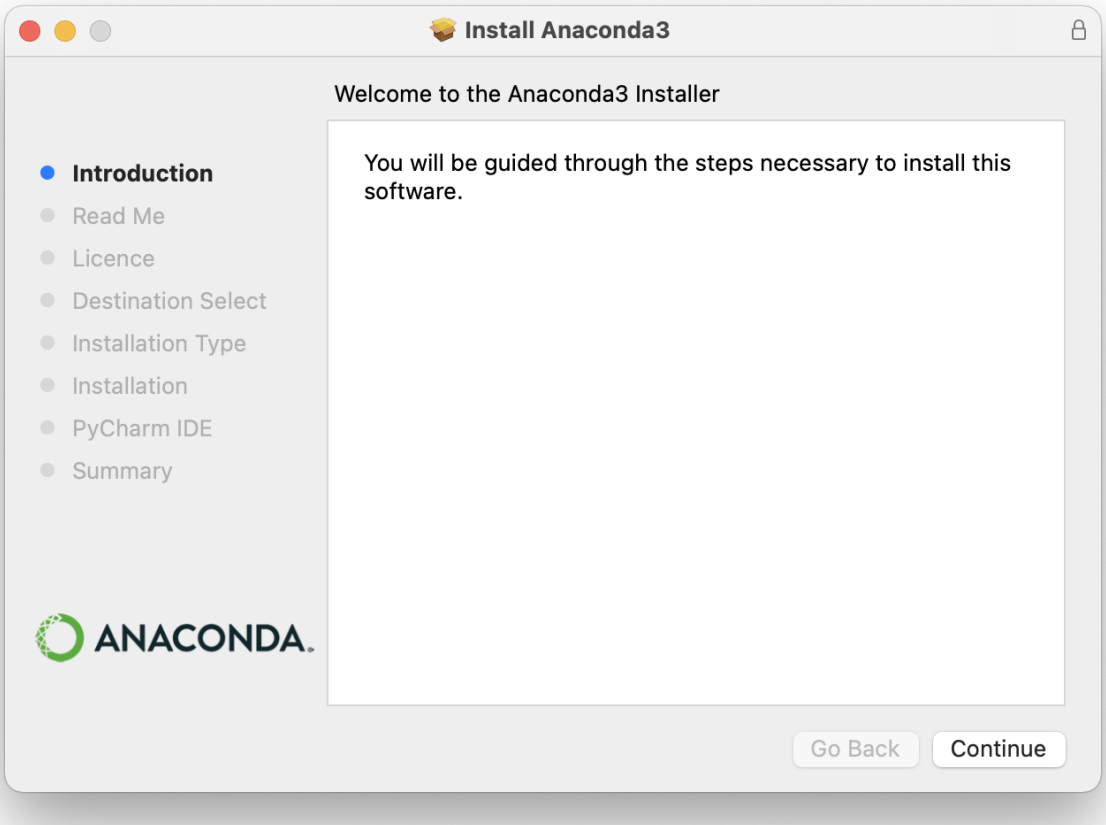


Рисунок В.3

Необходимо принять лицензионное соглашение и выбрать место для установки:

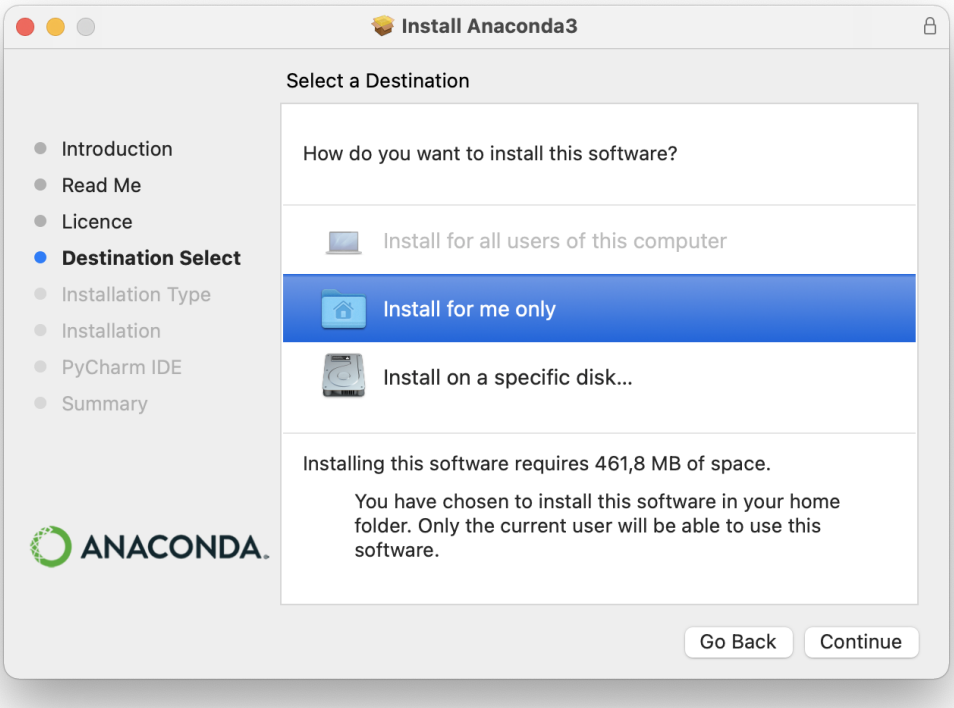


Рисунок B.4

Нажмите Continue, после чего начнется процесс установки.

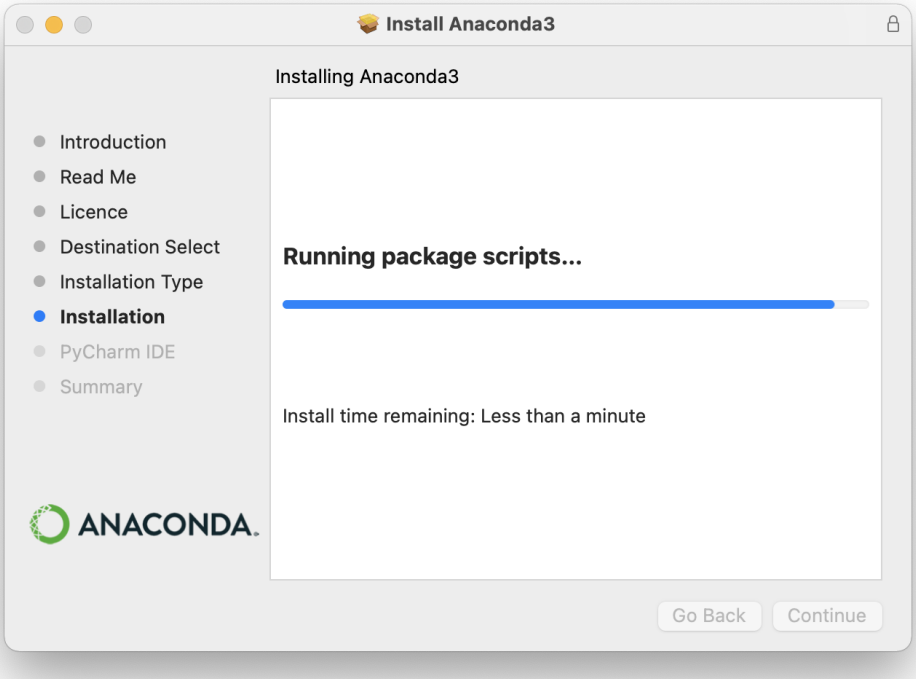


Рисунок В.5

По завершению установки основного дистрибутива, будет предложено установить среду разработки PyCharm IDE. Это программное обеспечение можно установить по желанию (его мы не будем использовать в рамках выполнения практических работ), и завершить процесс установки. После этого запустите Anaconda-Navigator, навигатор содержит достаточно большое количество полезных инструментов для работы с данными.

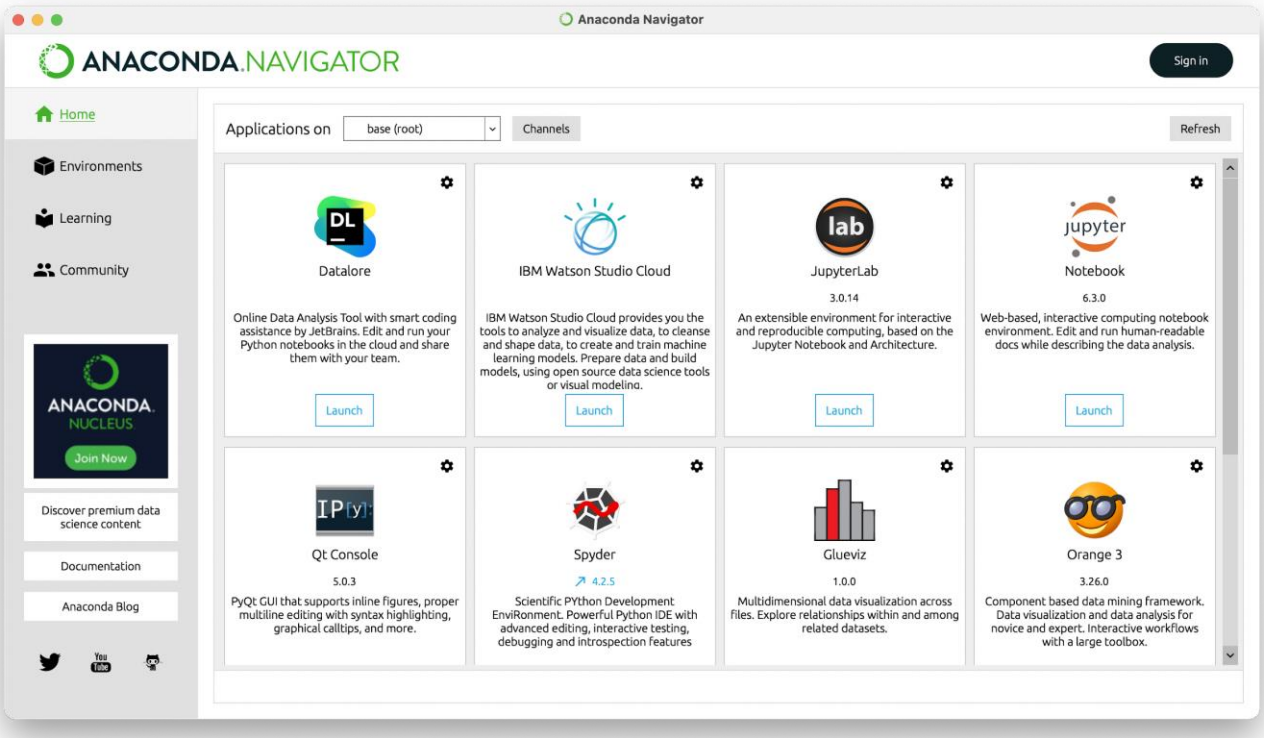


Рисунок B.6

Запустите инструмент Jupyter Notebook, нажав кнопку Launch. После этого откроется терминал и окно браузера по адресу http://localhost:8888/tree/ с деревом файлов на вашем локальном компьютере. В окне браузера создайте новую папку для работы с файлами проекта, например на рисунке ниже показана созданная папка UrFU/1-Anaconda, с созданным внутри файлом HelloWorld.jpynb. Для создания файлов и папок используйте элементы управления (New, Refresh…) в верхней части окна.

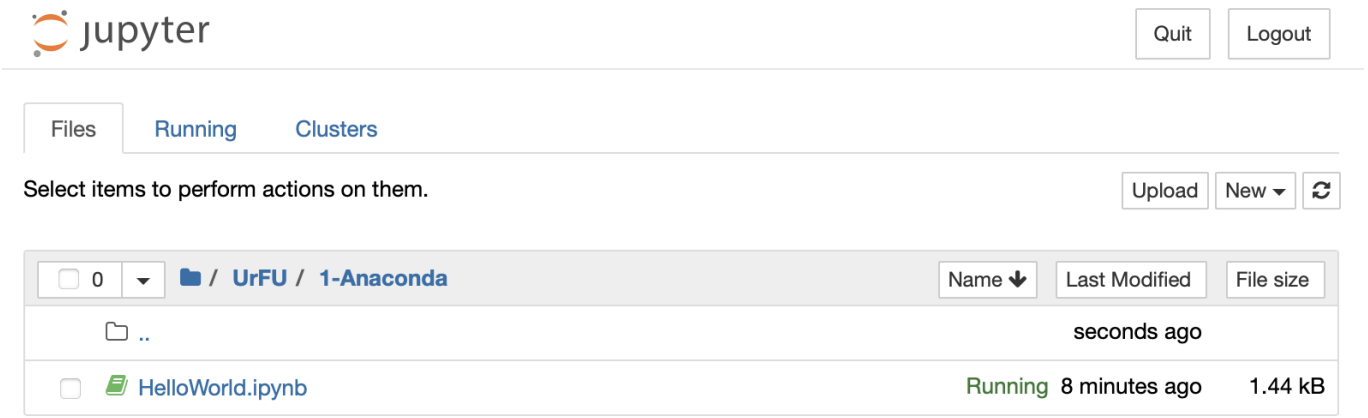


Рисунок В.7

Откройте созданный файл с именем HelloWorld, напишите в него команду print() с характерным содержанием и запустите выполнение, нажав Run. Результатом выполнения станет вывод сообщения Hello World.

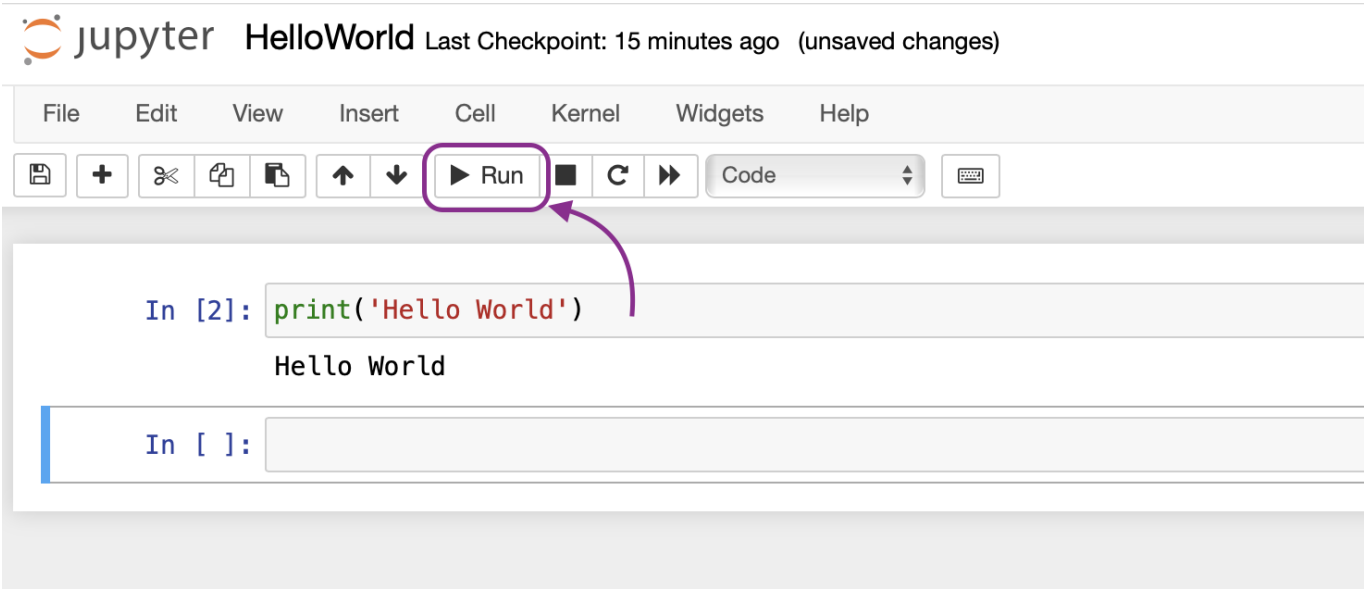


Рисунок В.8

Таким образом, подготовку необходимого программного обеспечения для выполнения лабораторных работ можно считать завершенной.

1. Лабораторная работа. Конструкции языка Python на примере реализации линейной регрессии

Цель работы: ознакомиться с основными операторами Python и освоить этапы реализации линейной регрессии.

Задание к работе.

1. В разделе «ход работы» пошагово выполнить каждый пункт с описанием и примера реализации задач по теме лабораторной работы.
2. После пункта выполнить «индивидуальное задание» при его наличии.
3. Ответьте на контрольные вопросы:
    1) должна ли величина loss стремиться к нулю при изменении исходных данных?
    2) какова роль параметра Lr (в качестве эксперимента можете попробовать изменить его значение)?

Ход работы.

1. Произвести подготовку данных для работы с алгоритмом линейной регрессии. 10 видов данных были установлены случайным образом, и данные находились в линейной зависимости. Данные преобразуются в формат массива, чтобы их можно было вычислить напрямую при использовании умножения и сложения.

In [ ]:

```
#Import the required modules, numpy for calculation, and Matplotlib for drawing
import numpy as np
import matplotlib.pyplot as plt
#This code is for jupyter Notebook only
%matplotlib inline

# define data, and change list to array
x = [3,21,22,34,54,34,55,67,89,99]
x = np.array(x)
y = [2,22,24,65,79,82,55,130,150,199]
y = np.array(y)

#Show the effect of a scatter plot
plt.scatter(x,y)
```

2. Определите связанные функции. Функция модели: определяет модель линейной регрессии wx+b. Функция потерь: функция потерь

среднеквадратичной ошибки. Функция оптимизации: метод градиентного спуска для нахождения частных производных w и b.

In [ ]:

```
#The basic linear regression model is wx+ b, and since this is a two-dimensional
space, the model is ax+ b

def model(a, b, x):
    return a*x + b

#Tahe most commonly used loss function of linear regression model is the loss
function of mean variance difference
def loss_function(a, b, x, y):
    num = len(x)
    prediction=model(a,b,x)
    return (0.5/num) * (np.square(prediction-y)).sum()

#The optimization function mainly USES partial derivatives to update two parameters a
and b
def optimize(a,b,x,y):
    num = len(x)
    prediction = model(a,b,x)
    #Update the values of A and B by finding the partial derivatives of the loss
function on a and b
    da = (1.0/num) * ((prediction -y)*x).sum()
    db = (1.0/num) * ((prediction -y).sum())
    a = a - Lr*da
    b = b - Lr*db
    return a, b

#iterated function, return a and b
def iterate(a,b,x,y,times):
    for i in range(times):
        a,b = optimize(a,b,x,y)
    return a,b
```

3. Начать итерацию

Шаг 1 Инициализация и модель итеративной оптимизации

In [ ]:

```
#Initialize parameters and display
a = np.random.rand(1)
print(a)
b = np.random.rand(1)
print(b)
Lr = 0.000001

#For the first iteration, the parameter values, losses, and visualization after the
iteration are displayed
a,b = iterate(a,b,x,y,1)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

Шаг 2 На второй итерации отображаются значения параметров, значения потерь и эффекты визуализации после итерации

In [ ]:

```
a,b = iterate(a,b,x,y,2)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

Шаг 3 Третья итерация показывает значения параметров, значения потерь и визуализацию после итерации

In [ ]:

```
a,b = iterate(a,b,x,y,3)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

Шаг 4 На четвертой итерации отображаются значения параметров, значения потерь и эффекты визуализации

In [ ]:

```
a,b = iterate(a,b,x,y,4)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

Шаг 5 Пятая итерация показывает значение параметра, значение потерь и эффект визуализации после итерации

In [ ]:

```
a,b = iterate(a,b,x,y,5)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

Шаг 6 10000-я итерация, показывающая значения параметров, потери и визуализацию после итерации

In [ ]:

```
a,b = iterate(a,b,x,y,10000)
prediction=model(a,b,x)
loss = loss_function(a, b, x, y)
```

```
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

Требования к отчету по лабораторной работе

Отчет по лабораторной работе должен быть в виде документа ipynb (документ Jupyter Notebook), ссылки на предоставленный доступ к документу google colab или другие аналогичные форматы. Отчет в формате pdf, word со структурой:

1. Название работы.
2. Краткое пояснение к содержанию.
3. Имя, фамилия, группа студента, выполнившего работу.
4. Задание на лабораторную работу.
5. Краткое описание теоретических сведений, соответствующих работе.
6. Код реализации выполнения задания.
7. Визуализация результатов выполнения (если применимо).
8. Выводы.
9. Приложение.

2. Лабораторная работа. Сбор, обработка и визуализация тестового набора данных

Цель работы: изучить работу с библиотекой matplotlib для визуализации данных, получить навыки подготовки данных для их дальнейшего анализа и представления.

Задание к работе.

1. В разделе «ход работы» пошагово выполнить каждый пункт с описанием и примера реализации задач по теме лабораторной работы.
2. После пункта выполнить «индивидуальное задание» при его наличии.

Ход работы.

1. Для графического анализа данных мы будем использовать популярную библиотеку matplotlib. Протестируйте работу приведенного ниже кода, который строит простейший график в прямоугольной системе координат:

```
In [24]: years = [1950, 1960, 1970, 1980, 1990, 2000, 2010] #год
         gdp = [300.2, 543.3, 1075.9, 2862.5, 5979.6, 10289.7, 14958.3] #ВВП
         plt.plot(years, gdp, color = 'green', marker = 'o', linestyle = 'solid')
         plt.title("Номинальный ВВП")
         plt.ylabel("Млрд $")
         plt.xlabel("г.")
Out[24]: Text(0.5, 0, 'г.')
```

Индивидуальное задание: построить график в прямоугольной системе координат для любого другого набора данных на свой выбор. Измените подписи осей, цвет и тип линии графика.

2. Основы работы с текстовыми файлами (чтение файлов). Python достаточно просто позволяет считывать и записывать файлы непосредственно из кода. В этом пункте мы рассмотрим работу с данными в популярном формате .csv (с разделением запятыми), в качестве тестовых данных будем использовать данные о погоде. Перейдите на сайт https://www.ncdc.noaa.gov/cdo-web/ и откройте вкладку Brouse Dataset:

3. Сохраните файл из раздела Climate Data Online - Daily Summaries – Documentation & Samples (Data Sample). По умолчанию файл имеет имя GHCND_sample_csv.csv, сохраните его на свой компьютер и переместите в корневую директорию, из которой осуществляется работа с Jupiter Notebook (т. е. скачанный Data Sample и файл Lab2.ipynb должны находиться в одной папке).

4. Первая строка файла .csv содержит серию заголовков данных, заголовки описывают информацию, хранящуюся в файле. Напишите код, который позволяет считать данные заголовков файла:

```
import csv
filename = 'GHCND_sample_csv.csv'
with open (filename) as f:
        reader = csv.reader(f)
        header_row = next(reader)
        print(header_row)
```

```
['STATION', 'STATION_NAME', 'ELEVATION', 'LATITUDE', 'LONGITUDE', 'DATE', 'TMAX', 'TMIN', 'PRCP']
```

Индивидуальное задание: скачать с сайта National Centers for Environmental Information (NCEI) 3 разных набора данных и считать из них данные заголовка файла.

Требования к отчету по лабораторной работе.

Отчет по лабораторной работе должен быть в виде документа ipynb (документ Jupyter Notebook), ссылки на предоставленный доступ к документу google colab или другие аналогичные форматы. Отчет в формате pdf, word со структурой:

1. Название работы.
2. Краткое пояснение к содержанию.
3. Имя, фамилия, группа студента, выполнившего работу.
4. Задание на лабораторную работу.
5. Краткое описание теоретических сведений, соответствующих работе.
6. Код реализации выполнения задания.
7. Визуализация результатов выполнения (если применимо).
8. Выводы.
9. Приложение.

3. Лабораторная работа. Разработка системы машинного обучения

Цель работы: изучить работу с библиотекой машинной обучения Scikit-learn.

Задание к работе.

1. Перед выполнением лабораторной работы, просмотрите программный код, который вы использовали в первой лабораторной работе.
2. В разделе «ход работы» пошагово выполнить каждый пункт с описанием и примера реализации задач по теме лабораторной работы.
3. После пункта выполнить «индивидуальное задание» при его наличии.

Ход работы.

1. Scikit-learn — это библиотека машинного обучения с открытым исходным кодом, которая поддерживает контролируемое и неконтролируемое обучение. Он также предоставляет различные инструменты для подбора подходящей модели обучения после предварительной обработки данных.
2. Подбор и прогнозирование: основы оценки. Scikit-learn предоставляет десятки встроенных алгоритмов и моделей машинного обучения, называемых оценками (Estimators). Каждый «оценщик» может быть приспособлен к некоторым данным, используя свой метод подбора.
3. Ниже приведен набор методов, предназначенных для регрессии, в которых целевое значение, как ожидается, будет линейной комбинацией функций. В математической записи, если $\hat{y}$ прогнозируемое значение:

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \ldots + w_p x_p$$

Обозначим вектор $w_0=(w_1,\ldots,w_p)$ как coef_ и w_0 как intercept_. LinearRegression соответствует линейной модели с коэффициентами для минимизации остаточной суммы квадратов между наблюдаемыми целями в наборе данных и целями, предсказанными линейным приближением. Математически это решает задачу вида:

$$min\|Xw - y\|_2^2$$

4. LinearRegression примет массивы методов массивов X, y и сохранит коэффициенты линейной модели в coef_:

In [ ]:

```
from sklearn import linear_model
X = [[0,4], [1,3], [2,2]]
y = [3, 5, 8]
reg = linear_model.LinearRegression()
reg.fit(X, y)
print(reg.coef_, reg.intercept_)
```

5. Трансформаторы и препроцессоры. В scikit-learn препроцессоры и преобразователи следуют тому же API, что и объекты оценки (фактически все они наследуются от одного и того же класса BaseEstimator). У объектов-преобразователей нет метода прогнозирования, а скорее метод, который выводит только что преобразованную матрицу выборки X:

```
from sklearn.preprocessing import StandardScaler
X = [[0, 0], [1, 1], [2, 2]]
StandardScaler().fit(X).transform(X)
```

```
from sklearn.preprocessing import MinMaxScaler
data = MinMaxScaler(feature_range=(0., 1.)).fit_transform(X)
data
```

6. Конвейеры (Pipelines): объединение препроцессоров (pre-processors) и оценщиков (estimators). Преобразователи и оценщики (предикторы) могут быть объединены в один объединяющий объект: конвейер. Конвейер предлагает тот же API, что и обычный оценщик: его можно подогнать и использовать для прогнозирования с помощью соответствия и прогнозирования. Как мы увидим позже, использование конвейера также предотвратит утечку данных, то есть раскрытие некоторых данных тестирования в ваших данных обучения.

```
from sklearn.pipeline import make_pipeline
# create a pipeline object
pipe = make_pipeline(
StandardScaler(),
linear_model.LinearRegression()
)
# fit the whole pipeline
pipe.fit(X, y)
# coefficient
pipe[1].coef_
```

7. Оценка модели. Подгонка модели к некоторым данным не означает, что она будет хорошо предсказывать невидимые данные. Это требует непосредственной оценки.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
pipe.fit(x_train, y_train)
mean_squared_error(y_test, pipe.predict(x_test))
```

Требования к отчету по лабораторной работе.

Отчет по лабораторной работе должен быть в виде документа ipynb (документ Jupyter Notebook), ссылки на предоставленный доступ к документу google colab или другие аналогичные форматы. Отчет в формате pdf, word со структурой:

1. Название работы.
2. Краткое пояснение к содержанию.
3. Имя, фамилия, группа студента, выполнившего работу.
4. Задание на лабораторную работу.
5. Краткое описание теоретических сведений, соответствующих работе.
6. Код реализации выполнения задания.
7. Визуализация результатов выполнения (если применимо).
8. Выводы.
9. Приложение.

4. Лабораторная работа. Разработка интеллектуальной системы для обработки естественного языка

Цель работы: изучить работу с платформой для создания NLP-программ на Python - NLTK (Natural Language Toolkit).

Задание к работе.

1. Перед выполнением лабораторной работы, просмотрите программный код, который вы использовали в первой лабораторной работе.
2. В разделе «ход работы» пошагово выполнить каждый пункт с описанием и примера реализации задач по теме лабораторной работы.
3. После пункта выполнить «индивидуальное задание» при его наличии.

Ход работы.

Natural Language Processing (далее – NLP) – обработка естественного языка – подраздел информатики и AI, посвященный тому, как компьютеры анализируют естественные (человеческие) языки. NLP позволяет применять алгоритмы машинного обучения для текста и речи. Обработка естественного языка имеет обширную область с участием самых разных вычислительных технологий. Одним из приемов визуализации слов и их частотностей – это построение облака слов (визуализация частотности слов в виде взвешенного списка). В таких списках слова располагаются в соответствии с их размерами, пропорциональными их частотностям. В лабораторной работе будет произведена подготовка данных с помощью библиотеки NLTK и построено облако частотности встречающихся слов.

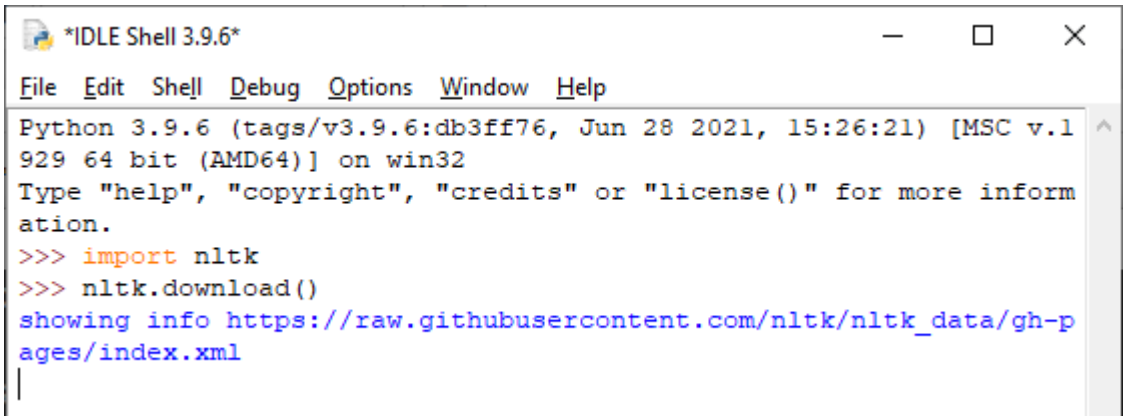
1. Для работы с платформой для обработки естественного языка, следует скачать и установить необходимые библиотеки и данные. Руководство по установке NLTK находится по адресу (https://www.nltk.org/install.html), кроме этого понадобятся Data-файлы, которые можно найти на официальном сайте (https://www.nltk.org/data.html). NLTK устанавливается автоматически вместе с Anaconda, однако могут возникнуть проблемы с установкой файлов Data, поэтому в пунктах ниже будут даны инструкции по их установке.

2. Убедиться в установке всех необходимых пакетом можно с помощью диспетчера пакетов conda. Просто запустите команды команды с CMD.exe Prompt (внутри Anaconda):

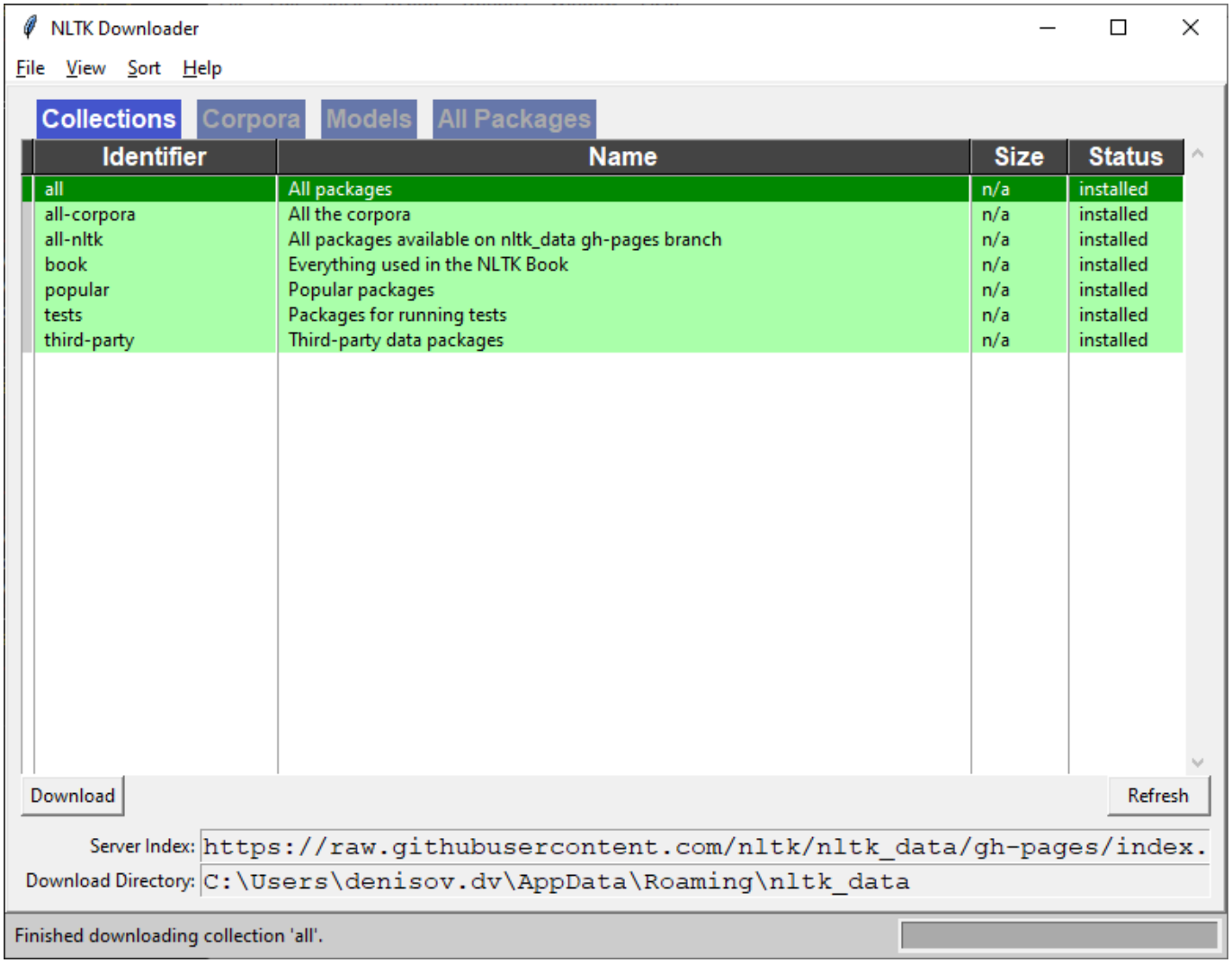
conda install -c https://conda.anaconda.org/conda-forge wordcloud

conda install -c anaconda nltk

3. Откройте командную строку Windows или IDLE Shell (Python) и введите команды *import nltk* и *nltk.download()*:



4. Откройте NLTK Downloader, нажмите кнопку Download и дождитесь окончания загрузки всех файлов. После завершения скачивания и установке можно перейти к импорту и анализу данных.



5. Частотный анализ русского текста. Подготовьте данные в формате .txt, содержащие обычный текст русского языка. Данные, которые вы подготовите будут использованы для дальнейшего анализа.

In [41]:

```
f = open('pushkin.txt', "r", encoding="utf-8")
text = f.read()
```

In [42]:

```
type(text)
```

Out[42]:

```
str
```

In [43]:

```
len(text)
```

Out[43]:

```
22968
```

In [44]:

```
text[:300]
```

Out[44]:

```
'Метель \n\n\nКони мчатся по буграм, \n\nТопчут снег глубокой \n\nВот, в сторонке
божий храм \n\nВиден одинокой. \n\nВдруг метелица кругом; \n\nСнег валит клоками;
```

```
\n\nЧерный вран, свистя крылом, \n\nВьется над санями; \n\nВещий стон гласит печаль!
\n\nКони торопливы \n\nЧутко смотрят в темну даль, \n\nВоздымая
гривы\n\n\xa0\xa0\xa0\xa0Жу'
```

6. Проведите предварительную обработку текста. Предварительная обработка текста включает в себя такие операции как удаление лишних символов, перевод слов в единый регистр и т.д.

In [45]:

```
# перевод в единый регистр (например, нижний)
text = text.lower()
```

In [46]:

```
import string
string.punctuation
```

Out[46]:

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [47]:

```
type(string.punctuation)
```

Out[47]:

```
str
```

In [48]:

```
spec_chars = string.punctuation + '\n\xa0«»\t—…'
```

In [49]:

```
%%time
text = "".join([ch for ch in text if ch not in spec_chars])
Wall time: 4 ms
```

In [51]:

```
import re
text = re.sub('\n', '', text)
```

In [52]:

```
def remove_chars_from_text(text, chars):
    return "".join([ch for ch in text if ch not in chars])
```

In [53]:

```
%%time
text = remove_chars_from_text(text, spec_chars)
Wall time: 4.02 ms
```

In [54]:

```
%%time
text = remove_chars_from_text(text, string.digits)
Wall time: 2.99 ms
```

7. Проведите тонизацию текста

In [55]:

```
from nltk import word_tokenize
text_tokens = word_tokenize(text)
```

In [56]:

```
print(type(text_tokens), len(text_tokens))
text_tokens[:10]
<class 'list'> 3402
```

Out[56]:

```
['метель',
 'кони',
 'мчатся',
 'по',
 'буграм',
 'топчут',
 'снег',
 'глубокой',
 'вот',
 'в']
```

In [57]:

```
import nltk
text = nltk.Text(text_tokens)
print(type(text))
text[:10]
```

```
<class 'nltk.text.Text'>
```

Out[57]:

```
['метель',
 'кони',
 'мчатся',
 'по',
 'буграм',
 'топчут',
 'снег',
 'глубокой',
 'вот',
 'в']
```

8. Расчет частоты встречаемости слов

In [59]:

```
%%time
from nltk.probability import FreqDist
fdist = FreqDist(text)
fdist
```

```
Wall time: 6.98 ms
```

Out[59]:

```
FreqDist({'и': 146, 'в': 101, 'не': 69, 'что': 54, 'с': 44, 'он': 42, 'она': 39,
'ее': 39, 'на': 31, 'было': 27, ...})
```

In [60]:

```
fdist.most_common(5)
```

Out[60]:

```
[('и', 146), ('в', 101), ('не', 69), ('что', 54), ('с', 44)]
```

In [61]:

```
fdist.plot(30,cumulative=False)
```

9. Удаление стоп-слов

In [62]:

```
from nltk.corpus import stopwords
russian_stopwords = stopwords.words("russian")
russian_stopwords.extend(['это', 'нею'])
```

In [63]:

```
print(len(russian_stopwords))
# russian_stopwords
153
```

In [64]:

```
%%time
text_tokens = [token.strip() for token in text_tokens if token not in
russian_stopwords]
Wall time: 6.98 ms
```

In [65]:

```
print(len(text_tokens))
2158
```

In [66]:

```
text = nltk.Text(text_tokens)
```

In [67]:

```
fdist_sw = FreqDist(text)
fdist_sw.most_common(10)
```

Out[67]:

```
[('владимир', 23),
 ('гавриловна', 20),
 ('марья', 17),
 ('поехал', 9),
 ('бурмин', 9),
 ('поминутно', 8),
 ('метель', 7),
 ('несколько', 6),
 ('сани', 6),
 ('владимира', 6)]
```

10. Построение облака слов

In [68]:

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
%matplotlib inline
```

In [71]:

```
text_raw = " ".join(text)
```

In [72]:

```
wordcloud = WordCloud().generate(text_raw)
```

In [73]:

```
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

11. В качестве результата выполнения работы должно получить облако слов следующего вида (облако будет отличаться для разного набора данных, рисунок ниже приведен только в качестве примера)

Требования к отчету по лабораторной работе.

Отчет по лабораторной работе должен быть в виде документа ipynb (документ Jupyter Notebook), ссылки на предоставленный доступ к документу google colab или другие аналогичные форматы. Отчет в формате pdf, word со структурой:

1. Название работы.
2. Краткое пояснение к содержанию.
3. Имя, фамилия, группа студента, выполнившего работу.
4. Задание на лабораторную работу.
5. Краткое описание теоретических сведений, соответствующих работе.
6. Код реализации выполнения задания.
7. Визуализация результатов выполнения (если применимо).
8. Выводы.
9. Приложение.

5. Лабораторная работа. Разработка системы для распределенных вычислений

Цель работы: изучить работу с библиотеками pandas и pickle, получить навыки работы с большими данными в парадигме MapReduce.

Задание к работе.

1. Перед выполнением лабораторной работы, просмотрите программный код, который вы использовали в первой лабораторной работе.
2. В разделе «ход работы» пошагово выполнить каждый пункт с описанием и примера реализации задач по теме лабораторной работы.
3. После пункта выполнить «индивидуальное задание» при его наличии.

Ход работы.

1. В работе показана простая реализация алгоритма Map-Reduce, реализованная в наборе данных W-Quality с использованием Python.
2. Нанесение данных на карту (функция Map) и вывод информационного сообщения «Данные нанесены на карту. Теперь запустите reducer, чтобы уменьшить содержимое файла»

In [1]:

```
import pandas as pd
import pickle

data = pd.read_csv('data.csv')

#Slicing Data
slice1 = data.iloc[0:399,:]
slice2 = data.iloc[400:800,:]
slice3 = data.iloc[801:1200,:]
slice4 = data.iloc[1201:,:]

def mapper(data):

    mapped = []

    for index,row in data.iterrows():

        mapped.append((row['quality'],row['volatile acidity']))

    return mapped

map1 = mapper(slice1)
map2 = mapper(slice2)
map3 = mapper(slice3)
map4 = mapper(slice4)
```

```
shuffled = {
    3.0: [],
    4.0: [],
    5.0: [],
    6.0: [],
    7.0: [],
    8.0: [],

}

for i in [map1,map2,map3,map4]:
    for j in i:
        shuffled[j[0]].append(j[1])

file= open('shuffled.pkl','ab')
pickle.dump(shuffled,file)
file.close()

print("Data has been mapped. Now, run reducer.py to reduce the contents in
shuffled.pkl file.")
```

3. Реализация функции Reduce

```
import pickle

file= open('shuffled.pkl','rb')
shuffled = pickle.load(file)

def reduce(shuffled_dict):
    reduced = {}

    for i in shuffled_dict:

        reduced[i] = sum(shuffled_dict[i])/len(shuffled_dict[i])

    return reduced
final = reduce(shuffled)

print("Average volatile acidity in different classes of wine: ")
for i in final:

       print(i,':',final[i])
```

Требования к отчету по лабораторной работе.

Отчет по лабораторной работе должен быть в виде документа ipynb (документ Jupyter Notebook), ссылки на предоставленный доступ к документу google colab или другие аналогичные форматы. Отчет в формате pdf, word со структурой:

1. Название работы.
2. Краткое пояснение к содержанию.
3. Имя, фамилия, группа студента, выполнившего работу.
4. Задание на лабораторную работу.

5. Краткое описание теоретических сведений, соответствующих работе.
6. Код реализации выполнения задания.
7. Визуализация результатов выполнения (если применимо).
8. Выводы.
9. Приложение.

6. Лабораторная работа. Разбор реализации системы для анализа социальных сетей

Цель работы: изучить работу с API и принципы визуализации данных с библиотекой plotly.

Задание к работе.

1. Перед выполнением лабораторной работы, просмотрите программный код, который вы использовали в первой лабораторной работе.
2. В разделе «ход работы» пошагово выполнить каждый пункт с описанием и примера реализации задач по теме лабораторной работы.
3. После пункта выполнить «индивидуальное задание» при его наличии.

Ход работы.

В этой работе будет реализована визуализация данных на базе информации с GitHub – сайта, организующего совместную работу программистов над проектами. Взаимодействие с сайтами реализуется через API (программный интерфейс).

1. Для визуализации мы будем использовать библиотеку plotly, скачайте и установите ее стандартными средствами, например через командную строку anaconda:

conda install -c plotly plotly=5.3.0

2. GitHub поддерживает программный интерфейс для запроса разнообразной информации посредством вызова API. Например, введите в строку браузера следующий адрес и нажмите Enter:

https://api.github.com/search/repositories?q=language:python&sort=stars

3. Данная команда вызова возвращает количество проектов Python, размещенных на GitHub в настоящее время, а также информацию о самых популярных репозиториях Python.
4. В качестве *«индивидуального задания»* разберите API-запрос на составляющие, дайте ответ на вопрос, какая часть запроса отвечает за сортировку проектов по количество присвоенных звезд.

5. С использованием библиотеки plotly постройте визуализацию данных, демонстрирующую популярность проектов Python на GitHub:

```
import requests
from plotly.graph_objs import Bar
from plotly import offline

# Make an API call and store the response.
url = 'https://api.github.com/search/repositories?q=language:python&sort=stars'
headers = {'Accept': 'application/vnd.github.v3+json'}
r = requests.get(url, headers=headers)
print(f"Status code: {r.status_code}")

# Process results.
response_dict = r.json()
repo_dicts = response_dict['items']
repo_links, stars, labels = [], [], []
for repo_dict in repo_dicts:
    repo_name = repo_dict['name']
    repo_url = repo_dict['html_url']
    repo_link = f"<a href='{repo_url}'>{repo_name}</a>"
    repo_links.append(repo_link)
    stars.append(repo_dict['stargazers_count'])
    owner = repo_dict['owner']['login']
    description = repo_dict['description']
    label = f"{owner}<br />{description}"
    labels.append(label)
# Make visualization.
data = [{
    'type': 'bar',
    'x': repo_links,
    'y': stars,
    'hovertext': labels,
    'marker': {
        'color': 'rgb(60, 100, 150)',
        'line': {'width': 1.5, 'color': 'rgb(25, 25, 25)'}
    },
    'opacity': 0.6,
}]
my_layout = {
    'title': 'Most-Starred Python Projects on GitHub',
    'titlefont': {'size': 28},
    'xaxis': {
        'title': 'Repository',
        'titlefont': {'size': 24},
        'tickfont': {'size': 14},
    },
    'yaxis': {
        'title': 'Stars',
        'titlefont': {'size': 24},
        'tickfont': {'size': 14},
    },
}
fig = {'data': data, 'layout': my_layout}
offline.plot(fig, filename='python_repos.html')
```

6. Реализуйте подобную визуализацию для любой интересной для вас социальной сети. Например, вы можете построить диаграмму

тематических групп Вконтакте, отсортированную по количеству участников.

Требования к отчету по лабораторной работе.

Отчет по лабораторной работе должен быть в виде документа ipynb (документ Jupyter Notebook), ссылки на предоставленный доступ к документу google colab или другие аналогичные форматы. Отчет в формате pdf, word со структурой:

1. Название работы.
2. Краткое пояснение к содержанию.
3. Имя, фамилия, группа студента, выполнившего работу.
4. Задание на лабораторную работу.
5. Краткое описание теоретических сведений, соответствующих работе.
6. Код реализации выполнения задания.
7. Визуализация результатов выполнения (если применимо).
8. Выводы.
9. Приложение.

ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ ЛИТЕРАТУРЫ:

Основная литература:

1. Грас Дж. Data Science. Наука о данных с нуля: Пер. с англ. – СПб.: БХВ-Петербург, 2020. – 336 с.: ил.

2. Николенко С.И., Кадурин А.А., Архангельская Е.О. Глубокое обучение. – СПб.: Питер, 2017. – 480 с.

3. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / пер. с анг. А. А. Слинкина. – 2-е изд., испр. – Москва : ДМК Пресс, 2018. – 652 с.: цв. ил.

4. Шолле Ф. Глубокое обучение на Python. – СПб.: Питер, 2018. – 400 с.

Дополнительная литература:

1. Клетте, Рейнхард. Компьютерное зрение. Теория и алгоритмы. – М.: ДМК Пресс, 2019.

2. Пойтнер, Ян. Программируем с PyTorch. Создание приложений глубокого обучения. – СПб: из-во Питер, 2020. – 256 с.

3. Траск Эндрю. Грокаем глубокое обучение. – СПб.: Питер, 2019. – 352 с.

4. Ян, Э. С. Программирование компьютерного зрения на языке Python / Э. С. Ян ; перевод с английского А. А. Слинкин. — Москва : ДМК Пресс, 2016. — 312 с.

5. Он-лайн курс «Нейронные сети и компьютерное зрение». – URL: https://stepik.org/course/50352/promo.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К САМОСТОЯТЕЛЬНОЙ РАБОТЕ

ВВЕДЕНИЕ

Во введении мы рассмотрим процесс установки необходимого программного обеспечения и напишем простейшую программу на языке Python, которая выводит сообщение Hello World. В качестве основного программного обеспечения, который будет использован для работы с инструментами анализа данных мы будем использовать Anaconda. Это дистрибутив языков программирования Python и R, включающий набор популярных свободных библиотек, объединенных проблематиками науки о данных и машинного обучения. Скачать дистрибутив можно с сайта anaconda.com.

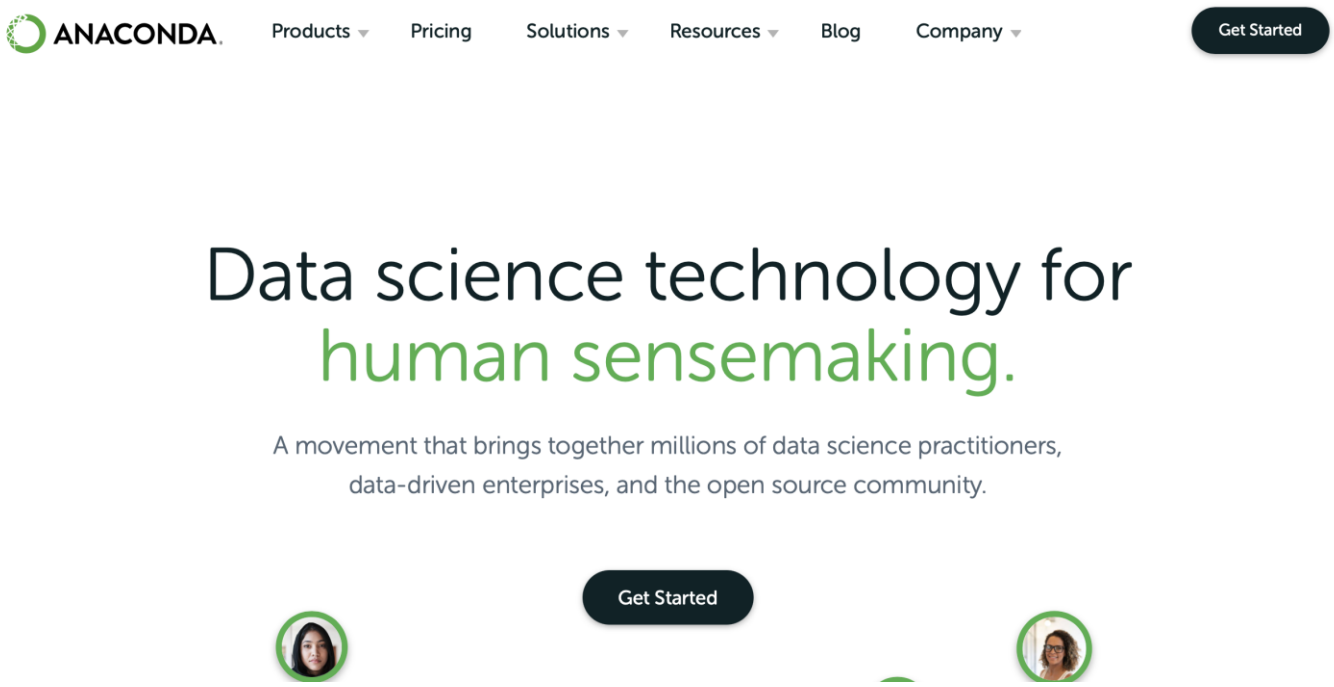


Рисунок В.1

Для скачивания переходим в верхнем меню сайта во вкладку Products – Individual Edition - Download:

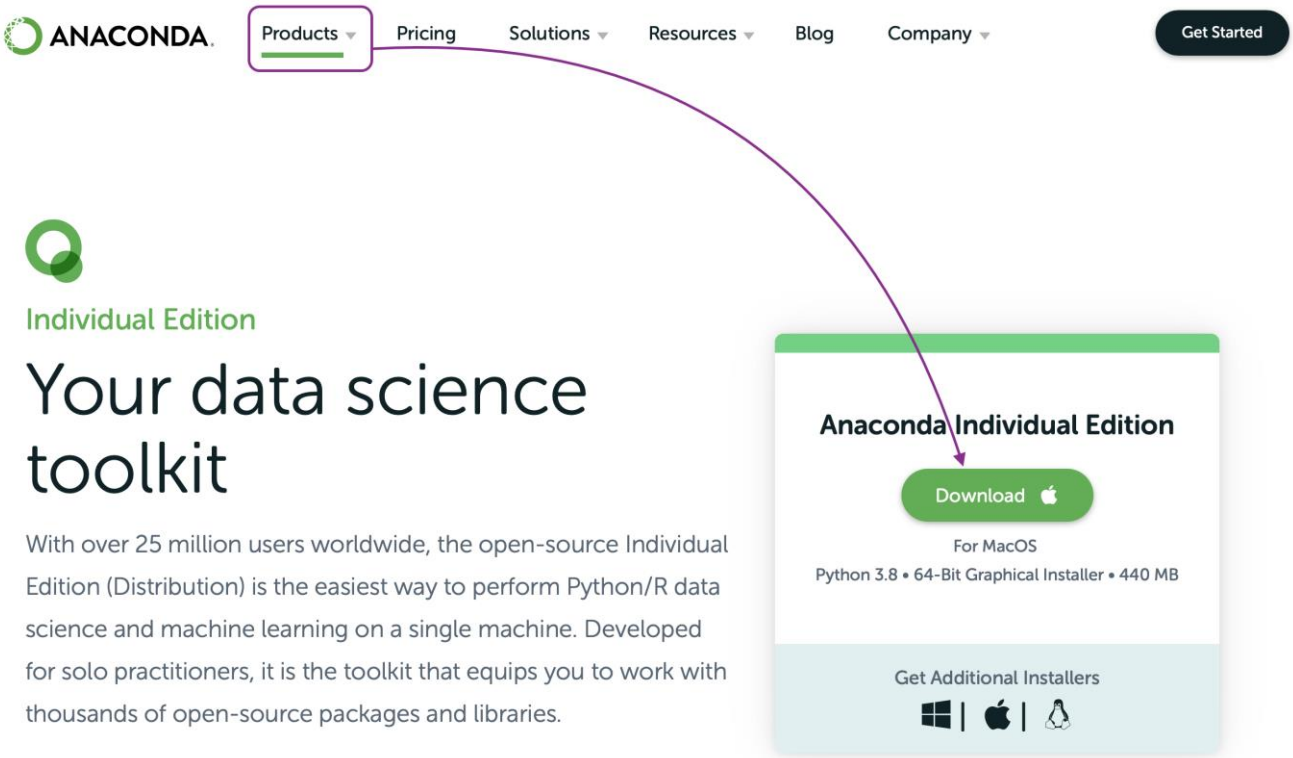


Рисунок В.2

Запускаем скачанный дистрибутив:

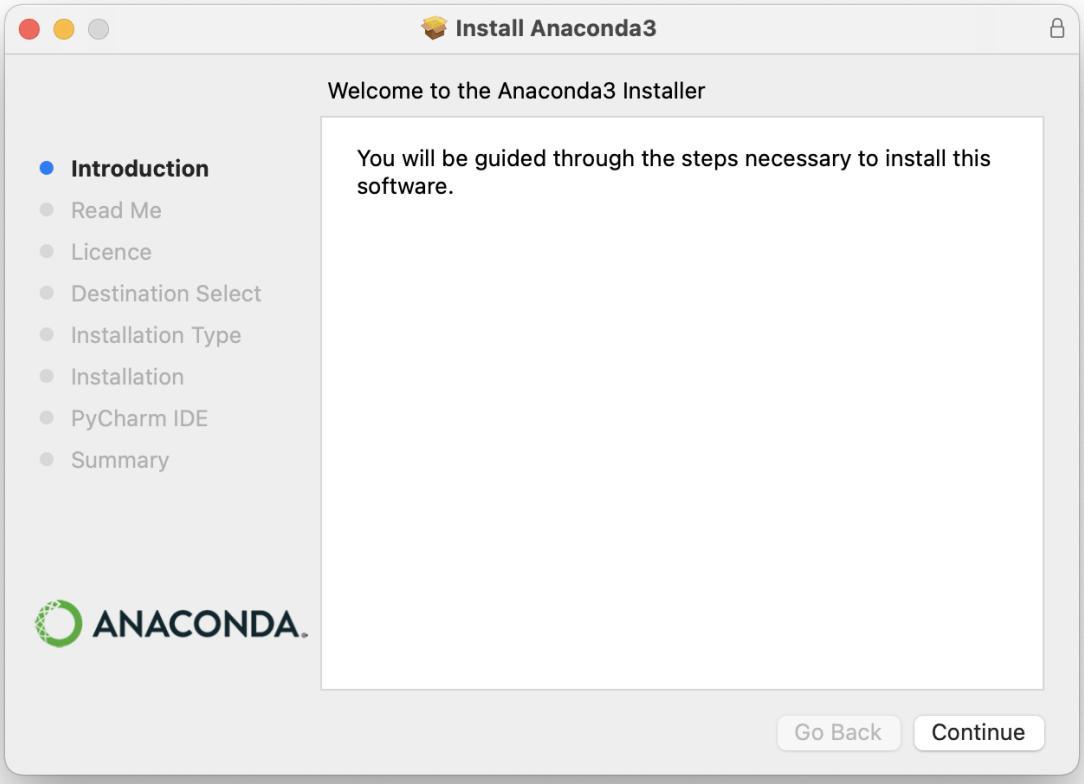


Рисунок В.3

Необходимо принять лицензионное соглашение и выбрать место для установки:

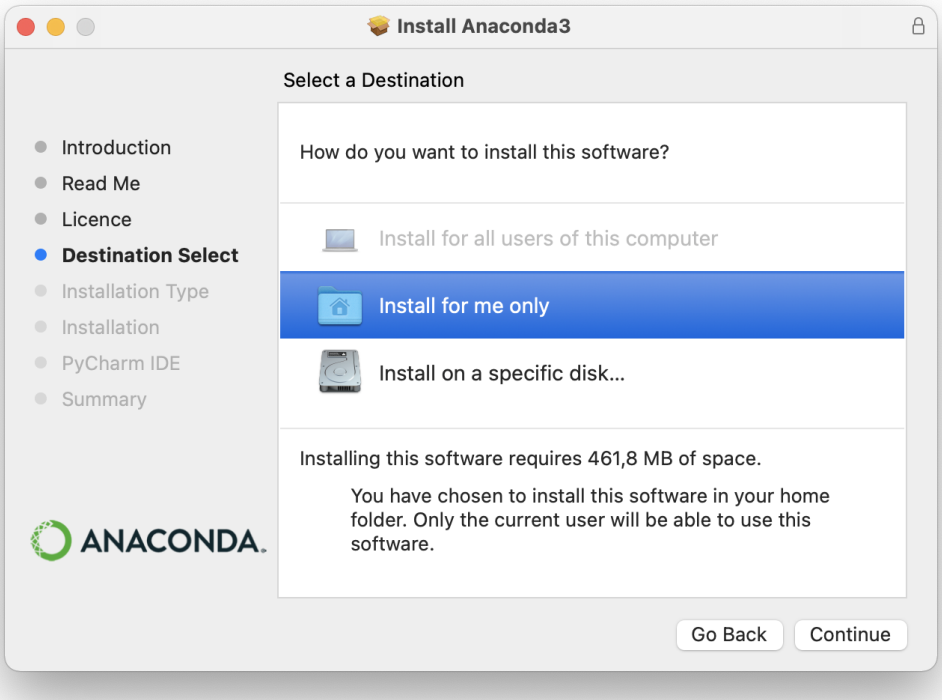


Рисунок В.4

Нажмите Continue, после чего начнется процесс установки.

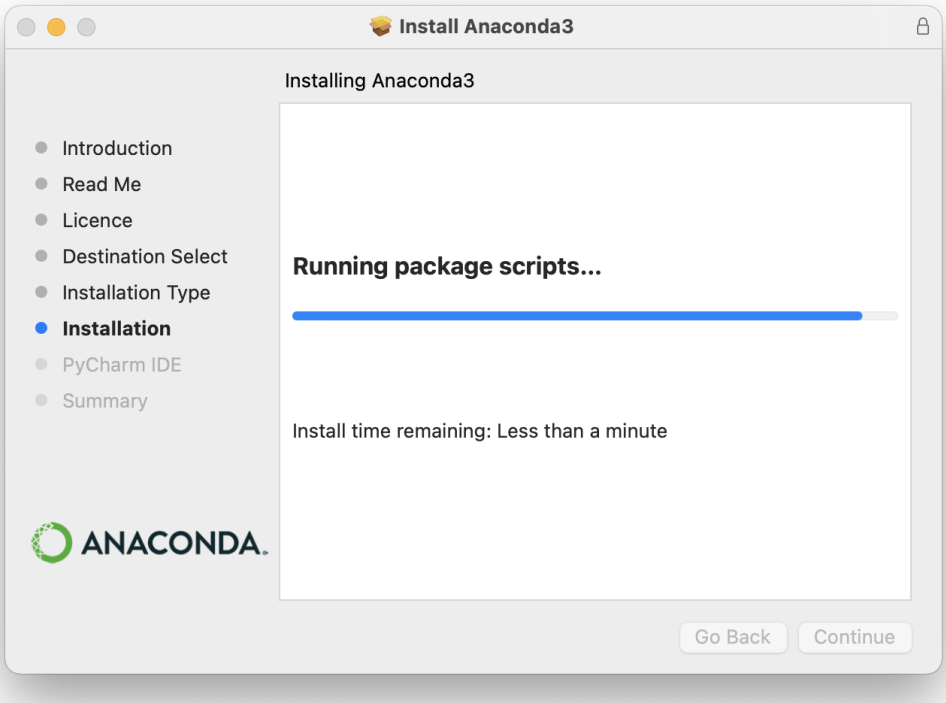


Рисунок В.5

По завершению установки основного дистрибутива, будет предложено установить среду разработки PyCharm IDE. Это программное обеспечение можно установить по желанию (его мы не будем использовать в рамках выполнения практических работ), и завершить процесс установки. После этого запустите Anaconda-Navigator, навигатор содержит достаточно большое количество полезных инструментов для работы с данными.

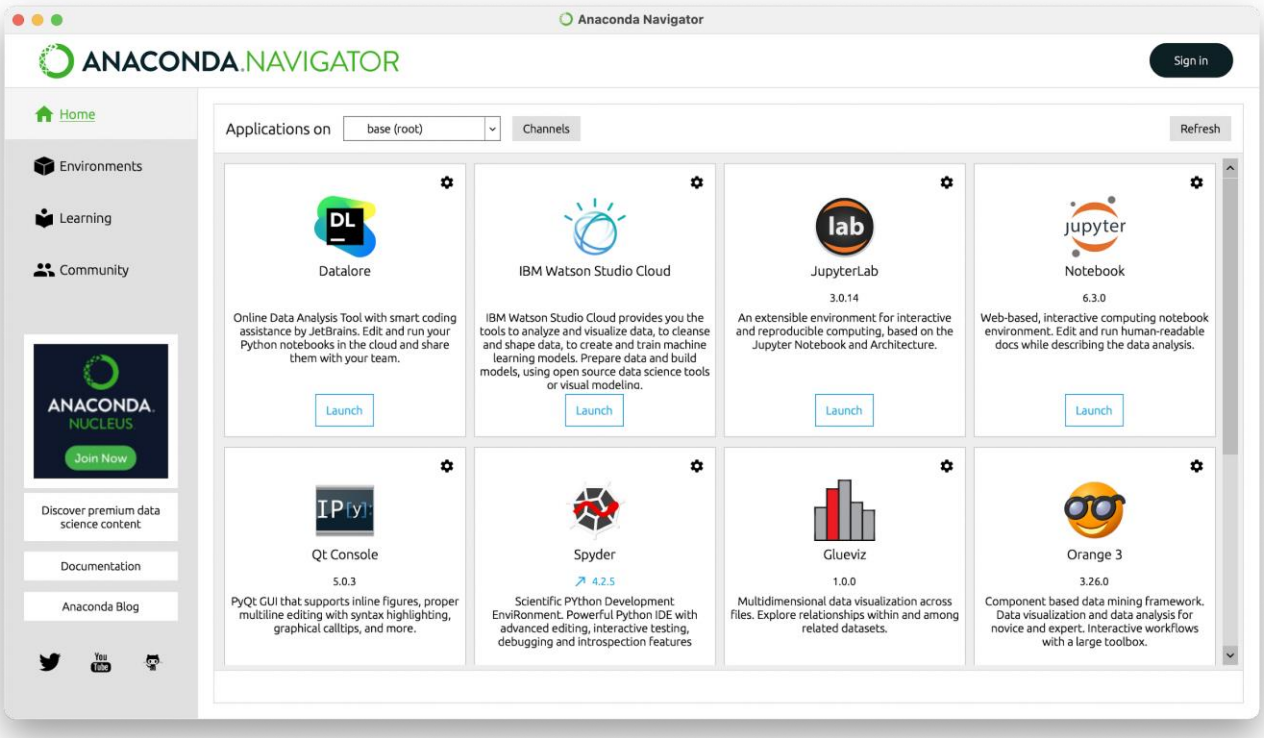


Рисунок B.6

Запустите инструмент Jupyter Notebook, нажав кнопку Launch. После этого откроется терминал и окно браузера по адресу http://localhost:8888/tree/ с деревом файлов на вашем локальном компьютере. В окне браузера создайте новую папку для работы с файлами проекта, например на рисунке ниже показана созданная папка UrFU/1-Anaconda, с созданным внутри файлом HelloWorld.jpynb. Для создания файлов и папок используйте элементы управления (New, Refresh…) в верхней части окна.

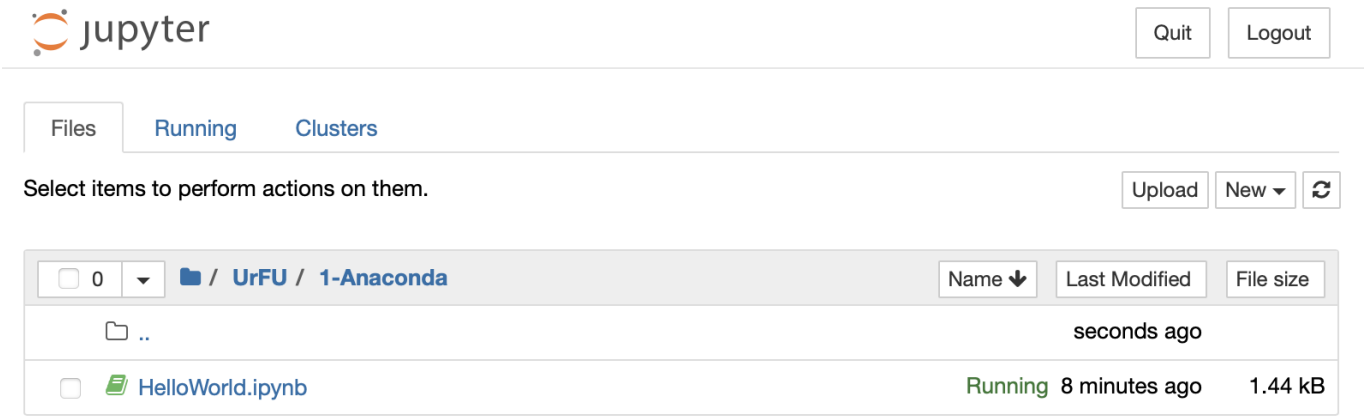


Рисунок В.7

Откройте созданный файл с именем HelloWorld, напишите в него команду print() с характерным содержанием и запустите выполнение, нажав Run. Результатом выполнения станет вывод сообщения Hello World.

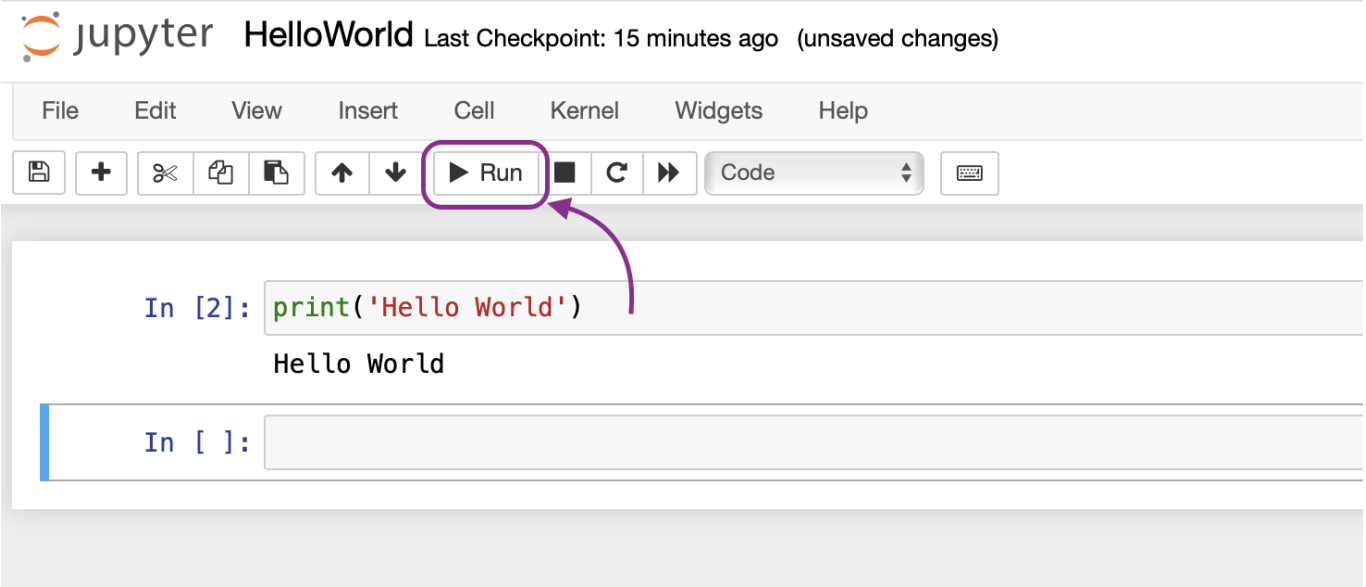


Рисунок В.8

Таким образом, подготовку необходимого программного обеспечения для выполнения лабораторных работ можно считать завершенной.

1. САМОСТОЯТЕЛЬНАЯ РАБОТА. КОНСТРУКЦИИ ЯЗЫКА PYTHON

Цель работы: ознакомиться с основными операторами Python, углубленно изучить работу этого языка программирования.

Задание к работе.

1. Ознакомиться с официальным руководством по языку Python, расположенном по адресу: https://docs.python.org/2/tutorial/
2. Ознакомьтесь с официальным руководством по Python: https://ipython.org/ipython-doc/2/interactive/tutorial
3. Составьте на одну из предложенных тем презентацию на 10-15 слайдов с примером реализации одной из функций на языке Python:
    a. Генераторы последовательностей. Функции-генераторы и генераторные выражения;
    b. Регулярные выражения;
    c. Объектно-ориентированное программирование в Python;
    d. Инструменты функционального программирования;
    e. Применение специальных функций: enumerate, zip и распаковка аргументов, переменные args и kwargs.

2. САМОСТОЯТЕЛЬНАЯ РАБОТА. СБОР, ОБРАБОТКА И ВИЗУАЛИЗАЦИЯ ТЕСТОВОГО НАБОРА ДАННЫХ

Цель работы: изучить работу с библиотеками для визуализации данных, получить навыки подготовки данных для их дальнейшего анализа и представления.

Задание к работе.

1. Ознакомиться с библиотекой seaborn (позволяет создавать более сложные визуализации по сравнению с matplotlib): https://web.stanford.edu/~mwaskom/software/seaborn.
2. Ознакомьтесь с библиотекой Bokeh (https://docs.bokeh.org/en/latest/), которая позволяет создавать визуализации в стиле D3,js, но на основе Python.
3. Выполните задания, данные в лабораторной работе №2 «сбор, обработка и визуализация тестового набора данных», но в качестве инструментов визуализации используйте библиотеки и пунктов 1 и 2 текущей самостоятельной работы.

3. САМОСТОЯТЕЛЬНАЯ РАБОТА. РАЗРАБОТКА СИСТЕМЫ МАШИННОГО ОБУЧЕНИЯ

Цель работы: изучить работу с библиотекой машинной обучения Scikit-learn.

Задание к работе.

1. Ознакомьтесь с учебником «Элементы статистического обучения», который можно скачать бесплатно с сайта: https://statweb.stanford.edu/~tibs/ElemStatLearn
2. Подготовьте презентацию на 10-15 слайдов на тему «Машинное обучение. Переобучение и недообучение».
3. Приведите пример кода, в котором модель достаточно обучена чтобы правильно классифицировать данные и недостаточно обучена (допускает значительное количество ошибок).

4. САМОСТОЯТЕЛЬНАЯ РАБОТА. РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ДЛЯ ОБРАБОТКИ ЕСТЕСТВЕННОГО ЯЗЫКА

Цель работы: изучить работу с платформой для создания NLP-программ на Python – NLTK (Natural Language Toolkit).

Задание к работе.

1. Подготовьте ответы на вопросы:
    a. Что такое Natural Language Processing?
    b. Какие задачи можно решать с помощью Python-библиотеки NLTK? Перечислите задачи и приведите один пример с кодом реализации одной из них.
    c. Что такое токенизация? Зачем она используется?
2. С помощью каких инструментов и библиотек Python можно подготовить облако частотности появления слов в виде произвольного изображения, как показано на рисунке ниже? Опишите порядок реализации данной задачи и инструменты, которые для этого используются

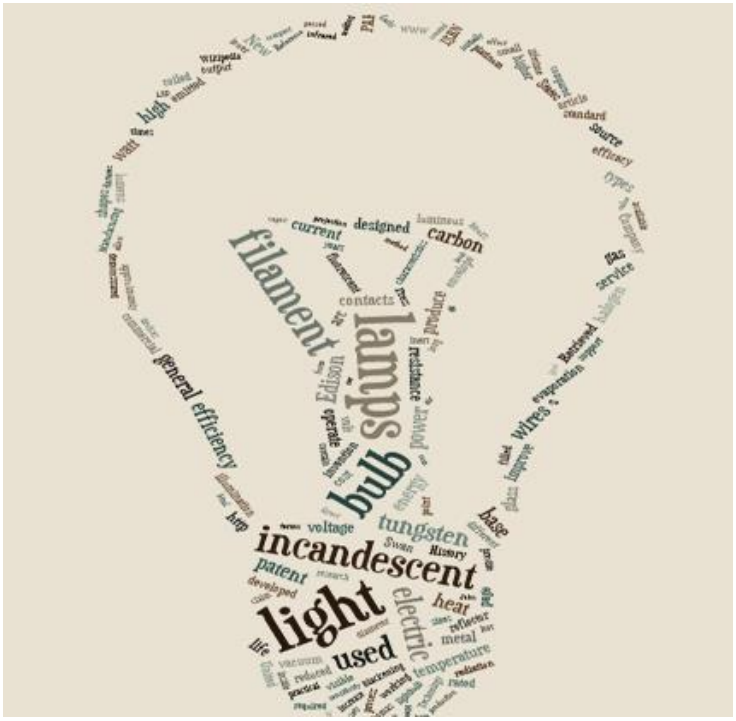


Рисунок 4.1

5. САМОСТОЯТЕЛЬНАЯ РАБОТА. РАЗРАБОТКА СИСТЕМЫ ДЛЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ

Цель работы: изучить работу с библиотеками pandas и pickle, получить навыки работы с большими данными в парадигме MapReduce.

Задание к работе.

1. Подготовьте презентацию на 10-15 слайдов с описанием технологии MapReduce.
2. Как MapReduce может быть использована для анализа обновлений ленты новостей? Дайте развернутый ответ, приведите ключевые части кода для реализации подобной задачи.
3. Опишите преимущества технологии MapReduce на примере умножения матриц? Дайте развернутый ответ, приведите ключевые части кода для реализации подобной задачи.

6. Самостоятельная работа. Принципы обработки естественного языка, подходы к анализу социальных сетей

Цель работы: изучить задачу анализа социальный сетей и обработки запросов на примере анализа данных с помощью графов.

Задание к работе.

Анализ социальных сетей – это процесс исследования различных систем с использованием теории сетей. Он начал широко применяться именно тогда, когда стало понятно, что огромное количество существующих сетей (социальных, экономических, биологических) обладают универсальными свойствами: изучив один тип, можно понять структуру и любых других сетей и научиться делать предсказания по ним.

Любые сети состоят из отдельных участников (людей или вещей в сети) и отношений между ними. Сети очень часто визуализируются с помощью графов – структур, состоящих из множества точек и линий, отображающих связи между этими точками. Участники представлены в виде узлов сети, а их отношения представлены в виде линий, их связывающих. Такая визуализация помогает получить качественную и количественную оценку сетей.

1. Опишите в произвольной форме задачу, в которой можно будет использовать визуализацию данных из социальных сетей для представления данных в виде рис. 6.1. Какие инструменты Python потребуется использовать?

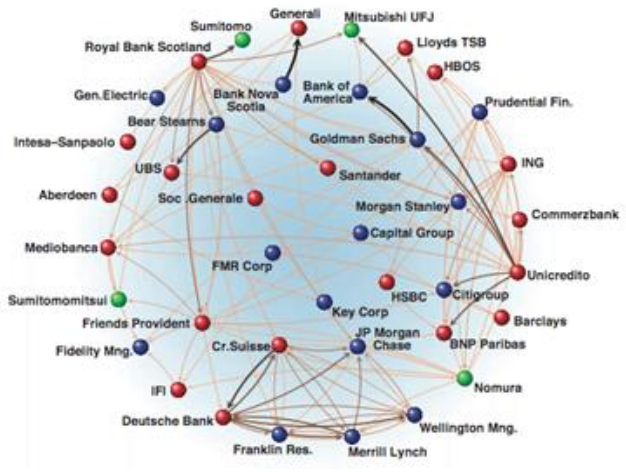


Рисунок 6.1

7. САМОСТОЯТЕЛЬНАЯ РАБОТА. РАЗБОР РЕАЛИЗАЦИИ СИСТЕМЫ ДЛЯ АНАЛИЗА СОЦИАЛЬНЫХ СЕТЕЙ

Цель работы: изучить работу с библиотеками requests, network и collections на примере решения задачи по анализу социальных сетей.

Задание к работе.

1. На примере Вконтакте API и языка Python построить эгоцентричный граф друзей:

```
import requests
import networkx
import time
import collections

def get_friends_ids(user_id):
    friends_url = 'https://api.vk.com/method/friends.get?user_id={}'
    # также вы можете добавить access_token в запрос, получив его через OAuth 2.0
    json_response = requests.get(friends_url.format(user_id)).json()
    if json_response.get('error'):
        print json_response.get('error')
        return list()
    return json_response[u'response']
graph = {}
friend_ids = get_friends_ids(1405906)  # ваш user id, для которого вы хотите
построить граф друзей.
for friend_id in friend_ids:
    print 'Processing id: ', friend_id
    graph[friend_id] = get_friends_ids(friend_id)
g = networkx.Graph(directed=False)
for i in graph:
    g.add_node(i)
    for j in graph[i]:
        if i != j and i in friend_ids and j in friend_ids:
            g.add_edge(i, j)

pos=networkx.graphviz_layout(g,prog="neato")
networkx.draw(g, pos, node_size=30, with_labels=False, width=0.2)
```

2. Превлиятельный результат выполнения изображен на рисунке 7.1

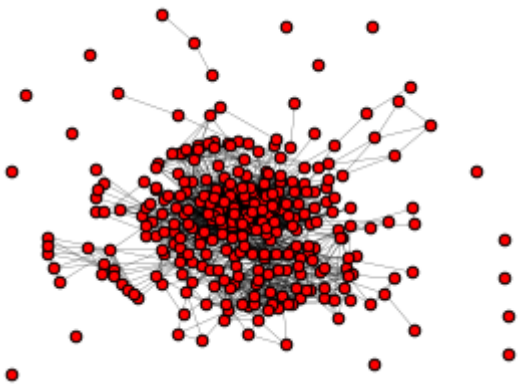


Рисунок 7.1

Перечень основной и дополнительной литературы

Основная литература:

1. Грас Дж. Data Science. Наука о данных с нуля: Пер. с англ. – СПб.: БХВ-Петербург, 2020. – 336 с.: ил.

2. Николенко С.И., Кадурин А.А., Архангельская Е.О. Глубокое обучение. – СПб.: Питер, 2017. – 480 с.

3. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / пер. с анг. А. А. Слинкина. – 2-е изд., испр. – Москва : ДМК Пресс, 2018. – 652 с.: цв. ил.

4. Шолле Ф. Глубокое обучение на Python. – СПб.: Питер, 2018. – 400 с.

Дополнительная литература:

1. Клетте, Рейнхард. Компьютерное зрение. Теория и алгоритмы. – М.: ДМК Пресс, 2019.

2. Пойтнер, Ян. Программируем с PyTorch. Создание приложений глубокого обучения. – СПб: из-во Питер, 2020. – 256 с.

3. Траск Эндрю. Грокаем глубокое обучение. – СПб.: Питер, 2019. – 352 с.

4. Ян, Э. С. Программирование компьютерного зрения на языке Python / Э. С. Ян ; перевод с английского А. А. Слинкин. — Москва : ДМК Пресс, 2016. — 312 с.

5. Он-лайн курс «Нейронные сети и компьютерное зрение». – URL: https://stepik.org/course/50352/promo.