Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

по домашней работе № 1

Название: Дескриптивный анализ данных.

Дисциплина: Методы машинного обучения.

Студент ИУ6-22М (Группа) _____ (Подпись, дата) М.И. Шаговитов (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) С.Ю. Папулин (И.О. Фамилия)

Москва, 2024

Цель работы:

Приобрести опыт решения практических задач по анализу данных, таких как загрузка, трансформация, вычисление простых статистик и визуализация данных в виде графиков и диаграмм, посредством языка программирования Python.

Вариант:

На рисунке 1 показан код, который вычисляет вариант относительно фамилии студента.

```
surname = "Шаговитов"

alp = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'
w = [1, 42, 21, 21, 34,  6, 44, 26, 18, 44, 38, 26, 14, 43,  4, 49, 45,
       7, 42, 29,  4,  9, 36, 34, 31, 29,  5, 30,  4, 19, 28, 25, 33]

# создаем словарь из двух списков, где ключами будут буквы, а значения - цифры
d = dict(zip(alp, w))

# потом суммируем цифры относительно букв введенной фамилии, делим на 40 и прибывляем 1
variant =  sum([d[el] for el in surname.lower()]) % 40 + 1

print("Задача № 1, шаг 5 - вариант: ", variant % 5 + 1)
print("Задача № 1, шаг 11 - вариант: ", variant % 2 + 1 )
print("задача № 2 - вариант: ", variant % 4 + 1)
```

```
Задача № 1, шаг 5 - вариант:  1
Задача № 1, шаг 11 - вариант:  2
задача № 2 - вариант:  2
```

Рисунок 1 – Вычисление варианта

В результате выполнения вышеприведенного кода были определены следующие варианты:

1) задача 1, шаг 5 – вариант 1;
2) задача 1, шаг 11 – вариант 2;
3) задача 2 – варианта 2.

Задание 1. Анализ индикаторов качества государственного управления (The Worldwide Government Indicators, WGI).

В качестве индикатора далее необходимо использовать контроль над коррупцией (Control of Corruption) и его показатели rank и estimate.

Набор данных:

1) WGI;

2) Регионы;

3) Описание WGI.

Замечание. Исходный файл с данными редактировать нельзя.

Выполнение:

1. Загрузите данные в DataFrame.

Для загрузки данный в DataFrame необходимо было предварительно скачать файл с данными, после чего, при чтении файла, определить какую строку взять за названия колонок. Эта загрузка и вывод результирующего DataFrame приведены на рисунке 2.

```
# импортируем библиотеки pandas и numpy
import pandas as pd
import numpy as np
```

```
# с помощью библиотеки pandas формируем датафрейм, где указываем -
# название файла, название используемой страницы и номер строки с заголовками
df = pd.read_excel("wgidataset.xlsx",
                   sheet_name='ControlofCorruption',
                  header=14)

# возврат первых 5-ти элементов датафрейма
df.head()
```

	Country/Territory	Code	Estimate	StdErr	NumSrc	Rank	Lower	Upper	Estimate.1	StdErr.1	...	NumSrc.22	Rank.22	Lower.22	Upper.2
0	Aruba	ABW	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2.0	78.571426	62.857143	89.04762
1	Andorra	ADO	1.318143	0.480889	1.0	87.096771	72.043015	96.774193	1.334759	0.453149	...	1.0	88.571426	70.000000	96.66666
2	Afghanistan	AFG	-1.291705	0.340507	2.0	4.301075	0.000000	27.419355	-1.176012	0.324013	...	8.0	12.380953	4.761905	20.95238
3	Angola	AGO	-1.167702	0.262077	4.0	9.677420	0.537634	27.419355	-1.180451	0.227055	...	10.0	29.047619	19.523809	40.47618
4	Anguilla	AIA	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.0	73.333336	50.952381	89.04762

5 rows × 146 columns

Рисунок 2 – Выполнение первого пункта первого задания

2. Отсортируйте данные по убыванию индекса DataFrame.

Для сортировки используем метод sort_index(ascending=0), в котором задаем условие сортировки, что и показано на рисунке 3.

```
# сортируем датафрейм по индексу (sort_index) по убыванию (ascending=0)
df.sort_index(ascending=0)
```

	Country/Territory	Code	Estimate	StdErr	NumSrc	Rank	Lower	Upper	Estimate.1	StdErr.1	...	NumSrc.22	Rank.22	Lower.22	Upper
213	Zimbabwe	ZWE	-0.278847	0.244907	5.0	47.849461	30.645161	60.752689	-0.504802	0.198134	...	12.0	10.000000	4.285714	17.619
212	Zambia	ZMB	-0.840641	0.262077	4.0	24.731182	5.913979	41.397850	-0.853156	0.227055	...	12.0	25.714285	17.619047	33.809
211	Congo, Dem. Rep.	ZAR	-1.647852	0.315914	3.0	0.000000	0.000000	12.365591	-1.416679	0.310343	...	11.0	4.285714	0.000000	6.666
210	South Africa	ZAF	0.732927	0.210325	6.0	76.344086	66.129036	81.182793	0.638809	0.188628	...	11.0	53.809525	46.190475	60.000
209	Serbia	SRB	-1.140072	0.262077	4.0	11.827957	0.537634	29.032259	-1.195605	0.191027	...	9.0	35.714287	26.190475	48.095
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4	Anguilla	AIA	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.0	73.333336	50.952381	89.047
3	Angola	AGO	-1.167702	0.262077	4.0	9.677420	0.537634	27.419355	-1.180451	0.227055	...	10.0	29.047619	19.523809	40.476
2	Afghanistan	AFG	-1.291705	0.340507	2.0	4.301075	0.000000	27.419355	-1.176012	0.324013	...	8.0	12.380953	4.761905	20.952
1	Andorra	ADO	1.318143	0.480889	1.0	87.096771	72.043015	96.774193	1.334759	0.453149	...	1.0	88.571426	70.000000	96.666

Рисунок 3 – Выполнение второго пункта первого задания

3. Отобразите данные по индексу WGI за 2022 год в виде горизонтального столбчатого графика (rank).

Для отображения данных воспользуемся графиком, который можно построить с помощью библиотеки matplotlib.

Для начала нужно отсортировать исходный датафрейм по нужному столбцу, что показано на рисунке 4.

```
# подключаем библиотеку для построение графиков
import matplotlib.pyplot as plt

# указываем, чтобы можно построить график в данной тетради
%matplotlib inline
```

```
# сортируем исходный датафрейм по значению Rank за 2022 год и записываем в новую переменную
df_sorted_rank = df.sort_values('Rank.23', ascending=1)
```

Рисунок 4 – Подключение необходимой библиотеки и сортировка датафрейма

Дальше выделяем необходимые данные в отдельные переменные, для дальнейшей обработке, что показано на рисунке 5.

```
# выделяем значения по y
offset = np.array(df_sorted_rank['Country/Territory'])

# выделяем значения по x
height = np.array(df_sorted_rank['Rank.23'])
```

Рисунок 5 – Выделение значений по y и x

После чего настраиваем график, что показано на рисунке 6.

```
# задаем график с определенными размерами
plt.figure("1", figsize=[10,60])

# выставляем отступы графика внутри заданной фигуры
plt.margins(0.2,0.001)

# задаем название для графика
plt.title("Rank за 2022 год")

# задаем видимость вертикальной сетки
plt.grid(True, axis="x")

# задаем сам график указывая его тип (столбчатый) и значения
barh = plt.barh(offset, height, color='red')

# выделяем подграфик
ax = plt.subplot(1,1,1)

# задаем лейблы для каждого столбца, которые показывают его значения
ax.bar_label(barh, padding=10)

# показываем график
plt.show()
```

Рисунок 6 – Настройка графика

На рисунках 7-9 показан полученный график.

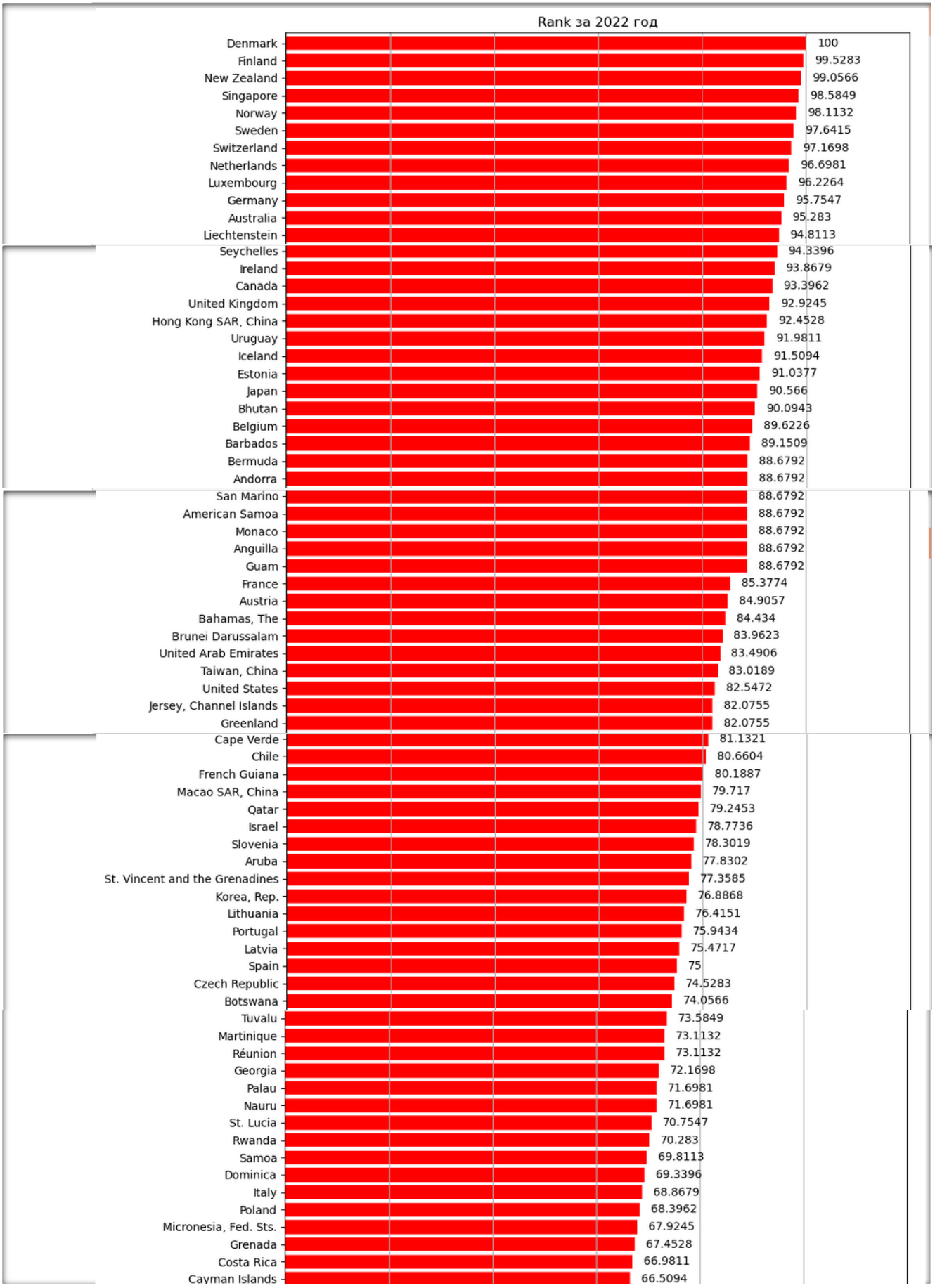

Рисунок 7 – Первая часть столбчатого графика (rank) по индексу WGI за 2022 год

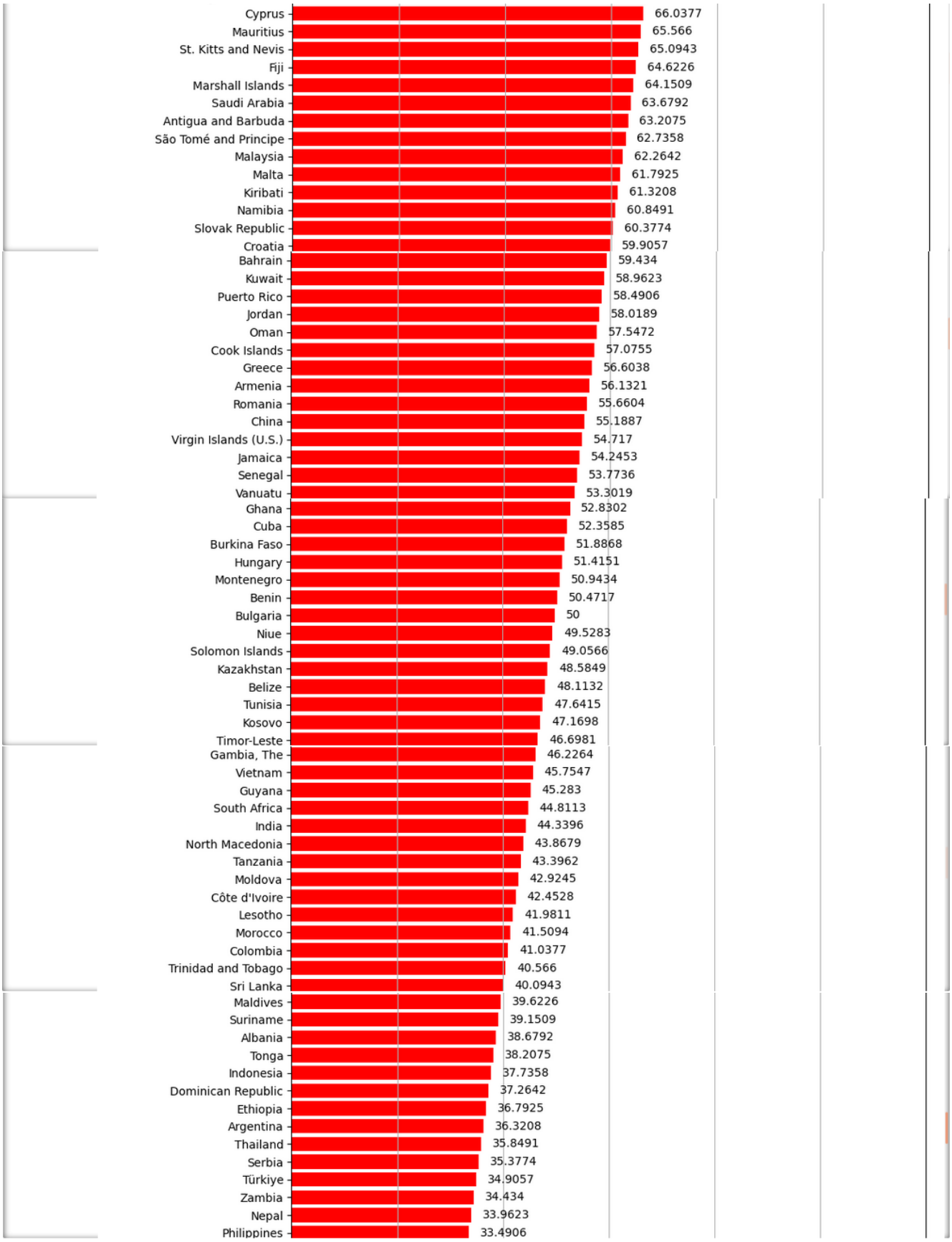

Рисунок 8 – Вторая часть столбчатого графика (rank) по индексу WGI за 2022 год

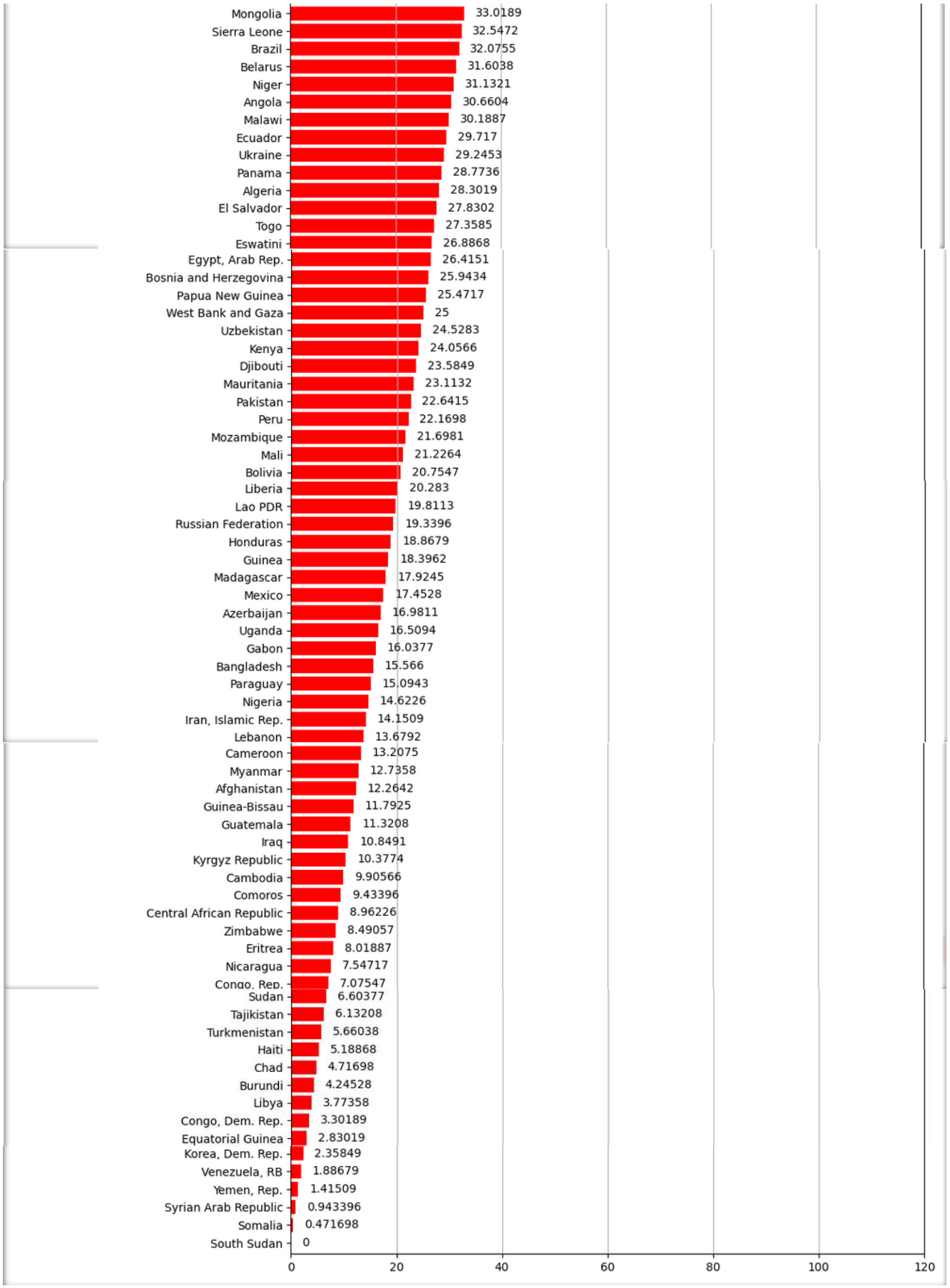

Рисунок 9 – Третья часть столбчатого графика (rank) по индексу WGI за 2022 год

4. Сформируйте DataFrame из исходного для региона в соответствии с Вашим вариантом.

Вариант 1. Asia Pacific.

В этом пункте необходимо сначала сделать DataFrame регионов из исходной таблицы регионов, после чего объединить с основным DataFrame и отсортировать, что показано на рисунке 10.

```
# с помощью библиотеки pandas формируем датафрейм, где указываем -
# название файла и номер строки с заголовками
df_regions = pd.read_excel("regions.xlsx",
                           header=0)
```

```
# объединяем два датафрейма по коду региона
df_merge = pd.merge(df, df_regions, on=['Code'])
```

```
# сделаем фильтрация по региону, код которой "AP"
# проходим по каждой строке и смотрим, чтобы в столбце "Region" ыыло значение "AP"
df_merge = df_merge[df_merge["Region"].apply(lambda item: item in 'AP')]
```

Рисунок 10 – Формирование датафрейма для региона AP

5. Выведите данные DataFrame'a.

Сформированный DataFrame показан на рисунке 11.

	Country/Territory	Code	Estimate	StdErr	NumSrc	Rank	Lower	Upper	Estimate.1	StdErr.1	...	Lower.22	Upper.22	Estimate.23	Std
0	Afghanistan	AFG	-1.291705	0.340507	2.0	4.301075	0.000000	27.419355	-1.176012	0.324013	...	4.761905	20.952381	-1.183776	0.1
6	Australia	AUS	1.877356	0.210325	6.0	93.548386	90.322578	99.462364	1.798130	0.188628	...	89.047623	96.666664	1.764490	0.1
13	Bangladesh	BGD	-0.969682	0.262077	4.0	17.741936	2.688172	34.946236	-0.773011	0.227055	...	10.952381	26.190475	-1.075527	0.1
22	Bhutan	BTN	0.942838	0.340507	2.0	81.182793	66.129036	90.322578	0.883641	0.324013	...	84.285713	95.714287	1.514259	0.2
29	China	CHN	-0.271190	0.188622	7.0	48.387096	32.258064	58.602150	-0.353955	0.172851	...	48.095238	62.380951	0.015970	0.1
53	Fiji	FJI	0.659303	0.340507	2.0	73.655914	59.677418	84.408600	0.663798	0.324013	...	54.285713	80.476189	0.388570	0.2
67	Hong Kong SAR, China	HKG	1.444894	0.204951	6.0	89.784943	81.182793	93.010750	1.303382	0.186292	...	89.047623	96.190475	1.610646	0.1
72	Indonesia	IDN	-0.864106	0.188622	7.0	22.043011	8.602151	34.408604	-1.160077	0.172851	...	27.619047	48.095238	-0.434911	0.1
73	India	IND	-0.381090	0.188622	7.0	43.010754	29.569893	53.763439	-0.258727	0.172851	...	31.904762	50.476189	-0.321811	0.1
82	Japan	JPN	1.192312	0.188622	7.0	84.408600	80.107529	90.322578	1.065931	0.172851	...	85.714287	95.714287	1.539972	0.1
86	Cambodia	KHM	-1.019842	0.275614	3.0	16.129032	2.150538	32.795700	-0.988312	0.232248	...	4.761905	18.571428	-1.241499	0.1
87	Korea, Rep.	KOR	0.382197	0.188622	7.0	65.591400	59.139786	74.731186	0.297208	0.172851	...	66.666664	80.952377	0.748462	0.1
89	Lao PDR	LAO	-0.722834	0.340507	2.0	28.494623	4.838710	50.537636	-0.689513	0.324013	...	6.190476	25.714285	-0.965779	0.1
94	Sri Lanka	LKA	-0.056539	0.262077	4.0	54.301075	37.096775	65.053764	-0.083365	0.227055	...	29.047619	50.476189	-0.383287	0.1
103	Maldives	MDV	-0.322941	0.340507	2.0	46.774193	19.354839	62.903225	-0.288015	0.324013	...	22.380953	58.095238	-0.401501	0.2
108	Myanmar	MMR	-1.500767	0.262077	4.0	1.612903	0.000000	14.516129	-1.334471	0.227055	...	6.666667	24.285715	-1.151623	0.1
109	Mongolia	MNG	0.111758	0.315914	3.0	60.752689	41.397850	73.118279	0.110406	0.310343	...	23.333334	46.190475	-0.560807	0.1
115	Malaysia	MYS	0.383065	0.188622	7.0	66.129036	59.139786	74.731186	0.363078	0.172851	...	50.000000	63.809525	0.249184	0.1
122	Nepal	NPL	-0.639209	0.340507	2.0	31.720430	7.526882	53.763439	-0.595586	0.324013	...	22.380953	46.190475	-0.532641	0.1
123	New Zealand	NZL	2.110246	0.210325	6.0	97.849464	93.010750	100.000000	2.148453	0.188628	...	96.190475	100.000000	2.163334	0.1
125	Pakistan	PAK	-1.220030	0.262077	4.0	7.526882	0.000000	25.268818	-1.080915	0.227055	...	14.761905	32.380951	-0.804033	0.1
128	Philippines	PHL	-0.358872	0.188622	7.0	45.698925	30.645161	54.301075	-0.381950	0.172851	...	24.285715	46.190475	-0.540948	0.1
129	Papua New Guinea	PNG	-0.433467	0.262077	4.0	40.322582	22.043011	57.526882	-0.698878	0.227055	...	14.761905	34.761906	-0.696246	0.1
131	Korea, Dem. Rep.	PRK	-1.284347	0.315914	3.0	4.838710	0.000000	25.806452	-1.248695	0.310343	...	0.000000	8.095238	-1.606099	0.2
140	Singapore	SGP	2.107434	0.188622	7.0	97.311829	93.010750	100.000000	2.057286	0.172851	...	96.190475	100.000000	2.094724	0.1
141	Solomon Islands	SLB	0.340782	0.439480	1.0	65.053764	43.010754	81.182793	0.361923	0.424762	...	31.904762	61.428570	-0.184422	0.2
155	Thailand	THA	-0.361192	0.188622	7.0	45.161289	30.645161	54.301075	-0.141036	0.172851	...	26.190475	48.095238	-0.452194	0.1
161	Taiwan, China	TWN	0.580821	0.188622	7.0	73.118279	62.903225	80.107529	0.583798	0.172851	...	80.000000	89.523811	1.148139	0.1
170	Vietnam	VNM	-0.489799	0.212363	6.0	37.096775	24.731182	52.150539	-0.491154	0.180076	...	31.904762	51.904762	-0.287283	0.1
171	Vanuatu	VUT	0.216309	0.439480	1.0	62.365593	36.559139	80.645164	0.238910	0.424762	...	36.666668	62.857143	-0.028941	0.2

30 rows × 148 columns

Рисунок 11 – Сформированный DataFrame

6. Постройте графики индекса WGI за 1996-2022 для стран своего региона (estimate).

Для выполнения этого пункта сначала необходимо отсортировать датафрейм по Estimate и Country, поменять индекс на столбец Country, а также, для удобства построения графика были переименованы столбцы Estimate на значения годов, что показано на рисунке 12.

```
 # фильтруем по показателю estimate и стране
df_wgi = df_merge.filter(regex='Estimate|Country')

# устанавливаем новый индекс в виде названия страны
df_wgi = df_wgi.set_index('Country')

# удаляем столбец 'Country/Territory'
df_wgi.drop('Country/Territory', axis=1, inplace=True)

# переименуем значение колонок на года
current_year = 1996
new_columns = []
step = 0
for item in df_wgi.columns.values:
    new_columns.append(str(current_year))
    if step < 3:
        current_year += 2
        step = step + 1
    else:
        current_year += 1

# ставим переименованные названия колонок
df_wgi.set_axis(new_columns, axis = 1, inplace = True)
```

Рисунок 12 – Подготовка датафрейма для построения графика

Далее производим настройку для построение графика, что показано на рисунке 13 и отображаем его, что показано на рисунке 14.

```
# транспонируем датафрейм, чтобы сделать по оси y выставить значение Estimate, а по x - года
df_wgi.T.plot(color='red', grid=1, figsize=(25,15), title='1996-2022 AP estimate', marker='o', legend = False)

# выставляем значения по оси x
x_ticks = range(len(df_wgi.columns.values))

# выставляем подписи значений по оси x
x_labels = df_wgi.columns.values

# вносим изменения в график
plt.xticks(ticks=x_ticks, labels=x_labels)
```

Рисунок 13 – Настройка графика

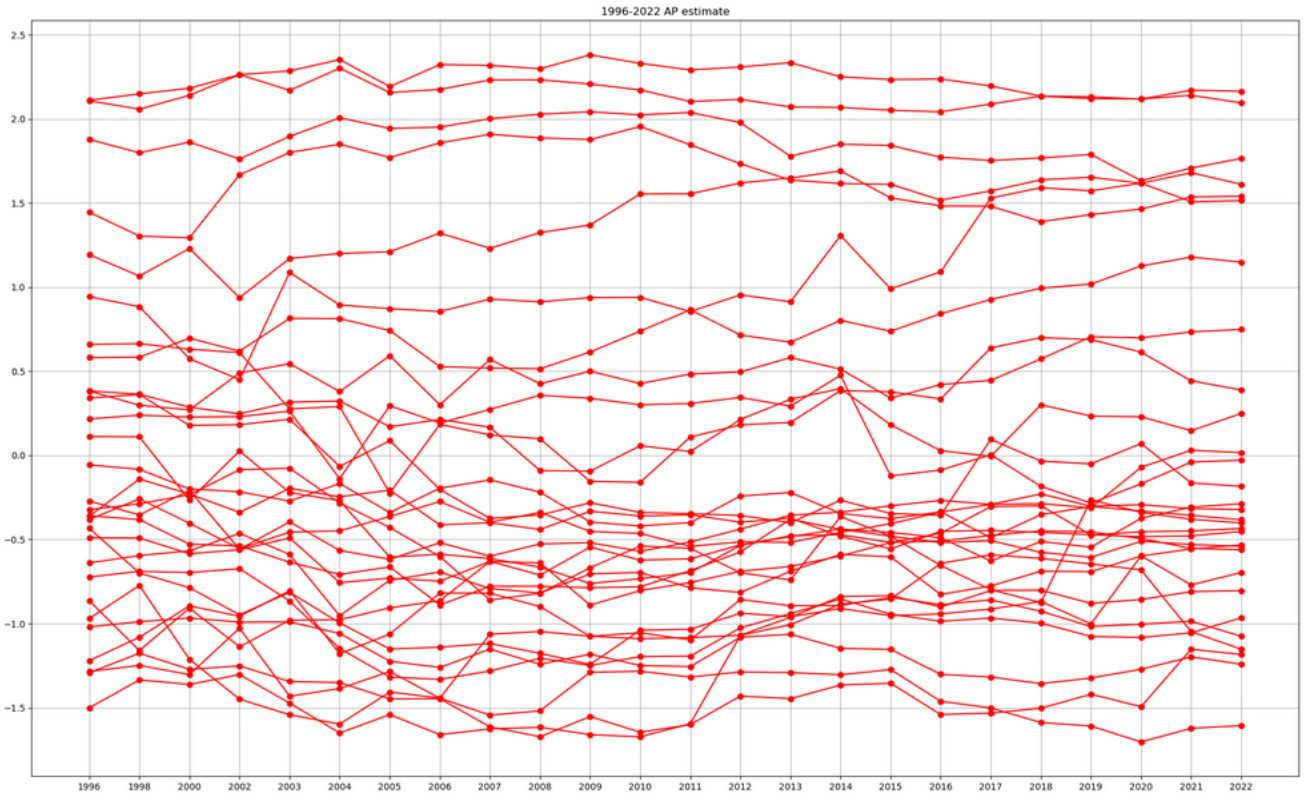

Рисунок 14 – Полученный график индекса WGI за 1996-2022

7. Найдите страны с наибольшим и наименьшим значением WGI региона за 2022 год (estimate).

Для нахождения стран с наибольшим и наименьшим значением WGI достаточно воспользоваться методами idxmax() и idxmin() соответственно, что показано на рисунке 15.

```
# находим наибольший WGI за 2022 год
max_wgi = df_wgi['2022'].idxmax()
max_wgi
```

```
'New Zealand'
```

```
# находим наименьший WGI за 2022 год
min_wgi = df_wgi['2022'].idxmin()
min_wgi
```

```
'Korea, North'
```

Рисунок 15 – Нахождение стран с минимальным и максимальным WGI

8. Определите средние значения региона за каждый год в период с 1996 по 2022 (estimate).

Для определения среднего необходимо пройтись по каждой колонке датафрейма и определить среднее число, что и показано на рисунке 16.

```
# находим средний WGI за 1996-2022 года
mean = df_wgi[df_wgi.columns.values].mean()
mean
```

```
1996    0.005390
1998    0.004409
2000   -0.035114
2002   -0.043584
2003   -0.024761
2004   -0.117873
2005   -0.141014
2006   -0.154418
2007   -0.139037
2008   -0.156720
2009   -0.154840
2010   -0.140798
2011   -0.131607
2012   -0.063054
2013   -0.046824
2014    0.012366
2015   -0.049325
2016   -0.073050
2017   -0.025685
2018   -0.012606
2019   -0.013031
2020    0.016584
2021    0.016733
2022    0.012814
dtype: float64
```

Рисунок 16 – Среднее значение WGI региона по годам

9. Постройте графики индекса WGI за 1996-2022 для стран своего региона и выделите страны с наибольшим и наименьшим значением WGI за 2022 год, а также отобразите среднее значение по региону и РФ.

Для выполнения этого задания, сначала необходимо транспонировать датафрейм, после чего производим настройку графика. Отображаем графики для всех стран, а дальше красим отдельными цветами необходимые, а именно:

1) красный – максимальный;
2) зеленый – РФ;
3) синий – минимальный;
4) желтый - средний.

Настройка графика показана на рисунке 17, а полученный график на рисунке 18.

```
# транспонируем датафрейм
df_wgi = df_wgi.T
```

```
# заного объединяем исходный датафрейм с датафреймом регионов для определения РФ
df_merge = pd.merge(df, df_regions, on=['Code'])

# строим предыдущий график
df_wgi.plot(color='grey', grid=1,figsize=(30,10), title='1996-2022 AP estimate', marker='o', legend=False)
x_ticks = range(len(df_wgi.columns.values))
x_labels = df_wgi.columns.values
plt.xticks(ticks=x_ticks, labels=x_labels)

# фильтруем объединенный датафрейм и находи РФ, и выделяем на графике отдельным цветом
Russia = df_merge
Russia.set_index('Country/Territory').T.get("Russian Federation").filter(regex='Estimate').plot(color = 'green', lege

# выделяем на графике отдельным цветом страну с минимальным WGI
df_wgi[min_wgi].plot(color='blue', legend=True, label="min WGI", marker='o')

# выделяем на графике отдельным цветом страну с максимальным WGI
df_wgi[max_wgi].plot(color='red', legend=True, label="max WGI", marker='o')

# выделяем на графике отдельным цветом страну с средний WGI
mean.plot(color='yellow', legend=True, label="mean WGI", marker='o')
```

Рисунок 17 – Настройка графика

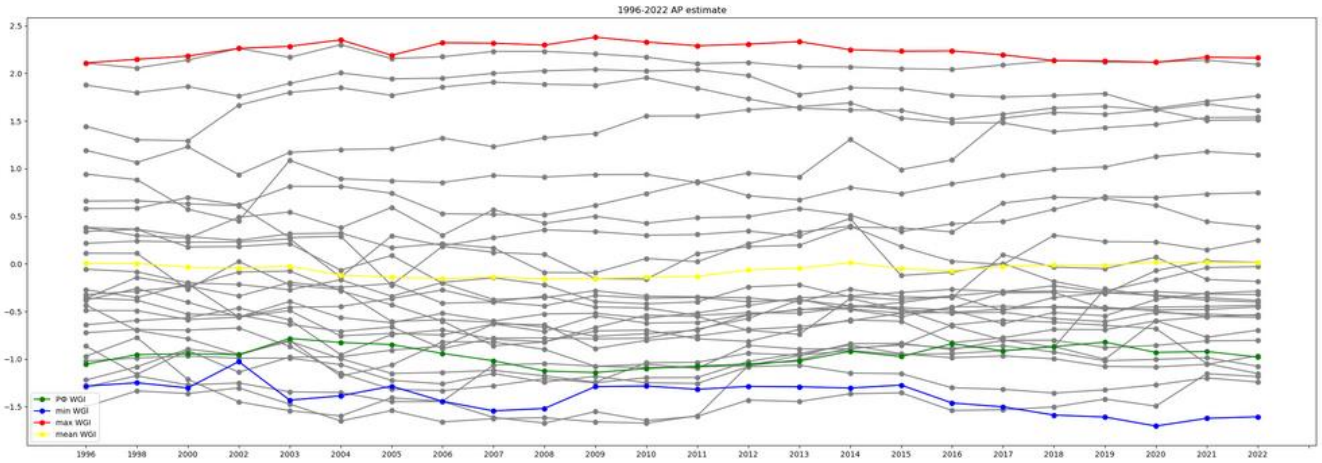

Рисунок 18 – Полученный график

11. Определите, как изменилось значение показателя rank с 1996 по 2022 (rank).

Для этого изначально необходимо заново составить датафрейм, так как в этом пункте идет смена варианта, а следовательно и необходимого региона, что показано на рисунке 19. На рисунке 20 показаны вычисление изменений значения показателя rank с 1996 по 2022 и полученный результат.

```
 # заного объединяем исходный датафрейм с датафреймом регионов для определения новой выборки по Америке для 2 вариант
df_merge = pd.merge(df, df_regions, on=['Code'])

# сделаем фильтрация по региону, код которой "AME"
# проходим по каждой строке и смотрим, чтобы в столбце "Region" ыыло значение "AME"
df_merge = df_merge[df_merge["Region"].apply(lambda item: item in 'AME')]
    # фильтруем по показателю rank и стране
df_rank = df_merge.filter(regex='Rank|Country')

# устанавливаем новый индекс в виде названия страны
df_rank = df_rank.set_index('Country')

# удаляем столбец 'Country/Territory'
df_rank.drop('Country/Territory', axis=1, inplace=True)
df_rank_host = df_rank
# переименуем значение колонок на года
current_year = 1996
new_columns = []
step = 0
for item in df_rank.columns.values:
    new_columns.append(str(current_year))
    if step < 3:
        current_year += 2
        step = step + 1
    else:
        current_year += 1

# ставим переименованные названия колонок
df_rank.set_axis(new_columns, axis = 1, inplace = True)
```

Рисунок 19 – Создание датафрейма для нового региона

```
# найдем изменение по разнице показателей rank за последний год и за первый
Difference = (df_rank['2022']-df_rank['1996'])
Difference
```

```
Country
Argentina                          -17.442684
Bahamas                              0.562996
Bolivia                             -4.514101
Brazil                             -24.913776
Barbados                            -1.709274
Canada                              -2.840332
Chile                               -9.662201
Colombia                             4.478596
Costa Rica                          -8.287682
Cuba                               -11.082371
Dominica                           -10.767906
Dominican Republic                  -4.133698
Ecuador                             -0.390545
Grenada                            -12.654701
Guatemala                          -12.335159
Guyana                              -7.405151
Honduras                             4.351794
Haiti                               -3.951106
Jamaica                             -7.582676
Saint Lucia                               NaN
Mexico                             -18.568678
Nicaragua                          -26.323798
Panama                             -21.764051
Peru                               -19.765671
Paraguay                             4.879286
El Salvador                          6.324812
Suriname                           -22.139378
Trinidad and Tobago                -40.079128
Uruguay                              9.723068
United States of America            -8.850677
Saint Vincent and the Grenadines          NaN
Venezuela                          -20.693852
dtype: float64
```

Рисунок 20 – Полученные изменения

12. Выведите таблицу для Вашего варианта (WGI - rank).

В этом пункте идет создание необходимой таблицы с вычислением необходимых показателей, что показано на рисунках 21-22. А полученная таблица показана на рисунке 23.

```
# заного объединяем исходный датафрейм с датафреймом регионов для определения новой выборки по Америке для 2 варианта
df_merge = pd.merge(df, df_regions, on=['Code'])
Russia = df_merge.copy()
# сделаем фильтрация по региону, код которой "AME"
# проходим по каждой строке и смотрим, чтобы в столбце "Region" было значение "AME"
df_merge = df_merge[df_merge["Region"].apply(lambda item: item in 'AME')]

df_merge.drop('Country/Territory', axis=1, inplace=True)
df_merge.set_index('Code')#выставление индекса
df_merge

#создание 0-го столбца таблицы
Rows = ['mean_2022', 'max_2022', 'min_2022', 'Russia_2022']
#создание 0-ой коллонки таблицы
Cols = ['Регион', 'Страна', 'WGI 1996', 'WGI 2022', 'Изменение']
Tabl_proc = pd.DataFrame(index=Rows,columns=Cols)

#создание первого столбка таблицы
Tabl_proc.loc['mean_2022', 'Регион'] = "AME"
Tabl_proc.loc['max_2022', 'Регион'] = "AME"
Tabl_proc.loc['min_2022', 'Регион'] = "AME"
Tabl_proc.loc['Russia_2022', 'Регион'] = "ECA"

#создание второго столбка таблицы
max_wgi = df_rank['2022'].idxmax()
min_wgi = df_rank['2022'].idxmin()
Tabl_proc.loc['mean_2022', 'Страна'] = ""
Tabl_proc.loc['max_2022', 'Страна'] = max_wgi
Tabl_proc.loc['min_2022', 'Страна'] = min_wgi
Tabl_proc.loc['Russia_2022', 'Страна'] = "Russian Federation"
```

Рисунок 21 – Первая часть создания таблицы

```
#получаем необходимые значения для третьего столбца
Tabl_proc.loc['mean_2022', 'WGI 1996'] = df_rank['1996'].mean()
max_wgi_count = df_rank.loc[max_wgi, '1996']
min_wgi_count = df_rank.loc[min_wgi, '1996']

#создание третьего столбца таблицы
Tabl_proc.loc['max_2022', 'WGI 1996'] = max_wgi_count
Tabl_proc.loc['min_2022', 'WGI 1996'] = min_wgi_count
Russia = Russia.set_index('Country/Territory').T.get("Russian Federation").filter(regex='Rank')
Tabl_proc.loc['Russia_2022', 'WGI 1996'] = Russia[0]

#получаем необходимые значения для четвертого столбца
Tabl_proc.loc['mean_2022', 'WGI 2022'] = df_rank['2022'].mean()
max_wgi_count = df_rank.loc[max_wgi, '2022']
min_wgi_count = df_rank.loc[min_wgi, '2022']

#создание четвертого столбца таблицы
Tabl_proc.loc['max_2022', 'WGI 2022'] = max_wgi_count
Tabl_proc.loc['min_2022', 'WGI 2022'] = min_wgi_count
Tabl_proc.loc['Russia_2022', 'WGI 2022'] = Russia[23]

#создание пятого столбца таблицы
#тут используем проценты, найденные выше
Tabl_proc.loc['mean_2022', 'Изменение'] = (df_rank['2022'].mean()-df_rank['1996'].mean()) # делаем среднее внутри, чт
Tabl_proc.loc['max_2022', 'Изменение'] = Difference.loc[max_wgi]
Tabl_proc.loc['min_2022', 'Изменение'] = Difference.loc[min_wgi]
Tabl_proc.loc['Russia_2022', 'Изменение'] = (Tabl_proc.loc['Russia_2022', 'WGI 2022'] - Tabl_proc.loc['Russia_2022',

Tabl_proc
```

Рисунок 22 – Вторая часть создания таблицы

	Регион	Страна	WGI 1996	WGI 2022	Изменение
mean_2022	AME		53.27957	45.59257	-7.686999
max_2022	AME	Canada	96.236557	93.396225	-2.840332
min_2022	AME	Venezuela	22.580645	1.886792	-20.693852
Russia_2022	ECA	Russian Federation	15.053763	19.339622	4.285859

Рисунок 23 –Полученная таблица

13. Отобразите диаграмму размаха (boxplot) индекса WGI за 2022 для всех стран и для каждого региона в отдельности (на одном графике) (estimate).

Для отображения этого графика, необходимо выделить каждый регион по отдельности и отсортировать его по необходимому значению, что и показано на рисунке 24.

```
# заного объединяем исходный датафрейм с датафреймом регионов для определения новой выборки по Америке для 2 варианта
merged_df = pd.merge(df, df_regions, on=['Code'])

# фильтруем по необходимому региону
#df_MENA
df_MENA = merged_df[merged_df.Region=='MENA']

#фильтруем по Estimate.23|Country, устанавливаем индекс и изменяем имя колонки
df_MENA_chart = df_MENA.filter(regex='Estimate.23|Country').set_index('Country').rename(columns={'Estimate.23': 'MENA
#удаляем столбец
df_MENA_chart = df_MENA_chart.drop('Country/Territory', axis=1, inplace=False)

# повторяем для других регионов
df_AP = merged_df[merged_df.Region=='AP']
df_AP_chart = df_AP.filter(regex='Estimate.23|Country').set_index('Country').rename(columns={'Estimate.23': 'AP'})
df_AP_chart = df_AP_chart.drop('Country/Territory', axis=1, inplace=False)
#df_ECA
df_ECA = merged_df[merged_df.Region=='ECA']
df_ECA_chart = df_ECA.filter(regex='Estimate.23|Country').set_index('Country').rename(columns={'Estimate.23': 'ECA'})
df_ECA_chart = df_ECA_chart.drop('Country/Territory', axis=1, inplace=False)
#df_SSA
df_SSA = merged_df[merged_df.Region=='SSA']
df_SSA_chart = df_SSA.filter(regex='Estimate.23|Country').set_index('Country').rename(columns={'Estimate.23': 'SSA'})
df_SSA_chart = df_SSA_chart.drop('Country/Territory', axis=1, inplace=False)
#df_AME
df_AME = merged_df[merged_df.Region=='AME']
df_AME_chart = df_AME.filter(regex='Estimate.23|Country').set_index('Country').rename(columns={'Estimate.23': 'AME'})
df_AME_chart = df_AME_chart.drop('Country/Territory', axis=1, inplace=False)
#df_WE_EU
df_WE_EU = merged_df[merged_df.Region=='WE/EU']
df_WE_EU_chart = df_WE_EU.filter(regex='Estimate.23|Country').set_index('Country').rename(columns={'Estimate.23': 'WE
df_WE_EU_chart = df_WE_EU_chart.drop('Country/Territory', axis=1, inplace=False)

plt.figure(figsize=(20,20))
df_chart_all_regions = df.filter(regex='Estimate.23|Country').set_index('Country/Territory').rename(columns={'Estimat
#объединение датафреймов в один
df_chart = pd.concat([df_MENA_chart, df_AP_chart, df_ECA_chart, df_SSA_chart, df_AME_chart, df_WE_EU_chart, df_chart_

df_chart.boxplot()#отображение
```

Рисунок 24 – Сортировка регионов и настройка графика

На рисунке 25 показан полученный график.

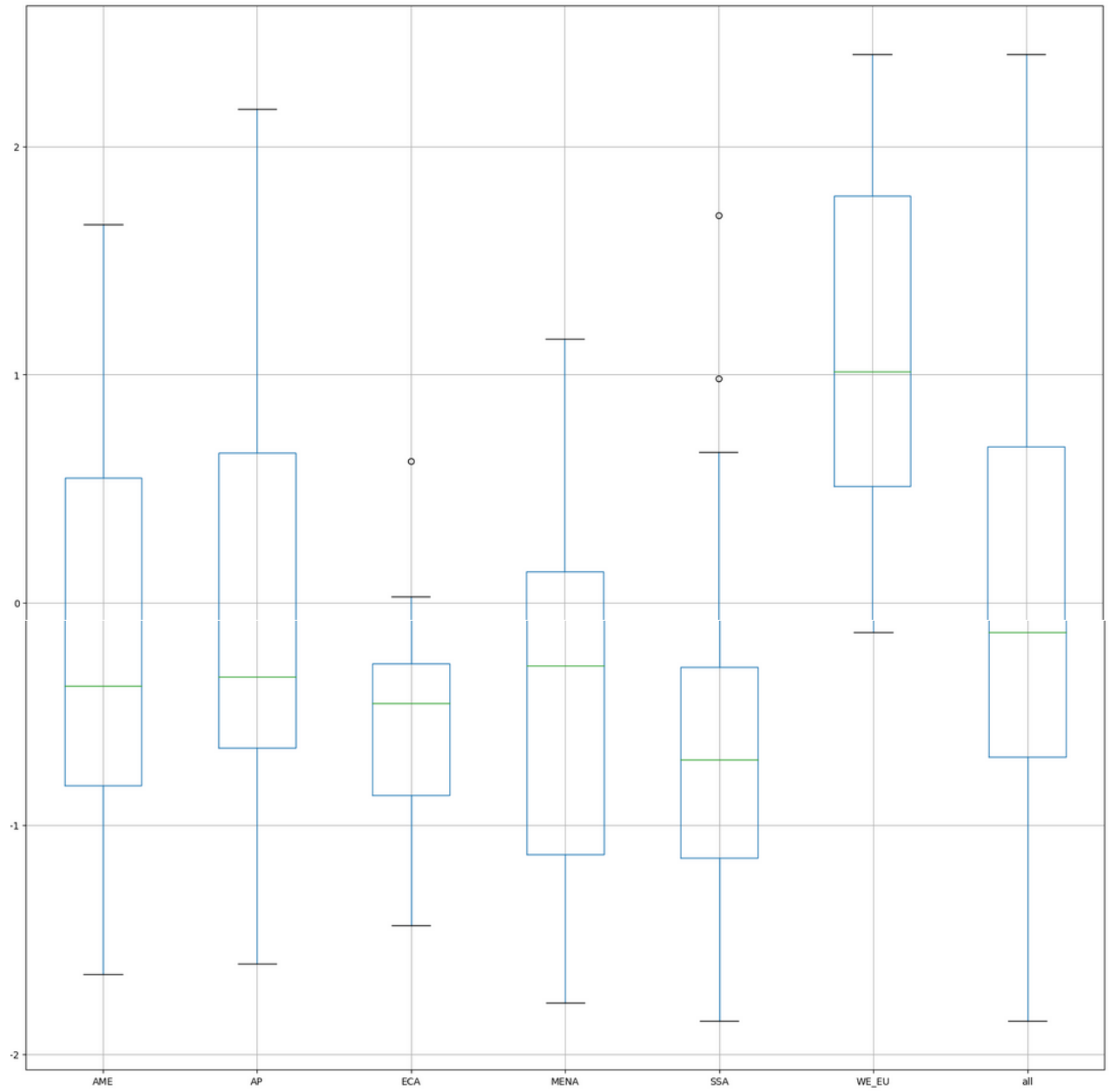

Рисунок 23 –Полученный график

Задача 2. Анализ рынка акций

1. Загрузите данные в один dataframe из всех файлов в папке /data/stock. Все файлы имеют одинаковую структуру, в том числе наименование столбцов. В качестве значений индекса dataframe'a необходимо указать значения столбца "Date". Название столбцов должные соответствовать названию акций (имя файла без .csv), а их значения - значениям цены закрытия (столбец "Close" в файлах .csv).

Выполнение:

Для загрузки данных необходимо преждевременно скачать таблицы. После чего был создан массив названий акций-таблиц и поочередно заполнена

результирующий датафрейм, что показано на рисунке 24. На рисунке 25 показан полученный датафрейм.

```
# делаем мписок имеющихся акций
cmls = ['AAPL','ABNB', 'ADBE', 'AMZN', 'CSCO',
        'DBX','EBAY', 'GOOGL', 'GTLB', 'HPQ',
       'INTC','META', 'MSFT', 'MU', 'NFLX',
        'NVDA', 'ORCL', 'PINS', 'SPOT',   # тут отсутствует один датафрейм так как там мешьше в два раза значений
        'TCOM', 'TSLA', 'TWLO', 'UBER', 'XIACY']

# делаем датафрейм из первого файла акций
df_1 = pd.read_csv("data/AAPL.csv",
                   header=0)

# создаем переменные размера результирующего датафрейма
cout_row = len(df_1)
cout_collums = len(cmls)

# создаем результирующий датафрейм необходимого размера и заполняем рандомными значениями
df_action = pd.DataFrame(data=np.random.randn(cout_row,cout_collums), columns=cmls)

# устанавливаем дату в качестве индекса
df_action = df_action.set_index(df_1['Date'])

# через цикл записываем неодходимые столбцы из данных акций в результирующий тадафрейм
for item in cmls:
    df = pd.read_csv("data/"+item+".csv",
                   header=0)
    df_action[item] = df['Close'].values
```

Рисунок 24 – Загрузка данных

Date	AAPL	ABNB	ADBE	AMZN	CSCO	DBX	EBAY	GOOGL	GTLB	HPQ	...	NFLX	NVDA	
2022-01-01	174.779999	153.970001	534.299988	149.573502	55.669998	24.750000	60.070000	135.303497	64.010002	36.730000	...	427.140015	244.860001	8
2022-02-01	165.119995	151.490005	467.679993	153.563004	55.770000	22.690001	54.590000	135.057007	58.270000	34.360001	...	394.519989	243.850006	7
2022-03-01	174.610001	171.759995	455.619995	162.997498	55.759998	23.250000	57.259998	139.067505	54.450001	36.299999	...	374.589996	272.859985	8
2022-04-01	157.649994	153.210007	395.950012	124.281502	48.980000	21.750000	51.919998	114.109497	47.930000	36.630001	...	190.360001	185.470001	7
2022-05-01	148.839996	120.870003	416.480011	120.209503	45.049999	20.840000	48.669998	113.762001	38.939999	38.840000	...	197.440002	186.720001	7
2022-06-01	136.720001	89.080002	366.059998	106.209999	42.639999	20.990000	41.669998	108.962997	53.139999	32.779999	...	174.869995	151.589996	6
2022-07-01	162.509995	110.980003	410.119995	134.949997	45.369999	22.740000	48.630001	116.320000	57.400002	33.389999	...	224.899994	181.630005	7
2022-08-01	157.220001	113.120003	373.440002	126.769997	44.720001	21.389999	44.130001	108.220001	59.869999	28.709999	...	223.559998	150.940002	7
2022-09-01	138.199997	105.040001	275.200012	113.000000	40.000000	20.719999	36.810001	95.650002	51.220001	24.920000	...	235.440002	121.389999	6
2022-10-01	153.339996	106.910004	318.500000	102.440002	45.430000	21.750000	39.840000	94.510002	48.459999	27.620001	...	291.880005	134.970001	7
2022-11-01	148.029999	102.139999	344.929993	96.540001	49.720001	23.559999	45.439999	100.989998	39.549999	30.040001	...	305.529999	169.229996	8
2022-12-01	129.929993	85.500000	336.529999	84.000000	47.639999	22.379999	41.470001	88.230003	45.439999	26.870001	...	294.880005	146.139999	8
2023-01-01	144.289993	111.110001	370.339996	103.129997	48.669998	23.230000	49.500000	98.839996	49.410000	29.139999	...	353.859985	195.369995	8
2023-02-01	147.410004	123.279999	323.950012	94.230003	48.419998	20.400000	45.900002	90.059998	44.040001	29.520000	...	322.130005	232.160004	8
2023-03-01	164.899994	124.400002	385.369995	103.290001	52.279999	21.620001	44.369999	103.730003	34.290001	29.350000	...	345.480011	277.769989	9
2023-04-01	169.679993	119.669998	377.559998	105.449997	47.250000	20.340000	46.430000	107.339996	30.360001	29.709999	...	329.929993	277.489990	9
2023-05-01	177.250000	109.769997	417.790009	120.580002	49.669998	23.020000	42.540001	122.870003	36.959999	29.059999	...	395.230011	378.339996	10
2023-06-01	193.970001	128.160004	488.989990	130.360001	51.740002	26.670000	44.689999	119.699997	51.110001	30.709999	...	440.489990	423.019989	11
2023-07-01	196.449997	152.190002	546.169983	133.679993	52.040001	26.950001	44.509998	132.720001	49.630001	32.830002	...	438.970001	467.290009	11
2023-08-01	187.869995	131.550003	559.340027	138.009995	57.349998	27.790001	44.779999	136.169998	47.369999	29.709999	...	433.679993	493.549988	12
2023-09-01	171.210007	137.210007	509.899994	127.120003	53.759998	27.230000	44.090000	130.860001	45.220001	25.700001	...	377.600006	434.989990	10
2023-10-01	170.770004	118.290001	532.059998	133.089996	52.130001	26.299999	39.230000	124.080002	43.279999	26.330000	...	411.690002	407.799988	10
2023-11-01	189.949997	126.339996	611.010010	146.089996	48.380001	28.180000	41.009998	132.529999	48.340000	29.340000	...	473.970001	467.700012	11
2023-12-01	192.529999	136.139999	596.599976	151.940002	50.520000	29.480000	43.619999	139.690002	62.959999	30.090000	...	486.880005	495.220001	10
2024-01-01	184.399994	144.139999	617.780029	155.199997	50.180000	31.680000	41.070000	140.100006	71.110001	28.709999	...	564.109985	615.270020	11
2024-02-01	180.750000	157.470001	560.280029	176.759995	48.369999	23.950001	47.279999	138.460007	72.120003	28.330000	...	602.919983	791.119995	11
2024-03-01	173.229996	166.669998	579.140015	175.389999	50.070000	23.840000	50.910000	138.500000	57.240002	30.500000	...	611.080017	919.130005	12
2024-03-12	173.229996	166.669998	579.140015	175.389999	50.070000	23.840000	50.910000	138.500000	57.240002	30.500000	...	611.080017	919.130005	12

28 rows × 24 columns

Рисунок 25 – Полученный датафрейм

2. Рассчитайте корреляционную матрицу для всех акций.

Получаем корреляционную матрицу и выводим полученный результат, что показано на рисунке 26.

```
# рассчитываем корреляционную матрицу
correlation_matrix = df_action.corr()
correlation_matrix
```

	AAPL	ABNB	ADBE	AMZN	CSCO	DBX	EBAY	GOOGL	GTLB	HPQ	...	NFLX	NVDA	ORCL	PINS
AAPL	1.000000	0.617430	0.833129	0.665715	0.589552	0.740429	0.115591	0.806847	0.282373	0.067074	...	0.701937	0.633114	0.769309	0.640294
ABNB	0.617430	1.000000	0.670509	0.830690	0.594365	0.332740	0.644140	0.780440	0.460602	0.390153	...	0.646901	0.649664	0.471504	0.554616
ADBE	0.833129	0.670509	1.000000	0.819614	0.554172	0.816359	0.180354	0.915440	0.496556	0.081518	...	0.821314	0.802739	0.785432	0.804657
AMZN	0.665715	0.830690	0.819614	1.000000	0.404820	0.478171	0.434078	0.912332	0.690644	0.235247	...	0.735466	0.765294	0.534556	0.666996
CSCO	0.589552	0.594365	0.554172	0.404820	1.000000	0.496982	0.494938	0.600025	0.068856	0.214262	...	0.497727	0.320159	0.463955	0.384233
DBX	0.740429	0.332740	0.816359	0.478171	0.496982	1.000000	-0.157363	0.669228	0.402517	-0.177013	...	0.635239	0.519374	0.667833	0.710191
EBAY	0.115591	0.644140	0.180354	0.434078	0.494938	-0.157363	1.000000	0.375794	0.251066	0.744560	...	0.138580	0.087027	-0.070414	-0.002757
GOOGL	0.806847	0.780440	0.915440	0.912332	0.600025	0.669228	0.375794	1.000000	0.535473	0.263251	...	0.717756	0.715287	0.618983	0.640675
GTLB	0.282373	0.460602	0.496556	0.690644	0.068856	0.402517	0.251066	0.535473	1.000000	0.094128	...	0.452625	0.404702	0.138574	0.525458
HPQ	0.067074	0.390153	0.081518	0.235247	0.214262	-0.177013	0.744560	0.263251	0.094128	1.000000	...	-0.203337	-0.160502	-0.260316	-0.285950
INTC	0.507251	0.738241	0.713875	0.816519	0.420854	0.390625	0.580047	0.826042	0.535441	0.591406	...	0.447049	0.458281	0.239485	0.452144
META	0.705358	0.723419	0.873388	0.830910	0.374998	0.552874	0.190361	0.808784	0.467641	-0.035611	...	0.897908	0.961389	0.821696	0.822643
MSFT	0.790691	0.679204	0.913842	0.838702	0.391476	0.648164	0.127010	0.845993	0.451366	-0.034581	...	0.900263	0.935386	0.847046	0.837576
MU	0.606787	0.842928	0.817961	0.906932	0.472688	0.440043	0.512637	0.867191	0.543109	0.308473	...	0.789551	0.796707	0.570765	0.717881
NFLX	0.701937	0.646901	0.821314	0.735466	0.497727	0.635239	0.138580	0.717756	0.452625	-0.203337	...	1.000000	0.910910	0.859397	0.930638
NVDA	0.633114	0.649664	0.802739	0.765294	0.320159	0.519374	0.087027	0.715287	0.404702	-0.160502	...	0.910910	1.000000	0.875089	0.815629
ORCL	0.769309	0.471504	0.785432	0.534556	0.463955	0.667833	-0.070414	0.618983	0.138574	-0.260316	...	0.859397	0.875089	1.000000	0.747754
PINS	0.640294	0.554616	0.804657	0.666996	0.384233	0.710191	-0.002757	0.640675	0.525458	-0.285950	...	0.930638	0.815629	0.747754	1.000000
SPOT	0.687415	0.753797	0.863827	0.875779	0.424007	0.525305	0.296858	0.821587	0.540113	0.005774	...	0.920771	0.925270	0.763100	0.842858
TCOM	0.439363	0.294269	0.533298	0.309545	0.257188	0.423136	-0.149330	0.322718	0.103614	-0.443806	...	0.766681	0.787859	0.836340	0.705551
TSLA	0.248385	0.353807	0.071508	0.302321	0.253808	0.037233	0.434899	0.326662	0.260908	0.568231	...	-0.251616	-0.277600	-0.310021	-0.253055
TWLO	0.042914	0.429915	0.067604	0.314869	0.383777	-0.113102	0.753732	0.315410	0.310273	0.728572	...	-0.102302	-0.244797	-0.393536	-0.141953
UBER	0.661323	0.680764	0.834611	0.796897	0.326346	0.595928	0.085736	0.737311	0.521399	-0.180970	...	0.937042	0.969790	0.832075	0.907751
XIACY	0.408747	0.564475	0.697612	0.654564	0.474311	0.382992	0.535223	0.680658	0.453669	0.378627	...	0.505430	0.445645	0.324511	0.524413

24 rows × 24 columns

Рисунок 26 – Полученная матрица корреляции

3. Отобразите корреляционную матрицу в виде диаграммы.

Для отображения графика подключим новую библиотеку, настроим отображение и покажем, что изображено на рисунке 27. Полученный график показан на рисунке 28.

```
# подключаем библиотеку Seaborn, которая используется для создания статистических графиков
import seaborn as sns

plt.figure(figsize=(20,20), dpi= 80)
sns.heatmap(correlation_matrix, xticklabels=correlation_matrix.columns, yticklabels=correlation_matrix.columns, cmap=

plt.title('Correlogram of action', fontsize=22)
plt.xticks(fontsize=12, rotation=60, horizontalalignment='right')
plt.yticks(fontsize=12, rotation=0, horizontalalignment='right')
plt.show()
```

Рисунок 27 – Настройка графика корреляционной матрицы

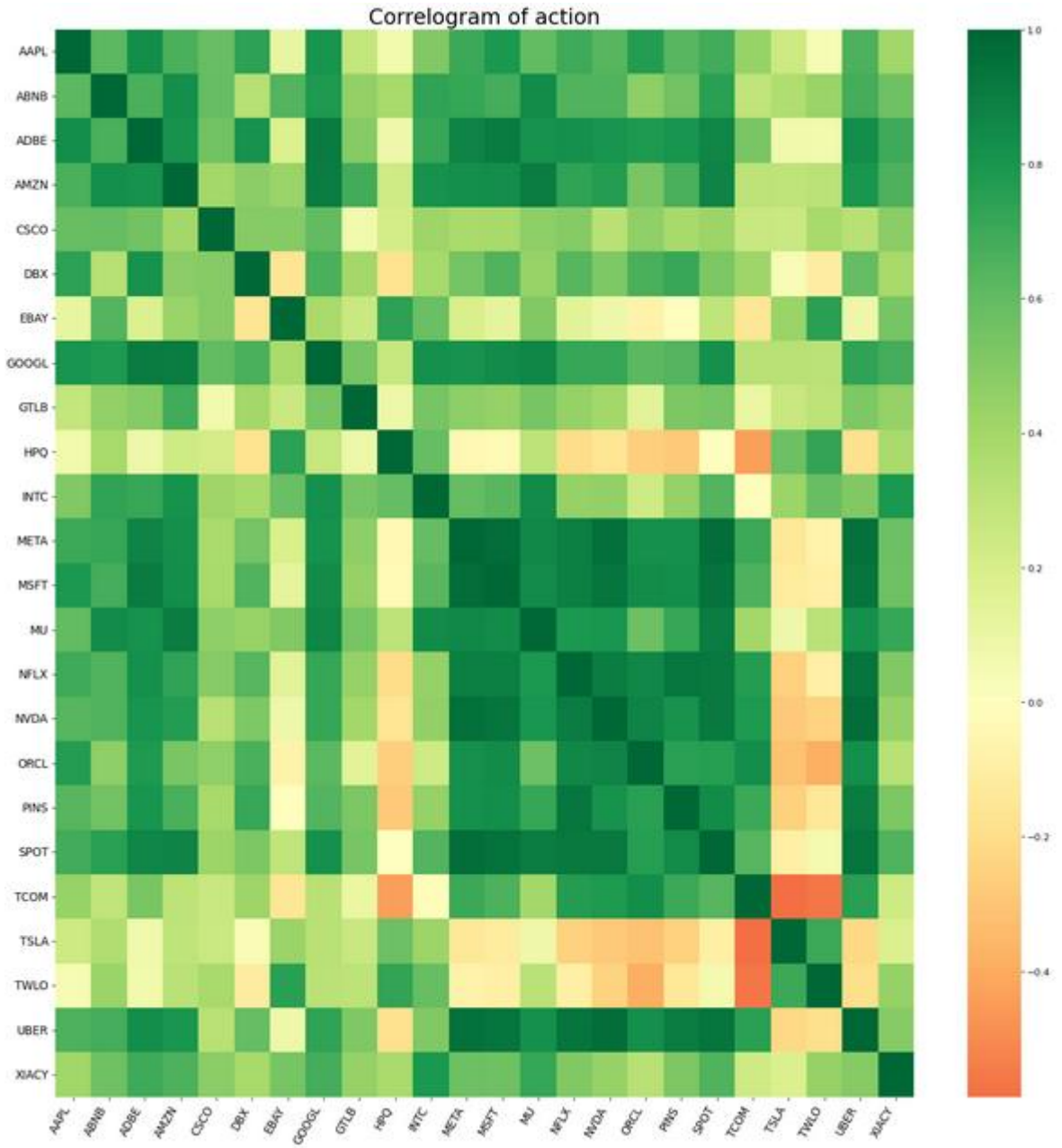

Рисунок 28 – График корреляционной матрицы

4. В соответствии Netflix (NFLX) определите:

- акцию с максимальной положительной корреляцией (max);

- акцию с максимальной отрицательной корреляцией (min);

- акцию с минимальной корреляцией (которая больше всего соответствует отсутствию какой-либо корреляции (none).

Для этого сделаем три переменных, которые будут хранить корреляционную матрицу, для удобства дальнейшей обработки. Далее Необходимо удалить столбец с акцией по варианту, чтобы не находило само себя. После чего станспонируем

матрицы и ищем максимальное, минимальное и нулевое, что показано на рисунках 29, 30 и 31 соответственно.

Акция с максимальной положительной корреляцией (max)

```
max_NFLX = df_action.corr()
min_NFLX = df_action.corr()
min_max = df_action.corr()

#удаление столбца NFLX, чтобы оно не было найденно максимальным
max_NFLX.drop(["NFLX"], axis=1, inplace=True)
min_NFLX = max_NFLX
```

```
max_NFLX = max_NFLX.T

# поиск максимального
max_NFLX = max_NFLX.filter(regex='NFLX').idxmax()
max_NFLX[0]
```

```
'UBER'
```

Рисунок 29 – Нахождение максимального

Акция с максимальной отрицательной корреляцией (min)

```
min_NFLX = min_NFLX.T

# поиск минимального
min_NFLX = min_NFLX.filter(regex='NFLX').idxmin()
min_NFLX[0]
```

```
'TSLA'
```

Рисунок 30 – Нахождение минимального

Акция с минимальной корреляцией (которая больше всего соответствует отсутствию какой-либо корреляции (none)

```
mass = min_max['NFLX'].values

# поиск числа наиболее приближенного к 0
max_to_zero = min(mass, key=lambda x: abs(x-0))

near = df_action.corr()
to_zero = near['NFLX']

# поиск акции с данным значением
to_zero = to_zero[to_zero.values == max_to_zero]
to_zero
```

```
TWLO   -0.102302
Name: NFLX, dtype: float64
```

Рисунок 31 – Нахождение нулевого

5. Постройте диаграммы разброса с компаниями, найденными в предыдущем пункте.

Полученный график с трема графиками разброса показан на рисунке 32.

```
NFLX = 'NFLX'

# отображение графика разброса с максимально отрицательной корреляцией
plt.scatter(df_action[NFLX], df_action[min_NFLX], label="NFLX-TSLA")

# отображение графика разброса с максимально положительной корреляцикй
plt.scatter(df_action[NFLX], df_action[max_NFLX], label="NFLX-UBER")

# отображение графика разброса с минимальной корреляцией
plt.scatter(df_action[NFLX], df_action[to_zero.keys()], label="NFLX-TWLO")
plt.legend()
plt.grid()
```

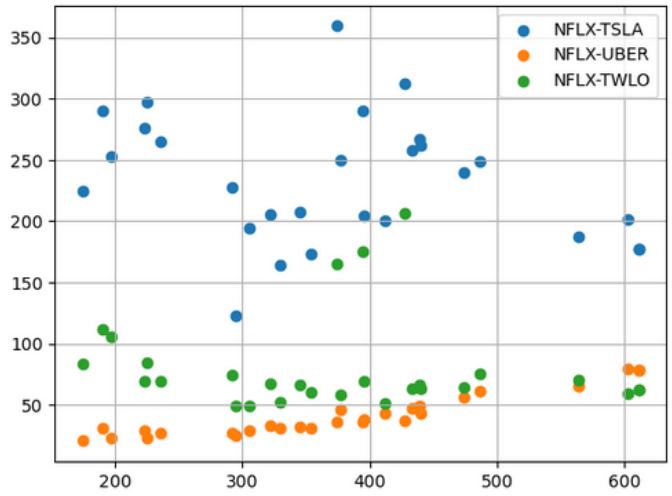

Рисунок 32 – График разброса

6. Рассчитайте среднюю цену акций для каждого месяца (исходные данные взяты с интервалом в месяц).

Для выполнения этого пункта, необходимо транспонировать датафрейм и найти средние, что и показано на рисунке 33.

```
# транспонируем, чтобы в столбце были значения по датам
df_action_T = df_action.T

# среднее значение за месяц всех акций
df_action_T.mean()
```

```
Date
2022-01-01    154.857167
2022-02-01    140.774723
2022-03-01    145.272287
2022-04-01    115.763514
2022-05-01    112.316034
2022-06-01     99.256929
2022-07-01    114.014999
2022-08-01    107.380833
2022-09-01     94.437083
2022-10-01     97.227501
2022-11-01    100.671666
2022-12-01     92.028958
2023-01-01    108.279540
2023-02-01    108.613126
2023-03-01    120.210832
2023-04-01    118.584166
2023-05-01    134.344584
2023-06-01    148.794582
2023-07-01    156.775000
2023-08-01    155.579584
2023-09-01    145.393333
2023-10-01    144.340623
2023-11-01    162.973751
2023-12-01    168.482916
2024-01-01    178.837501
2024-02-01    194.328293
2024-03-01    201.071667
2024-03-12    201.071667
dtype: float64
```

Рисунок 33 – Средняя цена акций по месяцам

7. Постройте графики для акций из пункта 4 и средней из пункта 6.

Для выполнение этого пункта делаем выборку по найденным данным в предыдущих пунктах и строим из них один график, что и показано на рисунке 34.

```
# отображение графика для среднего значения
df_action.T.mean().plot(grid=1, figsize=(25,7), color='blue', marker='o', label="Mean")

# отображение графика для акций с максимальным значения
df_action[max_NFLX].T.mean().plot(grid=1, figsize=(25,7), color='black', marker='o', label="UBER")

# отображение графика для акций с приближенным к 0 значения
df_action[to_zero.keys()].T.mean().plot(grid=1, figsize=(25,7), color='orange', marker='o', label="TWLO")

# отображение графика для акций с минимальным значения
df_action[min_NFLX].T.mean().plot(grid=1, figsize=(25,7), color='red', marker='o', label="TSLA")

# отображение графика для NFLX
df_action[NFLX].plot(grid=1, figsize=(25,7), color='green', marker='o', label="NFLX")

plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f3baf575eb0>
```

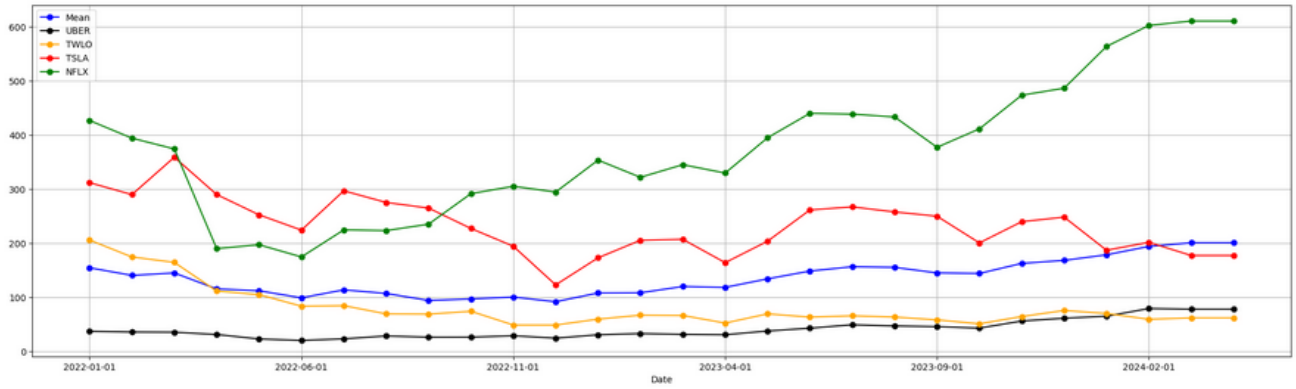

Рисунок 34 – Общий график акций из пунктов 4 и 6

Вывод:

В процессе выполнения домашнего задания 1 были основные принципы машинного обучения, изучены библиотеки pandas и matplotlib и их применение в практических задачах, а также знакомство с языком программирования python.