

**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**

**Методы машинного обучения**

**ОТЧЕТ**

**по самостоятельная работа № 1**

**Дескриптивный анализ данных**

**Студент**

ИУ6И-21М

(Группа)

\_\_\_\_\_  
(Подпись, дата)

Джабри А.Ш.

(И.О. Фамилия)

**Преподаватель**

\_\_\_\_\_  
(Подпись, дата)

Папулин С. Ю.

(И.О. Фамилия)

## 1- 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.

## 2 Платформа

### 2.1 Google Colaboratory

Google colaboratory, или более известный как Colab, - это бесплатная платформа, доступная из браузера, предоставляемая Google Research и позволяющая студентам и исследователям писать на python, имея при этом гибридные текстовые разделы и кодов. Google Colab можно рассматривать как инфраструктуру как услугу (IAAS), которая предоставляет своим пользователям аппаратные ресурсы в облаке, такие как хранилище, оперативная память и графический процессор. Это помогло нам преодолеть ограничения аппаратных возможностей наших машин.

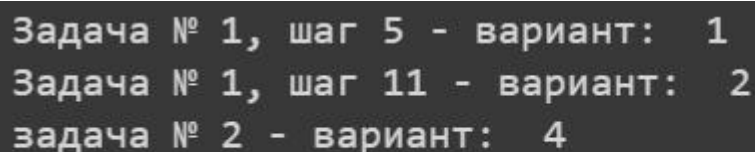
Преимуществом Colab является возможность делиться проектами между несколькими участниками без необходимости загружать или устанавливать программное обеспечение ; просто откройте проект в браузере и примите участие в его расширении.

### 2.2 Язык программирования и библиотеки

Поскольку язык, библиотеки и инструменты обсуждались на семинаре, мы не будем возвращаться к ним снова.

## 3 Выбирающий вариант

После запуска кода, приведенного в домашнем задании, мы получили следующий результат :



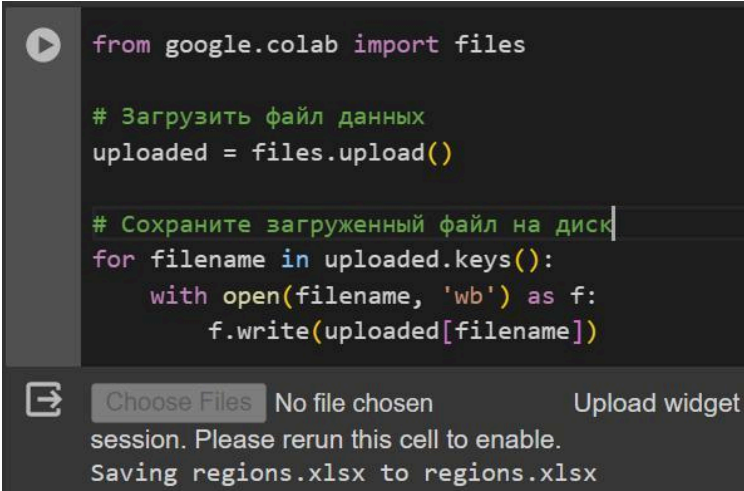
```
Задача № 1, шаг 5 - вариант: 1
Задача № 1, шаг 11 - вариант: 2
задача № 2 - вариант: 4
```

Рисунок 1 - выберите вариант.

## 4 Задание 1. Анализ индикаторов качества государственного управления

### 4.1 Загрузите данные в DataFrame

Для этого у нас есть два метода: первый - использовать кнопку загрузки, предоставляемую Google collab, или написать код на python для загрузки набора данных с хост-компьютера.



```
from google.colab import files

# Загрузить файл данных
uploaded = files.upload()

# Сохраните загруженный файл на диск
for filename in uploaded.keys():
    with open(filename, 'wb') as f:
        f.write(uploaded[filename])
```

Choose Files No file chosen Upload widget  
session. Please rerun this cell to enable.  
Saving regions.xlsx to regions.xlsx

Рисунок 2 - Загрузка набора данных.

В библиотеке pandas есть функции `read_excel()`, которые мы использовали для загрузки данных, но нам пришлось выполнить некоторые манипуляции с данными, чтобы привести наши данные в форму ранее использованных данных, поэтому мы не меняем код во всех вопросах.

```

# Теперь мы можем загрузить данные в DataFrame
import pandas as pd

# загруженный файл в формате Excel (xlsx)
regions = pd.read_excel('/content/regions.xlsx')
# загруженный файл имеет формат stata
wgidataset = pd.read_stata('/content/wgidataset.dta')
df = pd.read_excel("/content/wgidataset.xlsx", 'ControlofCorruption', skiprows=14,)
# Split the DataFrame
part1 = df.iloc[:, :2] # First two columns
part2 = df.iloc[:, 2:] # Rest of the columns

column_groups = [part2.columns[i:i+6] for i in range(0, len(part2.columns), 6)]
separate_datasets = []
for group in column_groups:
    separate_datasets.append(df[group])
# Initialize an empty list to store concatenated datasets
concatenated_datasets = []

# Concatenate part1 with each dataset separately
for dataset in separate_datasets:
    concatenated_dataset = pd.concat([part1, dataset], axis=1)
    concatenated_datasets.append(concatenated_dataset)
new_column_names = ['Estimate', 'StdErr', 'NumSrc', 'Rank', 'Lower', 'Upper']
# Loop through each dataset
for dataset in concatenated_datasets:
    # Identify the last 5 columns
    dataset.columns = list(dataset.columns[:-6]) + new_column_names

```

Рисунок 3 - Загрузка данных в DataFrame.

Теперь давайте посмотрим на наш набор данных в виде фреймов данных. Поскольку таблицы настолько велики, мы не можем увидеть здесь всю таблицу целиком, поэтому мы отобразим в отчете только фрагменты из таблиц, и их можно наблюдать при запуске кода, прилагаемого к отчет.

	Country/Territory	Code	Estimate	StdErr	NumSrc	Rank	Lower	Upper	Year
0	Aruba	ABW	NaN	NaN	NaN	NaN	NaN	NaN	1996
1	Andorra	ADO	1.318143	0.480889	1.0	87.096771	72.043015	96.774193	1996
2	Afghanistan	AFG	-1.291705	0.340507	2.0	4.301075	0.000000	27.419355	1996
3	Angola	AGO	-1.167702	0.262077	4.0	9.677420	0.537634	27.419355	1996
4	Anguilla	AIA	NaN	NaN	NaN	NaN	NaN	NaN	1996

Рисунок 4 - часть wgidataset в виде Dataframe.

	Country	Code	Region
0	Afghanistan	AFG	AP
1	Albania	ALB	ECA
2	Algeria	DZA	MENA
3	Angola	AGO	SSA
4	Argentina	ARG	AME

Рисунок 5 - часть region в виде Dataframe.

## 4.2 Отсортируйте данные по убыванию индекса Data Frame

Мы отсортировали набор данных, используя функцию `sort_index()`, как показано на следующем рисунке.

```
# Отсортируйте данные по показателю индекса WGI в порядке возрастания, чтобы получить ранги
sorted_dataset = Final_dataset.sort_values(by='Rank', ascending=False)
sorted_dataset.head()
```

	Country/Territory	Code	Estimate	StdErr	NumSrc	Rank	Lower	Upper	Year
4548	Denmark	DNK	2.236469	0.149797	10.0	100.0	95.714287	100.0	2020
1980	Denmark	DNK	2.376409	0.182742	8.0	100.0	96.116508	100.0	2008
2194	Denmark	DNK	2.435494	0.167893	9.0	100.0	97.607658	100.0	2009
2408	Denmark	DNK	2.352359	0.153291	10.0	100.0	97.619049	100.0	2010
4762	Denmark	DNK	2.333753	0.162209	10.0	100.0	97.142860	100.0	2021

Рисунок 6 - Сортировка набора данных.

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

После фильтрации набора данных, чтобы остались только данные за 2021 год, мы строим график в виде горизонтальной гистограммы, используя функции, предоставляемые библиотекой `matplotlib`.

```
import matplotlib.pyplot as plt

# Отфильтруйте набор данных, чтобы включить в него данные только за 2021 год
df_2021 = sorted_dataset[sorted_dataset['Year'] == 2021]

# Построение
plt.figure(figsize=(20, 50))
plt.barh(df_2021['Country/Territory'], df_2021['Rank'], color='skyblue')
plt.xlabel('WGI Index Score')
plt.ylabel('Country')
plt.title('WGI Index Scores for 2021')
plt.gca().invert_yaxis() # Инвертируйте ось y, чтобы получить самый высокий ранг сверху
plt.show()
```

Рисунок 8 - построение горизонтальной гистограммы.

Результат неясен из-за размера имеющегося у нас набора данных, и его можно четко наблюдать после запуска кода.

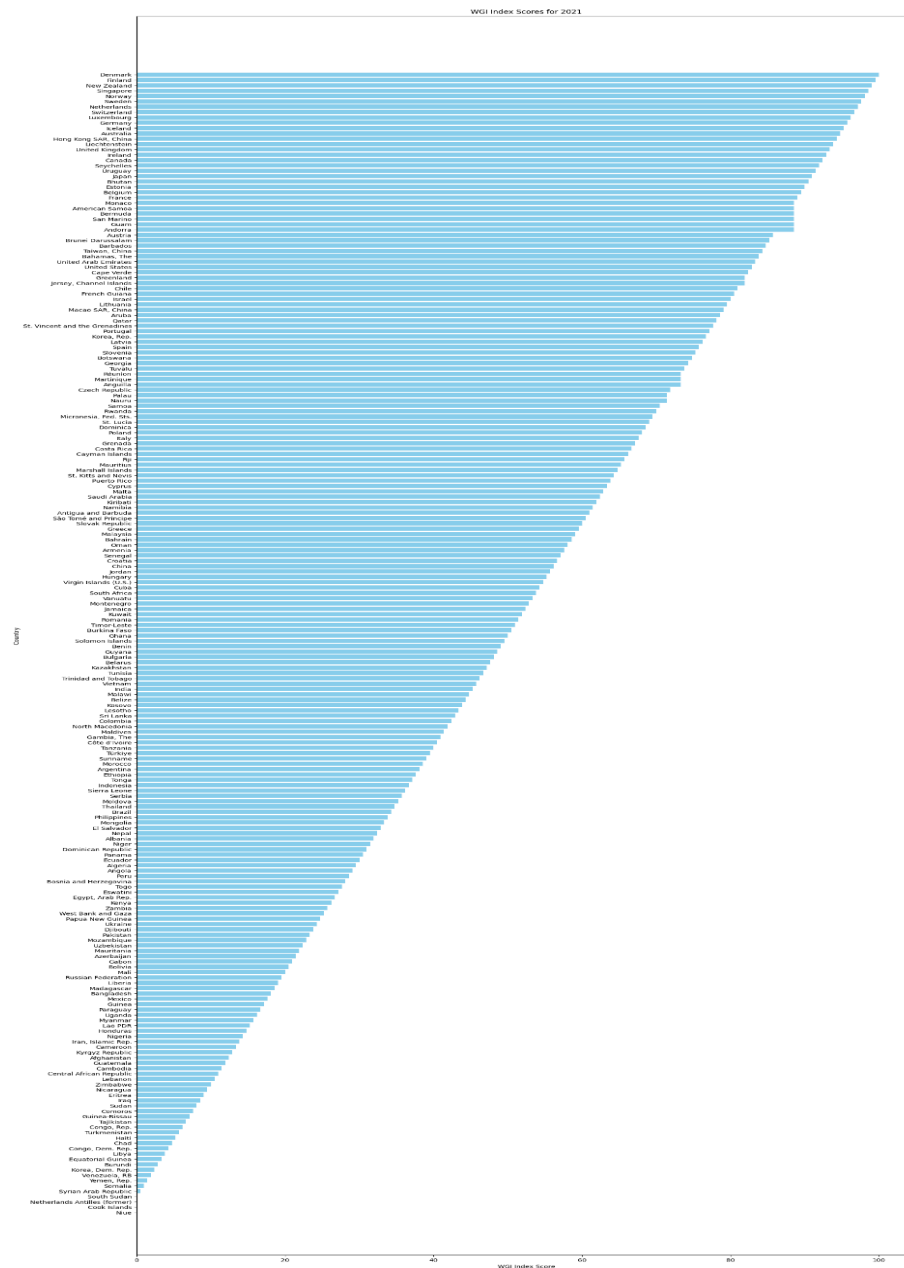


Рисунок 9 - горизонтальной гистограммы.

## 4.4 Создание базы данных для Азиатско-Тихоокеанского региона

Чтобы выполнить эту операцию, мы выполняем следующие действия:

1. Мы объединили два набора данных, используя код столбца
2. Затем мы отфильтровали данные, используя столбец Region

```
# 1 Отфильтруйте фрейм данных, чтобы сохранить только те строки,
# в которых значение в 'column_name' соответствует выбранному значению
merged_df = pd.merge(regions, Final_dataset, on='Code')
df_ap_region = merged_df[merged_df['Region'] == 'AP']
# # Теперь наш набор данных содержит только строки из набора данных wgi,
# которые относятся только к странам Азиатско-Тихоокеанского региона
df_ap_region.head()
```

	Country	Code	Region	Country/Territory	Estimate	StdErr	NumSrc	Rank	Lower	Upper	Year
0	Afghanistan	AFG	AP	Afghanistan	-1.291705	0.340507	2.0	4.301075	0.0	27.419355	1996
1	Afghanistan	AFG	AP	Afghanistan	-1.176012	0.324013	2.0	8.021390	0.0	33.689838	1998
2	Afghanistan	AFG	AP	Afghanistan	-1.271724	0.346906	2.0	4.787234	0.0	30.851065	2000
3	Afghanistan	AFG	AP	Afghanistan	-1.251137	0.352838	2.0	4.761905	0.0	32.804234	2002
4	Afghanistan	AFG	AP	Afghanistan	-1.344180	0.270215	3.0	4.761905	0.0	19.047619	2003

Рисунок 11- Набор данных для стран Азиатско-Тихоокеанского региона.

## 4.6 Постройте графики индекса WGI за 1996-2021 годы для стран Азиатско-Тихоокеанского региона (estimate)

Создайте базу данных за период с 1996 по 2021 год и только для стран Азиатско-Тихоокеанского региона.

```
# создать DF с 1996 по 2021 год
df_1996_2021 = df_ap_region[["countryname", "year", "pve"]]
df_1996_2021 = df_1996_2021[df_1996_2021["year"] != 2022]
df_1996_2021
```

Рисунок 12 - Создайте фрейм данных с 1996 по 2021 год.

Нам пришлось немного подкорректировать наши данные, представив каждый год в виде строки, а столбец - в виде оценки по стране.

```
# Pivot the data frame
pivoted_df = df_1996_2021.pivot_table(columns='Country/Territory', index='Year', values='Estimate')

# Rename the columns
pivoted_df.columns = [f'Estimate.{col}' for col in pivoted_df.columns]

pivoted_df.head()
```

Рисунок 12 - Подготовьте фрейм данных для построения графика.

```
pivoted_df.plot(grid=1,figsize=(25,10),title='WGI за 1996-2022 Asia Pacific estimate',marker='o',color='lightgrey')
```

Рисунок 12 -Постройте графики индекса WGI за 1996-2022.

И результат построения графика показан на рисунке ниже.

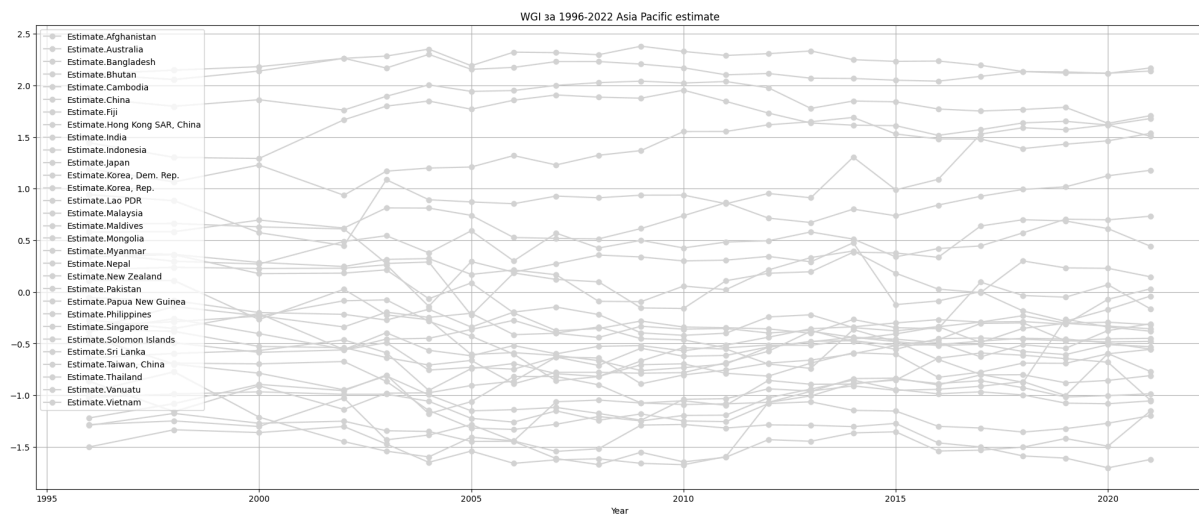


Рисунок 13 - Постройте графики индекса WGI за 1996-2021 годы для стран Азиатско-Тихоокеанского региона (оценка).

#### 4.7 Поиск стран с самыми высокими и самыми низкими значениями WGI для стран Азиатско-Тихоокеанского региона на 2022 год

Мы использовали функции `idmin()` и `idmax()` для вычисления максимума и минимума.

```
# Сохраняйте только строки с 2021 годом
df_2021 = df_1996_2021[df_1996_2021['year'] == 2021]
# Найдите страну с максимальным PVE
max_pve_country = df_2021.loc[df_2021['pve'].idxmax()]['countryname']
# Найдите страну с минимальным количеством PVE
min_pve_country = df_2021.loc[df_2021['pve'].idxmin()]['countryname']
```

Рисунок 14 - Ищите максимум и минимум

И результаты были следующими

```
#страна с максимальным PVE
max_pve_country

'New Zealand'

#страна с минимальным PVE
min_pve_country

'Korea, Dem. Rep.'
```

Рисунок 15 - Страны с самыми высокими и самыми низкими значениями WGI в варианте "Азиатско-тихоокеанский регион" на 2021 год



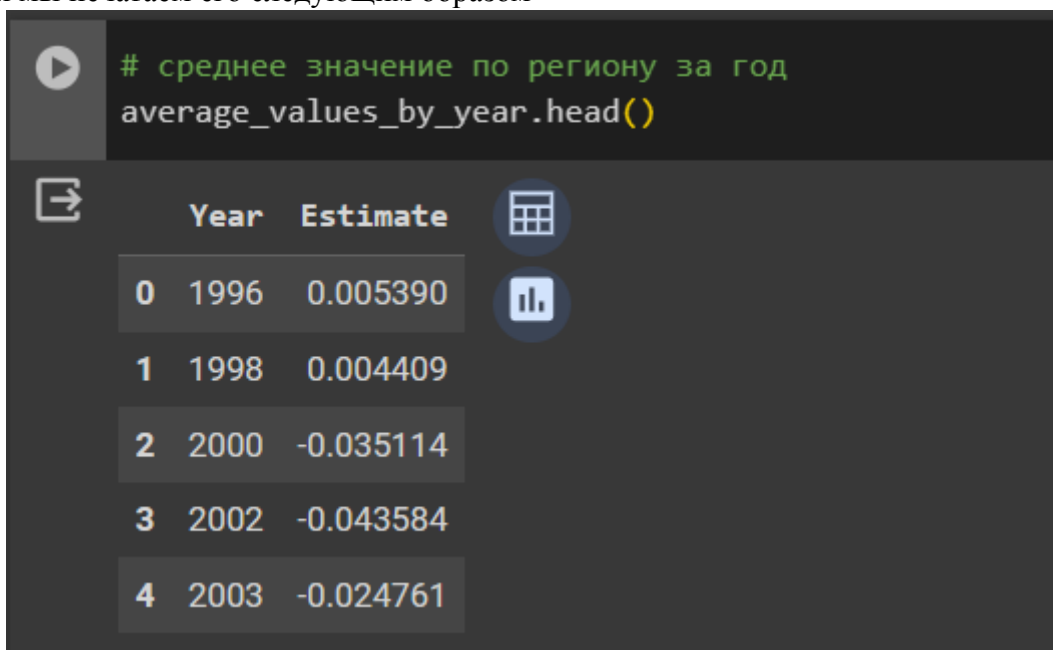
## 4.8 Средние значения по региону за каждый год с 1996 по 2021 год (estimate)

Сгруппируйте данные по годам и вычислите среднее значение для каждого года, и мы использовали функцию *mean()*.

```
[16] # Сгруппируйте данные по годам и рассчитайте среднее значение для каждого года
average_values_by_year = df_1996_2021.groupby('year').mean().reset_index()
```

Рисунок 16 - Рассчитайте среднее значение за каждый год с 1996 по 2021 год.

Затем мы печатаем его следующим образом



```
# среднее значение по региону за год
average_values_by_year.head()
```

	Year	Estimate
0	1996	0.005390
1	1998	0.004409
2	2000	-0.035114
3	2002	-0.043584
4	2003	-0.024761

Рисунок 16 - Средние значения по региону за каждый год с 1996 по 2021 год.

## 4.9 Постройте график индекса WGI, самого высокого, самого низкого, среднего для Азиатско-Тихоокеанского региона и Российской Федерации за годы (1996-2021)

Поскольку мы уже рассчитали максимальное и минимальное значения, а также среднее значение, нам нужны только значения по Российской Федерации.

```
# Российская Федерация
df_russia = wgidataset[wgidataset["countryname"] == "Russian Federation"]
df_russia = df_russia[df_russia['year'] != 2022]
df_russia = df_russia[["countryname", "year", "pve"]]

# DF страна с максимальным PVE
df_max = df_1996_2021[df_1996_2021["countryname"] == max_pve_country]
# DF страна с минимальным PVE
df_min = df_1996_2021[df_1996_2021["countryname"] == min_pve_country]
```

Рисунок 17 - Извлечение (estimate) значения для минимума, максимума и Российской Федерации.

Код для построения графика выглядит следующим образом.

```
# Plot the graph
plt.figure(figsize=(40, 20))
plt.plot(df_1996_2021['year'], df_1996_2021['pve'], marker='o', linestyle='-', color='aquamarine')
plt.plot(df_russia['year'], df_russia['pve'], marker='o', linestyle='-', color='orange', label='Russian Federation')
plt.plot(df_max['year'], df_max['pve'], marker='o', linestyle='-', color='green', label='Max')
plt.plot(df_min['year'], df_min['pve'], marker='o', linestyle='-', color='red', label='Min')
plt.plot(average_values_by_year['year'], average_values_by_year['pve'], marker='o', linestyle='-', color='blue', label='Mean')
plt.xlabel('Year')
plt.ylabel('Estimate')
plt.title('Plot of PVE over the Years')
plt.grid(True)
plt.legend()
plt.show()
```

Рисунок 18 - Извлечение (estimate) значения для минимума, максимума и Российской Федерации.

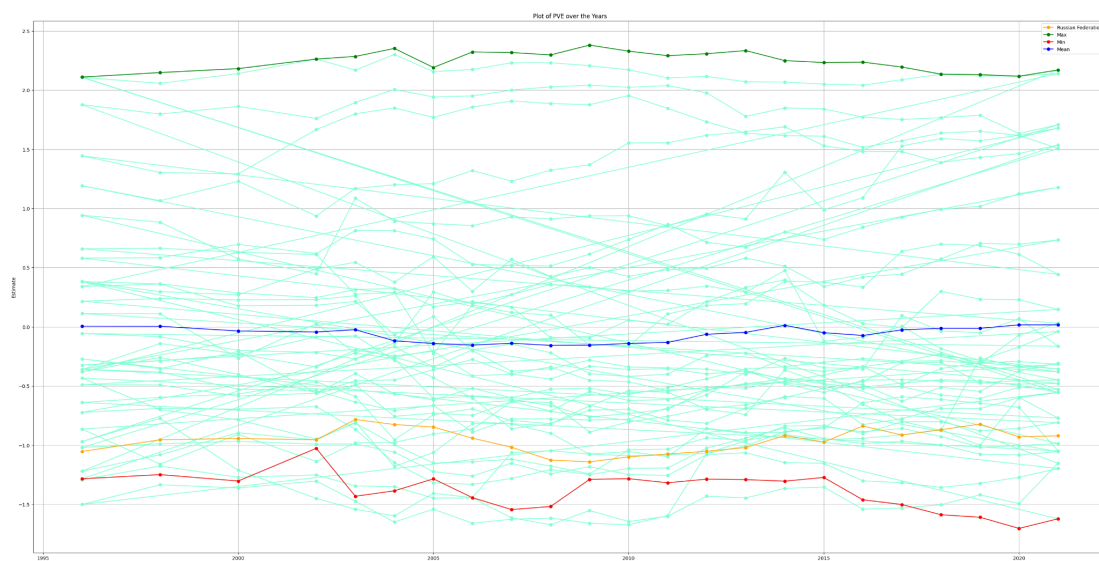


Рисунок 19 - Составьте график индекса WGI за 1996-2021 годы для страны в моем регионе и выберите страны с самыми высокими и самыми низкими значениями WQI на 2021 год.

## 4.10 Изменения значения показателя ранга с 1996 по 2022 год

Мы извлекли подмножество за 1996 год и за 2021 год, затем использовали следующую формулу для расчета изменений :

(финал wgi -инициал wgi)

И эта формула используется для всех стран, и операции выполняются так, как показано на рисунке

```
[32] #Название страны' - это имя столбца, содержащего названия стран в вашем наборе данных
unique_countries = df_1996_2021_rank['countryname'].unique().tolist()
# Инициализировать пустые списки для хранения элементов
percentage_change = []

for country in unique_countries:
    # Рассчитайте процентное изменение
    final_wgi = df_1996_2021_rank.loc[(df_1996_2021_rank['countryname'] == country) & (df_1996_2021_rank['year'] == 2021), 'var'].values
    initial_wgi = df_1996_2021_rank.loc[(df_1996_2021_rank['countryname'] == country) & (df_1996_2021_rank['year'] == 1996), 'var'].values
    percentage_change.append(final_wgi - initial_wgi)

# Создайте фрейм данных из списков
Changes = pd.DataFrame({'Country': unique_countries, 'Rank': percentage_change})
```

Рисунок 20 - Изменения значения показателя ранга с 1996 по 2022 год .

	Country	Changes
0	Aruba	[nan]
1	Andorra	[1.4746551513671875]
2	Afghanistan	[8.079877376556396]
3	Angola	[19.37019920349121]
4	Anguilla	[nan]

Рисунок 21 - Сколько процентов значение WGI с 1996 по 2021.

## 4.11 Таблица для (WGI - рейтинг)

В этом вопросе мы разделили код на фрагменты, и каждый фрагмент отвечает за хранение некоторой аналогичной информации в переменной, чтобы мы могли заполнить ею таблицу, в основном никаких вычислений не производилось, только извлечение из набора данных определенного значения и сохранение в конкретной переменной, которая будет использоваться в таблице

давайте начнем с первой части кода, где мы извлекли значения, запрошенные для таблицы за 2021 год.

```

# Сохраните только 2021 год
df_2021_rank = df_1996_2021[df_1996_2021["year"] == 2021]

# Рассчитайте среднее значение для WGI в 2021 году
df_2021_rank_mean = df_2021_rank['var'].mean()

# Сохраните только 1996 год
df_1996_rank = df_1996_2021[df_1996_2021["year"] == 1996]

# Рассчитайте среднее значение для WGI за 1996 год
df_1996_rank_mean = df_1996_rank['var'].mean()

# Рассчитайте процентное изменение среднего значения в период с 2021 по 1996 год
changes_mean = (df_2021_rank_mean - df_1996_rank_mean)

# Рассчитайте минимум и максимум для WGI в 2021 году
# Найдите страну с максимальным значением var
max_value_country = df_2021_rank.loc[df_2021_rank['var'] == df_2021_rank['var'].max(), 'countryname'].values[0]
max_value = df_2021_rank.loc[df_2021_rank['var'] == df_2021_rank['var'].max(), 'var'].values[0]

# Найдите страну с минимальным значением var
min_value_country = df_2021_rank.loc[df_2021_rank['var'] == df_2021_rank['var'].min(), 'countryname'].values[0]
min_value = df_2021_rank.loc[df_2021_rank['var'] == df_2021_rank['var'].min(), 'var'].values[0]

# Российская Федерация
df_russia = wgidataset[wgidataset["countryname"] == "Russian Federation"]
df_russia_2021 = df_russia[df_russia['year'] == 2021]

```

Рисунок 22 - Извлечение запрошенного значения из таблицы за 2021 год..

Во второй части кода мы извлекли значения, запрошенные для таблицы за 1996 год.

```

# Рассчитайте минимум и максимум для WGI в 2021 году
# Найдите страну с максимальным значением var
max_value_1996 = df_1996_rank.loc[df_1996_rank['countryname'] == max_value_country, 'var'].values[0]

# Найдите страну с минимальным значением var
min_value_1996 = df_1996_rank.loc[df_1996_rank['countryname'] == min_value_country, 'var'].values[0]
# Российская Федерация, 1996
df_russia_1996 = df_russia[df_russia['year'] == 1996]

```

Рисунок 23 - Извлечение по запрашиваемому значению в таблице на 1996 год.

В третьей части кода мы извлекли регионы страны с минимальным, максимальным рангом и для Российской Федерации.

```

# Извлечь максимальную область
max_region = regions.loc[regions['Country'] == max_value_country, 'Region'].values[0]
# Извлечь минимальный регион
min_region = regions.loc[regions['Country'] == "Korea, North", 'Region'].values[0]
# Извлечь российский регион
russian_region = regions.loc[regions['Country'] == "Russia", 'Region'].values[0]

```

Рисунок 24 - регионы для выделенных стран приведены в таблице.

в следующей части кода rf мы извлекли значения изменений, которые мы уже вычислили в вопросе 11.

```

# Извлечь изменения максимальной области
max_region_changes = Changes.loc[Changes['Country'] == max_value_country, 'Changes'].values[0]
# Извлечь изменения минимальной области
min_region_changes = Changes.loc[Changes['Country'] == "Korea, Dem. Rep.", 'Changes'].values[0]
# Выписка изменяет регион России
russian_region_changes = Changes.loc[Changes['Country'] == "Russian Federation", 'Changes'].values[0]

```

Рисунок 25 - Извлечение значений изменений для запрошенных случаев для таблицы.

и последняя часть - создание и заполнение таблицы всеми запрошенными значениями

```
import pandas as pd

# Создайте списки словарей для каждого столбца
data = {
    '' : ['Mean', 'Max', 'Min', 'Russia'],
    'Region' : ['- ', max_region, min_region, russian_region],
    'Country': ['- ', max_value_country, min_value_country, "Russian Federation"],
    'Rank 2021': [df_2021_rank_mean, max_value, min_value, df_russia_2021['var'].values[0]],
    'Rank 1996': [df_1996_rank_mean, max_value_1996, min_value_1996, df_russia_1996['var'].values[0]],
    'Changes': [changes_mean, max_region_changes, min_region_changes, russian_region_changes]
}

# # Создайте фрейм данных из словаря
table = pd.DataFrame(data)
# Заполните таблицу результатами
table
```

Рисунок 26 - Заполнение значений таблицы.

		Region	Country	Rank 2021	Rank 1996	Changes
0	Mean	AP	-	49.412698	50.197133	-0.784435
1	Max	AP	New Zealand	99.047623	97.849464	[1.1981582641601562]
2	Min	AP	Korea, Dem. Rep.	2.380952	4.838710	[-2.4577574729919434]
3	Russia	ECA	Russian Federation	19.523809	15.053763	[4.470046043395996]

Рисунок 27 - Результат работы таблицы.

## 4.12 Отобразите диаграмму размаха (boxplot) индекса WGI за 2022 для всех стран и для каждого региона в отдельности

Первым шагом было создание фрейма данных для каждой отдельной области, где это было сделано с помощью следующего кода

```
# Получите уникальные регионы
unique_regions = regions['Region'].unique()

# Создайте словарь для хранения фреймов данных для каждого региона
region_dfs = {}

# Разделите фрейм данных по регионам
for region in unique_regions:
    region_dfs[region] = regions[regions['Region'] == region].reset_index(drop=True)

# Создайте пустой словарь для хранения фреймов данных каждого региона
DataFrames = {}

df_2021 = wgidataset[wgidataset["year"] == 2021]
df_2021_estimate = df_2021[['countryname', 'pve']]
df_2021_estimate.rename(columns={'countryname': 'Country'}, inplace=True)

for region, region_df in region_dfs.items():
    # Создайте имя переменной на основе индекса
    DataFrames[region] = pd.merge(region_dfs[region], df_2021_estimate, on='Country', how='left')
```

Рисунок 28 - Создайте фрейм данных для каждой отдельной регион.



После декомпозиции страны по регионам мы можем отобразить сводный график WGI за 2021 год для всех стран и для каждого региона в отдельности.

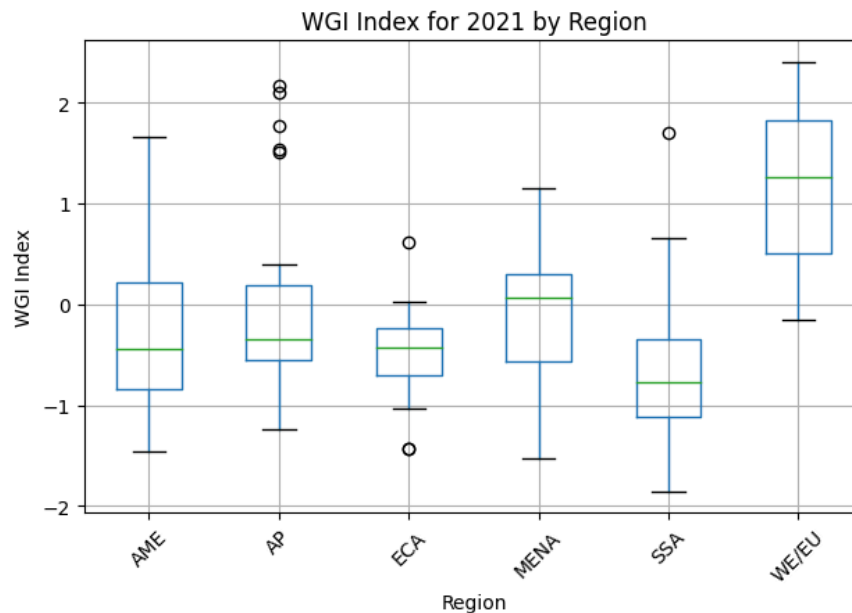


Рисунок 29 - Блок-график индекса WGI за 2022 год для всех стран и для каждого региона в отдельности (на одном графике).

## 5 Задача 2. Анализ рынка акций

Мы скачали данные из репозитория github и загрузили их в Google collab способом, который мы уже объясняли в первом задаче.

### 5.1 Загрузите данные в один Dataframe

мы просто перебрали все загруженные файлы и добавили один к другому с помощью функции `concat()`, следующий код показывает процедуру

```
# Путь к загруженному каталогу, содержащему CSV-файлы
folder_path = '/content/Stock'

# Инициализируйте пустой фрейм данных для хранения данных
combined_df = pd.DataFrame()

# Выполнить итерацию по каждому файлу в каталоге
for file_name in os.listdir(folder_path):
    # Проверьте, является ли файл CSV-файлом
    if file_name.endswith('.csv'):
        # Считайте CSV-файл во фрейм данных
        file_path = os.path.join(folder_path, file_name)
        df = pd.read_csv(file_path)

        # Извлеките название рекламной акции из имени файла
        promotion_name = os.path.splitext(file_name)[0]

        # Извлеките столбцы "Дата" и "Закреть" и установите "Дату" в качестве индекса
        df = df[['Date', 'Close']].set_index('Date')

        # Переименуйте столбец "Закреть" в название акции
        df.rename(columns={'Close': promotion_name}, inplace=True)

        # Объедините текущий фрейм данных с объединенным фреймом данных
        combined_df = pd.concat([combined_df, df], axis=1)

# Отображение объединенного фрейма данных
combined_df
```

Рисунок 30 - создайте глобальный фрейм данных.

	XIACY	NVDA	AMZN	MSFT	SPOT	ORCL	PINS	GOOGL	EBAY	AAPL	...	SHOP	META
Date													
2022-01-01	10.7350	244.860001	149.573502	310.980011	196.259995	81.160004	29.559999	135.303497	60.070000	174.779999	...	NaN	313.260010
2022-02-01	9.5500	243.850006	153.563004	298.790009	156.190002	75.970001	26.750000	135.057007	54.590000	165.119995	...	NaN	211.029999
2022-03-01	8.7199	272.859985	162.997498	308.309998	151.020004	82.730003	24.610001	139.067505	57.259998	174.610001	...	NaN	222.360001

Рисунок 31 - Часть глобального набора данных.

## 5.2 Вычисление корреляционной матрицы для всех акций

мы только что применили функцию `corr()` для вычисления корреляционной матрицы

```
# Вычислить корреляционную матрицу
correlation_matrix = combined_df.corr()

# Отобразить корреляционную матрицу
print(correlation_matrix)
```

Рисунок 32 - Вычислите корреляционную матрицу.

	XIACY	NVDA	AMZN	MSFT	SPOT	ORCL	PINS
XIACY	1.000000	0.445645	0.654564	0.565831	0.647331	0.324511	0.524413
NVDA	0.445645	1.000000	0.765294	0.935386	0.925270	0.875089	0.815629
AMZN	0.654564	0.765294	1.000000	0.838702	0.875779	0.534556	0.666996
MSFT	0.565831	0.935386	0.838702	1.000000	0.949380	0.847046	0.837576
SPOT	0.647331	0.925270	0.875779	0.949380	1.000000	0.763100	0.842858
ORCL	0.324511	0.875089	0.534556	0.847046	0.763100	1.000000	0.747754
PINS	0.524413	0.815629	0.666996	0.837576	0.842858	0.747754	1.000000
GOOGL	0.680658	0.715287	0.912332	0.845993	0.821587	0.618983	0.640675
EBAY	0.535223	0.087027	0.434078	0.127010	0.296858	-0.070414	-0.002757
AAPL	0.408747	0.633114	0.665715	0.790691	0.687415	0.769309	0.640294
TWLO	0.447846	-0.244797	0.314869	-0.094023	0.059969	-0.393536	-0.141953
TSLA	0.184629	-0.277600	0.302321	-0.117639	-0.092332	-0.310021	-0.253055
ADBE	0.697612	0.802739	0.819614	0.913842	0.863827	0.785432	0.804657
TCOM	0.237659	0.787859	0.309545	0.662193	0.640120	0.836340	0.705551
INTC	0.791377	0.458281	0.816519	0.627531	0.645555	0.239485	0.452144
SHOP	0.519367	0.713391	0.824934	0.842193	0.737909	0.635736	0.846115
META	0.573429	0.961389	0.830910	0.966868	0.973401	0.821696	0.822643
DBX	0.382992	0.519374	0.478171	0.648164	0.525305	0.667833	0.710191
NFLX	0.505430	0.910910	0.735466	0.900263	0.920771	0.859397	0.930638

Рисунок 33 - Часть корреляционной матрицы.

## 5.3 Отобразите корреляционную матрицу в виде диаграммы

Код, использованный для этого, был следующим

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

# Plotting the correlation matrix as a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, cmap='coolwarm')
plt.title('Correlation Matrix of Stock Closing Prices')
plt.show()
```

Рисунок 33 - Построение корреляционной матрицы цен закрытия акций.

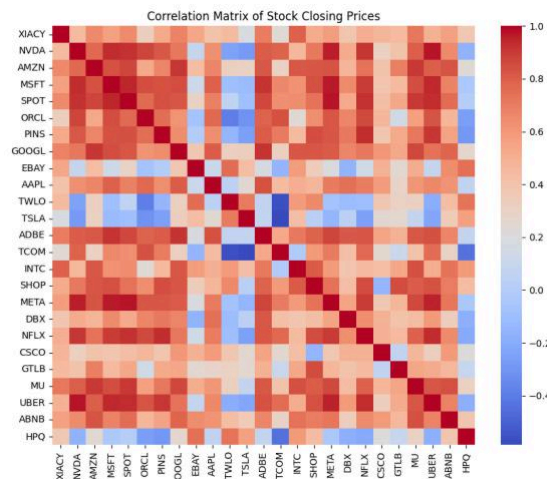


Рисунок 34 - Корреляционная матрица цен закрытия акций.

### 5.4.1 Аксию с максимальной положительной корреляцией (max)

Первым шагом было отфильтровать корреляционную матрицу, чтобы включить только корреляции с Uber (UBER), затем мы использовали *idxmax()*, чтобы найти максимальную долю

```
# Отфильтруйте корреляционную матрицу, чтобы включить только корреляции с Alibaba (BABA)
Uber_correlation = correlation_matrix['UBER'].drop('UBER') # Откажитесь от самокорреляции с Uber
# Найдите долю с максимальной положительной корреляцией
max_positive_correlation_stock = Uber_correlation.idxmax()

print("Share with the maximum positive correlation with Alibaba (Uber):", max_positive_correlation_stock)

Share with the maximum positive correlation with Alibaba (Uber): NVDA
```

Рисунок 35 - Рассчитайте долю с максимальным коэффициентом корреляции.

### 5.4.2 Аксию с максимальной отрицательной корреляцией (min)

Операция была аналогична предыдущей, мы только изменили функцию *idxmin()*

```
# Найдите долю с максимальной отрицательной корреляцией
max_negative_correlation_stock = Uber_correlation.idxmin()

print("Share with the maximum negative correlation with Uber (Uber):", max_negative_correlation_stock)

Share with the maximum negative correlation with Uber (Uber): TSLA
```

Рисунок 36 - Рассчитайте долю с минимальным коэффициентом корреляции.

### 5.4.3 Аксию с минимальной корреляцией

Мы рассчитали долю с минимальной корреляцией (ближайшей к нулю), используя функции *abs()* и *idxmin()*

```
# Найдите долю с минимальной корреляцией (ближайшей к нулю)
min_correlation_stock = Uber_correlation.abs().idxmin()

print("Share with the minimum correlation (closest to zero) with Uber (Uber):", min_correlation_stock)

Share with the minimum correlation (closest to zero) with Uber (Uber): EBAY
```

Рисунок 37 - Аксию с минимальной.



## 5.5 Постройте диаграммы разброса

Сначала мы начали с извлечения цен закрытия Uber (UBER) и выбранных компаний из предыдущего вопроса.

```
# Извлеките цены закрытия Uber (UBER) и выбранных компаний
baba_prices = combined_df['UBER']
max_positive_correlation_prices = combined_df[max_positive_correlation_stock]
max_negative_correlation_prices = combined_df[max_negative_correlation_stock]
min_correlation_prices = combined_df[min_correlation_stock]
```

Рисунок 38 - Извлеките цены закрытия Uber (UBER) и выбранных компаний.

код и результат построения точечных диаграмм показаны ниже.

```
# Создание точечных диаграмм
plt.figure(figsize=(12, 8))

# Точечный график для компании с минимальной корреляцией (ближайшей к нулю)
plt.subplot(2, 2, 1)
plt.scatter(baba_prices, min_correlation_prices, color='blue')
plt.title('Scatter Plot: Uber vs. ' + min_correlation_stock)
plt.xlabel('Uber (UBER) Closing Prices')
plt.ylabel(min_correlation_stock + ' Closing Prices')

# Точечный график для компании с максимальной положительной корреляцией
plt.subplot(2, 2, 2)
plt.scatter(baba_prices, max_positive_correlation_prices, color='green')
plt.title('Scatter Plot: Uber vs. ' + max_positive_correlation_stock)
plt.xlabel('Uber (UBER) Closing Prices')
plt.ylabel(max_positive_correlation_stock + ' Closing Prices')

# Точечный график для компании с максимальной отрицательной корреляцией
plt.subplot(2, 2, 3)
plt.scatter(baba_prices, max_negative_correlation_prices, color='red')
plt.title('Scatter Plot: Uber vs. ' + max_negative_correlation_stock)
plt.xlabel('Uber (UBER) Closing Prices')
plt.ylabel(max_negative_correlation_stock + ' Closing Prices')
plt.tight_layout()
plt.show()
```

Рисунок 40 – построение разброса диаграмм.

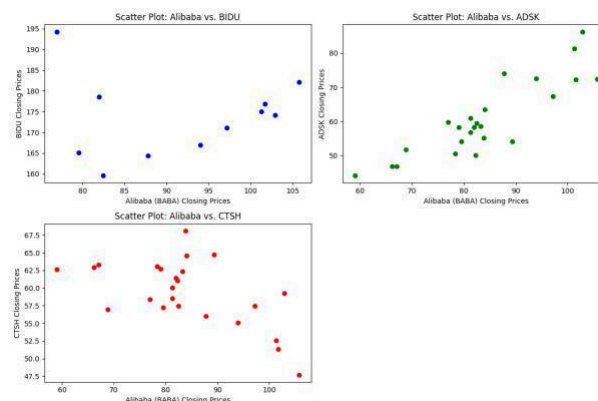


Рисунок 41 – диаграммы разброса.

## 5.6 Рассчитайте среднюю цену акций для каждого месяца

Преобразуйте индекс в формат даты и времени, затем повторно преобразуйте фрейм данных в месячные интервалы и вычислите среднее значение.

```
# Преобразуйте индекс в datetime, если он еще не в формате datetime
combined_df.index = pd.to_datetime(combined_df.index)

# Повторите выборку фрейма данных с месячными интервалами и вычислите среднее значение
monthly_avg_prices = combined_df.resample('M').mean()

# Отображение средних цен на акции за каждый месяц
print(monthly_avg_prices)
```

	XIACY	NVDA	AMZN	MSFT	SPOT	\
Date						
2022-01-31	10.7350	244.860001	149.573502	310.980011	196.259995	
2022-02-28	9.5500	243.850006	153.563004	298.790009	156.190002	
2022-03-31	8.7199	272.859985	162.097498	308.309998	151.020004	
2022-04-30	7.5400	185.470001	124.281502	277.519989	101.650002	
2022-05-31	7.6300	186.720001	120.209503	271.869995	112.769997	
2022-06-30	8.6300	151.589996	106.209999	256.829987	93.830002	
2022-07-31	7.8900	181.630005	134.949997	280.739990	113.019997	
2022-08-31	7.1900	150.940002	126.769997	261.470001	108.150002	
2022-09-30	5.5800	121.389999	113.000000	232.899994	86.300003	
2022-10-31	5.6300	134.970001	102.440002	232.130005	80.580002	
2022-11-30	6.7800	169.229996	96.540001	255.139999	79.419998	
2022-12-31	6.8750	146.139999	84.000000	239.820007	78.949997	

Рисунок 42 – Рассчитайте среднюю цену акций для каждого месяца.

## 5.7 Постройте графики для акций из пункта 4 и средней из пункта 6

Мы выполнили эту задачу, используя следующий код

```
import matplotlib.pyplot as plt

# Построение графика цен на отдельные акции и средних цен на акции за каждый месяц
plt.figure(figsize=(14, 8))

# Построение графиков цен на отдельные акции
for stock in [max_positive_correlation_stock, max_negative_correlation_stock, min_correlation_stock, "UBER"]:
    plt.plot(combined_df.index, combined_df[stock], label=stock)

# Построить график средних цен на акции за каждый месяц
plt.plot(monthly_avg_prices.index, monthly_avg_prices.mean(axis=1), color='black', linestyle='--', label='Average')

# Добавить метки и заголовков
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('Stock Prices and Average Price')
plt.legend()
plt.grid(True)

# Show plot
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Рисунок 43 – Построение графика цен на отдельные акции и средних цен на акции за каждый месяц.

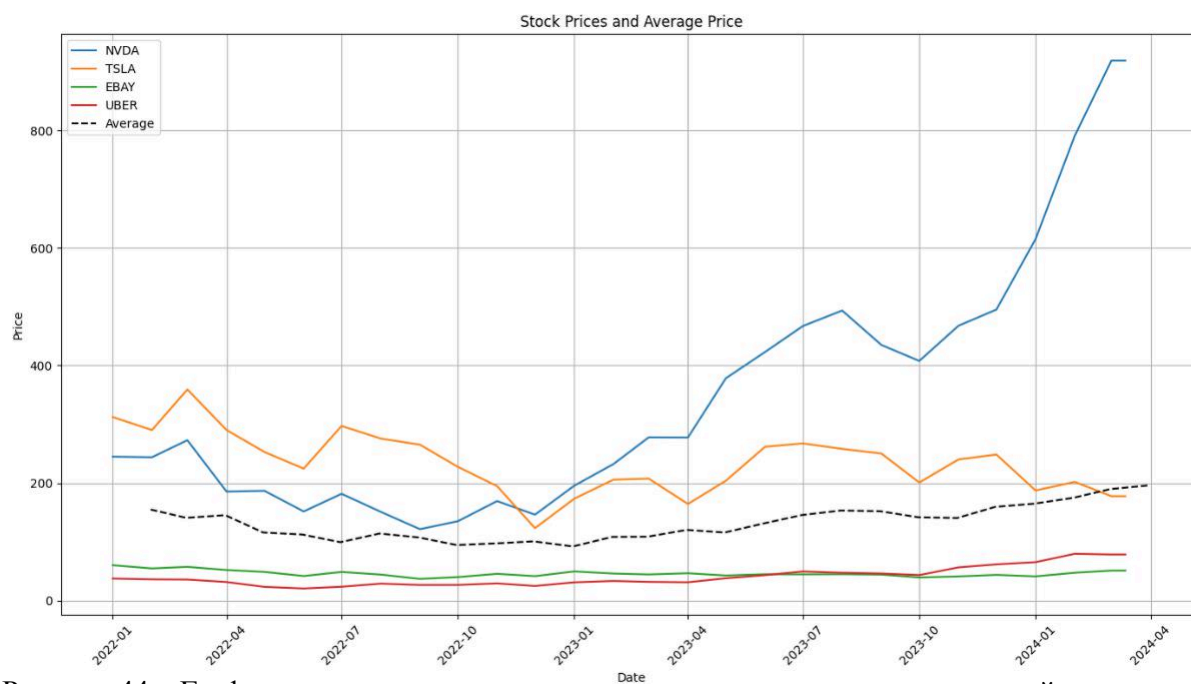


Рисунок 44 – График цен на отдельные акции и средних цен на акции за каждый месяц.