Федеральное государственное автономное образовательное учреждение
высшего образования

«Пермский государственный национальный исследовательский университет»

Институт компьютерных наук и технологий

Отчет
по лабораторной работе № 3

по дисциплине
«Введение в анализ данных»

Студент Панькова С.А.

Группа ИТ-13-2023

Пермь 2025

Задание 1

В файле «global-electricity-generation.csv» представлена информация о производстве электроэнергии странами с 1992 по 2021 год. В файле «global-electricity-consumption.csv» – информация о потреблении электроэнергии. Все данные – в млрд. кВт*ч.

1. Загрузите информацию из этих файлов.

Указание: рекомендуется воспользоваться функцией genfromtxt. Обратите внимание, что данные в файлах разделены запятой, в обоих файлах есть заголовок, который при загрузке нужно пропустить. Для корректной загрузки названий стран необходимо также указать тип данных (по умолчанию загружаемые данные преобразуются в вещественный тип). Вы можете либо собрать все данные в один массив записей, либо работать с тремя разными массивами: массив названий стран (одномерный, строки), массивы производства и потребления энергии (двумерные, вещественные).

Порядок стран в двух исходных файлах одинаковый.

```
import numpy as np

# Функция, которая заменяет некорректные данные на nan
def replace_unwanted_symbols(value):
    str_value = str(value)

    if str_value in ['--', 'ie', '']:
        return 'nan'

    return str_value


# 1. Загрузить информацию

arr_generation = np.array(
    np.genfromtxt('data/global-electricity-
generation.csv', dtype=None, comments='#', delimiter=',',
skip_header=1,
```

```
                skip_footer=0, converters={i:
replace_unwanted_symbols for i in range(31)}))
arr_consumption = np.array(
   np.genfromtxt('data/global-electricity-
consumption.csv', dtype=None, comments='#',
delimiter=',', skip_header=1,
                skip_footer=0, converters={i:
replace_unwanted_symbols for i in range(31)}))

# Массив названий стран из generation файла
arr_country_gen = []
for item in arr_generation:
   arr_country_gen.append(item[0])

# Массив названий стран из consumption файла
arr_country_con = []
for item in arr_consumption:
   arr_country_con.append(item[0])

if arr_country_gen == arr_country_con:
   print("Массивы названий стран идентичны - данные
загружены корректно!")
else:
   print("Массивы названий стран различаются!")

# Преобразовываем значения в числовой формат
arr_generation = arr_generation[:, 1:].astype(float)
arr_consumption = arr_consumption[:, 1:].astype(float)
```

- dtype=None - означает, что NumPy сам определит подходящие типы данных для каждого столбца
- comments='#' - строки, начинающиеся с #, будут пропущены при чтении
- delimiter=',' - указывает, что столбцы разделены запятыми (CSV формат)
- skip_header=1 - пропускает первую строку файла
- skip_footer=0 - не пропускает никакие строки в конце файла
- converters={i: replace_unwanted_symbols for i in range(31)} - применяет указанную функцию к каждому столбцу

Результат работы программы:

```
Массивы названий стран идентичны - данные загружены корректно!
```

2. Постройте одномерные массивы ежегодного производства потребления электроэнергии в среднем за последние 5 лет (один массив – производство по всем странам, второй – потребление по всем странам).

Указание: не забывайте, что в агрегирующих функциях для построчной обработки нужно указывать параметр axis=1.

```
# 2. Построение массивов средних значений за последние 5
лет
last_five_years_generation = arr_generation[:, -5:]
last_five_years_consumption = arr_consumption[:, -5:]

# Вычисление средних значений (игнорируя nan)
avg_generation = np.array([
    np.nanmean(row) if not np.isnan(row).all() else np.nan
    for row in last_five_years_generation
])
avg_consumption = np.array([
    np.nanmean(row) if not np.isnan(row).all() else np.nan
    for row in last_five_years_consumption
])

print("\nСреднее производство электроэнергии за последние
5 лет по странам:")
for i in range(len(arr_country_gen)):
    if not np.isnan(avg_generation[i]):
        print(f"{arr_country_gen[i]}:
{avg_generation[i]:.3f} млрд. кВт*ч")

print("\nСреднее потребление электроэнергии за последние
5 лет по странам:")
for i in range(len(arr_country_gen)):
    if not np.isnan(avg_consumption[i]):
        print(f"{arr_country_gen[i]}:
{avg_consumption[i]:.3f} млрд. кВт*ч")
```

- if not np.isnan(row).all() else np.nan - если НЕ все значения nan, то вычисляем среднее, иначе возвращаем nan

Результат работы программы:

```
Среднее производство электроэнергии за последние 5 лет по странам:
Algeria: 74.111 млрд. кВт*ч
Angola: 14.387 млрд. кВт*ч
Benin: 0.234 млрд. кВт*ч
Botswana: 2.619 млрд. кВт*ч
Burkina Faso: 1.685 млрд. кВт*ч
Burundi: 0.322 млрд. кВт*ч
Cabo Verde: 0.454 млрд. кВт*ч
Cameroon: 8.163 млрд. кВт*ч
Central African Republic: 0.151 млрд. кВт*ч
Chad: 0.305 млрд. кВт*ч
Comoros: 0.110 млрд. кВт*ч
Congo-Brazzaville: 3.466 млрд. кВт*ч
Congo-Kinshasa: 10.794 млрд. кВт*ч
Cote d'Ivoire: 10.242 млрд. кВт*ч
Djibouti: 0.056 млрд. кВт*ч
Egypt: 196.525 млрд. кВт*ч
Equatorial Guinea: 1.383 млрд. кВт*ч
Eritrea: 0.433 млрд. кВт*ч
Eswatini: 0.725 млрд. кВт*ч
Ethiopia: 14.012 млрд. кВт*ч
Gabon: 2.220 млрд. кВт*ч
Gambia: 0.304 млрд. кВт*ч
Ghana: 17.006 млрд. кВт*ч
Guinea: 2.262 млрд. кВт*ч
Guinea-Bissau: 0.082 млрд. кВт*ч
Kenya: 11.402 млрд. кВт*ч
```

и тд.

```
Среднее потребление электроэнергии за последние 5 лет по странам:
Algeria: 64.280 млрд. кВт*ч
Angola: 12.765 млрд. кВт*ч
Benin: 0.858 млрд. кВт*ч
Botswana: 3.588 млрд. кВт*ч
Burkina Faso: 2.419 млрд. кВт*ч
Burundi: 0.382 млрд. кВт*ч
Cabo Verde: 0.325 млрд. кВт*ч
Cameroon: 6.298 млрд. кВт*ч
Central African Republic: 0.141 млрд. кВт*ч
Chad: 0.258 млрд. кВт*ч
Comoros: 0.079 млрд. кВт*ч
Congo-Brazzaville: 1.893 млрд. кВт*ч
Congo-Kinshasa: 9.146 млрд. кВт*ч
Cote d'Ivoire: 7.137 млрд. кВт*ч
Djibouti: 0.486 млрд. кВт*ч
Egypt: 162.100 млрд. кВт*ч
Equatorial Guinea: 1.238 млрд. кВт*ч
Eritrea: 0.387 млрд. кВт*ч
Eswatini: 1.530 млрд. кВт*ч
Ethiopia: 9.381 млрд. кВт*ч
Gabon: 2.269 млрд. кВт*ч
Gambia: 0.249 млрд. кВт*ч
Ghana: 14.863 млрд. кВт*ч
Guinea: 1.972 млрд. кВт*ч
Guinea-Bissau: 0.077 млрд. кВт*ч
Kenya: 8.989 млрд. кВт*ч
```

и тд.

3. Напишите выражения, позволяющие получить ответ на следующие вопросы:

3.1 Суммарное (по всем странам) потребление электроэнергии за каждый год.

```
    # 3.1. Суммарное (по всем странам) потребление электроэнергии за каждый год (игнорируя nan)
    sum_consumption = np.nansum(arr_consumption, axis=0)
    print("\nСуммарное (по всем странам) потребление электроэнергии за каждый год")
    years = list(range(1992, 2022))
```

```
    for i, year in enumerate(years):
        print(f"{year}: {sum_consumption[i]:.3f} млрд.
кВт*ч")
```

Результат работы программы:

```
Суммарное (по всем странам) потребление электроэнергии за каждый год
1992: 10569.016 млрд. кВт*ч
1993: 10854.563 млрд. кВт*ч
1994: 11104.654 млрд. кВт*ч
1995: 11476.479 млрд. кВт*ч
1996: 11805.865 млрд. кВт*ч
1997: 12122.033 млрд. кВт*ч
1998: 12419.809 млрд. кВт*ч
1999: 12685.754 млрд. кВт*ч
2000: 13230.250 млрд. кВт*ч
2001: 13486.490 млрд. кВт*ч
2002: 13938.779 млрд. кВт*ч
2003: 14455.562 млрд. кВт*ч
2004: 15128.098 млрд. кВт*ч
2005: 15725.982 млрд. кВт*ч
2006: 16438.214 млрд. кВт*ч
2007: 17205.120 млрд. кВт*ч
2008: 17477.471 млрд. кВт*ч
2009: 17435.687 млрд. кВт*ч
2010: 18748.161 млрд. кВт*ч
2011: 19438.823 млрд. кВт*ч
2012: 19918.255 млрд. кВт*ч
2013: 20573.096 млрд. кВт*ч
2014: 20981.240 млрд. кВт*ч
2015: 21400.018 млрд. кВт*ч
2016: 22022.687 млрд. кВт*ч
2017: 22716.206 млрд. кВт*ч
2018: 23530.920 млрд. кВт*ч
2019: 23915.656 млрд. кВт*ч
2020: 23959.680 млрд. кВт*ч
2021: 25336.707 млрд. кВт*ч
```

3.2. Максимальное количество электроэнергии, которое произвела одна страна за один год (указание: чтобы не учитывать отсутствующие и

некорректные данные (nan) воспользуйтесь NaN-безопасной версией функции max, то есть nanmax).

```
# 3.2. Максимальное количество электроэнергии, которое произвела одна страна за один год (игнорируя nan)
max_generation_value = np.nanmax(arr_generation)
max_generation_indices = np.where(arr_generation == max_generation_value)
country_index = max_generation_indices[0][0]
year_index = max_generation_indices[1][0]
year = years[year_index]

print("\nМаксимальное количество электроэнергии, которое произвела одна страна за один год")
print(f"Страна: {arr_country_gen[country_index]}")
print(f"Год: {year}")
print(f"Производство: {max_generation_value:.3f} млрд. кВт*ч")
```

Результат работы программы:

```
Максимальное количество электроэнергии, которое произвела одна страна за один год
Страна: China
Год: 2021
Производство: 8151.518 млрд. кВт*ч
```

3.3. Список стран, которые производят более 500 млрд. кВт*ч электроэнергии ежегодно в среднем за последние 5 лет (воспользуйтесь массивом, полученным на шаге 2).

```
    # 3.3. Список стран, которые производят более 500 млрд. кВт*ч электроэнергии ежегодно в среднем за последние 5 лет
    print("\nСписок стран, которые производят более 500 млрд. кВт*ч электроэнергии ежегодно в среднем за последние 5 лет")
    for i in range(len(avg_generation)):
        if not np.isnan(avg_generation[i]) and avg_generation[i] > 500:
            print(arr_country_gen[i])
```

Результат работы программы:

```
Список стран, которые производят более 500 млрд. кВт*ч электроэнергии ежегодно в среднем за последние 5 лет
Russia
France
Germany
China
India
Japan
South Korea
Canada
United States
Brazil
```

3.4. 10% стран, которые потребляют больше всего электроэнергии ежегодно в среднем за последние 5 лет (указание: вначале определите соответствующую квантиль в массиве, построенном на шаге 2).

```
    # 3.4. 10% стран, которые потребляют больше всего
электроэнергии ежегодно в среднем за последние 5 лет

    print("\n10% стран, которые потребляют больше всего
электроэнергии ежегодно в среднем за последние 5 лет ")
    # Находим 90-й квантиль (игнорируя nan)
    quantile_90 = np.nanquantile(avg_consumption, 0.90)

    # Отбираем страны, которые превышают этот квантиль
    for i in range(len(arr_country_gen)):
        if not np.isnan(avg_consumption[i]) and
avg_consumption[i] > quantile_90:
            print(arr_country_gen[i])
```

- квантиль (0.90) - это значение, которое больше чем 90% всех значений в данных

Результат работы программы:

```
10% стран, которые потребляют больше всего электроэнергии ежегодно в среднем за последние 5 лет
South Africa
Russia
France
Germany
Italy
Spain
Turkiye
United Kingdom
Australia
China
India
Indonesia
Japan
South Korea
Taiwan
Vietnam
Iran
Saudi Arabia
Canada
Mexico
United States
Brazil
```

3.5. Список стран, которые увеличили производство электроэнергии в 2021 году по сравнению с 1992 годом более, чем в 10 раз.

```
    # 3.5. Список стран, которые увеличили производство электроэнергии в 2021 году по сравнению с 1992 годом более, чем в 10 раз
    print("\nСписок стран, которые увеличили производство электроэнергии в 2021 году по сравнению с 1992 годом более, чем в 10 раз")
    generation_2021 = arr_generation[:, -1]
    generation_1992 = arr_generation[:, 0]  # 1992 год - это первый столбец (индекс 0)
    for i in range(len(arr_country_gen)):
        # Игнорируем случаи, где есть nan значения
        if not np.isnan(generation_1992[i]) and not np.isnan(generation_2021[i]):
            if generation_1992[i] != 0 and generation_2021[i] > 10 * generation_1992[i]:
                print(arr_country_gen[i])
```

Результат работы программы:

```
Список стран, которые увеличили производство электроэнергии в 2021 году по сравнению с 1992 годом более, чем в 10 раз
Angola
Benin
Equatorial Guinea
Ethiopia
Mali
Mauritania
Mozambique
Sudan
Cambodia
China
Laos
Maldives
Vietnam
Turks and Caicos Islands
```

3.6. Список стран, которые в сумме за все годы потратили больше 100 млрд. кВт*ч электроэнергии и при этом произвели меньше, чем потратили.

```
    # 3.6. Список стран, которые в сумме за все годы потратили больше 100 млрд. кВт*ч электроэнергии и при этом произвели меньше, чем потратили
    print("\nСписок стран, которые в сумме за все годы потратили больше 100 млрд. кВт*ч электроэнергии и при этом произвели меньше, чем потратили")

    # Потребление энергии каждой страной за все годы (игнорируя nan)
    sum_consumption_all = np.nansum(arr_consumption, axis=1)

    # Производство энергии каждой страной за все годы (игнорируя nan)
    sum_generation_all = np.nansum(arr_generation, axis=1)

    for i in range(len(sum_generation_all)):
        if not np.isnan(sum_consumption_all[i]) and not np.isnan(sum_generation_all[i]):
            if sum_consumption_all[i] > 100 and sum_generation_all[i] < sum_consumption_all[i]:
                print(arr_country_gen[i])
```

Результат работы программы:

```
Список стран, которые в сумме за все годы потратили больше 100 млрд. кВт*ч электроэнергии и при этом произвели меньше, чем потратили
Zimbabwe
Belarus
Moldova
Belgium
Croatia
Finland
Hungary
Italy
Latvia
Luxembourg
Netherlands
North Macedonia
Hong Kong
```

3.7. Какая страна потратила наибольшее количество электроэнергии в 2020 году?

```
    # 3.7. Какая страна потратила наибольшее количество электроэнергии в 2020 году?
    print("\nКакая страна потратила наибольшее количество электроэнергии в 2020 году?")
    max_consumption_index = np.nanargmax(arr_consumption[:, -2])  # индекс максимального потребления в 2020
    print(arr_country_gen[max_consumption_index])
```

Результат работы программы:

```
Какая страна потратила наибольшее количество электроэнергии в 2020 году?
China
```

Задание 2

В файле «data2.csv» представлены данные наблюдений о прибыли (второй столбец) в зависимости от установленной скидки (первый столбец).

Комментарий. На самом деле эти данные сгенерированы синтетически, в учебных целях, но смысл значений в этой задаче не важен, и эти данные можно рассматривать как реальный датасет.

Бизнес-консультанты считают, что реальная зависимость прибыли от установленной скидки может быть описана квадратичной или кубической

функцией (то есть полиномом второй или третьей степени, f(x) = a2□^{2} + a1x + a0 или f(x) = a3□^{3} + a2□^{2} + a1x + a0).

1. Сформируйте систему линейных уравнений (СЛУ) для полинома 2й степени (для этого нужно выбрать 3 точки, в которых значение полинома должно совпадать с исходными данными; точки лучше выбирать равномерно «разбросанными» по исходным данным, то есть одна в начале имеющегося диапазона данных, одна в конце и одна в середине).

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import solve

# Загрузка данных
data = np.genfromtxt('data/data2.csv', delimiter=';',
skip_header=1, dtype=float)
x = data[:, 0]  # Скидки
y = data[:, 1]  # Прибыль

print(f"Средняя прибыль по исходным данным:
{np.mean(y):.2f}")

# Полином 2-й степени

# Выбор точек для квадратичного полинома
idx_quad = [0, len(x) // 2, -1]
x_quad = x[idx_quad]
y_quad = y[idx_quad]

# Матрица коэффициентов для квадратичного полинома
A_quad = np.array([
    [x_quad[0]**2, x_quad[0], 1],
    [x_quad[1]**2, x_quad[1], 1],
    [x_quad[2]**2, x_quad[2], 1]
])
```

2. Решите СЛУ (с помощью scipy.linalg.solve), тем самым найдя коэффициенты полинома.

```
# Решение системы уравнений
coeffs_quad = solve(A_quad, y_quad)

print("\nЗначения коэффициентов полинома 2-й степени")
print(coeffs_quad)
```

3. Получите вектор значений построенного полинома для заданных точек.

```
# Расчет значений полинома
y_pred_quad = coeffs_quad[0] * x**2 + coeffs_quad[1] * x
+ coeffs_quad[2]
```

4. Постройте в одной области два графика: один по заданным в файле точкам, другой – по полученному вектору.

```
# График для квадратичного полинома
plt.figure(figsize=(10, 6))
plt.plot(x, y, 'o', label='Исходные данные')
plt.plot(x, y_pred_quad, '-', label='Полином 2-й
степени')
plt.xlabel('Скидка')
plt.ylabel('Прибыль')
plt.legend()
plt.show()
```

5. Посчитайте значение квадратичного отклонения RSS (оно вычисляется по формуле, где $\gamma_{\square}$ – ожидаемые значения (из исходного файла), f($\square_{\square}$) – рассчитанные значения полинома.

```
# Расчет RSS
RSS_quad = np.sum((y - y_pred_quad) ** 2)
print("RSS для полинома 2-й степени:", RSS_quad)
```

6. Повторите шаги 1-5 для полинома 3-й степени (для этого нужно

будет выбрать 4 точки).

```
# Полином 3-й степени

# Выбор точек для кубического полинома
idx_cubic = [0, len(x) // 3, 2 * len(x) // 3, -1]
x_cubic = x[idx_cubic]
y_cubic = y[idx_cubic]

# Матрица коэффициентов для кубического полинома
A_cubic = np.array([
    [x_cubic[0]**3, x_cubic[0]**2, x_cubic[0], 1],
    [x_cubic[1]**3, x_cubic[1]**2, x_cubic[1], 1],
    [x_cubic[2]**3, x_cubic[2]**2, x_cubic[2], 1],
    [x_cubic[3]**3, x_cubic[3]**2, x_cubic[3], 1]
])

# Решение системы уравнений
coeffs_cubic = solve(A_cubic, y_cubic)

print("\nЗначения коэффициентов полинома 3-й степени")
print(coeffs_cubic)

# Расчет значений полинома
y_pred_cubic = coeffs_cubic[0] * x**3 + coeffs_cubic[1] *
x**2 + coeffs_cubic[2] * x + coeffs_cubic[3]

# График для кубического полинома
plt.figure(figsize=(10, 6))
plt.plot(x, y, 'o', label='Исходные данные')
plt.plot(x, y_pred_cubic, '-', label='Полином 3-й
степени')
plt.xlabel('Скидка')
plt.ylabel('Прибыль')
plt.legend()
plt.show()

# Расчет RSS
RSS_cubic = np.sum((y - y_pred_cubic) ** 2)
print("RSS для полинома 3-й степени:", RSS_cubic)
```

7. (по желанию). Можно поэкспериментировать с выбором точек для построения полиномов.

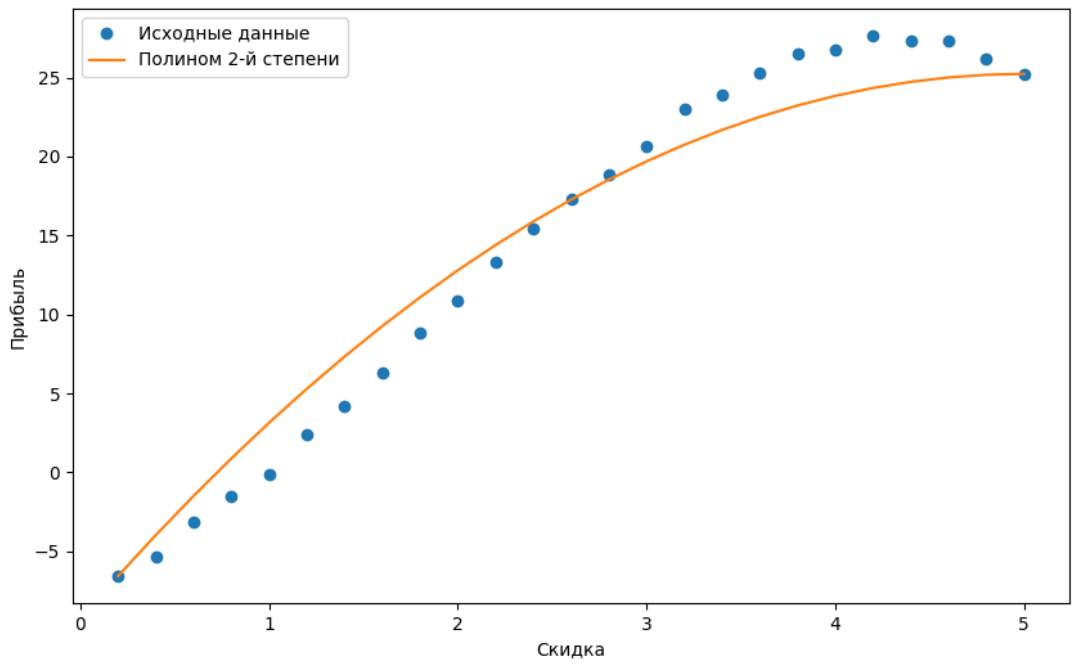
8. Выберите тот вариант, где значение отклонения (RSS) получается наименьшим. Для этого варианта посчитайте ожидаемое значение прибыли при значениях скидки в 6 и 8 процентов.

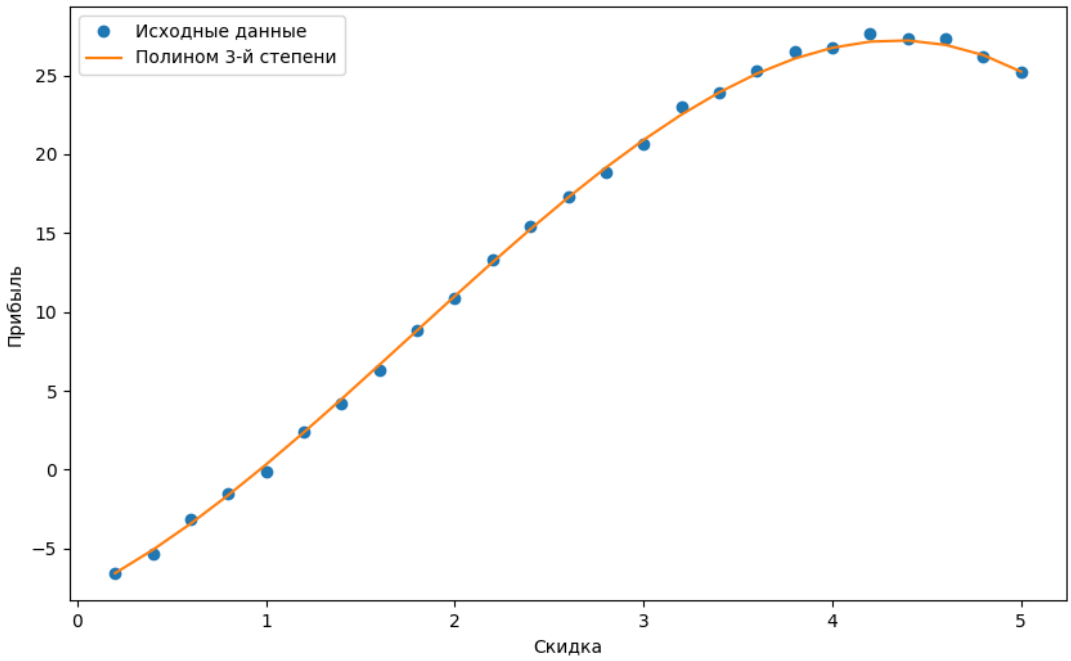
```
# Точки для прогноза
x_new = np.array([6, 8])

# Прогноз значений
y_pred_new = coeffs_cubic[0] * x_new**3 + coeffs_cubic[1]
* x_new**2 + coeffs_cubic[2] * x_new + coeffs_cubic[3]

print("Прогноз прибыли для скидки 6% и 8%:", y_pred_new)
```

Результат работы программы:





```
Средняя прибыль по исходным данным: 13.58

Значения коэффициентов полинома 2-й степени
[-1.25610119 12.86793993 -7.69796582]
RSS для полинома 2-й степени: 163.36349055625158

Значения коэффициентов полинома 3-й степени
[-0.58570762  3.22486627  5.10579311 -7.69796582]
RSS для полинома 3-й степени: 1.9542145464333431

Прогноз прибыли для скидки 6% и 8%: [ 12.51913288 -60.34248049]
```