МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Основы машинного обучения»

Тема: Регрессия

Вариант 4А

Студентка гр. 1304 Чернякова А.Д.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2024

Цель работы.

Изучить линейную и нелинейную регрессии на разном наборе данных, а также оценить качество моделей регрессии.

Задание.

1) Линейная регрессия
2) Нелинейная регрессия
3) Оценка модели регрессии

Выполнение работы.

1. Линейная регрессия

1.1. Загрузим набор данных lab3_lin4.csv с помощью метода read_csv, который принимает в качестве аргумента путь до файла. Убедимся, что загрузка прошла корректно с помощью метода head, который по умолчанию выводит первые 5 строк набора данных (см. листинг 1.1 и таблицу 1.1).

Листинг 1.1 - Загрузка и проверка данных lab3_lin4.csv

```
df = pd.read_csv('lab3_lin4.csv')
df.head()
```

Таблица 1.1 - Результат работы метода head

	x1	x2	y
0	1.1414	-0.0594	55.2686
1	-0.1517	-1.3385	-50.3732
2	-2.8140	0.1431	-145.5692
3	-0.6642	0.5702	-16.0423
4	-0.2069	-0.1609	-16.1340

Данные загружены корректно.

1.2. Используя train_test_split разобьем выборку на обучающую и тестовую в соотношении 80 на 20. Построим диаграмму рассеяния для проверки, что тестовая выборка соответствует обучающей (см. листинг 1.2, рисунок 1.2.1 для x1 и y, рисунок 1.2.2 для x2 и y).

Листинг 1.2 - Разделение выборки на обучающую и тестовую

```
x1 = np.array(df["x1"]).reshape(-1, 1)
x2 = np.array(df["x2"]).reshape(-1, 1)
y = np.array(df["y"]).reshape(-1, 1)

# Разобьем данные на обучающую и тестовую выборки
x1_train, x1_test, x2_train, x2_test, y_train, y_test  =
train_test_split(x1, x2, y, test_size=0.2)

# для x1
plt.scatter(x1_train, y_train, c="black", marker=".")
plt.scatter(x1_test, y_test, c="red", marker=".")
plt.legend(('обучающая выборка','тестовая выборка'))
plt.show()

# для x2
plt.scatter(x2_train, y_train, c="black", marker=".")
plt.scatter(x2_test, y_test, c="red", marker=".")
plt.legend(('обучающая выборка','тестовая выборка'))
plt.show()
```

В листинге 1.2 сначала извлекаются столбцы x1, x2 и y из датафрейма df и преобразуются в массивы NumPy. Затем используется метод reshape(-1, 1) для преобразования одномерного массива (вектора) в двумерный массив (матрицу). Это необходимо для некоторых операций с библиотекой scikit-learn, в данном случае для метода train_test_split. Так получаются матрицы x1, x2 и y.

Далее используется функция train_test_split для разбиения данных на обучающие и тестовые выборки. Параметр test_size=0.2 означает, что 20% данных будет использовано для тестовой выборки, а оставшиеся 80% — для обучающей. train_test_split принимает несколько массивов (x1, x2, y) и разбивает их синхронно, что гарантирует, что данные из разных массивов остаются согласованными по индексам после разбиения.

Затем строятся диаграммы рассеяния для x1 и y, x2 и y.

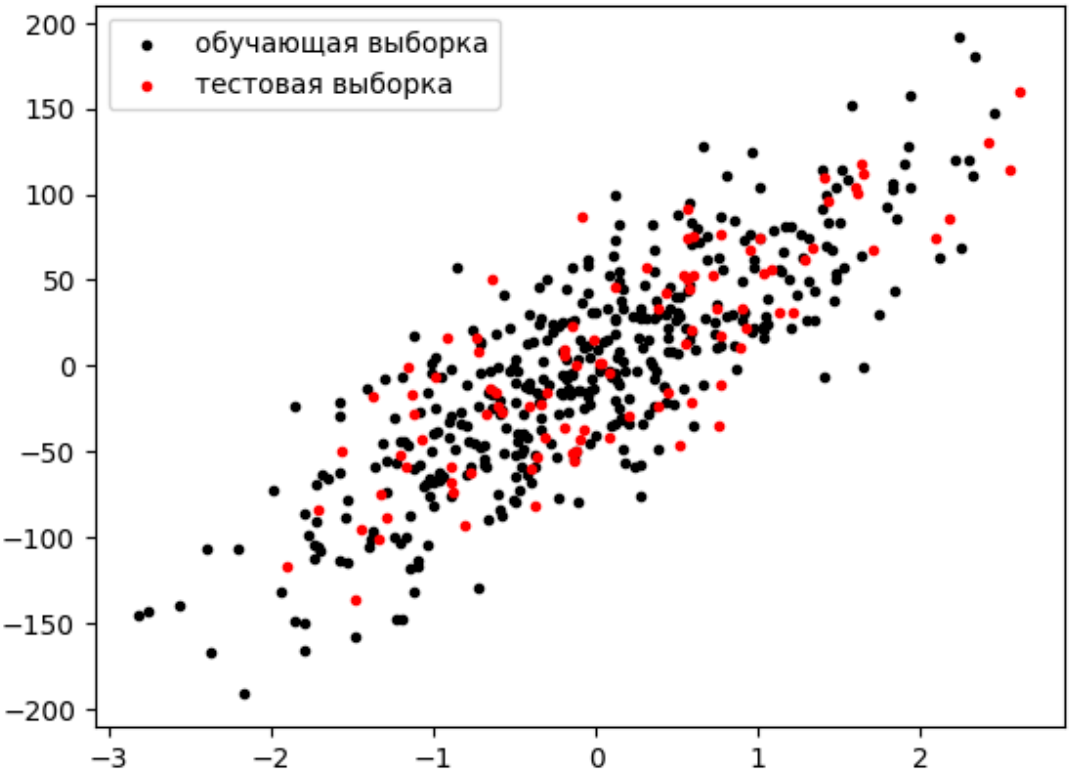


Рисунок 1.2.1 - Диаграмма рассеяния для x1 (train_test_split)

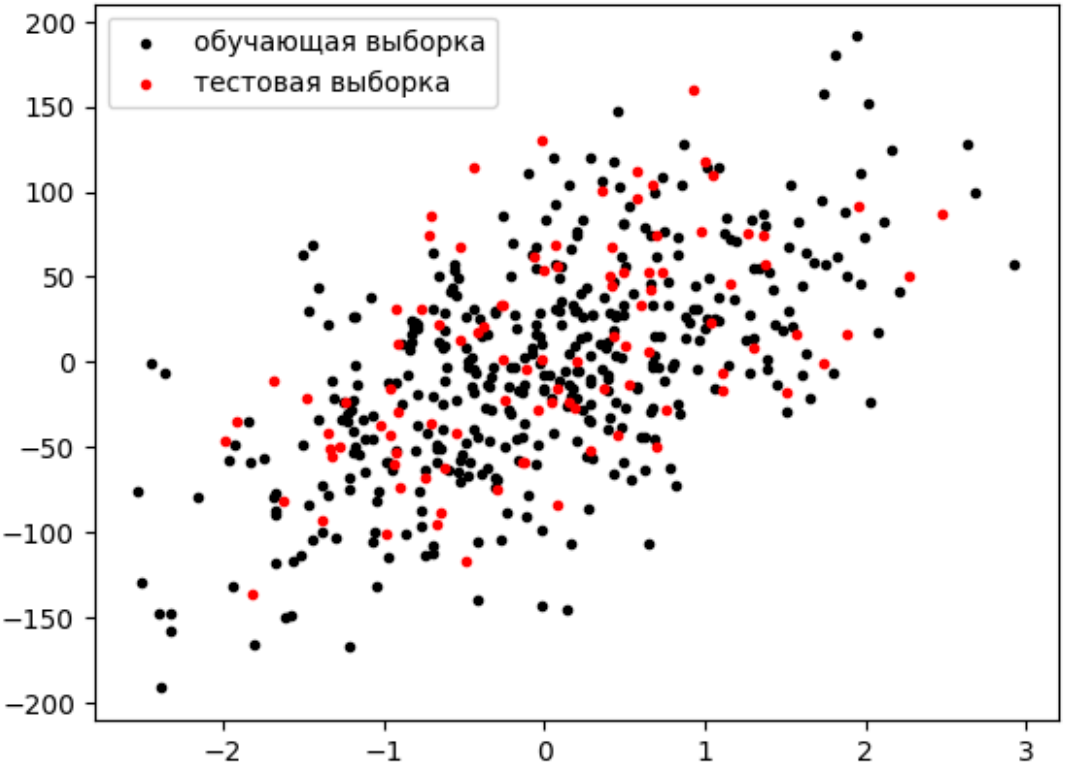


Рисунок 1.2.2 - Диаграмма рассеяния для x2 (train_test_split)

Точки тестовой выборки покрывают ту же область, что и точки обучающей выборки и плотность (концентрация) точек в тестовой выборке схожа с плотностью точек в обучающей выборке. Значит, по диаграмме рассеяния 1.2.1 и 1.2.2 можно сделать вывод, что тестовая выборка соответствует обучающей.

Если сравнивать диаграммы 1.2.1 и 1.2.2 между собой, то можно заметить, что на первой диаграмме распределение точек более сжато в линию.

1.3. Проведем линейную регрессию используя LinearRegression и получим коэффициенты регрессии (см. листинг 1.3).

Листинг 1.3 - Линейная регрессия с использованием LinearRegression

```
train_arr = np.column_stack([x1_train, x2_train])
test_arr = np.column_stack([x1_test, x2_test])
lin_reg = LinearRegression()
lin_reg.fit(train_arr, y_train)
y_train_pred = lin_reg.predict(train_arr)
y_test_pred = lin_reg.predict(test_arr)
print(lin_reg.coef_, lin_reg.intercept_)
```

В листинге 1.3 функция np.column_stack принимает несколько одномерных массивов и объединяет их по столбцам. В результате train_arr будет массивом, где каждый столбец соответствует одному из предикторов (x1_train и x2_train), test_arr будет массивом, где каждый столбец соответствует одному из предикторов (x1_test и x2_test). Далее вызывается конструктор класса LinearRegression из библиотеки scikit-learn, который создает экземпляр модели линейной регрессии. Объект lin_reg будет использоваться для обучения модели и предсказания значений. Метод fit обучает модель на обучающих данных. В результате обучения модель определяет оптимальные коэффициенты регрессии для предсказания значений y на основе x1 и x2. Метод predict использует обученную модель для предсказания значений y на основе обучающих и тестовых данных. В результате y_train_pred будет массивом предсказанных

значений для обучающей выборки, y_test_pred будет массивом предсказанных значений для тестовой выборки.

Коэффициенты регрессии: при первом предикторе - 51.55501708, при втором - 35.22042183, свободный член - 0.12859405. По данным цифрам можно сделать вывод, что первый предиктор оказывают влияние на отклик больше чем второй.

1.4. Для обучающей и тестовой выборки рассчитаем коэффициент детерминации, MAE, MAPE (см. листинг 1.4).

Все функции принимают аргументы: 1) массив истинных значений отклика (в нашем случае, это y_train для обучающей выборки и y_test для тестовой выборки) 2) массив предсказанных значений отклика(в нашем случае, это y_train_pred для обучающей выборки и y_test_pred для тестовой выборки).

Коэффициент детерминации (R^2): Число от 0 до 1, где 1 означает идеальное соответствие предсказанных значений реальным данным.

Средняя абсолютная ошибка (MAE): это значение всегда неотрицательное, чем меньше значение, тем лучше модель.

MAPE: чем меньше значение, тем точнее модель предсказывает значения в процентном выражении. Значение 0% означает, что модель предсказывает идеально.

Листинг 1.4 - Расчет коэффициентов детерминации, MAPE, MAE

```
# Коэффициент детерминации (R²) для обучающей выборки
print(r2_score(y_train, y_train_pred))

# Коэффициент детерминации (R²) для тестовой выборки
print(r2_score(y_test, y_test_pred))

# Средняя абсолютная ошибка (MAE) для обучающей выборки
print(mean_absolute_error(y_train, y_train_pred))

# Средняя абсолютная ошибка (MAE) для тестовой выборки
print(mean_absolute_error(y_test, y_test_pred))
```

```
# Средний абсолютный процент ошибки (MAPE) для обучающей выборки
print(mean_absolute_percentage_error(y_train, y_train_pred)*100)

# Средний абсолютный процент ошибки (MAPE) для тестовой выборки
print(mean_absolute_percentage_error(y_test, y_test_pred)*100)
```

Результат работы функций представлен на рисунке 1.4.

```
0.995999256617365
0.996311327786271 1
3.160417318076089
3.1144923624698424
20.364501886098783
16.83869147117745
```

Рисунок 1.4 - Результат расчета коэффициентов детерминации, MAPE, MAE

Данные метрики по сути нужны, чтобы сравнить их на обучающей и тестовой выборке, и они должны быть близки.

Можно заметить, что коэффициенты детерминации для обучающей и тестовой выборки практически совпали, а также 0.99 очень близко к 1, значит модель хорошо приближает.

Средняя абсолютная ошибка приблизительно равна 3 для обеих выборок, что свидетельствует о хорошем качестве предсказаний модели.

Наибольшее отличие значений наблюдается у метрики MAPE (20% и 17%), это связано с тем, что имеются выбросы данных. MAPE означает, насколько в среднем процентов модель ошибается от реальных значений на обучающих данных. 20% и 17% великовато для ошибки, однако, несмотря на относительно высокие значения MAPE, остальные метрики показывают хорошие результаты.

В целом можно говорить о хорошем качестве модели и ее способности обобщать.

1.5. Построим диаграмму рассеяния между предикторами и откликом. На диаграмме изобразим какое значение должно быть, и какое предсказывается (см. листинг 1.5, рис. 1.5.1 и 1.5.2).

Листинг 1.5 - Построение диаграмм рассеяния между предикторами и откликом.

```
# Предсказание значений
y_pred = lin_reg.predict(np.column_stack([x1,x2]) )

plt.scatter(x1, y, c="red", marker='.', label='train')
plt.scatter(x1, y_pred, c="blue", marker='.', label='predict')
plt.legend()
plt.grid()
plt.show()

plt.scatter(x2, y, c="red", marker='.', label='train')
plt.scatter(x2, y_pred, c="blue", marker='.', label='predict')
plt.legend()
plt.grid()
plt.show()
```

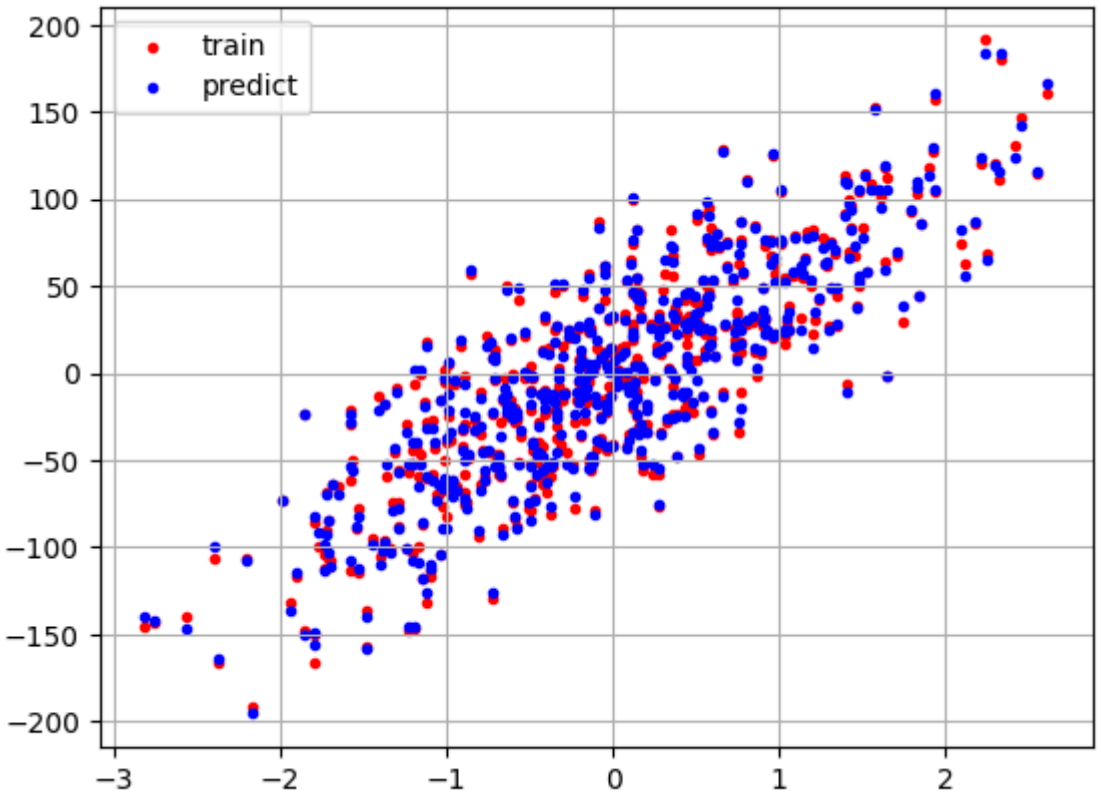


Рисунок 1.5.1 - Диаграмма рассеяния между предиктором x1 и откликом y

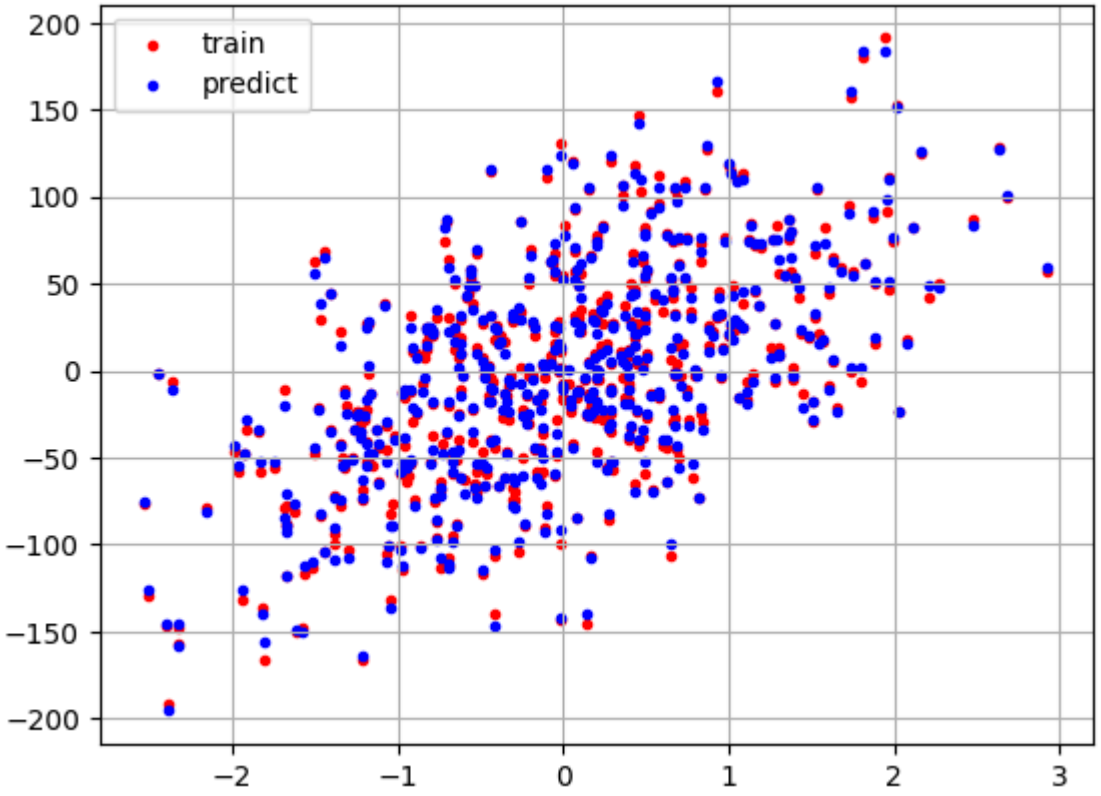


Рисунок 1.5.2 - Диаграмма рассеяния между предиктором x2 и откликом y

Визуально видно, что линейная регрессия прошла успешна,

2. Нелинейная регрессия.

2.1. Загрузим набор данных lab3_poly1.csv с помощью метода read_csv, который принимает в качестве аргумента путь до файла. Убедимся, что загрузка прошла корректно с помощью метода head, который по умолчанию выводит первые 5 строк набора данных (см. листинг 2.1 и таблицу 2.1).

Листинг 2.1 - Загрузка и проверка данных lab3_poly1.csv

```
df = pd.read_csv('lab3_poly1.csv')
df.head()
```

Таблица 2.1 - Результат работы метода head

	x	y
0	-1.2547	3.4811
1	0.8172	-7.4786
2	-0.4300	-3.8623
3	1.4550	-20.3935
4	-0.6722	-5.8812

Данные загружены корректно.

2.2. Используя train_test_split разобьем выборку на обучающую и тестовую в соотношении 80 на 20. Построим диаграмму рассеяния для проверки, что тестовая выборка соответствует обучающей (см. листинг 2.2, рисунок 2.2).

Листинг 2.2 - Разделение выборки на обучающую и тестовую

```
x = np.array(df["x"]).reshape(-1, 1)
y = np.array(df["y"]).reshape(-1, 1)

# Разобьем данные на обучающую и тестовую выборки
x_train, x_test, y_train, y_test  = train_test_split(x, y,
test_size=0.2)

plt.scatter(x_train, y_train, c="black", marker=".")
plt.scatter(x_test, y_test, c="red", marker=".")
plt.legend(('обучающая выборка','тестовая выборка'))
plt.show()
```

В листинге 2.2 сначала извлекаются столбцы x и y из датафрейма df и преобразуются в массивы NumPy. Затем используется метод reshape(-1, 1) для преобразования одномерного массива (вектора) в двумерный массив (матрицу). Это необходимо для некоторых операций с библиотекой scikit-learn, в данном случае для метода train_test_split. Так получаются матрицы x и y.

Далее используется функция train_test_split для разбиения данных на обучающие и тестовые выборки. Параметр test_size=0.2 означает, что 20% данных будет использовано для тестовой выборки, а оставшиеся 80% — для обучающей. train_test_split принимает несколько массивов (x, y) и разбивает их синхронно, что гарантирует, что данные из разных массивов остаются согласованными по индексам после разбиения.

Затем строится диаграмма рассеяния для предиктора x.

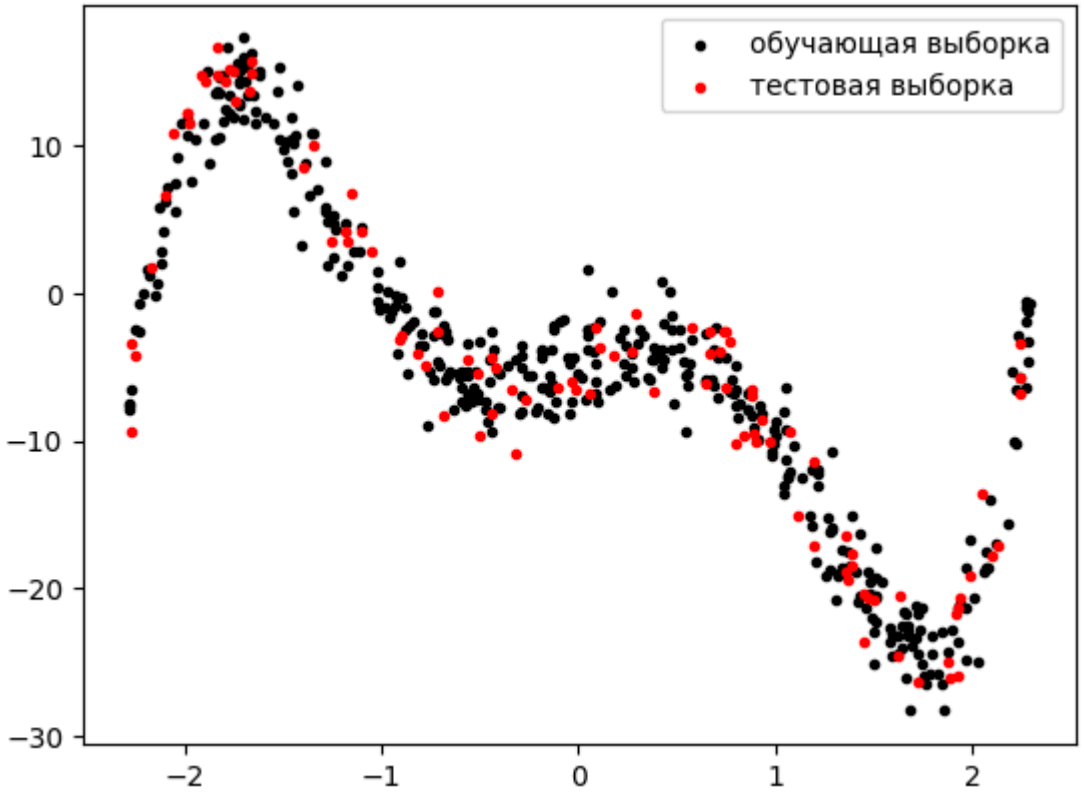


Рисунок 2.2 - Диаграмма рассеяния для предиктора x (train_test_split)

Точки тестовой выборки покрывают ту же область, что и точки обучающей выборки и плотность (концентрация) точек в тестовой выборке схожа с плотностью точек в обучающей выборке. Значит, по диаграмме рассеяния 2.2 можно сделать вывод, что тестовая выборка соответствует обучающей.

2.3. Проверим работу стандартной линейной регрессии на загруженных данных. Построим диаграмму рассеяния данных с выделенной полученной линией регрессии (см. листинг 2.3 и рисунок 2.3).

Листинг 2.3 - Построение диаграммы рассеяния с выделенной полученной линией регрессии

```
lin_reg = LinearRegression()
lin_reg.fit(x_train, y_train)
y_train_pred = lin_reg.predict(x_train)
y_test_pred = lin_reg.predict(x_test)

print(lin_reg.coef_, lin_reg.intercept_)

plt.scatter(x_train, y_train, c="black", marker='.')
plt.scatter(x_test, y_test, c="red", marker='.')
plt.plot(x_test, y_test_pred, 'b-')
plt.legend(('train', 'test', 'predict'))
plt.grid()
plt.show()
```

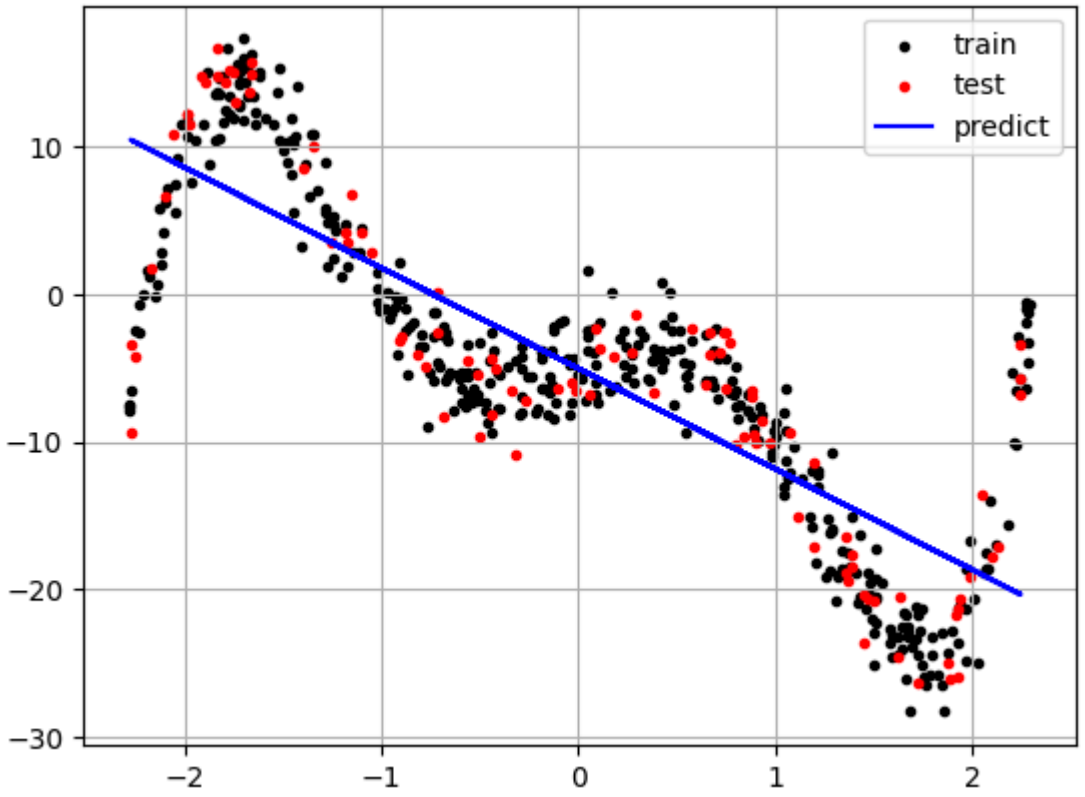


Рисунок 2.3 - Диаграмма рассеяния с выделенной полученной линией регрессии

Анализируя рисунок 2.3 можно сделать вывод, стандартная линейная регрессия на загруженных нелинейных данных сработает, однако у многих точек остаток линейной регрессии очень большой. И данную нелинейную зависимость никак нельзя описать линией.

2.4. Конструируя полиномиальный признаки для разных степеней полинома, найдем степень полинома наилучшим образом аппроксимирующая данные (см. листинг 2.4). Создаются пустые списки train_r2 и test_r2, в которых будут храниться значения коэффициента детерминации для обучающей и тестовой выборок соответственно. Цикл перебора степеней полинома: для каждой степени полинома от 1 до 50 выполняются следующие действия: 1) создаются полиномиальные признаки с помощью PolynomialFeatures, 2) модель линейной регрессии обучается на обучающих данных, 3) для обучающей и тестовой выборок делаются предсказания с помощью обученной модели, 4) рассчитывается коэффициент детерминации для обучающей и тестовой выборок, 5) значения коэффициента детерминации добавляются в соответствующие списки train_r2 и test_r2. Далее происходит построение графика.

График зависимости коэффициента детерминации от степени полинома(на одном графике изобразим линии для обучающей и тестовой выборки отдельно) смотреть на рисунке 2.4.

Листинг 2.4 - Построение графика зависимости коэффициента детерминации от степени полинома

```
# Массив для хранения значений R^2
train_r2 = []
test_r2 = []

# Перебираем степени полинома от 1 до 50
degrees = np.arange(1, 51)
for degree in degrees:
    # Создаем полиномиальные признаки
    poly = PolynomialFeatures(degree)
```

```
    x_train_poly = poly.fit_transform(x_train)
    x_test_poly = poly.fit_transform(x_test)

    # Обучаем модель
    model = LinearRegression()
    model.fit(x_train_poly, y_train)

    # Предсказываем значения
    y_train_pred = model.predict(x_train_poly)
    y_test_pred = model.predict(x_test_poly)

    # Вычисляем R^2
    train_r2.append(r2_score(y_train, y_train_pred))
    test_r2.append(r2_score(y_test, y_test_pred))

# Строим график
plt.plot(degrees, train_r2, label='train')
plt.plot(degrees, test_r2, label='test')
plt.xlabel('Степень полинома')
plt.ylabel('Коэффициент детерминации (R^2)')
plt.title('Зависимость R^2 от степени полинома')
plt.grid()
plt.legend()
plt.show()
```

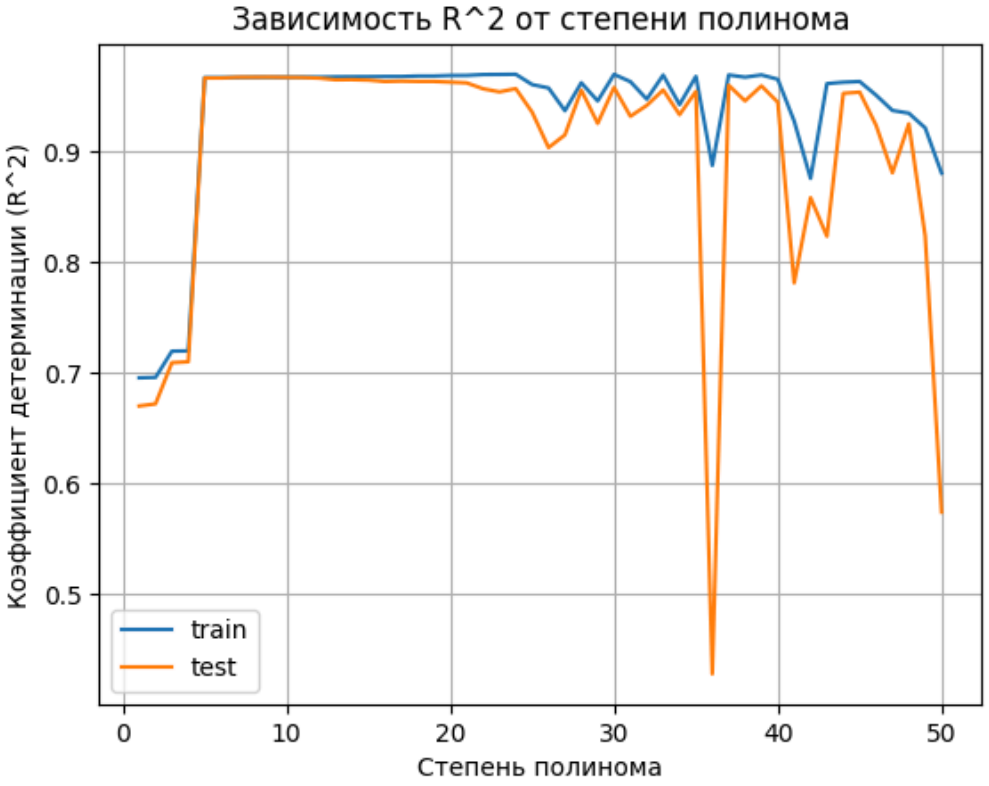


Рисунок 2.4 - Графика зависимости коэффициента детерминации от степени полинома

Чтобы более точно проанализировать, увеличим рисунок, рассмотрев только до 20 степени (смотреть рисунок 2.4.1).

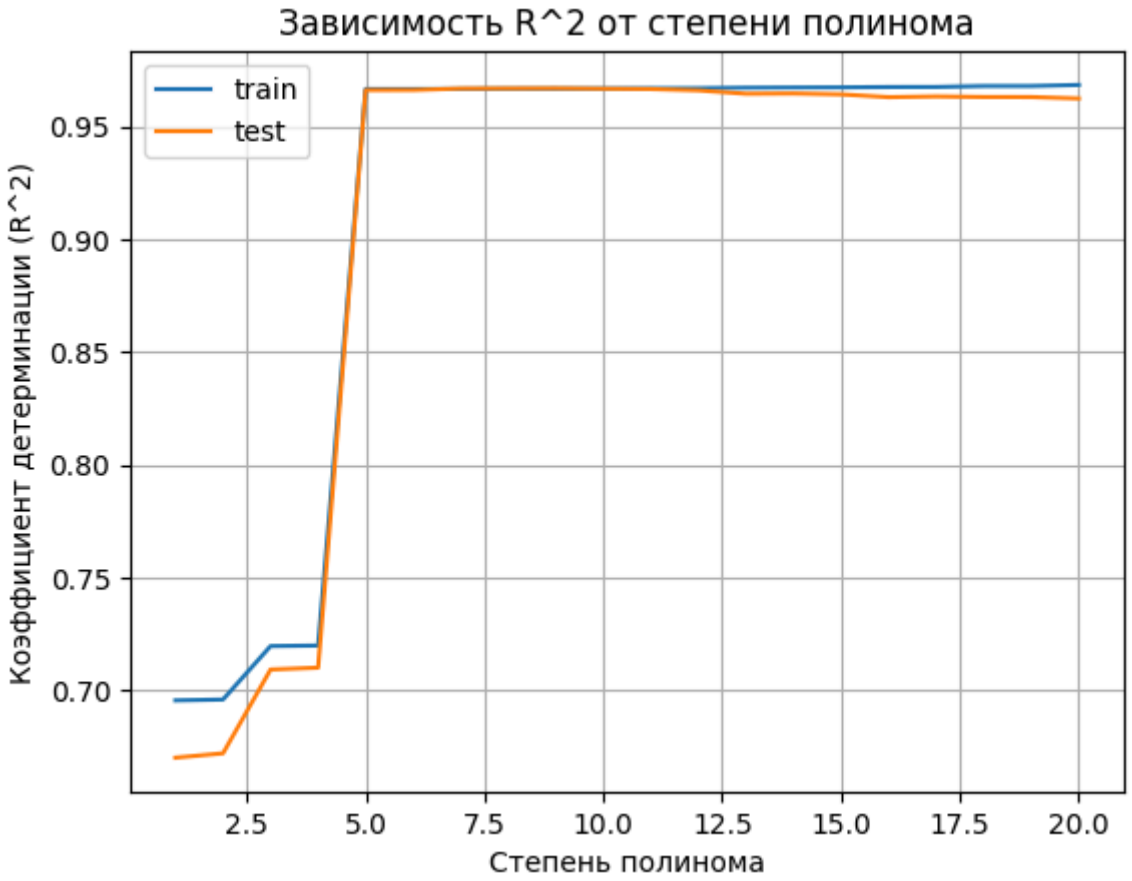


Рисунок 2.4.1 - Графика зависимости коэффициента детерминации от степени полинома (увеличенный)

По рисунку 2.4 видно, что при степени полинома выше 5, коэффициент детерминации не растет. Можно сделать вывод, что степень полинома наилучшим образом аппроксимирующая данные равна 5.

Один из признаков переобучения - высокий R^2 на обучающих данных по сравнению с тестовыми данными. Это указывает на то, что модель выучила специфические детали обучающего набора данных, которые не присутствуют в тестовом наборе. В нашем случае примерно с 12 степени начинается переобучение.

Если сравнивать графики зависимости коэффициента детерминации от степени полинома для обучающей и тестовой выборки, то можно сказать, что начиная примерно с четвертой по 12-ую степень полинома они практически совпадают. На более низких степенях они разнятся, но все равно разность небольшая - в пределах 0.05, а на более высоких разность достигает 0,8, что очень велико.

2.5. Для пятой степени полинома (так как по рисунку 2.4.1 степени выше не дают результат лучше) рассчитаем и проанализируем полученные коэффициенты. Рассчитаем значение метрик коэффициент детерминации, MAPE, MAE (см. листинг 2.5 и рис. 2.5).

Листинг 2.5 - Расчет коэффициентов и метрик

```
# Выбираем степень полинома
degree = 5

# Создаем полиномиальные признаки
poly = PolynomialFeatures(degree)
x_train_poly = poly.fit_transform(x_train)
x_test_poly = poly.transform(x_test)

# Обучаем модель линейной регрессии
model = LinearRegression()
model.fit(x_train_poly, y_train)

# Предсказываем значения
y_train_pred = model.predict(x_train_poly)
y_test_pred = model.predict(x_test_poly)

# Вычисляем коэффициент детерминации (R^2)
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)

# Вычисляем MAE
train_mae = mean_absolute_error(y_train, y_train_pred)
test_mae = mean_absolute_error(y_test, y_test_pred)

# Вычисляем MAPE
train_mape = mean_absolute_percentage_error(y_train,
y_train_pred)*100
test_mape = mean_absolute_percentage_error(y_test,
y_test_pred)*100
```

```
print("Коэффициенты модели:")
print(model.coef_[:,1:], model.intercept_)
print("\nКоэффициент детерминации (R^2):")
print("Обучающая выборка:", train_r2)
print("Тестовая выборка:", test_r2)
print("\nMAPE (Mean Absolute Percentage Error):")
print("Обучающая выборка:", train_mape)
print("Тестовая выборка:", test_mape)
print("\nMAE (Mean Absolute Error):")
print("Обучающая выборка:", train_mae)
print("Тестовая выборка:", test_mae)
```

```
Коэффициенты модели:
[[  4.26337074  -0.18000955 -11.1762102    0.01458334   2.03768706]] [-4.84396719]

Коэффициент детерминации (R^2):
Обучающая выборка: 0.9662792009409327
Тестовая выборка: 0.9664485201119905

MAPE (Mean Absolute Percentage Error):
Обучающая выборка: 83.9805848342875
Тестовая выборка: 88.11425391035903

MAE (Mean Absolute Error):
Обучающая выборка: 1.5538324712554772
Тестовая выборка: 1.723002337815209
```

Рисунок 2.5 - Коэффициенты и метрики

4.26337074 -0.18000955 -11.1762102   0.01458334   2.03768706]] [-4.84396719]

Анализируя коэффициенты, можно сказать, что самое сильное влияние оказывает третий коэффициент (в отрицательную сторону), далее первый коэффициент (в положительную сторону) и пятый (в положительную сторону). Меньше всего оказывают влияние второй (в десятки раз меньше остальных, в отрицательную сторону) и четвертый (в сотни раз меньше остальных, в положительную сторону). Так как второй и четвертый коэффициент слишком маленькие, то можно считать их равными 0.

Итого уравнение примет вид: $y = -5 + 4x - 11x^3 + 2x^5$.

Данные метрики (коэффициент детерминации, MAPE, MAE) по сути нужны, чтобы сравнить их на обучающей и тестовой выборке, и они должны быть близки.

Можно заметить, что коэффициенты детерминации для обучающей и тестовой выборки практически совпали, а также 0.97 очень близко к 1, значит модель хорошо приближает.

Средняя абсолютная ошибка приблизительно равна равна 1.55 для обучающей выборки и 1.72 (значения отличаются на 0.17).

MAPE означает, насколько в среднем процентов модель ошибается от реальных значений на обучающих данных. 88% и 84% очень большое значения для ошибки, это связано с тем, что имеются большие выбросы данных (в пункте 2.7 рассмотрим где именно расположены эти выбросы).

2.6. Для выбранной степени полинома построим диаграмму рассеяния данных с линией соответствующей полученному полиному (см. листинг 2.6 и рис. 2.6). В листинге 2.6 сначала отрисовываются диаграммы рассеивания обучающих и тестовых данных. Затем linspace создает массив x_range из 100 равномерно распределенных значений от минимального до максимального значения x и преобразует этот массив в двумерный массив с одной колонкой, чтобы он соответствовал формату, ожидаемому моделью. Далее poly.transform() преобразует массив x_range в полиномиальные признаки пятой степени. Предсказываются значения y для преобразованного массива x_range_poly с использованием обученной модели. Строится линия на графике для предсказанных значений y.

Листинг 2.6 - Диаграмма рассеяния с линией соответствующей полученному полиному

```
# Диаграмма рассеяния обучающих данных
plt.scatter(x_train, y_train, color='blue', label='Обучающая
выборка', marker='.')
```

```
# Диаграмма рассеяния тестовых данных
plt.scatter(x_test, y_test, color='green', label='Тестовая
выборка', marker='.')

# Линия полинома
x_range = np.linspace(x.min(), x.max(), 100).reshape(-1, 1)
x_range_poly = poly.transform(x_range)
y_range_pred = model.predict(x_range_poly)
plt.plot(x_range, y_range_pred, color='red',
label='Полиномиальная регрессия (степень 5)')

plt.title('Диаграмма рассеяния с линией полиномиальной
регрессии')
plt.legend()
plt.grid()
plt.show()
```

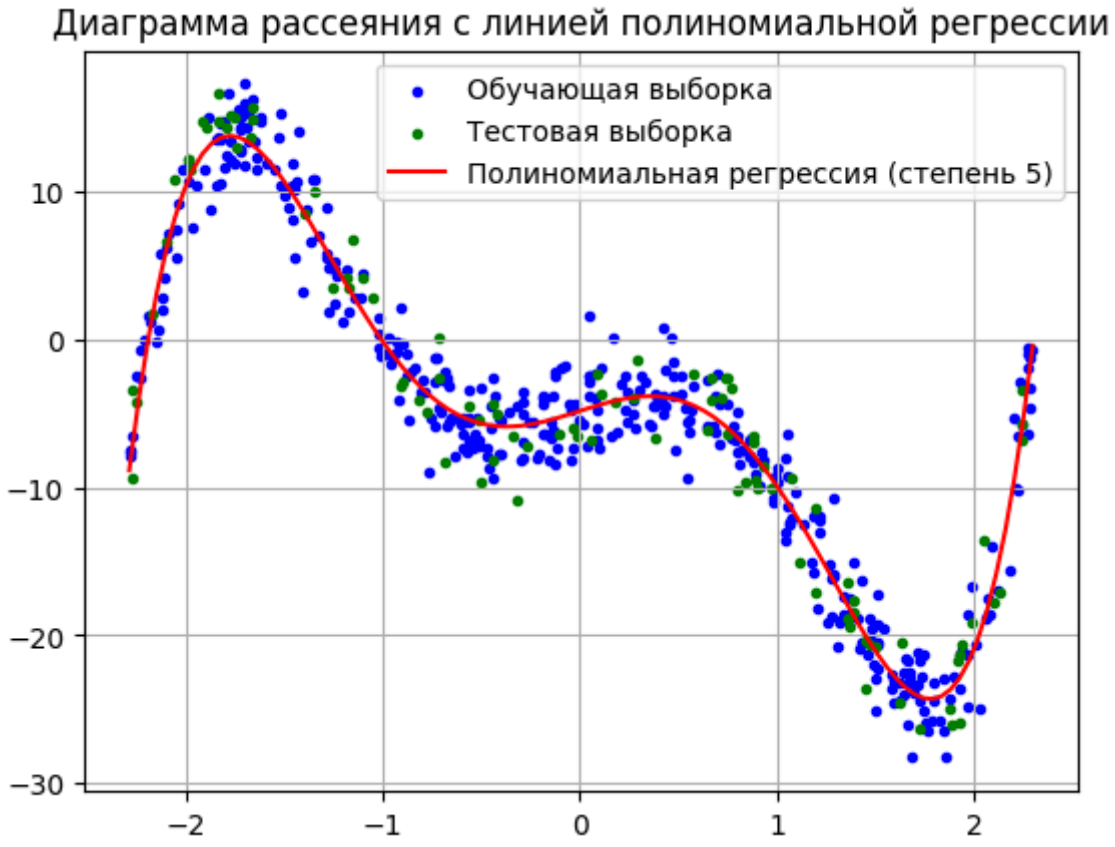


Рисунок 2.6 - Диаграмма рассеяния с линией соответствующей полученному полиному

Если говорить о качестве аппроксимации, то она выполнена успешно, однако все же присутствуют шумы приблизительно на промежутках [-1.8, -1.4], [-0.8, 0.5], [1.5, 2], что и повлияло на плохое значение MAPE.

3. Оценка модели регрессии

3.1. Загрузим набор данных Student_Performance.csv с помощью метода read_csv, который принимает в качестве аргумента путь до файла. Убедимся, что загрузка прошла корректно с помощью метода head, который по умолчанию выводит первые 5 строк набора данных (см. листинг 3.1 и таблицу 3.1).

Листинг 3.1 - Загрузка и проверка данных Student_Performance.csv

```
df = pd.read_csv('Student_Performance.csv')
df.head()
```

Таблица 3.1 - Результат работы метода head

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	Yes	9	1	91.0
1	4	82	No	4	2	65.0
2	8	51	Yes	7	2	45.0
3	5	52	Yes	5	2	36.0
4	7	75	No	8	5	66.0

Данные загружены корректно.

3.2. Проведем предобработку набора данных - замена текстовых данных, удаление null значений, удаление дубликатов (см. листинг 3.2.1 и таблицу 3.2.1). В нашем наборе данных только один столбец имел текстовые

данные (Extracurricular Activities), поэтому в нем произвели замену: “Yes” на 1, “No” на 0. Для удаления null значений использовался метод dropna. Для удаления дупликтов использовался метод drop_duplicates. Аргумент inplace=True в методах pandas означает, что операция будет выполнена непосредственно на оригинальном DataFrame, а не создаст его копию.

Листинг 3.2.1 - Предобработка данных

```
# 1. Замена текстовых данных на числовые ("Yes" на 1, "No" на 0)
df['Extracurricular Activities'] = df['Extracurricular
Activities'].map({'Yes': 1, 'No': 0})

# 2. Удаление null значений (если такие есть)
df.dropna(inplace=True)

# 3. Удаление дубликатов
df.drop_duplicates(inplace=True)

df.head()
```

Таблица 3.2.1 - Результат работы метода head после предобработки данных

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	1	9	1	91.0
1	4	82	0	4	2	65.0
2	8	51	1	7	2	45.0
3	5	52	1	5	2	36.0
4	7	75	0	8	5	66.0

Используя train_test_split разобьем выборку на обучающую и тестовую в соотношении 80 на 20 (см. листинг 3.2.2).

Листинг 3.2.2 - Разбиение выборки на обучающую и тестовую

```
# 'Performance Index' - это целевая переменная (отклик)
X = df.drop('Performance Index', axis=1)
y = df['Performance Index']

# Разделяем данные на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
```

Параметр test_size=0.2 означает, что 20% данных будет использовано для тестовой выборки, а оставшиеся 80% — для обучающей.

3.3. Построим модель, которая будет предсказывать значение признака Performance Index на основе остальных признаков. Чтобы выбрать модель, нужно сначала решить какая у нас регрессия, линейная или нет. Для этого построим график зависимости R^2 от степени полинома (см. листинг 3.3). График представлен на рисунке 3.3.

Листинг 3.3 - Построение графика зависимости R^2 от степени полинома

```
# Массив для хранения значений R^2
test_r2 = []

degrees = np.arange(1, 11)
for degree in degrees:
    # Создаем полиномиальные признаки
    poly = PolynomialFeatures(degree)
    x_train_poly = poly.fit_transform(X_train)
    x_test_poly = poly.fit_transform(X_test)

    # Обучаем модель
    model = LinearRegression()
    model.fit(x_train_poly, y_train)

    # Предсказываем значения
    y_train_pred = model.predict(x_train_poly)
    y_test_pred = model.predict(x_test_poly)

    # Вычисляем R^2
    test_r2.append(r2_score(y_test, y_test_pred))

# Строим график
plt.plot(degrees, test_r2)
```

```
plt.xlabel('Степень полинома')
plt.ylabel('Коэффициент детерминации (R^2)')
plt.title('Зависимость R^2 от степени полинома')
plt.grid()
plt.show()
```

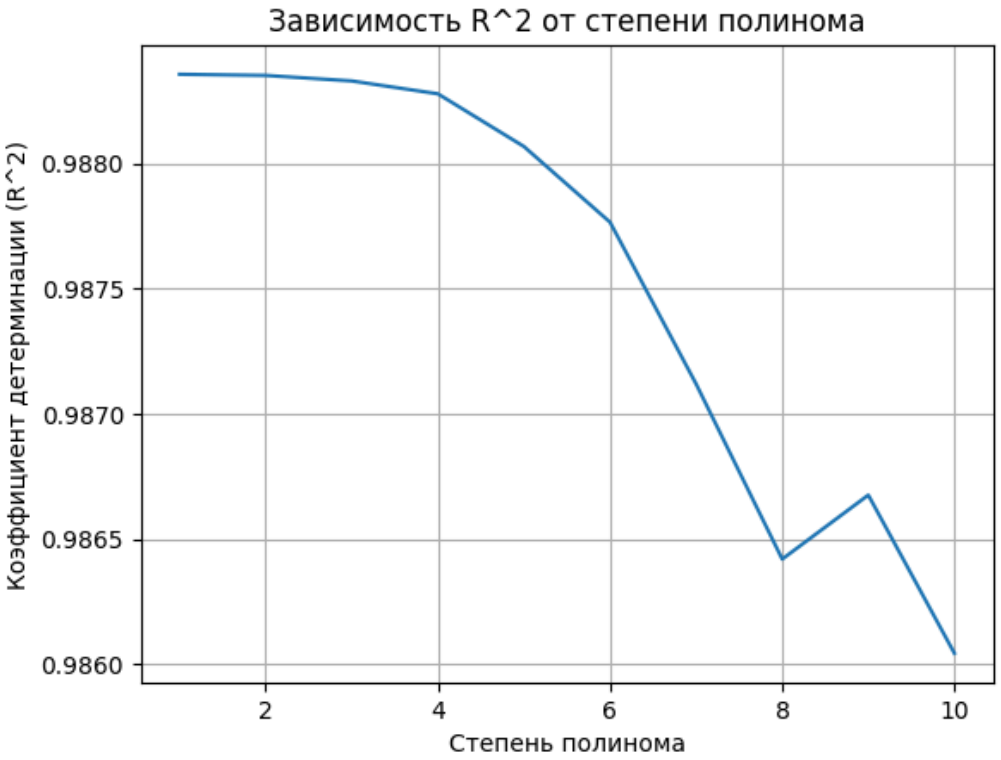


Рисунок 3.3 - График зависимости R^2 от степени полинома

По графику видно, что наибольший коэффициент детерминации при степени равной 1. В итоге получаем, что для данного набора данных регрессия линейная.

Проведем линейную регрессию используя LinearRegression и получим коэффициенты регрессии, а также рассчитаем метрики (см. листинг 3.3 и см. рис. 3.3).

Листинг 3.3 - Линейная регрессия с использованием LinearRegression

```
# Обучение модели линейной регрессии
model = LinearRegression()
```

```
model.fit(X_train, y_train)

# Предсказание значений
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)

# Вычисление метрик
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)
train_mae = mean_absolute_error(y_train, y_train_pred)
test_mae = mean_absolute_error(y_test, y_test_pred)
train_mape = mean_absolute_percentage_error(y_train,
y_train_pred) * 100
test_mape = mean_absolute_percentage_error(y_test, y_test_pred) *
100

# Вывод результатов
print("Коэффициенты модели:")
print(model.coef_, model.intercept_)
print("Свободный член (intercept):", model.intercept_)
print("\nКоэффициент детерминации (R²):")
print("Обучающая выборка:", train_r2)
print("Тестовая выборка:", test_r2)
print("\nMAE (Mean Absolute Error):")
print("Обучающая выборка:", train_mae)
print("Тестовая выборка:", test_mae)
print("\nMAPE (Mean Absolute Percentage Error):")
print("Обучающая выборка:", train_mape)
print("Тестовая выборка:", test_mape)
```

```
Коэффициенты модели:
[2.84857195 1.01876122 0.59757815 0.4771185  0.19456297]
Свободный член (intercept): -34.050793230676

Коэффициент детерминации (R²):
Обучающая выборка: 0.9886469263981209
Тестовая выборка: 0.9888105950354005

MAE (Mean Absolute Error):
Обучающая выборка: 1.6283808000294349
Тестовая выборка: 1.6029225858863907

MAPE (Mean Absolute Percentage Error):
Обучающая выборка: 3.482796717514352
Тестовая выборка: 3.411380457167048
```

Рисунок 3.3 - Рассчитанные коэффициенты и метрики

В листинге 3.3 метод fit обучает модель на обучающих данных. В результате обучения модель определяет оптимальные коэффициенты регрессии

для предсказания значений y. Метод predict использует обученную модель для предсказания значений y на основе обучающих и тестовых данных. В результате y_train_pred будет массивом предсказанных значений для обучающей выборки, y_test_pred будет массивом предсказанных значений для тестовой выборки.

Коэффициенты регрессии: 2.84857195, 1.01876122, 0.59757815, 0.4771185, 0.19456297, свободный член: -34.050793230676. По данным цифрам можно сделать вывод, что предикторы Hours Studied и Previous Scores оказывают влияние на отклик (Performance Index) больше всего. Затем чуть меньше оказывают влияние Extracurricular Activities и Sleep Hours. Меньше всего влияет предиктор Sample Question Papers Practiced.

В листинге 3.3 для обучающей и тестовой выборки также рассчитаны коэффициент детерминации, MAE, MAPE (результат смотреть на рисунке 3.3).

Коэффициент детерминации (R^2): Число от 0 до 1, где 1 означает идеальное соответствие предсказанных значений реальным данным.

Средняя абсолютная ошибка (MAE): это значение всегда неотрицательное, чем меньше значение, тем лучше модель.

MAPE: чем меньше значение, тем точнее модель предсказывает значения в процентном выражении. Значение 0% означает, что модель предсказывает идеально.

Данные метрики по сути нужны, чтобы сравнить их на обучающей и тестовой выборке, и они должны быть близки.

Можно заметить, что коэффициенты детерминации для обучающей и тестовой выборки практически совпали, а также 0.988 очень близко к 1, значит модель хорошо приближает.

Средняя абсолютная ошибка приблизительно равна 1.6 для обеих выборок (отличие на 2 сотых), что свидетельствует о хорошем качестве предсказаний модели (значения почти совпадают).

Метрика MAPE приблизительно равна 3.5% для обеих выборок (отличие на 7 сотых), что свидетельствует о хорошем качестве предсказаний модели (значения почти совпадают и ошибка очень мала, в пунктах 1 и 2 ошибки достигали 20%).

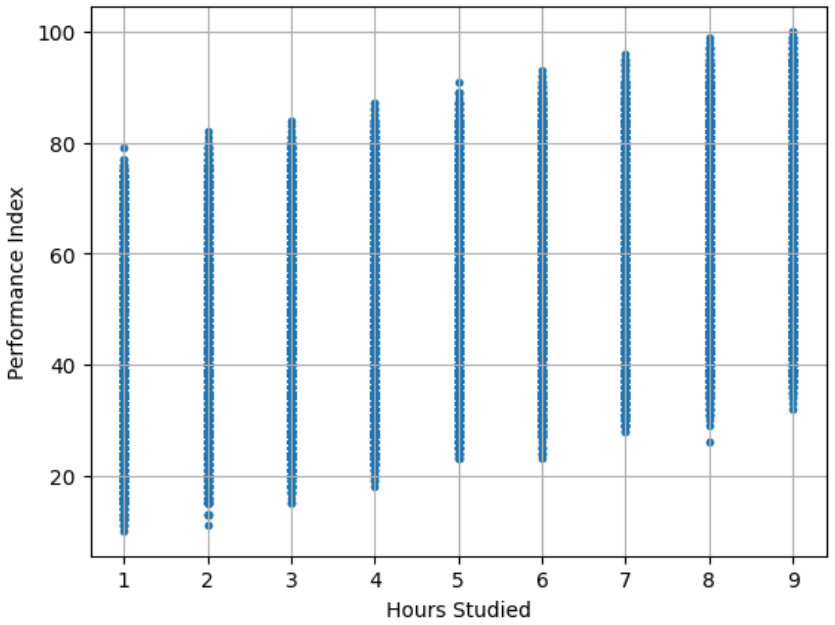
Можно говорить о хорошем качестве модели и ее способности обобщать.

Линейная регрессия на данном наборе данных сработала отлично, показания всех метрик очень хорошие и практически совпадают для обучающей и тестовой выборки.

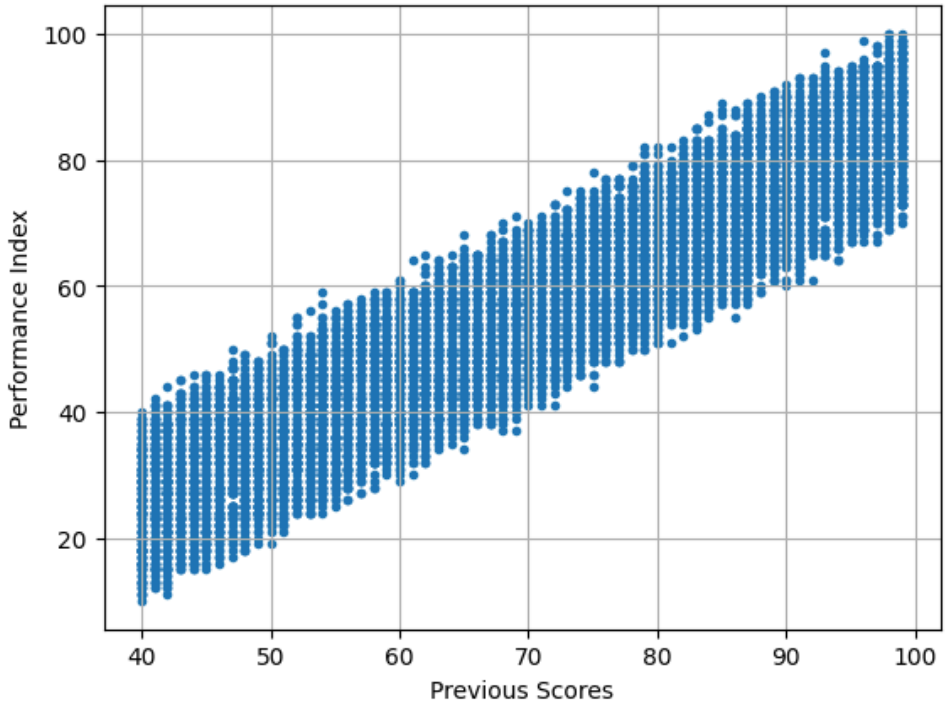
3.4. Проанализируем полученную модель. Вначале построим графики рассеяния без предсказания (см. листинг 3.4.1 и рисунки 3.4.1 – 3.4.5).

Листинг 3.4.1 - Графики рассеяния без предсказания

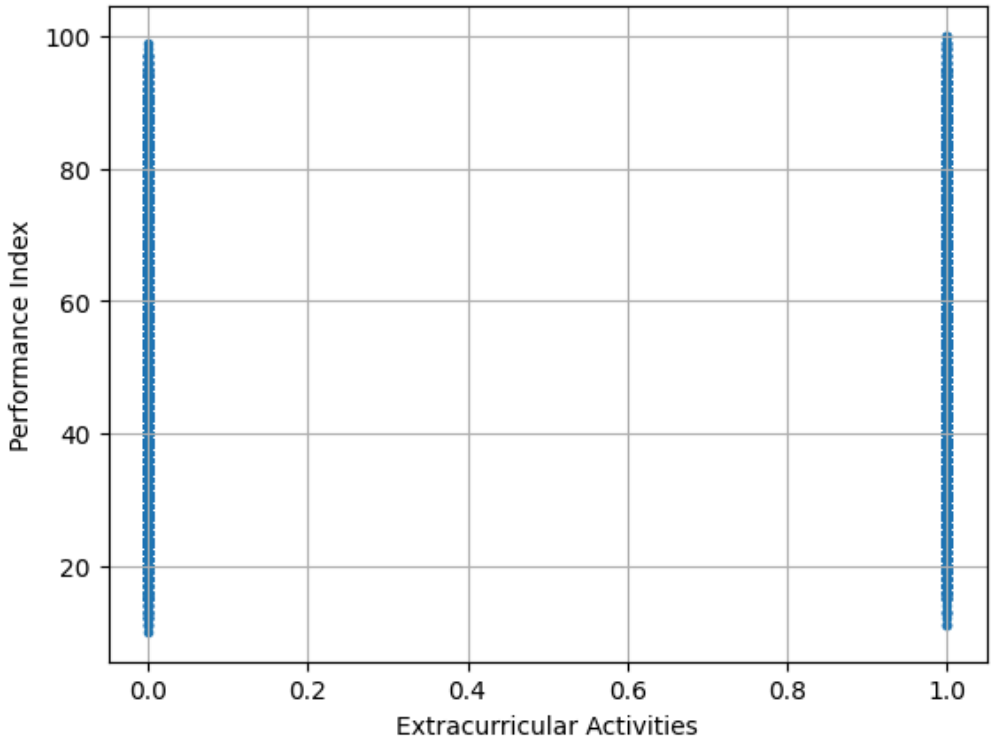
```
# Построение диаграмм рассеяния для каждого признака
for column in X.columns:
    plt.scatter(X[column], y, marker='.')
    plt.xlabel(column)
    plt.ylabel('Performance Index')
    plt.grid()
    plt.show()
```



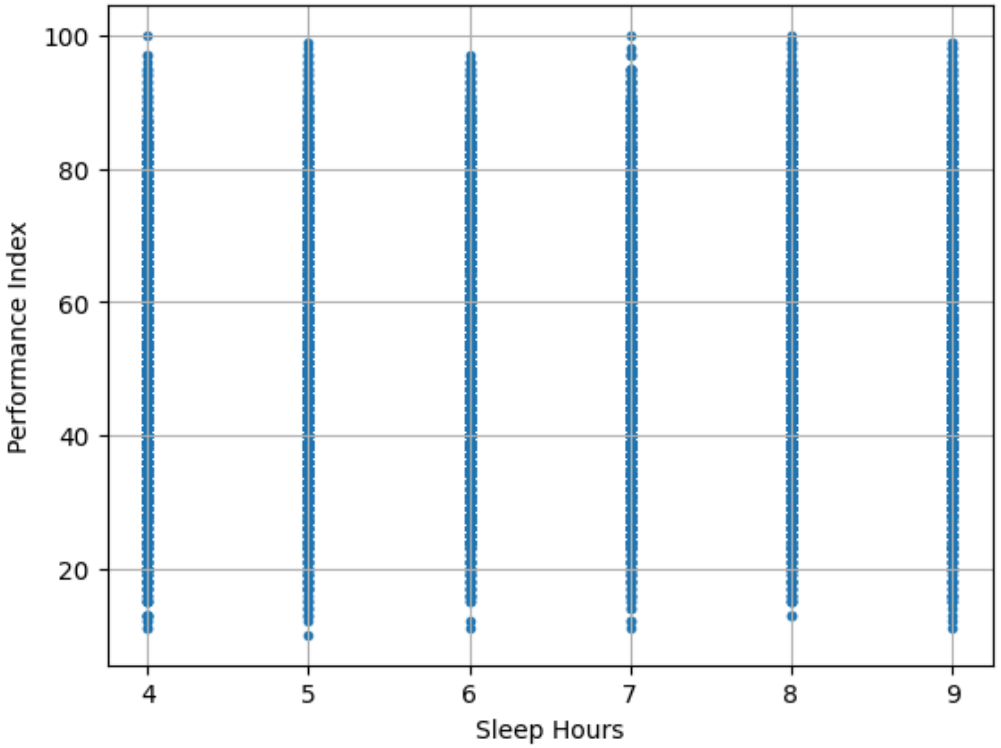
Листинг 3.4.1 - График рассеяния без предсказания для Hours Studied



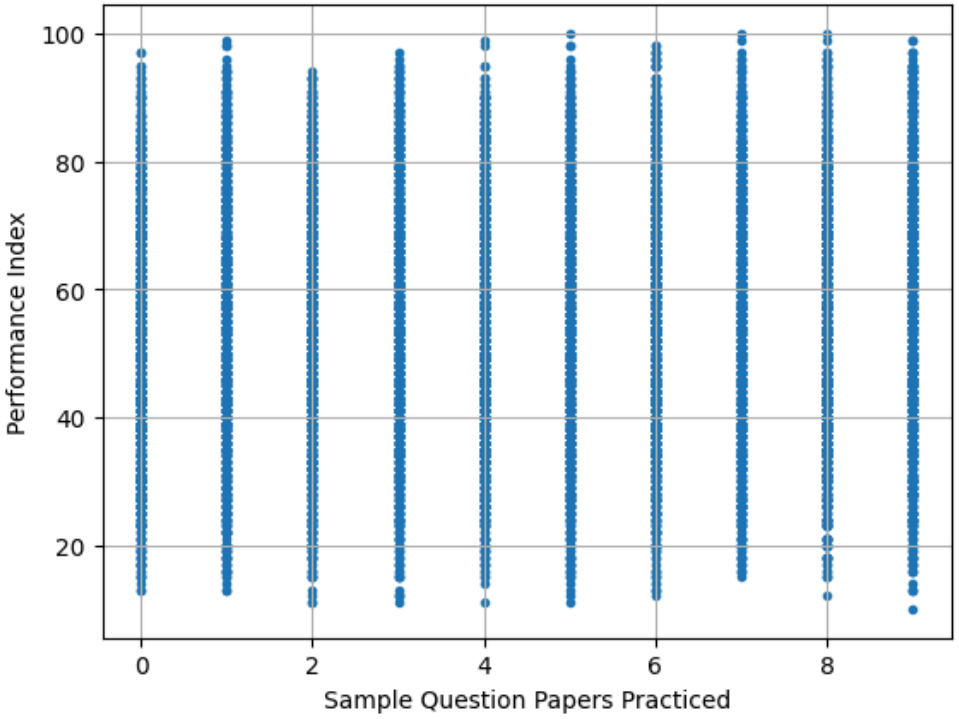
Листинг 3.4.2 - График рассеяния без предсказания для Previous Scores



Листинг 3.4.3 - График рассеяния без предсказания для Extracurricular Activities



Листинг 3.4.4 - График рассеяния без предсказания для Sleep Hours

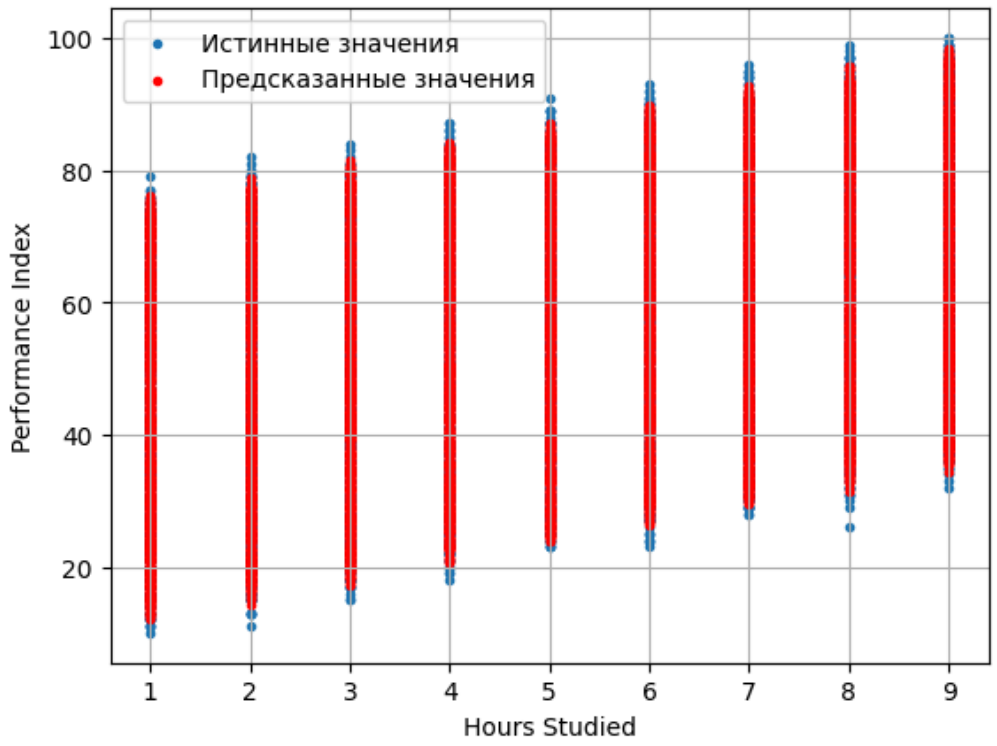


Листинг 3.4.5 - График рассеяния без предсказания для Sample Question Papers Practiced

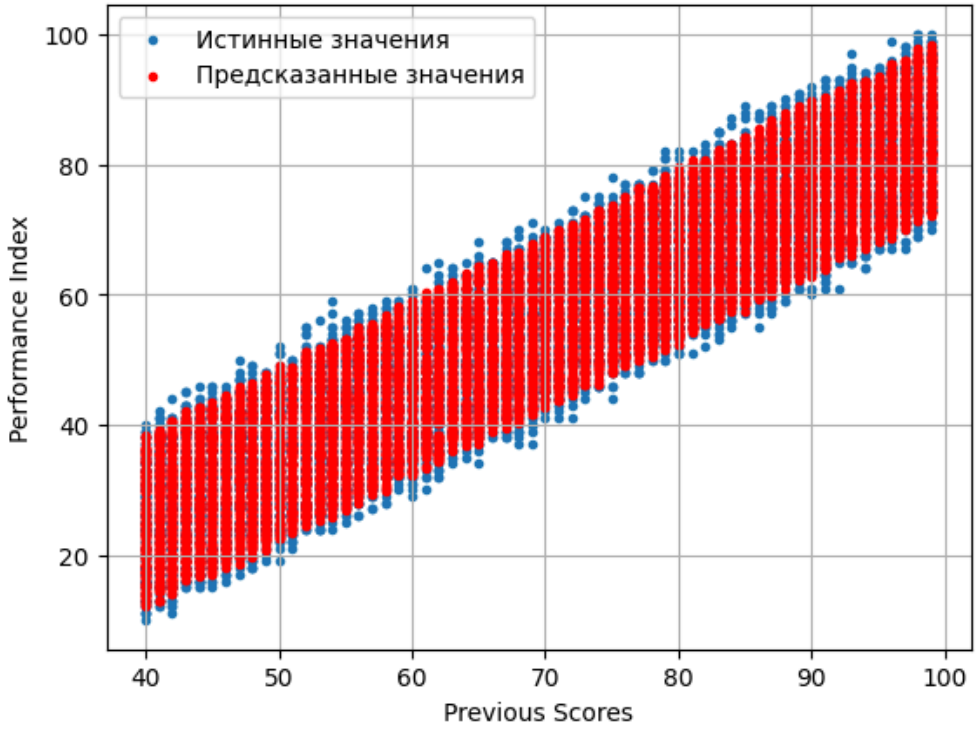
Теперь построим графики рассеяния с предсказанием (см. листинг 3.4.2 и рисунки 3.4.6 – 3.4.10).

Листинг 3.4.2 - Графики рассеяния с предсказанием

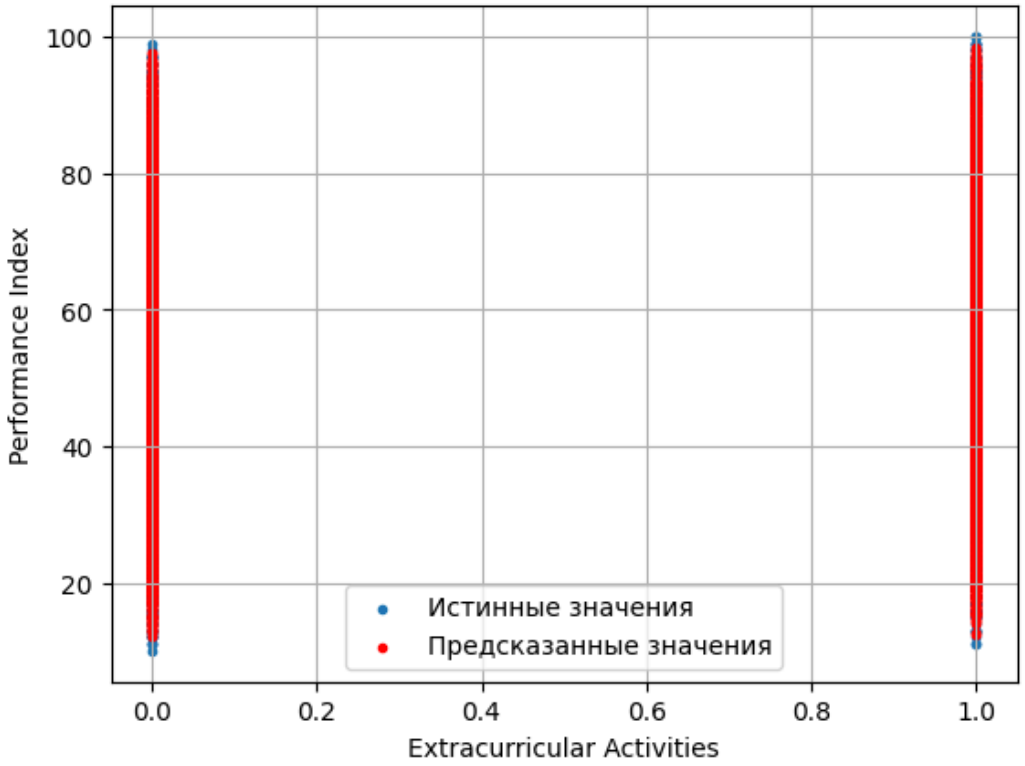
```
# Предсказание значений
y_pred = model.predict(X)

# Построение диаграмм рассеяния для каждого признака с
предсказаниями
for column in X.columns:
    plt.scatter(X[column], y, marker='.', label='Истинные
значения')
    plt.scatter(X[column], y_pred, marker='.', color='red',
label='Предсказанные значения')
    plt.xlabel(column)
    plt.ylabel('Performance Index')
    plt.grid()
    plt.legend()
    plt.show()
```



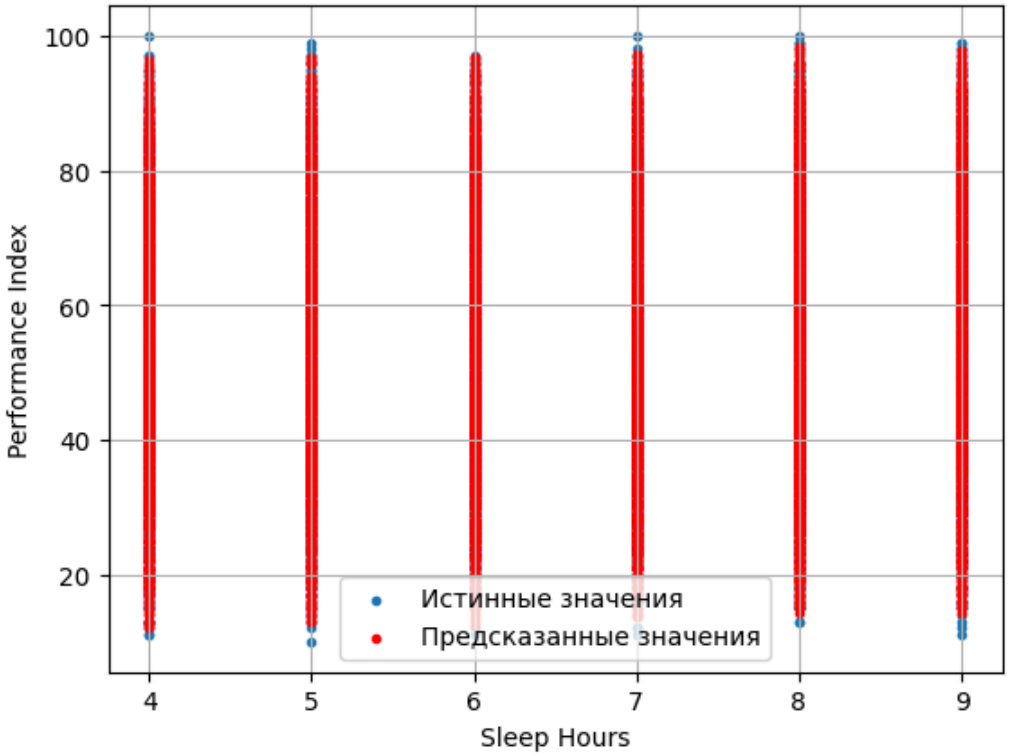
Листинг 3.4.6 - График рассеяния с предсказанием для Hours Studied



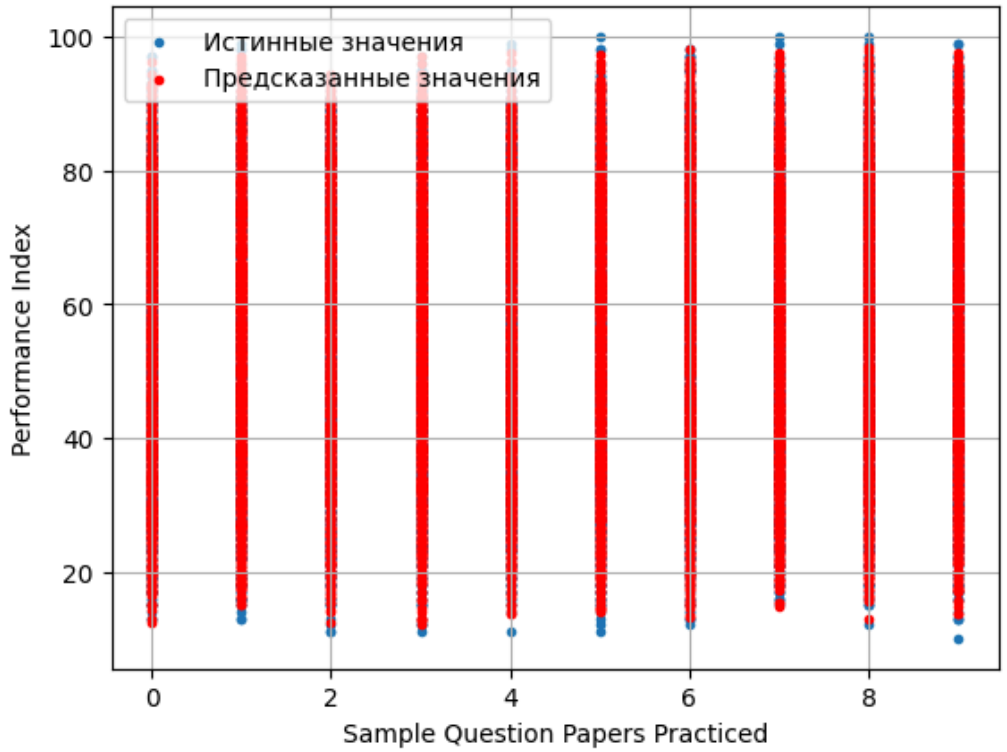
Листинг 3.4.7 - График рассеяния с предсказанием для Previous Scores



Листинг 3.4.8 - График рассеяния с предсказанием для Extracurricular Activities



Листинг 3.4.9 - График рассеяния с предсказанием для Sleep Hours



Листинг 3.4.10 - График рассеяния с предсказанием для Sample Question Papers Practiced

Визуально видно, что линейная регрессия прошла успешна, так как для всех признаков кроме Previous Scores значения практически полностью совпали, у Previous Scores есть больше не совпадений, однако все равно результат очень хороший.

Как отмечалось в пункте 3.3 коэффициенты регрессии: 2.84857195, 1.01876122, 0.59757815, 0.4771185, 0.19456297, свободный член: -34.050793230676. По данным цифрам можно сделать вывод, что предикторы Hours Studied и Previous Scores оказывают влияние на отклик (Performance Index) больше всего, то есть они самые значимые для предсказания отклика. Затем чуть меньше оказывают влияние Extracurricular Activities и Sleep Hours. Меньше всего влияет предиктор Sample Question Papers Practiced.

Проблемы линейной регрессии:

- Наличие выбросов - могут сильно сместить расположение линии
- Наличие мультиколлинеарности - наличие корреляции между предикторами. Для проверки можно делать каждый признак целевым, и проводить линейную регрессию и оценивать коэффициент детерминации
- Автокорреляция остатков - если целевой признак зависит от самого себя

Вывод.

В ходе выполнения работы были изучены линейная и нелинейная регрессии на разном наборе данных, а также на еще одном наборе данных было самостоятельно определено, какая регрессия необходима. Для нелинейной регрессии был проведен поиск степени полинома, необходимой для наивысшего значения коэффициента детерминации.