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 3А

Студент гр. 1303 Самохин К. А.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2024

Цель работы.

Изучить линейную и нелинейную регрессии. Оценить модель регрессии на реальных данных.

Задание.

1. Линейная регрессия.
   1.1. Загрузите набор данных соответствующей цифре вашего варианта. Убедитесь, что загрузка прошла корректно.
   1.2. Используя train_test_split разбейте выборку на обучающую и тестовую. Проверьте, что тестовая выборка соответствует обучающей. *Можно использовать диаграммы рассеяния / нормированные гистограммы/boxplot/violin plot.*
   1.3. Проведите линейную регрессию используя LinearRegression. Получите коэффициенты регрессии и объясните полученные результаты.
   1.4. Для обучающей и тестовой выборки рассчитайте коэффициент детерминации, MAPE, MAE. Объясните полученные значений метрик. Сравните метрики для обучающей и тестовой выборки, сделайте выводы о качестве обобщения полученной модели.
   1.5. Повторите пункты 1.3 и 1.4 для модификаций линейной регрессии:
      1.5.1. Лассо регрессия. Самостоятельно подберите гиперпараметры
      1.5.2. Гребневая регрессия. Самостоятельно подберите гиперпараметры
      1.5.3. ElasticNet. Самостоятельно подберите гиперпараметры
      1.5.4. Регрессия, оптимизируемая градиентным спуском
   1.6. Постройте сводную таблицу для рассчитываемых метрик, и используемых методов (с разделением на обучающую и тестовую выборку). По таблице сделайте выводы о том, какой вид регрессии дал лучшую модель. Опишите какие проблемы могут возникнуть при применении каждой модели. *Для объяснения результатов*

можно построить «карту высот (heat map)» с отображением корреляции признаков.

1.7. Для модели, которая дала лучшие результаты, постройте диаграмму рассеяния между предикторами и откликом. На диаграмме изобразите какое значение должно быть, и какое предсказывается. Визуально оцените качестве построенного регрессора.

2. Нелинейная регрессия.

2.1. Загрузите набор данных соответствующей букве вашего варианта. Убедитесь, что загрузка прошла корректно.

2.2. Используя train_test_split разбейте выборку на обучающую и тестовую. Проверьте, что тестовая выборка соответствует обучающей.

2.3. Проверьте работу стандартной линейной регрессии на загруженных данных. Постройте диаграмму рассеяния данных с выделенной полученной линией регрессии. Объясните полученный результат.

2.4. Конструируя полиномиальный признаки для разных степеней полинома найдите степень полинома наилучшим образом аппроксимирующая данные. Постройте график зависимости коэффициента детерминации от степени полинома (на одном графике изобразите линии для обучающей и тестовой выборки отдельно). Сделайте вывод о том, при какой степени полинома модель начинает переобучаться.

2.5. Для выбранной степени полинома, рассчитайте и проанализируйте полученные коэффициенты. Рассчитай значение метрик коэффициент детерминации, MAPE, MAE.

2.6. Для выбранной степени полинома, постройте диаграмму рассеяния данных с линией соответствующей полученному полиному. Сделайте выводы о качестве аппроксимации.

2.7. Для выбранной степени полинома, решите задачу нелинейной регрессии без конструирования полиномиальных признаков и используя библиотеку TensorFlow.

2.8. Рассчитай метрики коэффициент детерминации, MAPE, MAE, а также постройте диаграмму рассеяния для данных линией соответствующей полученному полиному, для модели, полученной при помощи TensorFlow.

2.9. Сравните результаты, полученные с/без конструирования полиномиальных признаков.

3. Оценка модели регрессии.

3.1. Загрузите набор данных Student_Performance.csv. Данный набор данных содержит информацию о характеристиках студента, а также качестве его обучения.

3.2. Проведите предобработку набора данных - замена текстовых данных, удаление null значений, удаление дубликатов. Разделите на обучающую и тестовую выборку.

3.3. Постройте модель, которая будет предсказывать значение признака Performance Index на основе остальных признаков. *Модель выберите самостоятельно*.

3.4. Проанализируйте полученную модель. Сделайте выводы о значимости/информативности признаков. Опишите какие проблемы могут возникнуть при применении модели.

Выполнение работы.

1. Линейная регрессия.

1.1. Загрузим данные из файла *lab3_lin3.csv*, используя метод *read_csv* (см. листинг 1.1.1). Корректность загрузки датасета проверим, вызвав метод *head* (см. листинг 1.1.2). Результат работы метода представлен в таблице 1.1.1.

Листинг 1.1.1 – Считывание датасета из файла lab3_lin3.csv.

```
1  df = pd.read_csv('lab3_lin3.csv')
```

Листинг 1.1.2 – Вызов метода *head.*

```
1  df.head()
```

Таблица 1.1.1 – Результат работы метода *head.*

	x1	x2	x3	x5	y
0	-0.9037	-1.0798	-3.1031	-0.3140	-23.4544
1	-0.3493	-0.0351	-0.7074	-0.3945	-8.0183
2	-0.6300	0.3800	-0.7846	0.7628	-8.7186
3	-1.1512	-0.2577	-0.5640	1.3832	-4.2153
4	-1.0127	-0.1675	1.6817	-0.1974	15.3587

1.2. Используя метод *train_test_split* разобьём выборку на тестовую и обучающую (см. листинг 1.2.1). Для удобства дальнейшей визуализации у всех наблюдений из полученных разбиений укажем их тип, после чего объединим разбиения с указанными типами в один датафрейм. Также, при помощи диаграмм рассеяния, проверим, что тестовая выборка соответствует обучающей (см. листинг 1.2.2). Полученные диаграммы см. на рисунке 1.2.1.

Листинг 1.2.1 – Разбиение выборки на тестовую и обучающую.

```
1    df_train, df_test = train_test_split(df, test_size = 0.3, shuffle
     = False)
2
3    train = ["train" for _ in range(len(df_train))]
4    test = ["test" for _ in range(len(df_test))]
5    df_train.insert(loc=5, column="type", value=train)
6    df_test.insert(loc=5, column="type", value=test)
7    df_result = pd.concat([df_train, df_test])
```

Листинг 1.2.2 – Рисование диаграмм рассеяния.

```
1    g = sb.PairGrid(data=df_result, hue="type", palette="tab10")
2    g.map(sb.scatterplot)
3    g.add_legend()
```

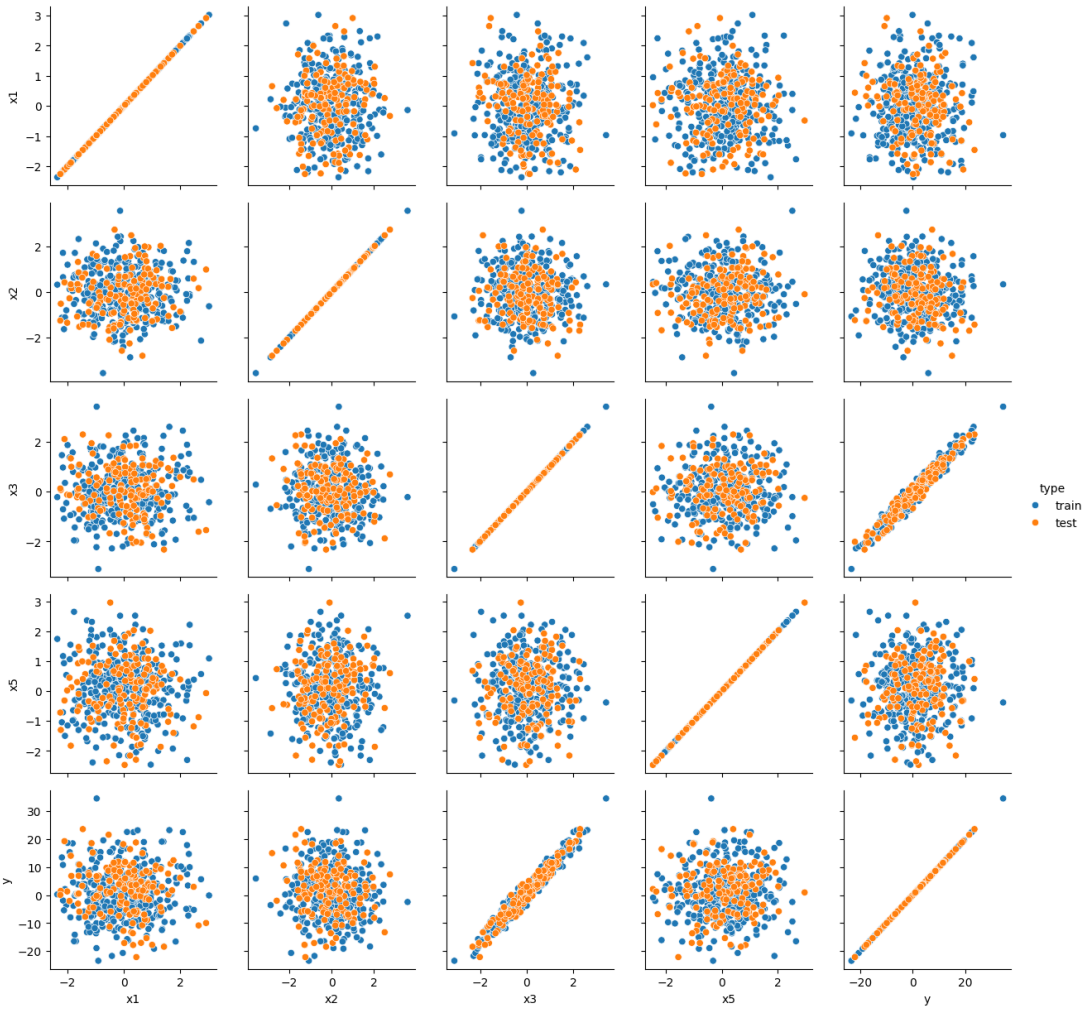

Рисунок 1.2.1 – Диаграммы рассеяния признаков обучающей и тестовой выборок.

По рисунку 1.2.1 можно сказать, что разделение прошло успешно, тестовая выборка соответствует обучающей для всех предикторов. Также можно заметить, что признак *y* имеет линейную зависимость от признака *x3*. При этом, относительно признаков *x1*, *x2* и *x5* признак *y* не имеет ни линейной, ни нелинейной зависимости.

1.3. Проведём линейную регрессию, используя *LinearRegression* (см. листинг 1.3.1). При проведении регрессии были получены соответственные коэффициенты регрессии, представленные в листинге 1.3.2.

Листинг 1.3.1 – Проведение линейной регрессии.

```
linear_reg = LinearRegression()
X_train = df_train[["x1", "x2", "x3", "x5"]]
X_test = df_test[["x1", "x2", "x3", "x5"]]
```

```
4
5     linear_reg.fit(X=X_train, y=df_train["y"])
6     y_train_linear_pred = linear_reg.predict(X_train)
7     y_test_linear_pred = linear_reg.predict(X_test)
8     print(f"Коэффициенты: {linear_reg.coef_}")
9     print(f"Свободный член: {linear_reg.intercept_}")
```

Листинг 1.3.2 – Коэффициенты линейной регрессии.

```
Коэффициенты:   [-0.04996487 -0.04685254  8.84795721 -0.01990352]
Свободный член: 0.9341647877193847
```

После применения линейной регрессии были получены следующие коэффициенты: -0.04996487, -0.04685254, 8.84795721, -0.01990352. Свободный член равен 0.9341647877193847. Полученные коэффициенты позволяют сделать вывод, что признак *x3* влияет на отклик значительно сильнее других признаков.

1.4. Для обучающей и тестовой выборок рассчитаем коэффициент детерминации, MAPE, MAE (см. листинг 1.4.1). Результаты вычислений см. в листинге 1.4.2.

Листинг 1.4.1 – Вычисление коэффициентов для оценки качества линейной регрессии.

```
1     print(f"R^2 train:\t {r2_score(y_true=df_train['y'],
      y_pred=y_train_linear_pred)}")
2     print(f"R^2 test:\t {r2_score(y_true=df_test['y'],
      y_pred=y_test_linear_pred)}")
3     print(f"MAE train:\t {mean_absolute_error(y_true=df_train['y'],
      y_pred=y_train_linear_pred)}")
4     print(f"MAE test:\t {mean_absolute_error(y_true=df_test['y'],
      y_pred=y_test_linear_pred)}")
5     print(f"MAPE train:\t
      {mean_absolute_percentage_error(y_true=df_train['y'],y_pred=y_tra
      in_linear_pred)}")
6     print(f"MAPE test:\t
      {mean_absolute_percentage_error(y_true=df_test['y'],y_pred=y_test
      _linear_pred)}")
```

Листинг 1.4.2 – Результат вычисления коэффициентов.

```
R^2 train: 0.951696151812519
R^2 test: 0.9468388415760167
MAE train: 1.618735197645929
MAE test: 1.6317915803520227
MAPE train: 1.5216926101603092
MAPE test: 1.1517705694718097
```

Метрика R^2 – коэффициент детерминации, принимающий значения от 0 до 1 и равный 1 в том случае, если метрика хорошо приближает. Метрика MAE – средний абсолютная ошибка, измеряется в тех же величинах, что и отклик. Метрика MAPE – процентное отклонение.

Из листинга 1.4.2 видно, что коэффициент детерминации для тестовой и обучающей выборок близок к 1, это означает практически полное соответствие модели данным. Учитывая диапазон значений признака *y*, можно сказать, что и значения коэффициента MAE находятся в допустимых пределах. Значения метрики MAPE (процентное отклонение) очень велики: 152% для обучающей выборки и 115% для тестовой.

Значение метрики R^2 для обучающей выборки выше, чем для тестовой, а MAE, наоборот, выше для тестовой выборки, чем для обучающей. В свою очередь, значение метрик MAPE для тестовой выборки значительно меньше, чем для обучающей. Как итог, из-за большого значения процентного отклонения, можно сказать, что модель плохо предсказывает данные на обеих выборках.

1.7. Построим диаграмму рассеяния между предиктором и откликом (см. листинги 1.7.1 и 1.7.2). Полученные диаграммы представлены на рисунках 1.7.1 и 1.7.2.

Листинг 1.7.1 – Построение диаграмм рассеяния для модели обучающих данных.

```
1    df_train_pred = pd.DataFrame({"x1": X_train["x1"], "x2":
     X_train["x2"], "x3": X_train["x3"], "x5": X_train["x5"], "y":
     y_train_linear_pred, "type": "train_pred"})
2    train = pd.concat([df_train, df_train_pred])
```

```
3     g = sb.PairGrid(data=train, hue="type", palette="tab10")
4     g.map(sb.scatterplot, style=train["type"])
5     g.add_legend()
```

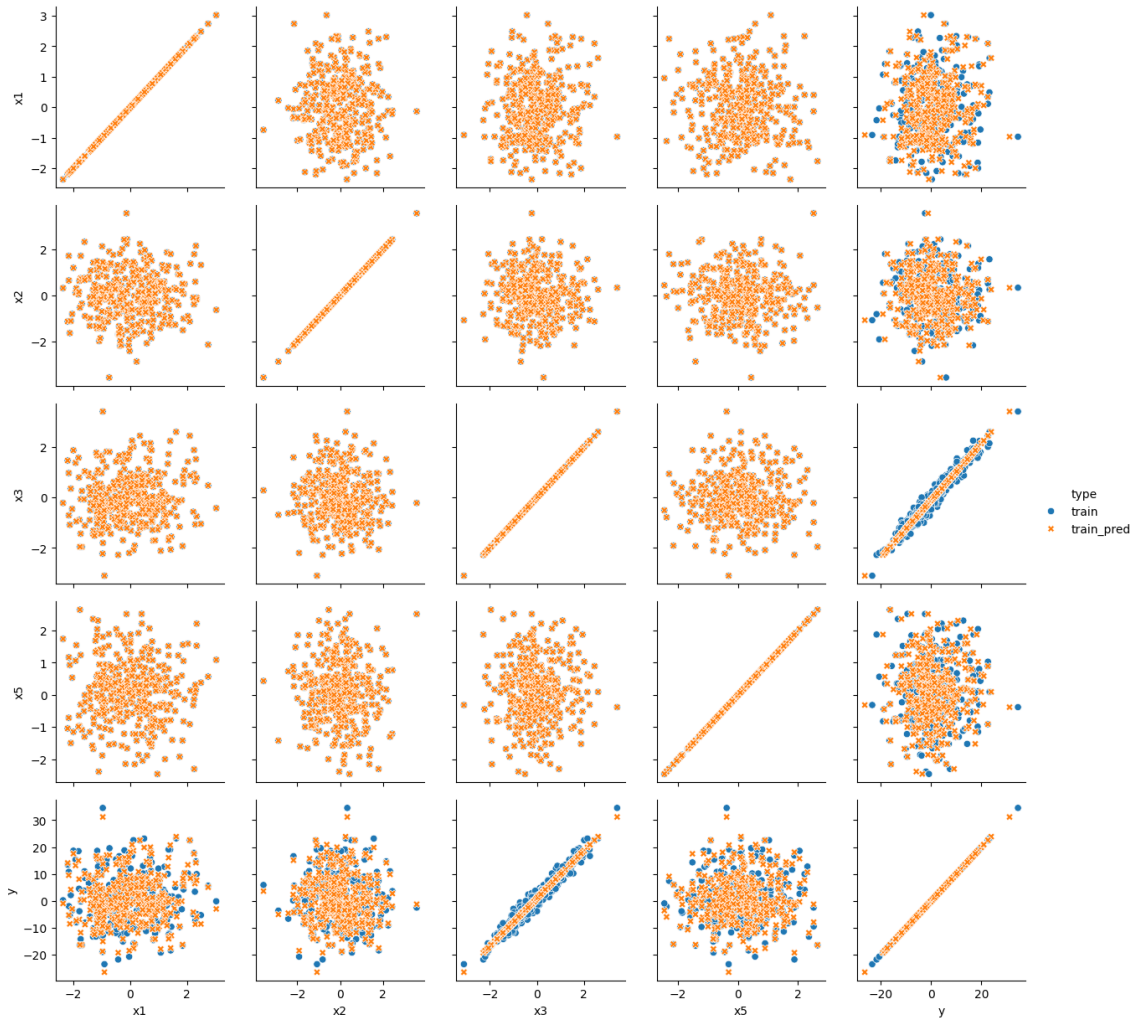

Рисунок 1.7.1 - Диаграммы рассеяния для модели обучающих данных.

Листинг 1.7.2 – Построение диаграмм рассеяния для модели тестовых данных.

```
1     df_test_pred = pd.DataFrame({"x1": X_test["x1"], "x2":
      X_test["x2"], "x3": X_test["x3"], "x5": X_test["x5"], "y":
      y_test_linear_pred, "type": "test_pred"})
2     test = pd.concat([df_test, df_test_pred])
3     g = sb.PairGrid(data=test, hue="type", palette="tab10")
4     g.map(sb.scatterplot, style=test["type"])
5     g.add_legend()
```

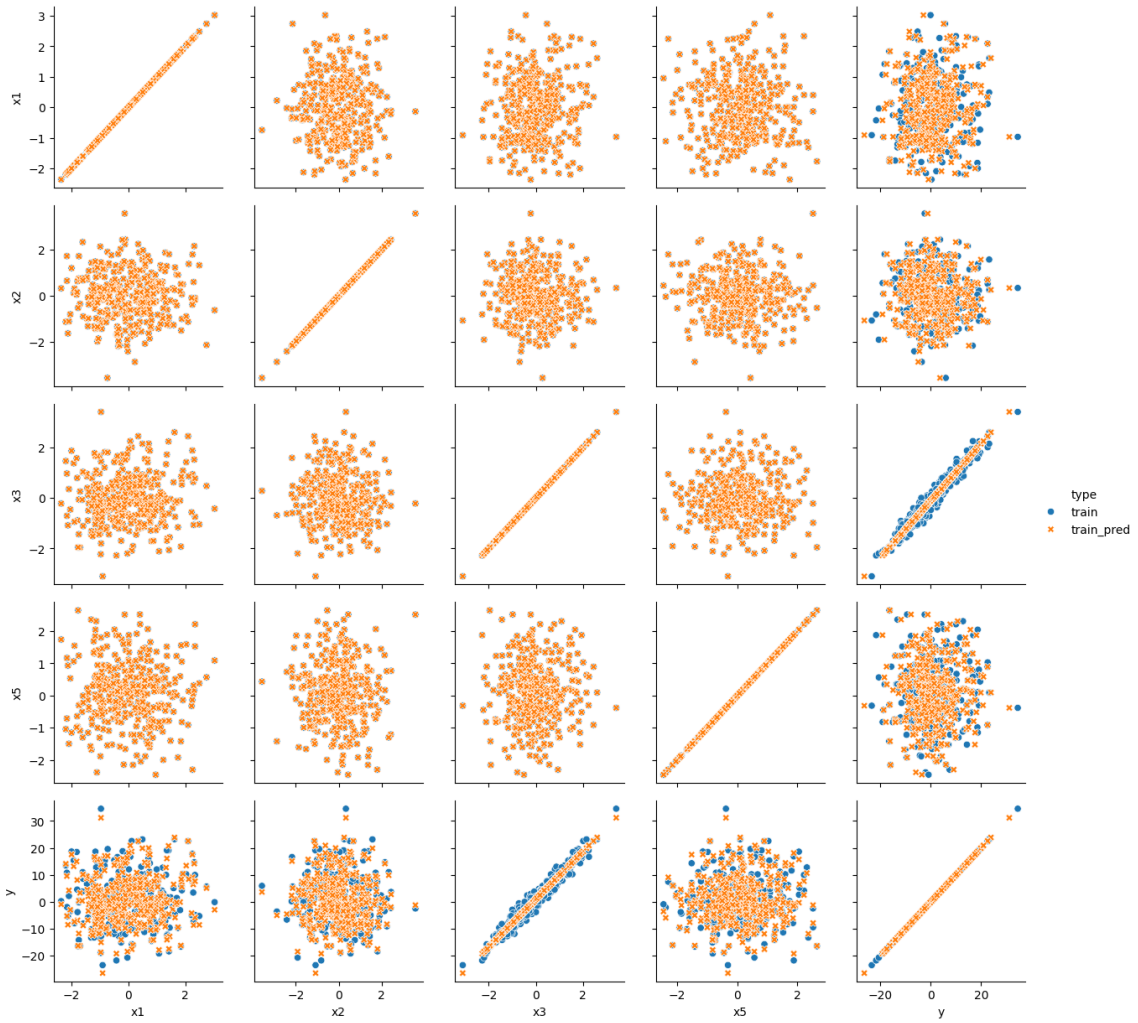

Рисунок 1.7.2 - Диаграммы рассеяния для модели тестовых данных.

Изображенные на рисунках 1.7.1 и 1.7.2 диаграммы позволяют оценить качество построенного регрессора как среднее. Так, и на обучающей, и на тестовой выборках регрессор для предиктора *x3* предсказывает значения вдоль линии тренда с небольшими отклонениями. Форма распределения других предикторов примерно повторяет форму распределения изначальных данных, однако присутствует некоторое отклонение.

2. Нелинейная регрессия.

2.1 Загрузим данные из файла *lab3_poly1.csv*, используя метод *read_csv* (см. листинг 2.1.1). Корректность загрузки датасета проверим, вызвав метод *head* (см. листинг 2.1.2). Результат работы метода представлен в таблице 2.1.1.

Листинг 2.1.1 – Считывание датасета из файла lab3_poly1.csv.

```
poly_df = pd.read_csv('lab3_poly1.csv')
```

Листинг 2.1.2 – Вызов метода *head.*

```
1  poly_df.head()
```

Таблица 2.1.1 – 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

2.2. Используя метод *train_test_split* разобьём выборку на тестовую и обучающую (см. листинг 2.2.1). Для удобства дальнейшей визуализации у всех наблюдений из полученных разбиений укажем их тип, после чего объединим разбиения с указанными типами в один датафрейм. Также, при помощи диаграмм рассеяния, проверим, что тестовая выборка соответствует обучающей (см. листинг 2.2.2). Полученные диаграммы см. на рисунке 2.2.1.

Листинг 2.2.1 – Разбиение выборки на тестовую и обучающую.

```
1     poly_df_train, poly_df_test = train_test_split(poly_df,
      test_size=0.3)
2
3     train = ["train" for _ in range(len(poly_df_train))]
4     test = ["test" for _ in range(len(poly_df_test))]
5     poly_df_train.insert(loc=2, column="type", value=train)
6     poly_df_test.insert(loc=2, column="type", value=test)
7     poly_df_result = pd.concat([poly_df_train, poly_df_test])
```

Листинг 2.2.2 – Рисование диаграммы рассеяния.

```
1     sb.scatterplot(data=poly_df_result, x="x", y="y", hue="type")
```

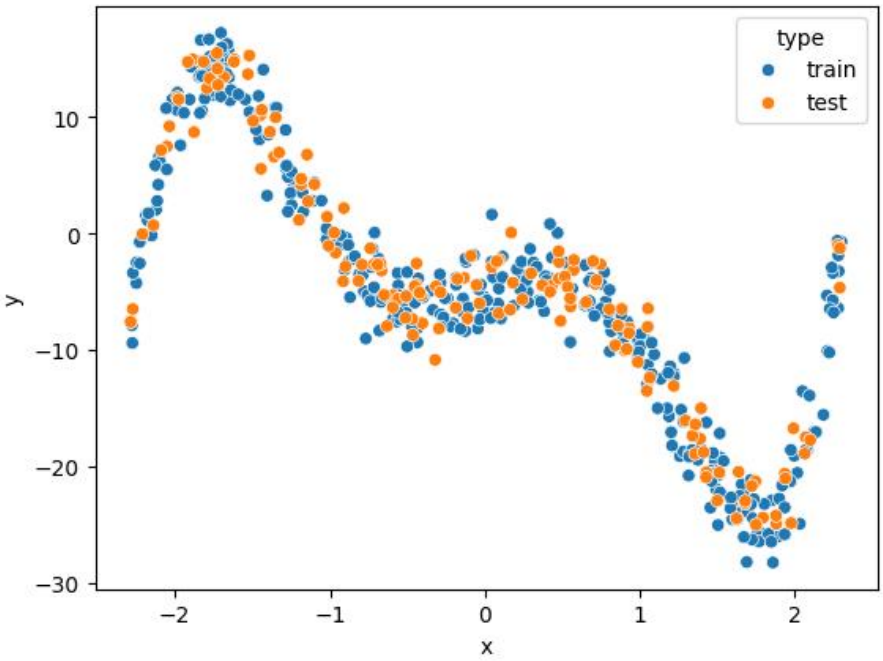

Рисунок 2.2.1 – Диаграмма рассеяния обучающей и тестовой выборок.

На рисунке 2.2.1 видно, что разбиение на обучающую и тестовую выборки прошло успешно. Тестовая выборка полностью соответствует обучающей.

2.3. Проверим работу стандартной линейной регрессии на загруженных данных (см. листинг 2.3.1), полученные коэффициенты см. в листинге 2.3.2. Далее построим диаграмму рассеяния данных с полученной линией регрессии (см. листинг 2.3.3). Полученная диаграмма представлена на рисунке 2.3.1.

Листинг 2.3.1 – Проведение линейной регрессии.

```
linear_reg = LinearRegression()
poly_df_train = poly_df_train.sort_values("x")
poly_df_test = poly_df_test.sort_values("x")
X_poly_train = poly_df_train[["x"]]
X_poly_test = poly_df_test[["x"]]

linear_reg.fit(X=X_poly_train, y=poly_df_train["y"])
y_train_linear_pred = linear_reg.predict(X_poly_train)
y_test_linear_pred = linear_reg.predict(X_poly_test)
print(f"Коэффициенты:\t{linear_reg.coef_}")
print(f"Свободный член:\t{linear_reg.intercept_}")
```

Листинг 2.3.2 – Коэффициенты линейной регрессии.

```
Коэффициенты:   [-6.85133144]
Свободный член: -5.036452190145947
```

В результате применения линейной регрессии были получены коэффициент -6.85133144 и свободный член -5.036452190145947.

Листинг 2.3.3 – Рисование диаграммы рассеяния с линией регрессии.

```
1    sb.scatterplot(data=poly_df_result, x="x", y="y", hue="type")
2    plt.plot(X_poly_train, y_train_linear_pred, 'red')
3    plt.title('Линейная регрессия')
4    plt.grid()
5    plt.show()
```

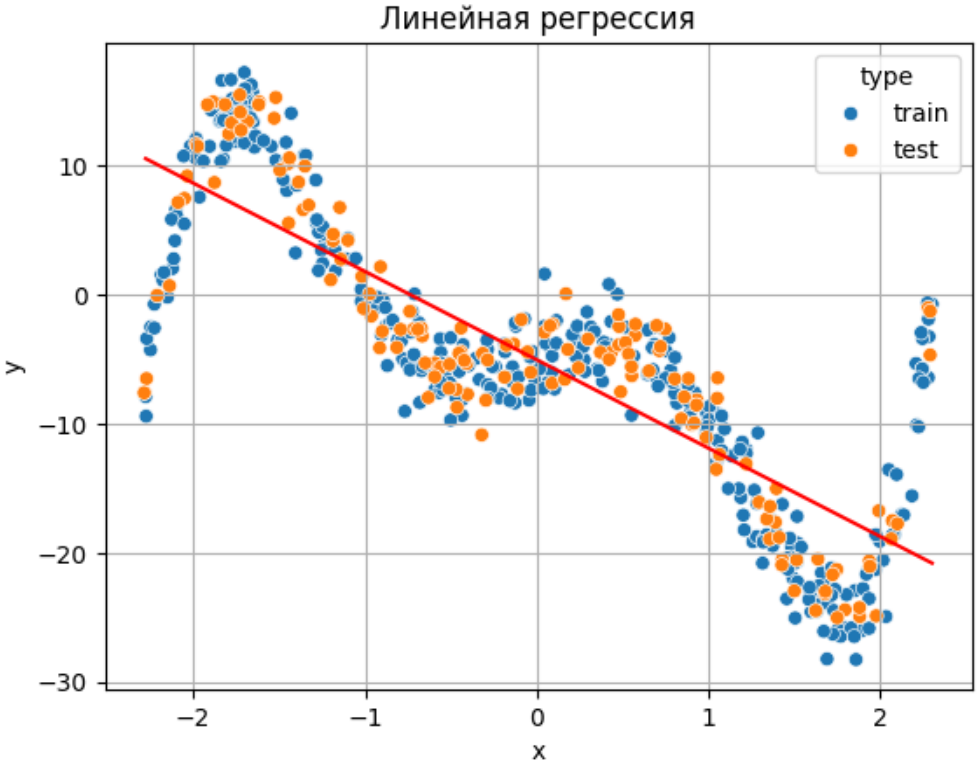

Рисунок 2.3.1 – Диаграмма рассеяния с линией регрессии.

По диаграмме, изображенной на рисунке 2.3.1 видно, что линия регрессии очень плохо предсказывает распределение данных, так как они имеют сложную форму и больше похожи на полином, чем на прямую.

2.4. Сконструируем полиномиальные признаки для разных степеней полинома (см. листинг 2.4.1) и построим график зависимости коэффициента

детерминации от степени полинома (см. листинг 2.4.2). Полученный график см. на рисунке 2.4.1.

Листинг 2.4.1 – Вычисление коэффициента детерминации для разных степеней полинома.

```
r2_train = []
r2_test = []

for degree in range(1, 11):
    poly_f = PolynomialFeatures(degree)
    linear_reg = LinearRegression()
    poly_x_train = poly_f.fit_transform(X=X_poly_train)
    poly_x_test = poly_f.fit_transform(X=X_poly_test)

    linear_reg.fit(X=poly_x_train, y=poly_df_train[["y"]])
    y_train_pred = linear_reg.predict(X=poly_x_train)
    y_test_pred = linear_reg.predict(X=poly_x_test)
    r2_train.append(r2_score(y_true=poly_df_train[["y"]],
     y_pred=y_train_pred))
    r2_test.append(r2_score(y_true=poly_df_test[["y"]],
     y_pred=y_test_pred))
```

Листинг 2.4.2 – Рисование графика зависимости детерминации от степени полинома.

```
plt.plot(range(1, 11), r2_train, 'blue')
plt.plot(range(1, 11), r2_test, 'orange')
plt.legend(["train", "test"], title="type")
plt.xlabel('Степень')
plt.ylabel('R^2')
plt.grid()
```

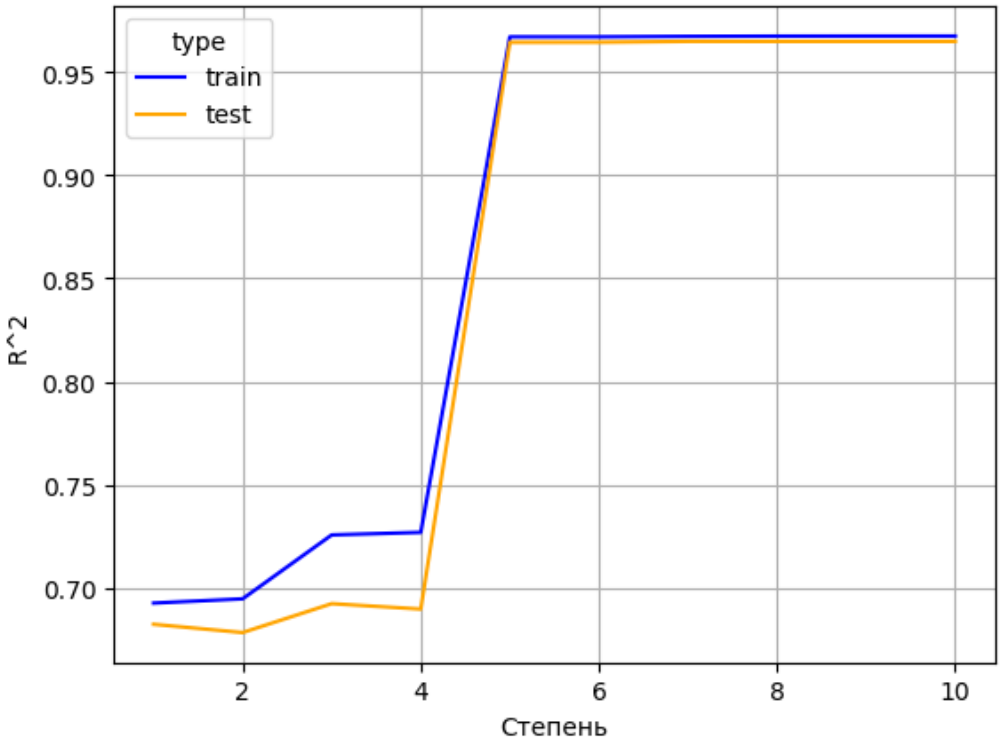

Рисунок 2.4.1 – График зависимости детерминации от степени полинома.

На рисунке 2.4.1 видно, что для всех степеней ниже 5-й коэффициент детерминации очень мал, а для степеней выше 5-й увеличение коэффициента не происходит. Попытка взять степень из промежутка, на котором нет явного увеличения детерминации, может привести к переобучению. Поэтому в качестве оптимальной будет выбрана степень 5.

Построим диаграмму рассеяния с наложенной кривой регрессии (см. листинг 2.4.5). Обучение модели с выбранным числом полиномиальных признаков см. в листинге 2.4.3, полученные коэффициенты полинома см. в листинге 2.4.4. Полученная диаграмма представлена на рисунке 2.4.2.

Листинг 2.4.3 – Обучение модели с 5-ю полиномиальными признаками.

```
poly_f = PolynomialFeatures(5, include_bias = False)
poly_linear_reg = LinearRegression()
poly_X_train = poly_f.fit_transform(X=X_poly_train)
poly_X_test = poly_f.fit_transform(X=X_poly_test)

poly_linear_reg.fit(X=poly_X_train, y=poly_df_train["y"])
y_train_poly_pred = poly_linear_reg.predict(poly_X_train)
y_test_poly_pred = poly_linear_reg.predict(poly_X_test)
```

```
9     print(f"Коэффициенты:\t{poly_linear_reg.coef_}")
10    print(f"Свободный член:\t{poly_linear_reg.intercept_}")
```

Листинг 2.4.4 – Коэффициенты полиномиальной регрессии.

```
Коэффициенты:    [   4.5880179    -0.17367913 -11.4828389      0.02648908
2.08663091]
Свободный член: -4.976815488767255
```

Получены 5 коэффициентов: 4.5880179, -0.17367913, -11.4828389, 0.02648908, 2.08663091. Свободный член равен -4.976815488767255. Значения коэффициентов позволяют судить о том, что наибольшее влияние на отклик оказывает третий коэффициент.

Листинг 2.4.5 – Построение диаграммы рассеяния с кривой регрессии.

```
1     sb.scatterplot(data=poly_df_result, x="x", y="y")
2     plt.plot(X_poly_train, y_train_poly_pred, 'red')
3     plt.grid()
4     plt.show()
```

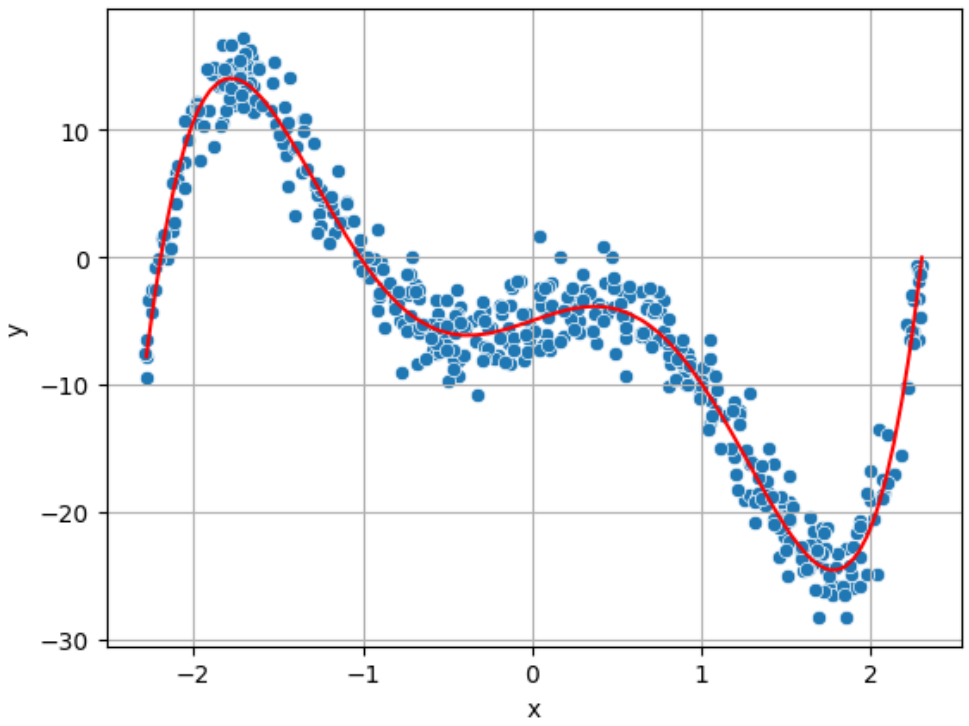

Рисунок 2.4.2 - Диаграмма рассеяния с кривой регрессии.

На рисунке 2.4.2 видно, что полином 5-й степени хорошо аппроксимирует данные в выборке, кривая совпадает с формой распределения.

2.5. Для выбранной степени полинома рассчитаем коэффициент детерминации, MAE, MAPE (см. листинг 2.5.1). Полученные значения см. в листинге 2.5.2.

Листинг 2.5.1 – Вычисление коэффициентов для оценки качества полиномиальной регрессии.

```
1    print(f"R^2 train:\t{r2_score(y_true=poly_df_train['y'],
     y_pred=y_train_poly_pred)}")
2    print(f"R^2 test:\t{r2_score(y_true=poly_df_test['y'],
     y_pred=y_test_poly_pred)}")
3    print(f"MAE
     train:\t{mean_absolute_error(y_true=poly_df_train['y'],
     y_pred=y_train_poly_pred)}")
4    print(f"MAE test:\t{mean_absolute_error(y_true=poly_df_test['y'],
     y_pred=y_test_poly_pred)}")
5    print(f"MAPE
     train:\t{mean_absolute_percentage_error(y_true=poly_df_train['y']
     , y_pred=y_train_poly_pred)}")
6    print(f"MAPE
     test:\t{mean_absolute_percentage_error(y_true=poly_df_test['y'],
     y_pred=y_test_poly_pred)}")
```

Листинг 2.5.2 – Результат вычисления коэффициентов.

```
R^2 train: 0.9669942172350254
R^2 test:  0.9643691655915924
MAE train: 1.5864610898648426
MAE test:  1.5657032092027494
MAPE train: 0.810433627305602
MAPE test: 0.9801964068857081
```

Из листинга 2.5.2 видно, что коэффициент детерминации для тестовой и обучающей выборок близок к 1, это означает практически полное соответствие модели данным. Учитывая диапазон значений признака y, можно сказать, что средняя абсолютная ошибка обеих выборок также достаточно мала. Значения

метрики MAPE (процентное отклонение) принимают значения: 81% для обучающей и 98% для тестовой, это говорит о том, что модель при прогнозировании допускает достаточно большую ошибку.

Как итог, можно сказать, что модель плохо предсказывает данные на обеих выборках.

2.6. Для выбранной степени полинома, построим диаграмму рассеяния с кривой регрессии (см. листинг 2.6.1). Полученная диаграмма представлена на рисунке 2.6.1.

Листинг 2.6.1 – Построение диаграммы рассеяния с кривой регрессии.

```
x = poly_df.sort_values("x")["x"].values
y_pred = sum([coef * x**(i+1) for i, coef in
enumerate(poly_linear_reg.coef_)])
sb.scatterplot(data=poly_df_result, x="x", y="y", hue="type")
plt.plot(x, y_pred+poly_linear_reg.intercept_, 'red')
```

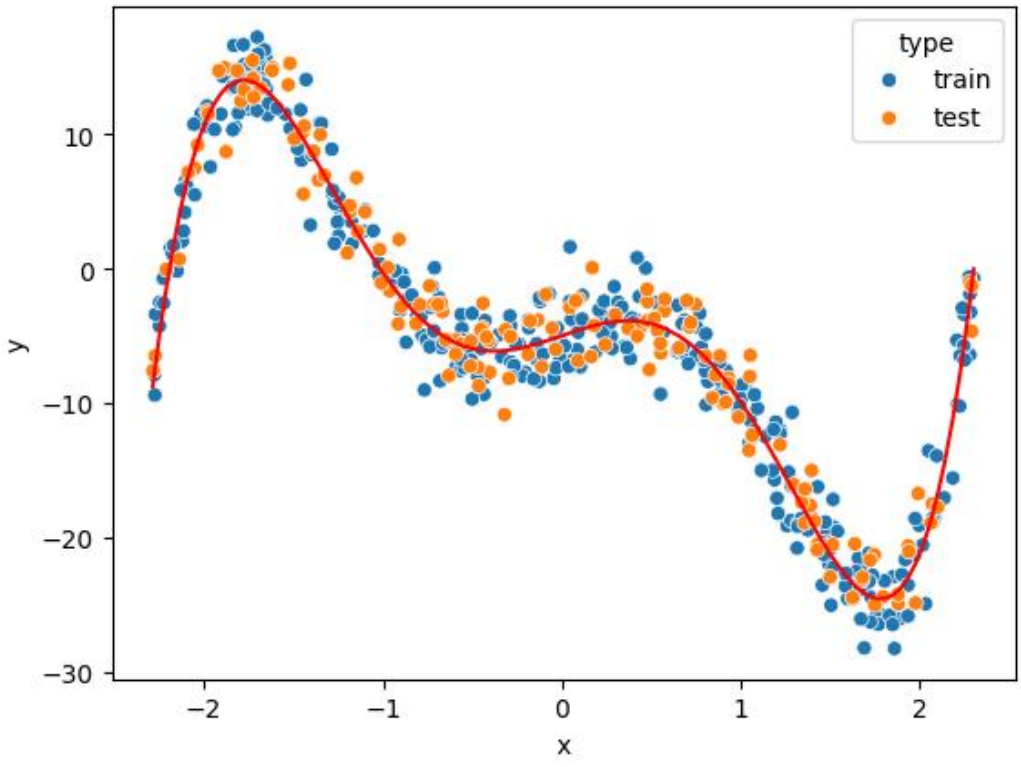

Рисунок 2.6.1 – Диаграмма рассеяния с кривой регрессии.

Из рисунка 2.6.1 видно, что кривая регрессии полностью повторяет форму распределения данных, однако, как мы выяснили выше, ошибка предсказания у построенной линии очень велика.

3. Оценка модели регрессии.

3.1. Загрузим набор данных *Student_Performance.csv* (см. листинг 3.1.1). Корректность загрузки датасета проверим, вызвав метод *head* (см. листинг 3.1.2). Результат работы метода представлен в таблице 3.1.1.

Листинг 3.1.1 – Считывание датасета из файла Student_Performance.csv.

```
1    student_df = pd.read_csv("Student_Performance.csv")
```

Листинг 3.1.2 – Вызов метода *head.*

```
1    student_df.head()
```

Таблица 3.1.1 – 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

3.2. Проведём предобработку данных - замену текстовых данных, удаление null значений, удаление дубликатов (см. листинг 3.2.1). Разделим на обучающую и тестовую выборку (см. листинг 3.2.2).

Листинг 3.2.1 – Предобработка датафрейма *Student_Performance.csv*.

```
1    student_df.dropna(inplace=True)
2    student_df.drop_duplicates(subset=student_df.columns,
     inplace=True)
3    le = slp.LabelEncoder()
4    student_df["Extracurricular Activities"] =
     le.fit_transform(student_df["Extracurricular Activities"])
```

Листинг 3.2.2 - Разбиение выборки на тестовую и обучающую.

```
1    student_df_train, student_df_test = train_test_split(student_df,
     test_size=0.3)
2
3    train = ["train" for _ in range(len(student_df_train))]
4    test = ["test" for _ in range(len(student_df_test))]
5    student_df_train.insert(loc=6, column="type", value=train)
6    student_df_test.insert(loc=6, column="type", value=test)
7    student_df_result = pd.concat([student_df_train, student_df_test])
8    student_df_result.sort_index(inplace=True)
```

Чтобы проверить соответствие тестовой выборке обучающей, построим диаграммы рассеяния разбиений по всем признакам (см. листинг 3.2.3). Полученные диаграммы представлены на рисунке 3.2.1.

Листинг 3.2.2 - Рисование диаграмм рассеяния.

```
1    g = sb.PairGrid(data=student_df_result, hue="type",
     palette="tab10")
2    g.map(sb.scatterplot)
3    g.add_legend()
```

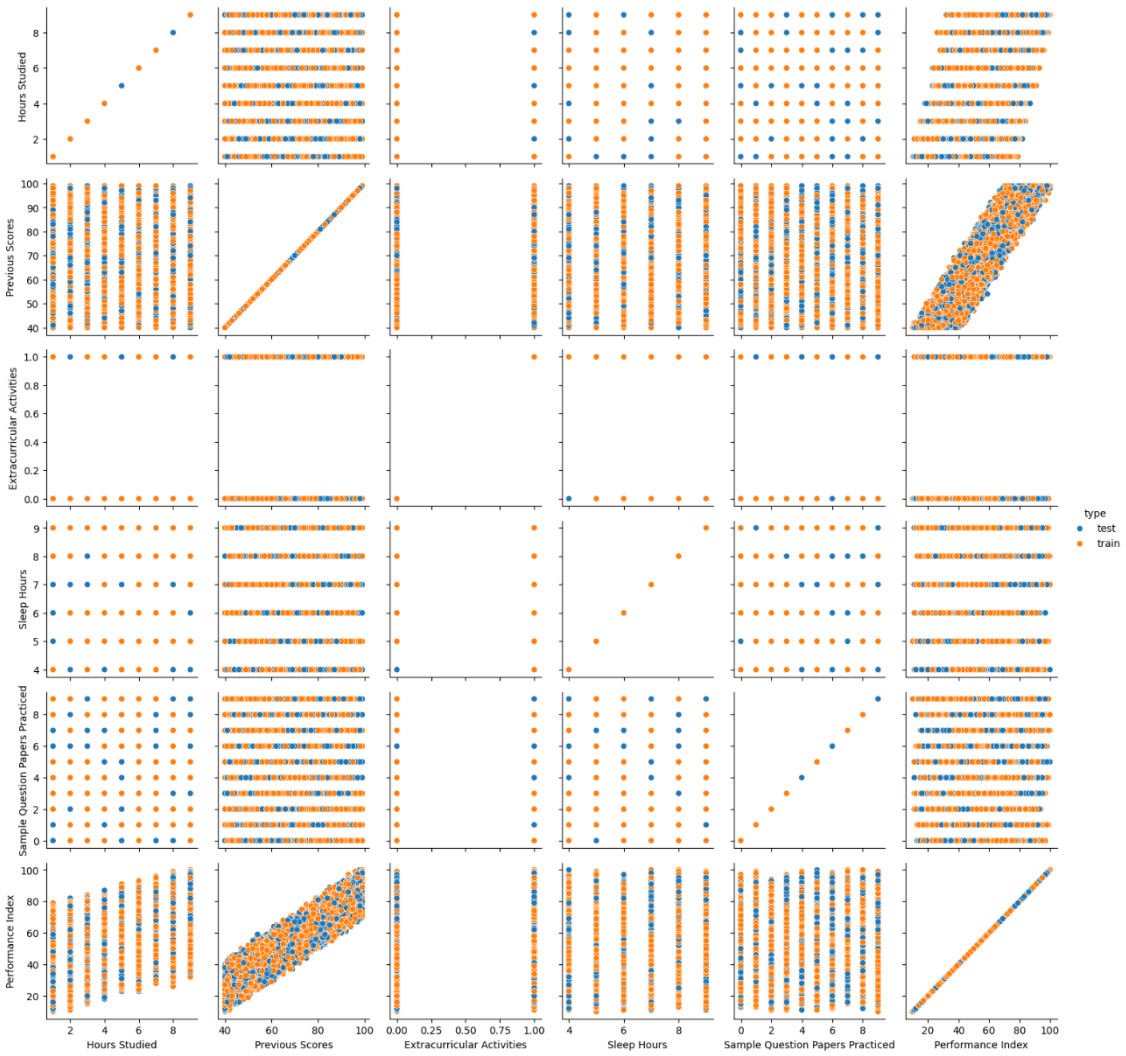

Рисунок 3.2.1 – Диаграммы рассеяния обучающей и тестовой выборок.

Как можно видеть на рисунке 3.2.1, явная корреляция данных заметная только у предикторов *Hours Studied*, *Previous Scores* относительно отклика *Performance Index*. Это значит, что большая часть предикторов (3 из 5) плохо коррелирует с признаком *Performance Index*.

3.3. Построим модель, которая будет предсказывать значение признака *Performance Index* на основе остальных признаков. В предыдущем пункте мы заметили, что только у двух предикторов прослеживается явная корреляция с откликом. Чтобы убедиться в правильности наших наблюдений, вычислим коэффициенты корреляции по каждому признаку относительно *Performance Index* (см. листинг 3.3.1). Полученные коэффициенты представлены в листинге 3.3.2.

Листинг 3.3.1 – Вычисление коэффициентов корреляции.

```
correlation = r_regression(X=student_df_result[student_df_result.columns[:-2]], y=student_df_result["Performance Index"])
correlation
```

Листинг 3.3.2 – Значения коэффициентов корреляции.

```
array([0.37533203, 0.91513508, 0.02607459, 0.05035247, 0.04343571])
```

В листинге 3.3.2 видно, что, как и было замечено ранее, явно коррелируют лишь 2 признака, в то время как остальные представляются практически не значимыми. На основании этих наблюдений было принято решение использовать лассо регрессию (см. листинг 3.3.3). Полученные коэффициенты регрессии представлены в листинге 3.3.4.

Листинг 3.3.3 – Применение лассо регрессии.

```
student_train = student_df_train[["Hours Studied", "Previous Scores", "Extracurricular Activities", "Sleep Hours", "Sample Question Papers Practiced"]]
student_test = student_df_test[["Hours Studied", "Previous Scores", "Extracurricular Activities", "Sleep Hours", "Sample Question Papers Practiced"]]
lasso_reg = Lasso(alpha=2)

lasso_reg.fit(X=student_train, y=student_df_train["Performance Index"])
y_train_lasso_pred = lasso_reg.predict(student_train)
y_test_lasso_pred = lasso_reg.predict(student_test)
print(f"Коэффициенты:\t{lasso_reg.coef_}")
print(f"Свободный член:\t{lasso_reg.intercept_}")
```

Листинг 3.3.4 –Коэффициенты лассо регрессии.

```
Коэффициенты:   [2.56098801 1.01099676 0. 0. 0.]
Свободный член: -27.73994170765156
```

Как можно видеть в листинге 3.3.4, признаки *Hours Studied* и *Previous Scores* имеют наибольшее значение, в то время как коэффициенты для других признаков были занулены, это может повлиять на потерю информации и точность полученного предсказания.

3.4. Проанализируем полученную модель. Для этого рассчитаем коэффициенты детерминации, MAE и MAPE для обучающей и тестовой выборок (см. листинг 3.4.1). Полученные коэффициенты см. в листинге 3.4.2.

Листинг 3.4.1 – Вычисление коэффициентов для оценки качества лассо регрессии.

```
1    print(f"R^2
train:\t{r2_score(y_true=student_df_train['Performance     Index'],
y_pred=y_train_lasso_pred)}")
2    print(f"R^2 test:\t{r2_score(y_true=student_df_test['Performance
     Index'], y_pred=y_test_lasso_pred)}")
3    print(f"MAE
     train:\t{mean_absolute_error(y_true=student_df_train['Performance
     Index'], y_pred=y_train_lasso_pred)}")
4    print(f"MAE
     test:\t{mean_absolute_error(y_true=student_df_test['Performance
     Index'], y_pred=y_test_lasso_pred)}")
5    print(f"MAPE
     train:\t{mean_absolute_percentage_error(y_true=student_df_train['
     Performance Index'], y_pred=y_train_lasso_pred)}")
6    print(f"MAPE
     test:\t{mean_absolute_percentage_error(y_true=student_df_test['Pe
     rformance Index'], y_pred=y_test_lasso_pred)}")
```

Листинг 3.4.2 – Результат вычисления коэффициентов.

```
R^2 train: 0.9838969626867072
R^2 test:  0.984707371998405
MAE train: 1.9356470428878538
MAE test:  1.9014056807727058
MAPE train: 0.041884435713075624
MAPE test: 0.041583088925412105
```

Коэффициент детерминации для обеих выборок очень близок к 1, это означает почти полное соответствие модели данным. Значение коэффициента MAE для обеих выборок, с учётом диапазона значений отклика, также достаточно мало. Процентное отклонение обеих выборок примерно равно 4%, это значит, что модель допускает крайне малую ошибку при прогнозировании данных.

Для визуальной оценки качества приближения построим диаграммы рассеяния между предикторами и откликом (см. листинги 3.4.3 и 3.4.4). Полученные диаграммы см. на рисунках 3.4.1 и 3.4.2.

Листинг 3.4.3 - Построение диаграмм рассеяния для модели обучающих данных.

```
1    student_df_train_pred = pd.DataFrame({"Hours Studied":
     student_train["Hours Studied"], "Previous Scores":
     student_train["Previous Scores"], "Extracurricular Activities":
     student_train["Extracurricular Activities"], "Sleep Hours":
     student_train["Sleep Hours"], "Sample Question Papers Practiced":
     student_train["Sample Question Papers Practiced"], "Performance
     Index": y_train_lasso_pred, "type": "train_pred"})
2    train = pd.concat([student_df_train, student_df_train_pred])
3    g = sb.PairGrid(data=train, hue="type", palette="tab10")
4    g.map(sb.scatterplot, style=train["type"])
5    g.add_legend()
```

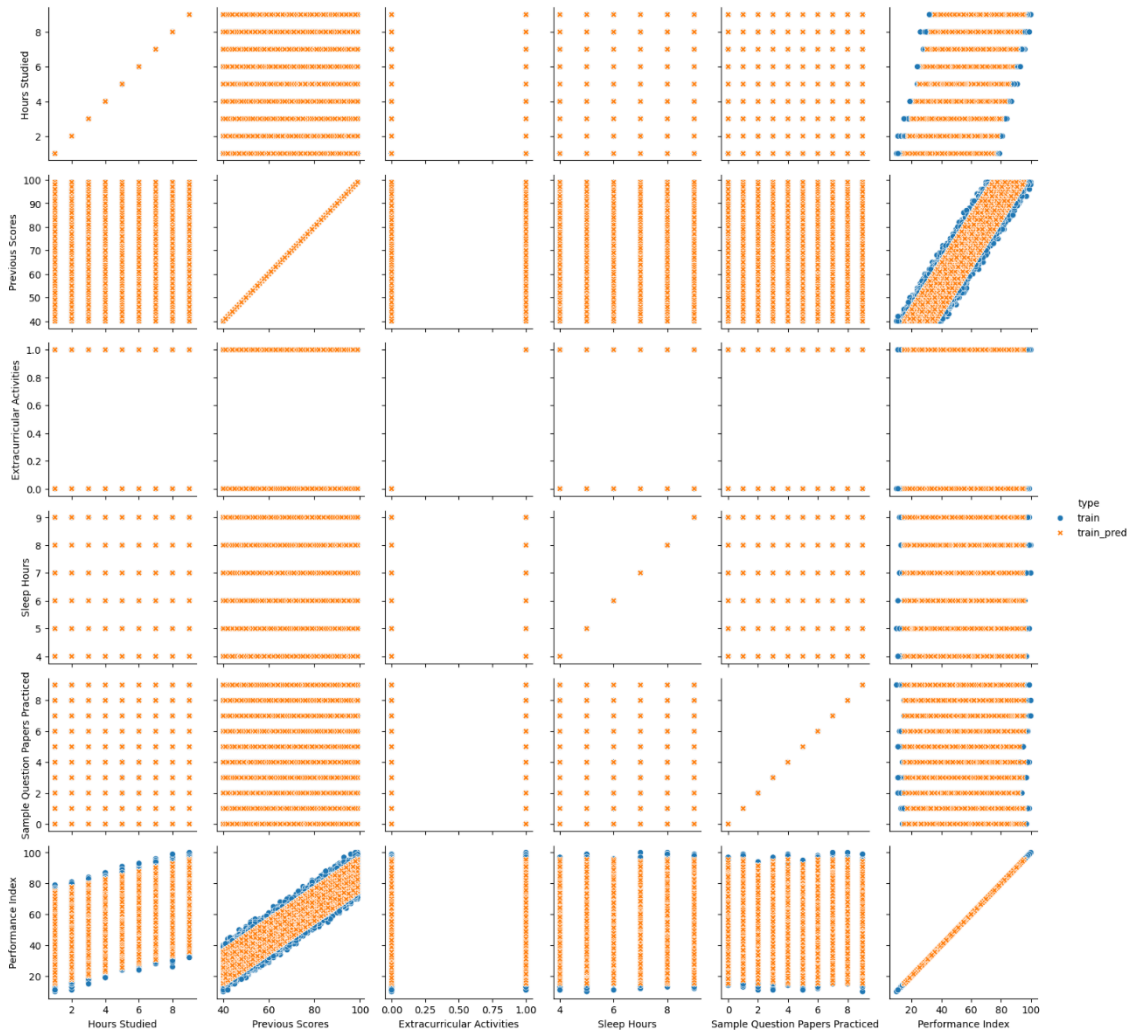

Рисунок 3.4.1 – Диаграммы рассеяния для модели обучающих данных.

Листинг 3.4.4 - Построение диаграмм рассеяния для модели тестовых данных.

```
1    student_df_test_pred = pd.DataFrame({"Hours Studied":
     student_test["Hours Studied"], "Previous Scores":
     student_test["Previous Scores"], "Extracurricular Activities":
     student_test["Extracurricular Activities"], "Sleep Hours":
     student_test["Sleep Hours"], "Sample Question Papers Practiced":
     student_test["Sample Question Papers Practiced"], "Performance
     Index": y_test_lasso_pred, "type": "test_pred"})
2    test = pd.concat([student_df_test, student_df_test_pred])
3    g = sb.PairGrid(data=test, hue="type", palette="tab10")
4    g.map(sb.scatterplot, style=test["type"])
5    g.add_legend()
```

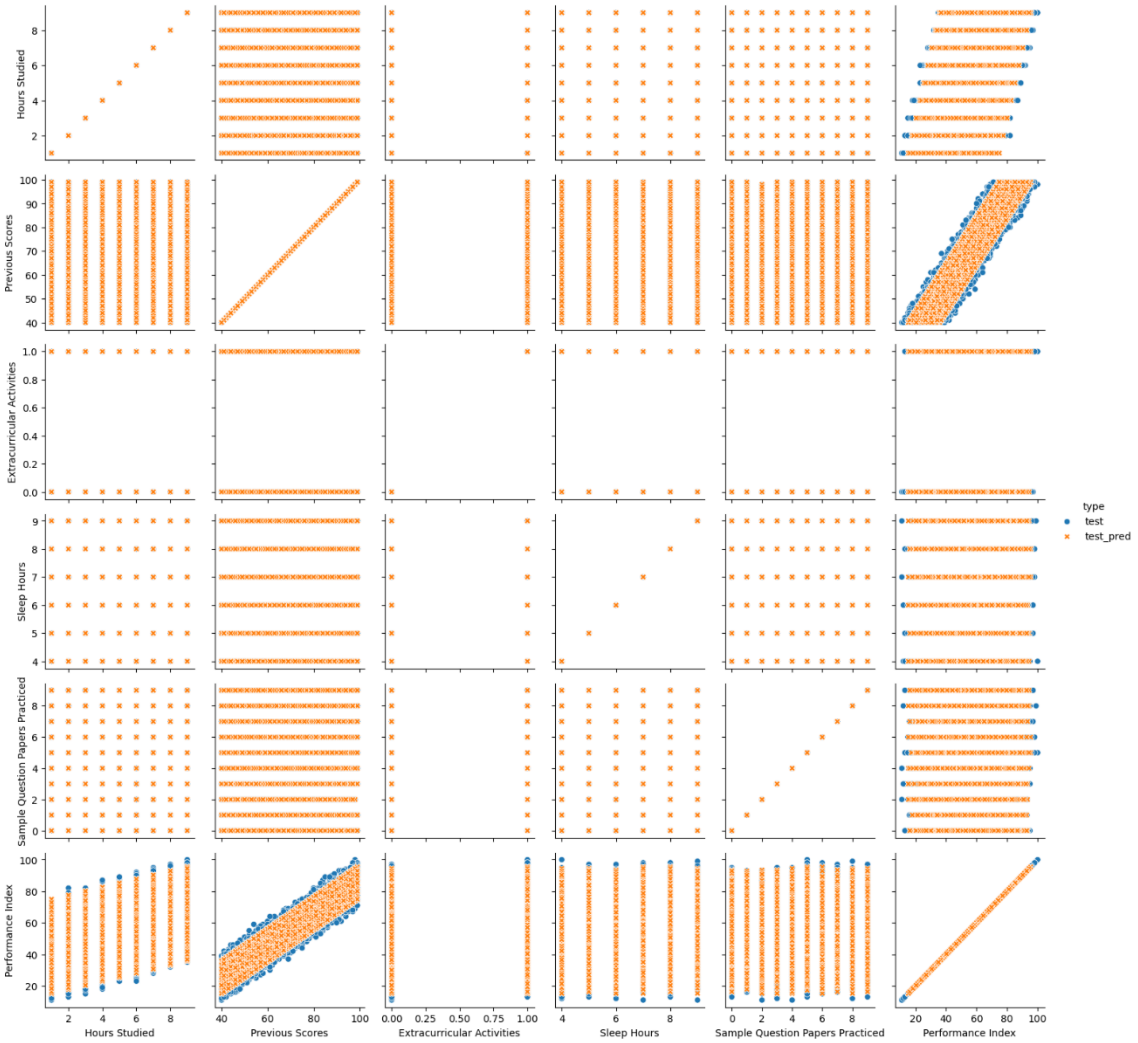

Рисунок 3.4.2 – Диаграммы рассеяния для модели тестовых данных.

Как видно по рисункам 3.4.1 и 3.4.2, предсказанные данные почти полностью покрывают исходные, что позволяет судить о высоком качестве предсказания использованной модели. Высокое качество построенного регрессора также подтверждается метриками, приведёнными в листинге 3.4.2.

Говоря об информативности признаков, можно отметить, что наиболее значимым оказался признак *Hours Studied*, а сразу после него по значимости расположился *Previous Scores*. Коэффициенты остальных признаков лассо регрессия занулила, из-за чего сложно судить об их значимости. Однако, даже без их использования при построении модели предсказание прошло достаточно точно, поэтому можно предположить, что значимость признаков *Extracurricular Activities*, *Sleep Hours* и *Sample Question Papers Practiced* мала.

При использовании лассо регрессии возможна потеря данных и снижение точности предсказания из-за того, что регрессия зануляет коэффициенты некоторых признаков.

Выводы.

В ходе выполнения лабораторной работы были изучены линейная и нелинейная регрессии. Оценена эффективность разных моделей на реальных данных.

Выяснилось, что линейная регрессия плохо себя показывает на наборах данных, в которых не все предикторы хорошо коррелируют с откликом. Этой проблемы нет в методе *Lasso* линейной регрессии, так как он позволяет не учитывать не информативные признаки.

Нелинейная регрессия намного лучше показала себя при описании сложных форм данных, чем линейная.