**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 1Б**

| Студент гр. 1303 |  | Чубан Д.В. |
| --- | --- | --- |
| 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. Загрузите набор данных соответствующей цифре вашего варианта. Убедитесь, что загрузка прошла корректно.
   2. Используя train\_test\_split разбейте выборку на обучающую и тестовую. Проверьте, что тестовая выборка соответствует обучающей. *Можно испльзовать диаграммы рассеяния/нормированные гистограммы/boxplot/violin plot*.
   3. Проведите линейную регрессию используя LinearRegression. Получите коэффициенты регрессии и объясните полученные результаты.
   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.
   5. Повторите пункты 1.3 и 1.4 для модификаций линейной регрессии:
      1. Лассо регрессия. Самостоятельно подберите гиперпараметры
      2. Гребневая регрессия. Самостоятельно подберите гиперпараметры
      3. ElasticNet. Самостоятельно подберите гиперпараметры
      4. Регрессия оптимизируемая градиентным спуском
   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.*
   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.**
   1. Загрузите набор данных соответствующей букве вашего варианта. Убедитесь, что загрузка прошла корректно.
   2. Используя train\_test\_split разбейте выборку на обучающую и тестовую. Проверьте, что тестовая выборка соответствует обучающей.
   3. Проверьте работу стандартной линейной регрессии на загруженных данных. Постройте диаграмму рассеяния данных с выделенной полученной линией регрессии. Объясните полученный результат.
   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.
   5. Для выбранной степени полинома, рассчитайте и проанализируйте полученные коэффициенты. Рассчитай значение метрик коэффициент детерминации, MAPE, MAE.
   6. Для выбранной степени полинома, постройте диаграмму рассеяния данных с линией соответствующей полученному полиному. Сделайте выводы о качестве аппроксимации.
   7. Для выбранной степени полинома, решите задачу нелинейной регрессии без конструирования полиномиальных признаков и используя библиотеку TensorFlow.
   8. Рассчитай метрики коэффициент детерминации, MAPE, MAE, а также постройте диаграмму рассеяния для данных линией соответствующей полученному полиному, для модели полученной при помощи TensorFlow.
   9. Сравните результаты полученные с/без конструирования полиномиальных признаков.
3. **Оценка модели регрессии**
   1. Загрузите набор данных Student\_Performance.csv . Данный набор данных содержит информацию о характеристиках студента, а также качестве его обучения.
   2. Проведите предобработку набора данных - замена текстовых данных, удаление null значений, удаление дубликатов. Разделите на обучающую и тестовую выборку.
   3. Постройте модель, которая будет предсказывать значение признака Performance Index на основе остальных признаков. *Модель выберите самостоятельно.*
   4. Проанализируйте полученную модель. Сделайте выводы о значимости/информативности признаков. Опишите какие проблемы могут возникнуть при применении модели.

**Выполнение работы.**

1. Линейная регрессия
   1. Загрузим набор данных.

Загрузим набор данных и убедимся, что загрузка прошла корректно. Результат представлен в таблице 1.1.

Таблица 1.1 – данные lab3\_lin1.csv

|  | **x1** | **x2** | **x3** | **x4** | **x5** | **y** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.4241 | 1.2065 | -0.0992 | 1.1363 | -0.4517 | 2.5081 |
| **1** | 1.4999 | -1.8712 | -0.1863 | 0.1274 | 1.6150 | 69.0924 |
| **2** | -1.9402 | 1.0229 | 0.0152 | 0.6839 | 0.5239 | 48.9160 |
| **3** | -0.3690 | 0.2127 | -0.5173 | 1.3148 | 0.1116 | -1.6333 |
| **4** | 1.3925 | 1.9357 | -1.0630 | -0.0942 | -0.1154 | -99.9433 |

Из таблицы видно, что имеются 4 предиктора и 1 отклик.

* 1. Разобьем выборку на обучающую и тестовую.

Используем train\_test\_split (листинг 1.2.1), а также проверим на диаграмме рассеяния, что тестовая выборка соответствует обучающей (рис. 1.2.1).

Листинг 1.2.1 – Использование train\_test\_split

| x = df\_lin[["x1", "x2", "x3", "x5"]].to\_numpy()  y = df\_lin["y"].to\_numpy()  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.3, shuffle = False) |
| --- |

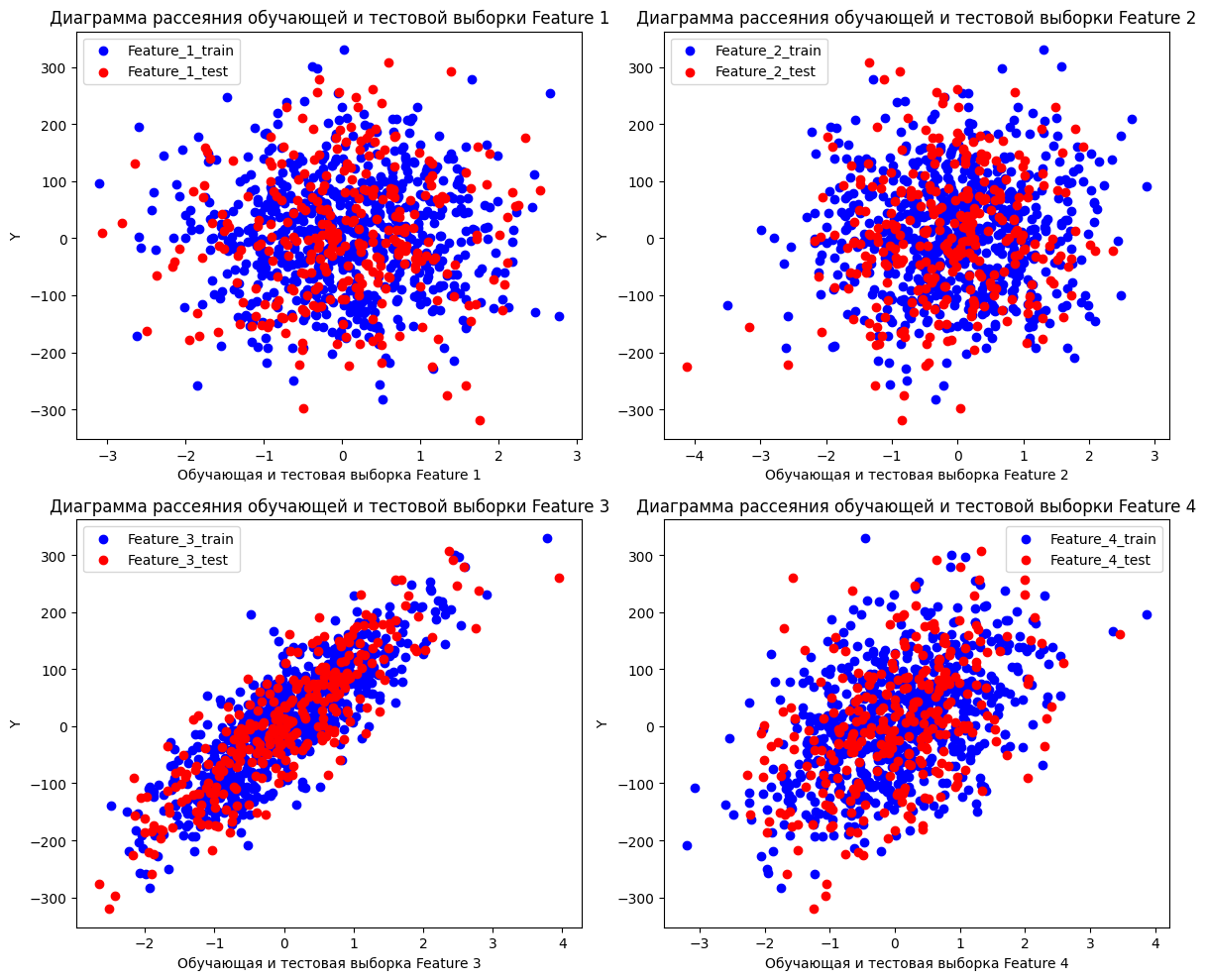


Рисунок 1.2.1 – диаграммы рассеяния обучающей и тестовой выборок для каждого из предикторов

По рисунку 1.2.1 можно увидеть, что тестовая выборка соответствует обучающей для каждого предиктора.

* 1. Проведение линейной регрессии

Проведем линейную регрессию используя LinearRegression (листинг 1.3.1).

Листинг 1.3.1 – Использование LinearRegression

| lin\_reg = LinearRegression()  lin\_reg.fit(x\_train, y\_train)  print(lin\_reg.coef\_)  print(lin\_reg.intercept\_) |
| --- |

Коэффициенты регрессии равны: [ 1.09051103 0.15354934 89.93428583 47.63530072]. Из этого можно понять, что третий коэффициент регрессии сильнее всего влияет на отклик в положительную сторону. Также достаточно сильное влияние в положительную сторону оказывает 4 коэффициент.

Свободный член равен 3.6409562645010265. Это число представляет собой значение прогнозируемой переменной, когда все предикторы равны 0.

* 1. Для обучающей и тестовой выборки рассчитаем коэффициент детерминации, MAPE, MAE.

Для этого напишем функцию metrics, принимающую на вход модель и выводящую рассчитанные метрики для обеих выборок (листинг 1.4.1, таб. 1.4.1).

Листинг 1.4.1 – Функция metrics

| def metrics(model):  y\_train\_pred = model.predict(x\_train)  y\_test\_pred = model.predict(x\_test)  r2\_train = r2\_score(y\_train, y\_train\_pred)  mae\_train = mean\_absolute\_error(y\_train, y\_train\_pred)  mape\_train = mean\_absolute\_percentage\_error(y\_train, y\_train\_pred)  r2\_test = r2\_score(y\_test, y\_test\_pred)  mae\_test = mean\_absolute\_error(y\_test, y\_test\_pred)  mape\_test = mean\_absolute\_percentage\_error(y\_test, y\_test\_pred)  print("обучающая:")  print("R^2:", r2\_train)  print("MAE:", mae\_train)  print("MAPE:", mape\_train)  print()  print("тестовая:")  print("R^2:", r2\_test)  print("MAE:", mae\_test)  print("MAPE:", mape\_test)  return r2\_train, mae\_train, mape\_train, r2\_test, mae\_test, mape\_test |
| --- |

Таблица 1.4.1 – Вывод metrics

|  | Обучающая выборка | Тестовая выборка |
| --- | --- | --- |
| R2 | 0.9322865306693603 | 0.9414085497438694 |
| MAE | 20.82540974662524 | 22.380587662042284 |
| MAPE | 2.3183910839503303 | 1.117974724088372 |

Метрика R^2 – коэффициент детерминации, изменятся от 0 до 1, и равен 1 если модель хорошо приближает. Метрика MAE – средняя абсолютная ошибка. Метрика MAPE – процентное отклонение.

Для данной модели можно сказать, что она хорошо приближает, т.к. значение R2 близко к 1. Средняя абсолютная ошибка достаточно большая, средняя абсолютная ошибка достигает 231% для обучающей и 111% для тестовой, что является плохим показателем.

R2 и MAE у обучающей выборки ниже, чем у тестовой, а MAPE выше.

В целом, вероятно можно сказать, что модель обобщает на обеих выборках неудовлетворительно.

1.7. Для модели, которая дала лучшие результаты, построим диаграмму рассеяния между предикторами и откликом (листинг 1.7.1, рис. 1.7.1).

Листинг 1.7.1 – построение диаграммы рассеяния между предикторами и откликом

| y\_pred\_train = lin\_reg.predict(x\_train)  y\_pred\_test = lin\_reg.predict(x\_test)  fig, axs = plt.subplots(2, 4, figsize=(18, 8))  for i in range(x\_train.shape[1]):  axs[0, i].scatter(x\_train[:, i], y\_train, color='blue', label='Фактические значения')  axs[0, i].scatter(x\_train[:, i], y\_pred\_train, color='red', label='Предсказанные значения')  axs[0, i].set\_xlabel(f'Предиктор {i+1}')  axs[0, i].set\_ylabel('Отклик')  axs[0, i].set\_title(f'Предиктор {i+1}')  axs[0, i].legend()  for i in range(x\_test.shape[1]):  axs[1, i].scatter(x\_test[:, i], y\_test, color='blue', label='Фактические значения')  axs[1, i].scatter(x\_test[:, i], y\_pred\_test, color='orange', label='Предсказанные значения')  axs[1, i].set\_xlabel(f'Предиктор {i+1}')  axs[1, i].set\_ylabel('Отклик')  axs[1, i].set\_title(f'Предиктор {i+1}')  axs[1, i].legend()  plt.tight\_layout()  plt.show() |
| --- |

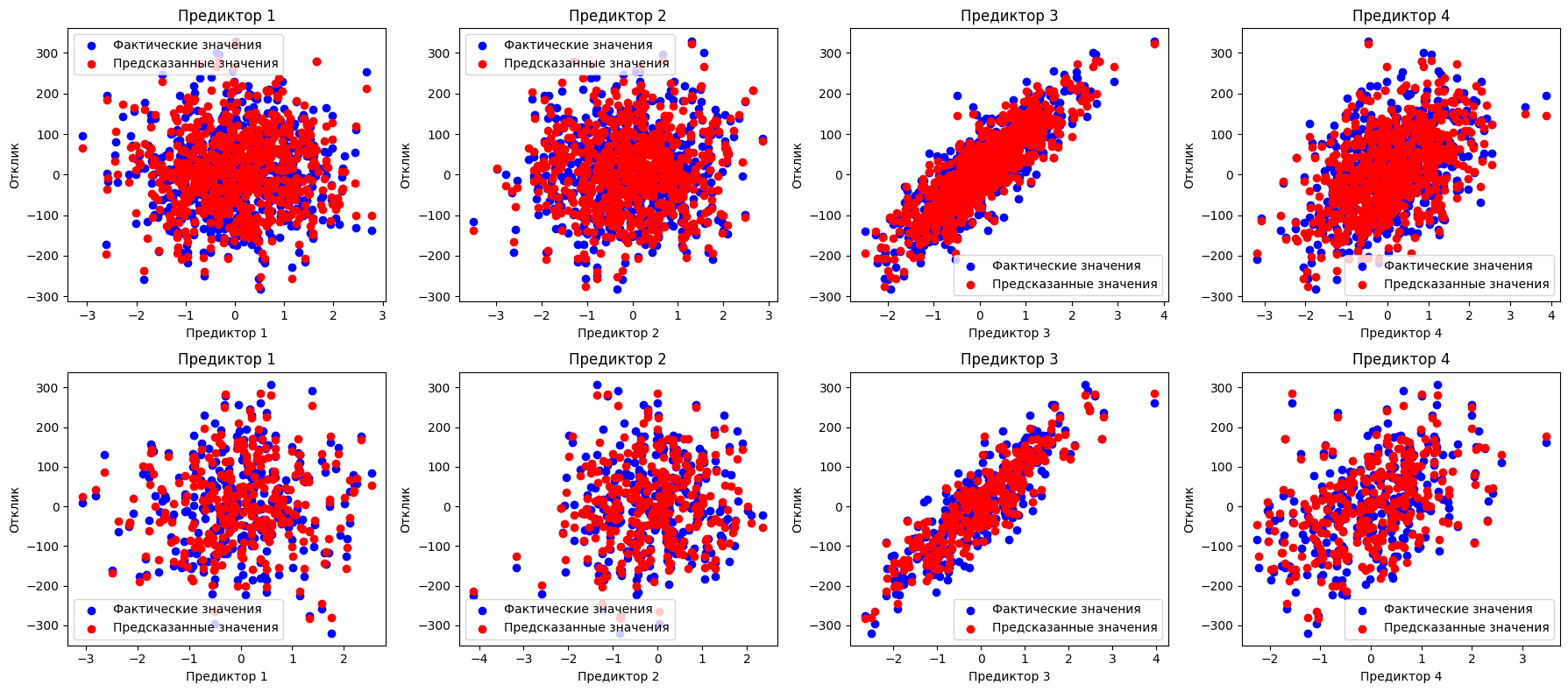


Рисунок 1.7.1 – диаграмма рассеяния между предикторами и откликом.

Визуально качество построенного регрессора можно оценить как высокое, так как на обучающей и тестовой выборках предсказанные данные практически идентичны фактическим.

1. Нелинейная регрессия
   1. Загрузим набор данных и убедимся в корректности загрузки (табл. 2.1).

Таблица 2.1 – данные lab3\_poly2.csv

|  | x | y |
| --- | --- | --- |
| **0** | 1.2429 | 0.2452 |
| **1** | -0.6314 | -1.0334 |
| **2** | 0.9256 | 1.9695 |
| **3** | 0.6894 | 0.3605 |
| **4** | -0.1864 | -0.1788 |

Из таблицы видно, что имеются 1 предиктор и 1 отклик.

* 1. Используя train\_test\_split разобьем выборку на обучающую и тестовую.

Используем train\_test\_split (листинг 2.2.1), а также проверим на диаграмме рассеяния, что тестовая выборка соответствует обучающей (рис. 2.2.1).

Листинг 2.2.1 – Использование train\_test\_split

| x = df\_poly["x"].to\_numpy().reshape(-1, 1)  y = df\_poly["y"].to\_numpy().reshape(-1, 1)  x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.3, shuffle = False) |
| --- |

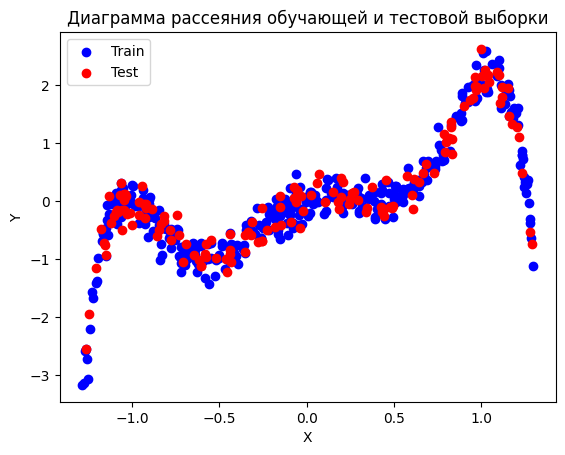


Рисунок 2.2.1 – диаграммы рассеяния обучающей и тестовой выборок

По рисунку 2.2.1 можно увидеть, что тестовая выборка соответствует обучающей для каждого предиктора.

* 1. Проверим работу стандартной линейной регрессии на загруженных данных. Построим диаграмму рассеяния данных с выделенной полученной линией регрессии (листинг 2.3.1, рис. 2.3.1).

Листинг 2.3.1 – Использование линейной регрессии.

| lin\_reg = LinearRegression()  lin\_reg.fit(x\_train, y\_train)  y\_pred\_train = lin\_reg.predict(x\_train)  y\_pred\_test = lin\_reg.predict(x\_test)  plt.scatter(x\_test, y\_test, color='green', label='Фактические значения')  plt.scatter(x\_test, y\_pred\_test, color='orange', label='Предсказанные значения')  plt.title(f'Применение стандартной линейной регрессии')  plt.xlabel('X')  plt.ylabel('Y')  plt.legend() |
| --- |

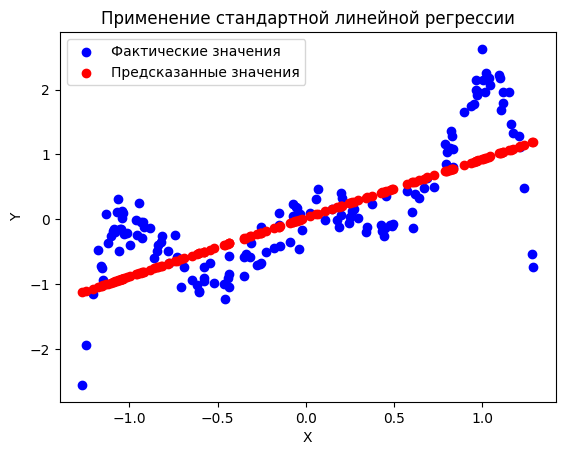


Рисунок 2.3.1 – диаграмма рассеяния данных с выделенной линией регрессии

По рисунку можно сказать, что линейная регрессия плохо обобщает данные, так как они не имеют линейной зависимости.

* 1. Найдем степень полинома наилучшим образом аппроксимирующую данные. Построим график зависимости коэффициента детерминации от степени полинома (листинг 2.4.1, рис. 2.4.1).

Листинг 2.4.1 – Нахождение степени полинома и построение графика.

| train\_r2 = []  test\_r2 = []  for degree in range(1, 11):  poly = PolynomialFeatures(degree)  x\_poly\_train = poly.fit\_transform(x\_train)  x\_poly\_test = poly.transform(x\_test)  model = LinearRegression(fit\_intercept = False)  model.fit(x\_poly\_train, y\_train)  y\_train\_pred = model.predict(x\_poly\_train)  y\_test\_pred = model.predict(x\_poly\_test)  train\_r2.append(r2\_score(y\_train, y\_train\_pred))  test\_r2.append(r2\_score(y\_test, y\_test\_pred))  plt.plot(range(1, 11), train\_r2, label='Train R2')  plt.plot(range(1, 11), test\_r2, label='Test R2')  plt.xlabel('Степень полинома')  plt.ylabel('R2 Score')  plt.title('Зависимость R2 Score от степени полинома')  plt.legend()  plt.show() |
| --- |

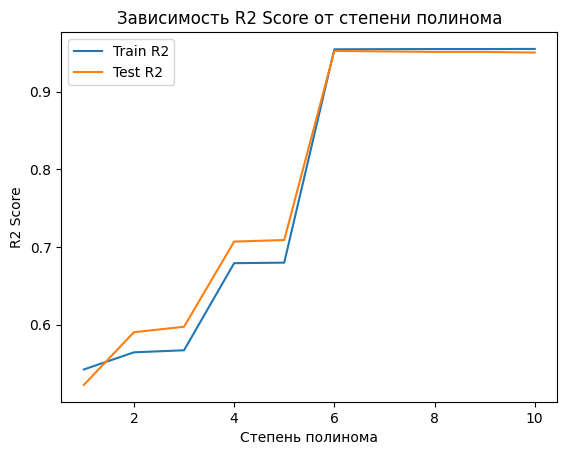


Рисунок 2.4.1 – график зависимости коэффициента детерминации от степени полинома

Из рисунка 2.4.1 можно сказать, что наилучшим образом аппроксимирует данные полином со степенью 6, а при степенях 7+ модель начнет переобучаться.

* 1. Для полинома степени 6 рассчитаем коэффициенты и метрики R^2, MAE и MAPE (листинг 2.5.1, табл. 2.5.1).

Листинг 2.5.1 – Вычисление коэффициентов для полинома степени 6

| def poly\_metrics(poly, model):  x\_poly\_train = poly.fit\_transform(x\_train)  x\_poly\_test = poly.transform(x\_test)  y\_train\_pred = model.predict(x\_poly\_train)  y\_test\_pred = model.predict(x\_poly\_test)  r2\_train = r2\_score(y\_train, y\_train\_pred)  mae\_train = mean\_absolute\_error(y\_train, y\_train\_pred)  mape\_train = mean\_absolute\_percentage\_error(y\_train, y\_train\_pred)  r2\_test = r2\_score(y\_test, y\_test\_pred)  mae\_test = mean\_absolute\_error(y\_test, y\_test\_pred)  mape\_test = mean\_absolute\_percentage\_error(y\_test, y\_test\_pred)  print("Метрики для обучающей выборки:")  print("R^2:", r2\_train)  print("MAE:", mae\_train)  print("MAPE:", mape\_train)  print()  print("Метрики для тестовой выборки:")  print("R^2:", r2\_test)  print("MAE:", mae\_test)  print("MAPE:", mape\_test)  return r2\_train, mae\_train, mape\_train, r2\_test, mae\_test, mape\_test |
| --- |

Коэффициенты регрессии равны: [1.00660965 -4.25846968 0.01230578 10.59249166 0.01936263 -5.29303717]. По коэффициентам можно сказать, что первый и четвертый коэффициенты влияют на отклик положительно, отрицательно влияют второй и шестой коэффициенты, а третий и пятый коэффициенты незначительны.

Таблица 2.5.1 – метрики

|  | Обучающая выборка | Тестовая выборка |
| --- | --- | --- |
| R2 | 0.954562295549346 | 0.9524960976884708 |
| MAE | 0.1553334716839232 | 0.1652783557205586 |
| MAPE | 1.0608265815790738 | 0.8286128358039425 |

По метрикам для обеих выборок можно сказать, что модель обобщает удовлетворительно, т.к. у обеих выборок R2 близок к 1, MAE низок, а MAPE равен 106% у обучающей и 82% у тестовой.

* 1. Для 6 степени полинома построим диаграмму рассеяния данных с линией соответствующей полученному полиному (рис. 2.7.1).

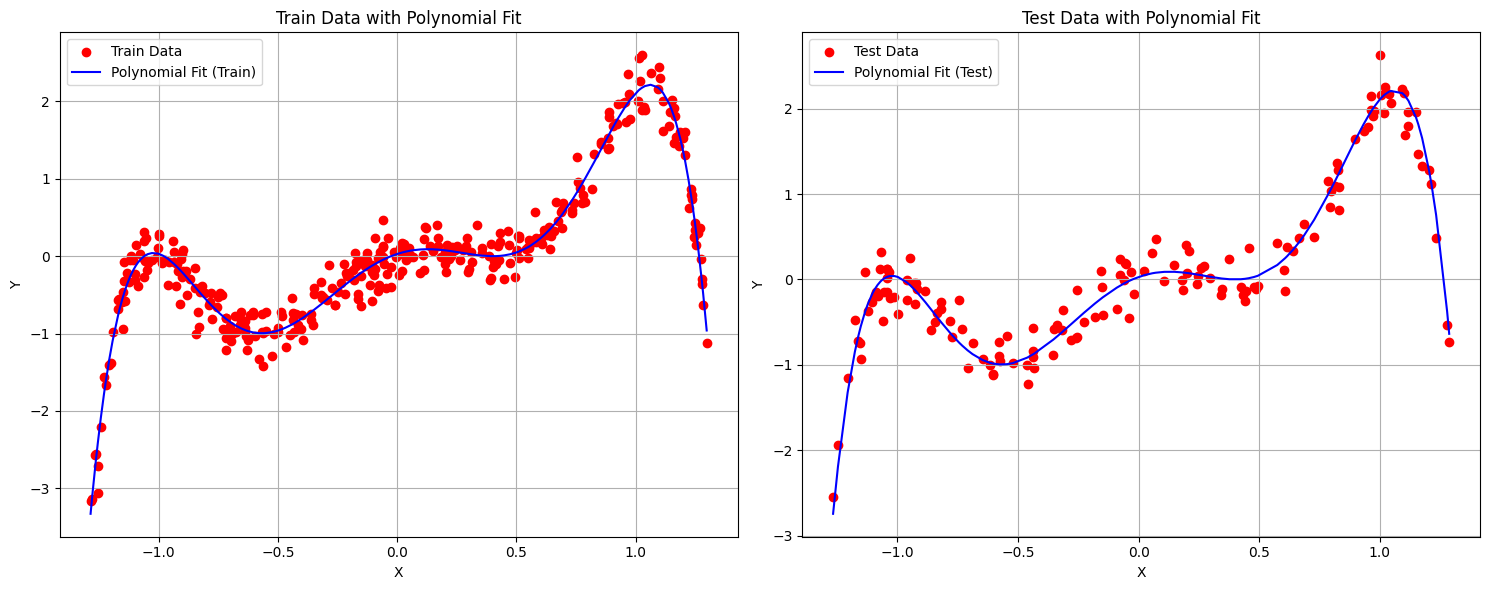


Рисунок 2.7.1 – диаграмма рассеяния данных с линией соответствующей полученному полиному

Из рисунка 2.7.1 видно, что полином 6 степени хорошо аппроксимирует данные с нелинейной зависимостью, т.к. линия полинома идет практически идеально вдоль распределения данных.

1. Оценка модели регрессии.
   1. Загрузим набор данных Student\_Performance (табл. 3.1.1).

Таблица 3.1.1 – содержимое первых строк Student\_Performance

|  | **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 |

Из таблицы можно понять, что Performance Index – это отклик, а значит остальные 5 колонок – предикторы.

* 1. Предобработка данных

Проведем предобработку данных (листинг 3.2.1, табл. 3.2.1).

Листинг 3.2.1 – предобработка данных

| label\_encoders = {}  for column in df\_student\_performance.columns:  if df\_student\_performance[column].dtype == 'object':  label\_encoders[column] = LabelEncoder()  df\_student\_performance[column] = label\_encoders[column].fit\_transform(df\_student\_performance[column])  df\_student\_performance.dropna(inplace=True)  df\_student\_performance.drop\_duplicates(inplace=True)  df\_student\_performance |
| --- |

Таблица 3.2.1 – содержимое первых строк Student\_Performance после предобработки

|  | **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 |

Так как данные из разного диапазона, то нормируем данные при помощи StandardScaler (листинг 3.2.2, рис. 3.2.1).

Листинг 3.1.2 – нормировка данных

| label\_encoders = {}  for column in df\_student\_performance.columns:  if df\_student\_performance[column].dtype == 'object':  label\_encoders[column] = LabelEncoder()  df\_student\_performance[column] = label\_encoders[column].fit\_transform(df\_student\_performance[column])  df\_student\_performance.dropna(inplace=True)  df\_student\_performance.drop\_duplicates(inplace=True)  df\_student\_performance |
| --- |

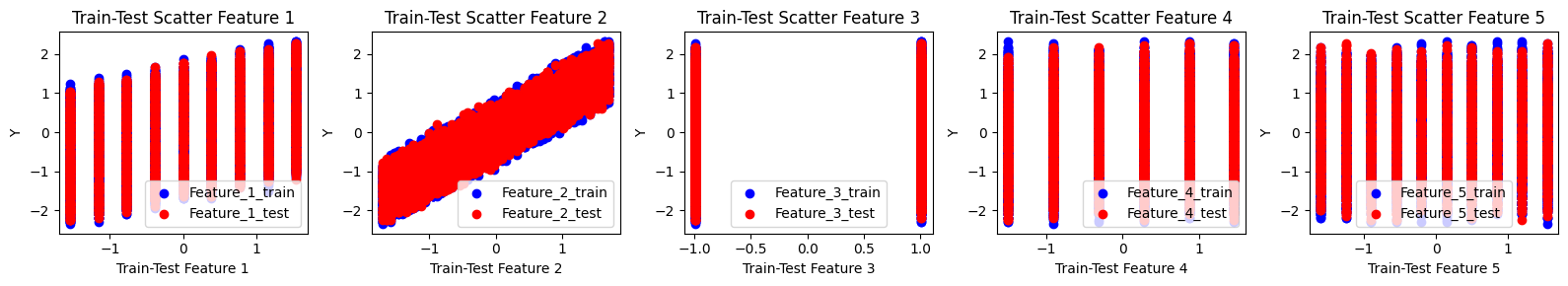


Рисунок 3.2.1 – диаграммы рассеяния двух выборок для каждого предиктора

Из рисунка 3.2.1 можно сказать, что только у второго предиктора есть линейная зависимость, остальные – дискретные.

3.3-3.4. Построим модель, которая будет предсказывать значение признака Performance Index на основе остальных признаков. Проанализируем полученную модель.

Для этого применим SGDRegressor с гиперпараметрами penalty='elasticnet', l1\_ratio=0.5, learning\_rate='adaptive', eta0=0.01, max\_iter=1000, tol=1e-3. Обучим модель, рассчитаем коэффициенты и метрики. Данные метрик находятся в таблице 3.3.1

Таблица 3.3.1 – Метрики

|  | Обучающая выборка | Тестовая выборка |
| --- | --- | --- |
| R2 | 0.9887625742643993 | 0.9884846414554459 |
| MAE | 0.08407098487194044 | 0.0854962577186096 |
| MAPE | 0.3182224371974515 | 0.3423933570697417 |

По метрикам из табл. 3.3.1 можно сказать, что модель хорошо обобщает, т.к. R2 практически равен 1, MAE низок, а MAPE не превышает 35% у обеих выборок.

Коэффициенты регрессии: [0.38480025 0.91853487 0.01688545 0.04176842 0.02903068]

Свободный член: -0.00089884

По коэффициентам можно сказать, что первый и второй предикторы положительно влияют на отклик (второй сильнее). Остальные же не оказывают существенного влияния на отклик.

Информативность признаков – кол-во времени за учебой и предыдущая оценка работы в большей степени влияют на итоговую оценку.

Построим диаграммы рассеяния (рис. 3.3.1)

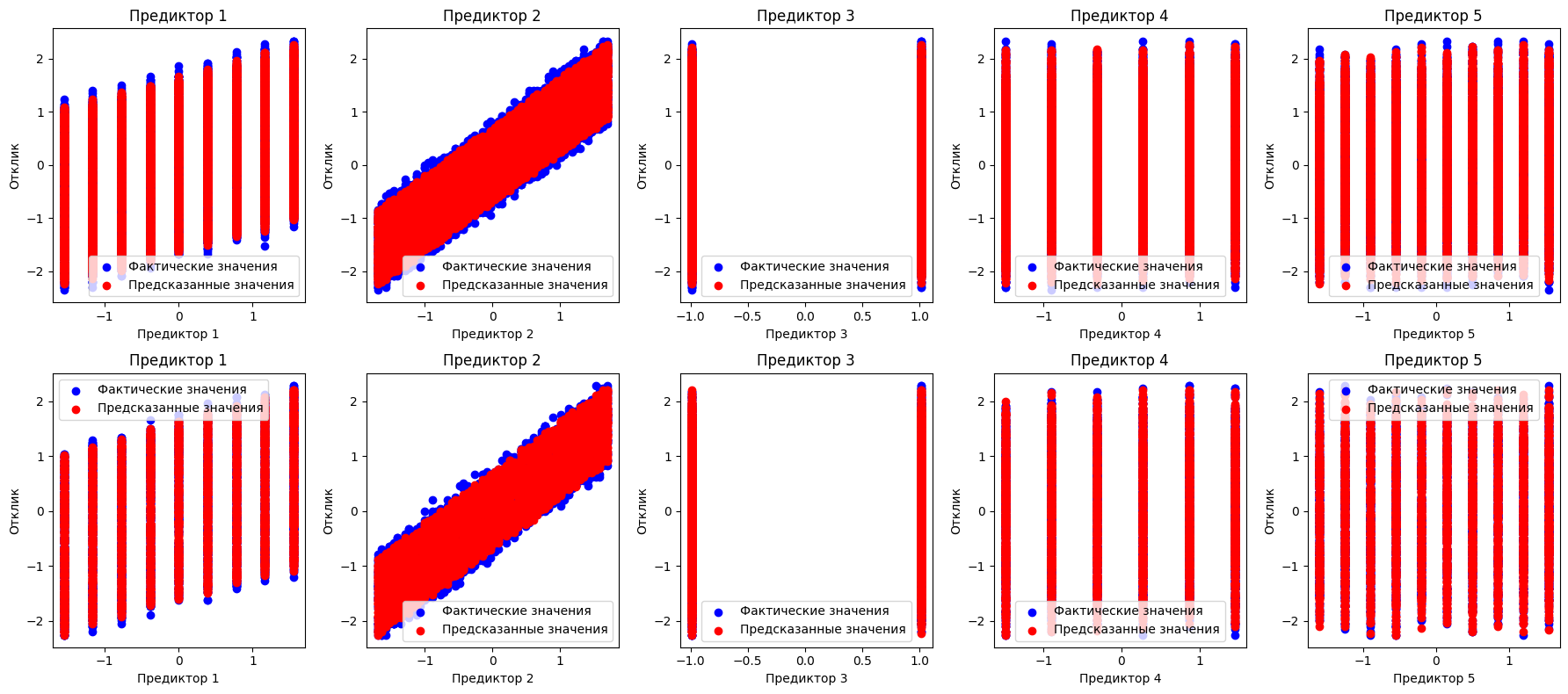


Рисунок 3.3.1 – диаграммы рассеяния между предикторами и откликом

Из рисунка 3.3.1 видно, что модель хорошо обобщает предикторы, но она может столкнуться с проблемой выбросов, переизбытка (1 предиктор) и недостатке данных (2 предиктор). А т.к. 2 предиктор является более значимым, то это может повлечь неверное предсказание итоговой оценки студента.

**Вывод.**

Были изучены линейная и нелинейная регрессии:

* Изучено разбиение на тестовую и обучающую выборки
* Изучено проведение линейной регрессии
* Вычислены коэффициенты детерминации, MAPE, MAE
* Найдена степень полинома, дающая наилучшую аппроксимацию данных, найдена степень, с которой начинается переобучение

Линейная регрессия успешно применятся для данных с линейной зависимостью.

Нелинейная регрессия требует использования полиномиальных признаков для лучшего обобщения данных моделью.

Использование модели регрессии на реальных данных позволяет предсказывать конечное значение основываясь на многих признаках.