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

ОТЧЕТ

по лабораторной работе № 3

по дисциплине «Основы машинного обучения»

Тема: регрессия

Вариант 3В

Студентка гр. 1304 _____ Хорошкова А.С.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2024

Задание.

Провести линейную регрессию.

Провести нелинейную регрессию.

Оценить модель регрессии.

Выполнение работы.

1. Линейная регрессия

Согласно 3 варианту были загружены данные из файла lab3_lin3.csv как Pandas DataFrame с помощью функции upload_df, представленной на Листинг 1.1.

Листинг 1.1

```
def upload_df(file_name, delete_first=True):
    df = pd.read_csv(file_name, delimiter=',')
    if delete_first:
        return df.drop(df.columns[[0]], axis=1)
    else:
        return df
```

С помощью метода print_df_data (Листинг 1.2) был вызван у датафрейма метод head, выводящий первые 5 строк датафрейма, и метод describe, выводящий характеристики датафрейма.

Листинг 1.2

```
def print_df_data(df):
    print("\n============== head ==============")
    print(df.head())
    print("\n============= describe ===============")
    print(df.describe())
```

Результат вывода представлен на Рисунок 1.

```
============== 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

============= describe ===============
                x1          x2          x3          x5           y
count  500.000000  500.000000  500.000000  500.000000  500.000000
mean     0.016720   -0.031923    0.006915    0.085608    1.009930
std      1.031982    1.032431    0.979696    0.980627    8.905402
min     -2.357100   -3.583200   -3.103100   -2.466600  -23.454400
25%     -0.749500   -0.727450   -0.650825   -0.540450   -5.012700
50%      0.025650   -0.036600   -0.051800    0.113450    0.623250
75%      0.710550    0.597925    0.641725    0.740850    6.798875
max      3.023300    3.582300    3.415100    2.957800   34.554500
```

Рисунок 1 - вывод первых пяти строк датафрейма и описания датафрейма

С помощью функции split_and_show, представленной на Листинг 1.3, данные были разбиты на обучающую и тестовую выборки, и были выведены графики, демонстрирующие разбиение данных (Рисунок 2, Рисунок 3, Рисунок 4 и Рисунок 5). По графикам видно, что тестовая выборка соответствует обучающей.

Листинг 1.3

```
def split_and_show(xp_split, yp_split, test_size=0.3, shuffle=False,  alpha=1
    ):
        xp_train, xp_test, yp_train, yp_test = train_test_split(xp_split,
    yp_split, test_size=test_size, shuffle=shuffle)
        print(xp_test)
        print(xp_train)
        print(yp_test)
        print(yp_train)
        for title in xp_split.columns.tolist():
            plt.scatter(xp_train[title], yp_train, c='b', marker='.'
            , alpha =alpha)
      plt.scatter(xp_test[title], yp_test, c='r', marker='.'
            , alpha =alpha)
      plt.grid()
            plt.xlabel(title)
            plt.ylabel("y")
            plt.legend(('train', 'test'))
            plt.show()
      return xp_train, xp_test, yp_train, yp_test
```

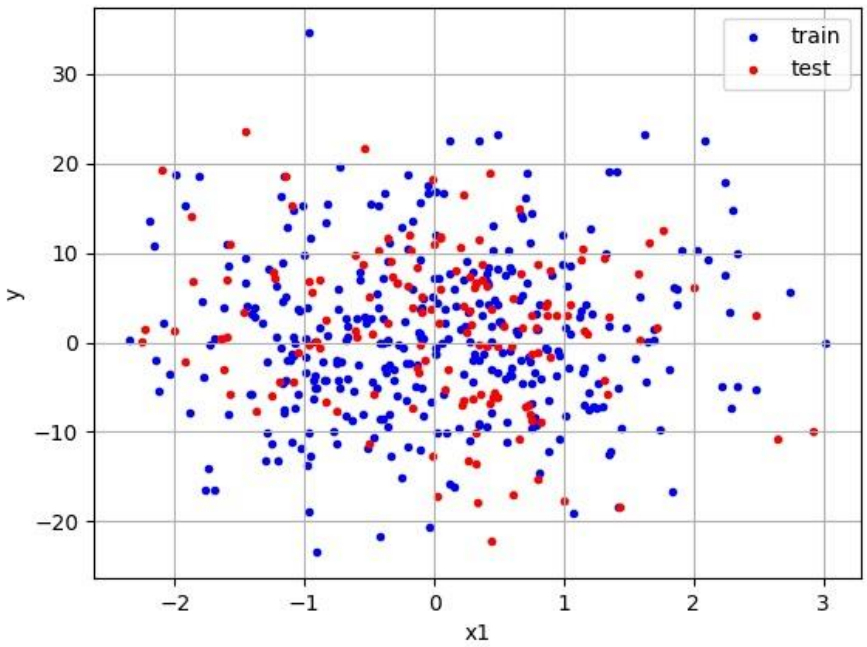

Рисунок 2 - разбиение на обучающую и тестовую выборку для x1 и y

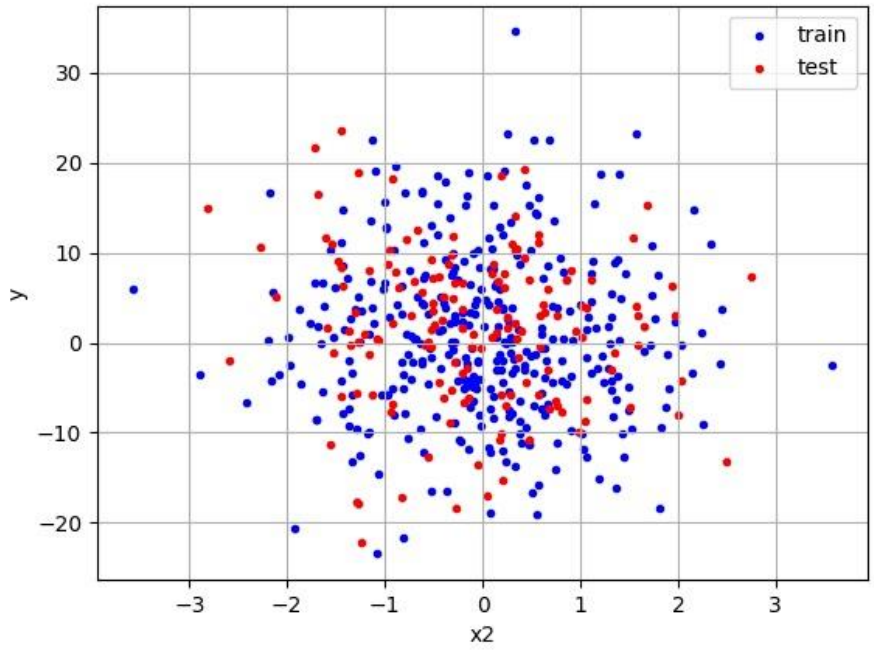

Рисунок 3 - разбиение на обучающую и тестовую выборку для x2 и y

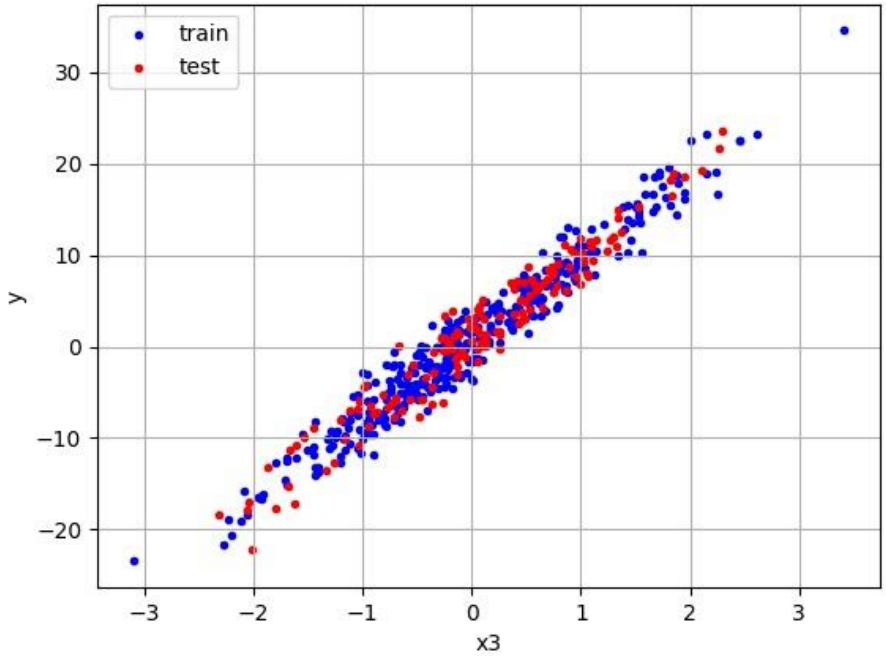

Рисунок 4 - разбиение на обучающую и тестовую выборку для x3 и y

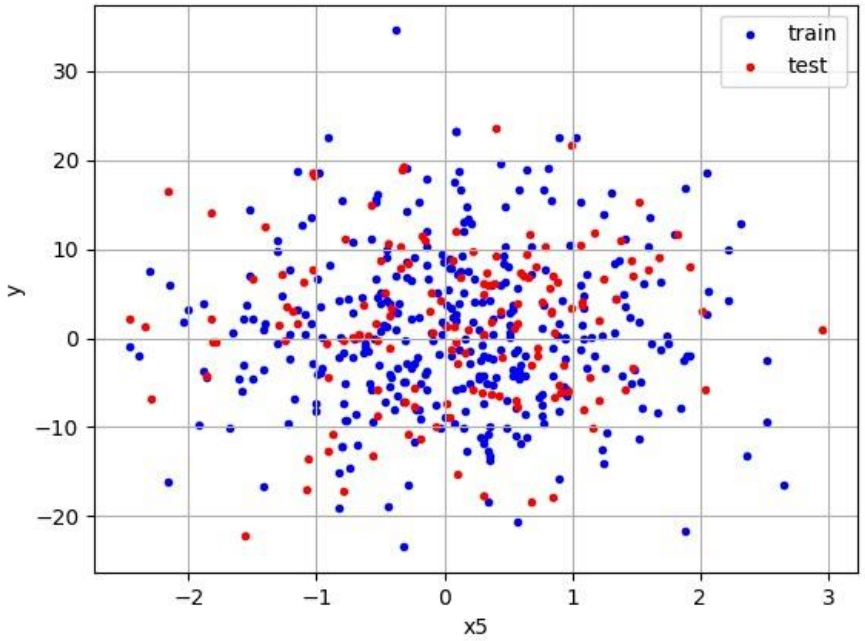

Рисунок 5 - разбиение на обучающую и тестовую выборку для x5 и y

Для разбиения была использована функция train_test_split из библиотеки sklearn. В качестве размера тестовой выборки было выбрано 30% от общего числа данных. Было выбрано значение параметра shuffle=False, так как данные в файле уже были перемешаны, что подтверждается графиками.

Для проведения регрессии различными способами, была написана функция make_regression (Листинг 1.4).

Листинг 1.4

```
def make_regression(xp_train, yp_train, regression, regression_name):
    regression.fit(xp_train, yp_train)
    print("С помощью", regression_name, "получены коэффициенты при x",
regression.coef_,
          "Полученный свободный член ", regression.intercept_)
    return regression
```

Была проведена линейная регрессия с помощью LinearRegression. С помощью линейной регрессии были получены следующие коэффициенты при xn: [-0.04996487 -0.04685254 8.84795721 -0.01990352]. Полученный свободный член равен 0.9341647877193847. Таким образом, линейная регрессия дала следующий многочлен (коэффициенты были округлены до сотых): y=-0.045*x1-0.045*x2+8.85*x3-0.02*x+0.93. Из уравнения видно, что наибольшим образом на y влияет x3 (прямая зависимость), влияние остальных предикторов намного меньше.

Для обучающей и тестовой выборки были рассчитаны коэффициент детерминации, MAPE, MAE. Для вывода метрик была использована функция metrics (Листинг 1.5).

Листинг 1.5

```
def metrics(y_real, y_predicted):
    print("r: ", r2_score(y_real, y_predicted))
    print("MAPE: ", mean_absolute_percentage_error(y_real, y_predicted) *
100)
    print("MAE: ", mean_absolute_error(y_real, y_predicted))
```

Рассчитанные метрики представлены на Рисунок 6. Как видно на рисунке, коэффициент детерминации для тестовой и обучающей выборки примерно равен 0.95, что говорит о хорошем обучении модели и отсутствии переобучения. Средняя абсолютная ошибка для обучающей выборки равна 1.62, для тестовой 1.63. Процентное отклонение для обучающей выборки равно 152%, для тестовой 115%. Скорее всего такие значения были получены из-за того, что, судя по Рисунок 2, Рисунок 3, Рисунок 5, значение y не зависит линейно от x1, x2 и x5, но они вошли в итоговое уравнение, хоть и с небольшими коэффициентами. Также на Рисунок 4 видно, что точки x3 расположены в форме полосы ненулевой ширины вдоль определённой линии, то есть возможна дисперсия данных.

```
========== linear train metrics ==========
r:  0.9516961518125189
MAPE:  152.16926101603093
MAE:  1.618735197645929
========== linear test metrics ==========
r:  0.9468388415760167
MAPE:  115.17705694718097
MAE:  1.6317915803520229
```

Рисунок 6 – метрики для linear regression

Была проведена линейная регрессия с помощью lasso regression с параметром alpha=0.09. Данный параметр был выбран, так как при меньших альфа в итоговое уравнение входили предикторы x1, x2 и x5, но, судя по Рисунок 2, Рисунок 3, Рисунок 5, значение y не зависит линейно от x1, x2 и x5. При больших значениях альфа начинали ухудшаться значения метрик, в частности значение коэффициента детерминации.

Были получены следующие коэффициенты при xn: [-0 -0 8.75459696 -0]. Полученный свободный член равен 0.9303482403687283. Таким образом, регрессия дала следующий многочлен (коэффициенты были округлены до

сотых): y=8.75*x3+0.93. Из уравнения видно, что на y влияет только x3 (прямая зависимость), влияние остальных переменных с данным параметром альфа не выявлено.

Для обучающей и тестовой выборки были рассчитаны коэффициент детерминации, MAPE, MAE. Для вывода метрик была использована функция metrics (Листинг 1.5).

Рассчитанные метрики представлены на Рисунок 7.

```
========== lasso train metrics ==========
r:  0.9515301715143185
MAPE:  150.88418261447362
MAE:  1.6230475932732225
========== lasso test metrics ==========
r:  0.9466495824229048
MAPE:  113.58705571955655
MAE:  1.639928688398066
```

Рисунок 7 – метрики для lasso regression

Была проведена линейная регрессия с помощью ridge regression с параметром alpha=0.1. Данный параметр был выбран, так как при меньших или больших альфа немного ухудшались значения метрик.

Были получены следующие коэффициенты при xn: -0.04980784 -0.04690077 8.84538475 -0.01984256]. Полученный свободный член равен 0.9341079720418788. Таким образом, регрессия дала следующий многочлен (коэффициенты были округлены до сотых): y=-0.050*x1-0.047*x2+8.85*x3-0.02*x5+0.93. Из уравнения видно, что наибольшим образом на y влияет x3 (прямая зависимость), влияние остальных переменных по сравнению с x3 относительно невелико.

Для обучающей и тестовой выборки были рассчитаны коэффициент детерминации, MAPE, MAE. Для вывода метрик была использована функция metrics (Листинг 1.5).

Рассчитанные метрики представлены на Рисунок 8.

```
========== ridge train metrics ==========
r:  0.9516960716834572
MAPE:  152.11914331734428
MAE:  1.6188066025487042
========== ridge test metrics ==========
r:  0.9468366315664933
MAPE:  115.14671746344243
MAE:  1.6318804695564777
```

Рисунок 8 – метрики для ridge regression

Была проведена линейная регрессия с помощью elasticNet regression с параметром alpha=0.13. Данный параметр был выбран, так как при меньших альфа в итоговое уравнение начинали входить предикторы x1, x2 и x5, но, судя по Рисунок 2, Рисунок 3, Рисунок 5, значение y не зависит линейно от x1, x2 и x5. При больших значениях альфа начинали ухудшаться значения метрик, в частности значение коэффициента детерминации.

Были получены следующие коэффициенты при xn: [-0 -0 8.23745844 -0]. Полученный свободный член равен 0.920188537157424. Таким образом, регрессия дала следующий многочлен (коэффициенты были округлены до сотых): y=8.23*x3+0.92. Из уравнения видно, что на y влияет только x3 (прямая зависимость), влияние остальных переменных с выбранным параметром не выявлено.

Для обучающей и тестовой выборки были рассчитаны коэффициент детерминации, MAPE, MAE. Для вывода метрик была использована функция metrics (Листинг 1.5).

Рассчитанные метрики представлены на Рисунок 6.

```
========== elastic train metrics ==========
r:  0.9471307790449968
MAPE:  141.47067721183402
MAE:  1.6959027102997997
========== elastic test metrics ==========
r:  0.9415568033756476
MAPE:  108.77640144700891
MAE:  1.7257307275826337
```

Рисунок 9 – метрики для elasticNet regression

Была проведена линейная регрессия с помощью gradient regression со значениями по умолчанию. Были получены следующие коэффициенты при xn: [0 -0.04432464 -0.04223644 8.84991897 -0.0187903]. Полученный свободный

член равен 0.92701116. Таким образом, регрессия дала следующий многочлен (коэффициенты были округлены до сотых): y=-0.04*x1-0.04*x2+8.85*x3-0.02*x5+0.93. Из уравнения видно, что наибольшим образом на y влияет x3 (прямая зависимость), влияние остальных переменных по сравнению с x3 относительно невелико.

Для обучающей и тестовой выборки были рассчитаны коэффициент детерминации, MAPE, MAE. Для вывода метрик была использована функция metrics (Листинг 1.5).

Рассчитанные метрики представлены на Рисунок 10.

```
========== gradient train metrics ==========
r:  0.9516947691109088
MAPE:  152.5968007540086 %
MAE:  1.6178552961364163
========== gradient test metrics ==========
r:  0.9468407618578378
MAPE:  115.12701743430239 %
MAE:  1.6322206283157952
```

Рисунок 10 – метрики для gradient regression

Итоговая таблица со сравнением всех регрессий и метрик представлена на Таблица 1.

Таблица 1

	коэффициенты					метрики			
	при X1*	при X2*	при X3*	при X5*	свободный член*	выборка	R***	MAPE **	MAE *
linear	-0.05	-0.05	8.85	-0.02	0.93	train	0.952	152	1.62
						test	0.947	115	1.63
lasso	-0	-0	8.75	-0	0.93	train	0.952	150	1.62
						test	0.947	113	1.64
ridge	-0.05	-0.05	8.85	-0.02	0.93	train	0.952	152	1.62
						test	0.947	115	1.63
Elastic Net	-0	-0	8.23	-0	0.92	train	0.947	141	1.70
						test	0.942	108	1.72
gradient	-0.04	-0.04	8.85	-0.02	0.93	train	0.952	152	1.62
						test	0.947	115	1.63

* - округлено до сотых

** - округлено до целых

*** - округлено до тысячных

Основные недостатки метода линейной регрессии заключаются в невозможности работы с данными сложной формы, а также в сильной чувствительности от выбросов. Недостатком lasso является то, что он может отбросить в действительности нужные предикторы при увеличении значения альфа. Недостатком ridge метода является то, что он не способен отбросить ненужные предикторы, а может только уменьшить их значение, приблизив к нулю. Недостатком метода Elastic Net является то, что подбор оптимальных параметров может быть нетривиальным. Недостатком gradient также является сложность подбора оптимальных параметров.

В Таблица 1 видно, что все способы линейной регрессии дали схожие результаты. В качестве лучшей модели была выбрана модель, полученная с помощью lasso regression, так как эта модель даёт хорошее значение коэффициента детерминации и относительно неплохие остальные метрики, и, судя по графикам, набор данных содержит предикторы, от которых у не зависит линейно, поэтому метод должен уметь отбрасывать лишние предикторы.

С помощью метода show_predicted_plot (Листинг 1.6) были построены диаграммы рассеяния между предикторами и откликом. Диаграмма была построена по тестовой выборке. Результаты представлены на Рисунок 11, Рисунок 12, Рисунок 13 и Рисунок 14. По рисункам видно, что предсказанные данные похожи на тестовые.

Листинг 1.6

```
def show_predicted_plot(x, y, y_predicted, x_lable='x', y_lable='y'):
    plt.plot(x, y, 'k.')
    plt.plot(x, y_predicted, 'rx')
    plt.legend(('real', 'predict'))
    plt.title("test sample prediction")
    plt.xlabel(x_lable)
    plt.ylabel(y_lable)
    plt.grid()
    plt.show()
```

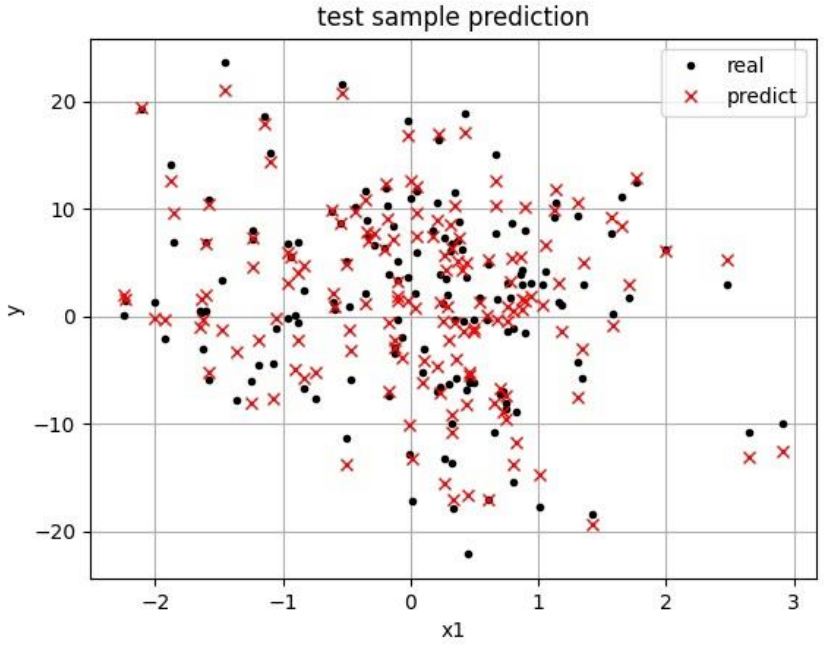

Рисунок 11 – зависимость x1 и y

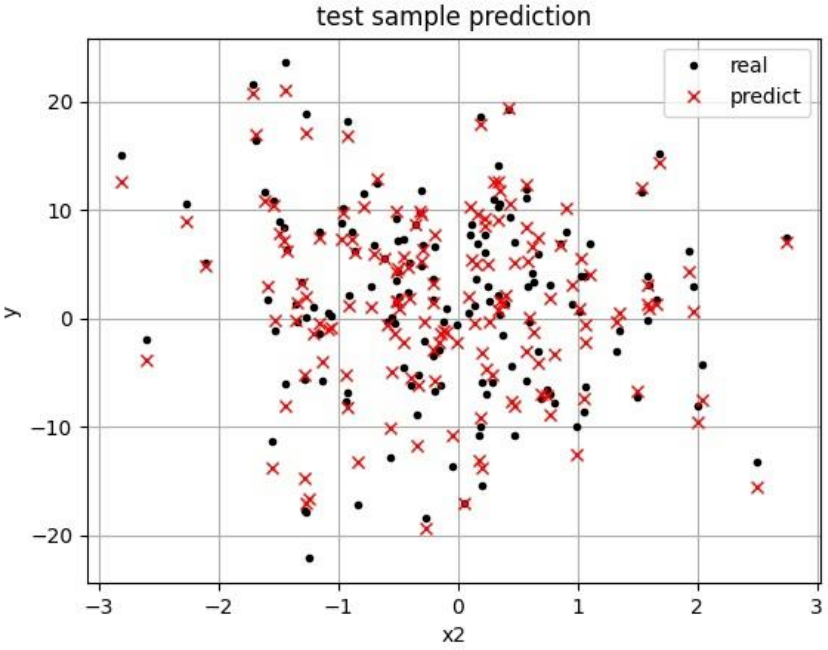

Рисунок 12 – зависимость x2 и y

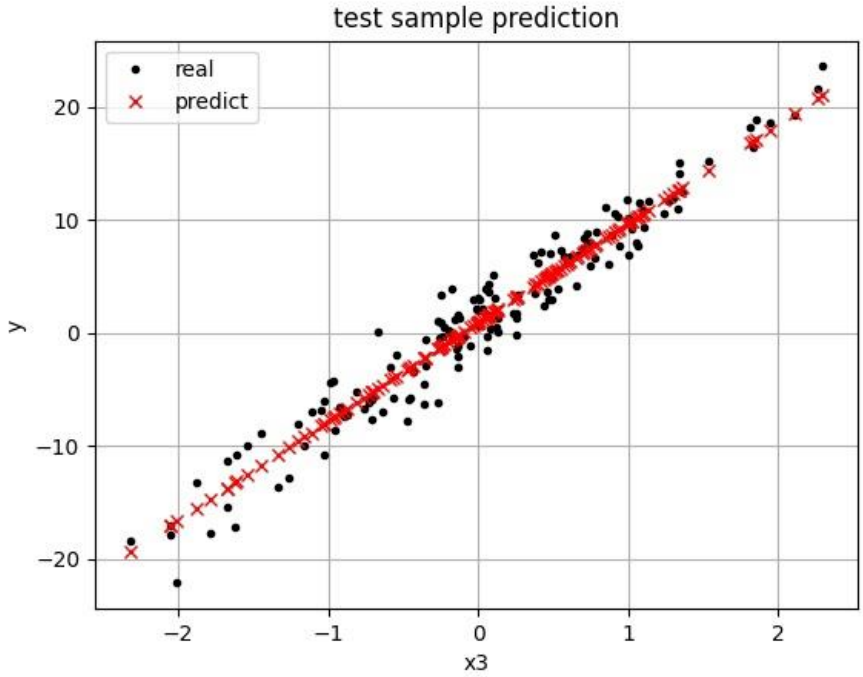

Рисунок 13 – зависимость x3 и y

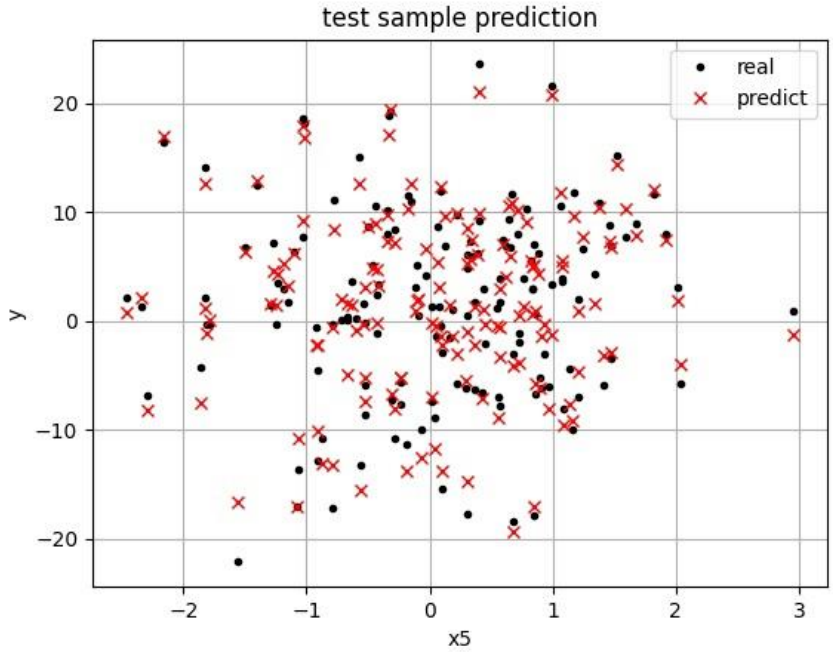

Рисунок 14 – зависимость x5 и y

Таким образом, с помощью lasso regression удалось построить модель, которая предсказывает похожие на тестовые значения.

2. Нелинейная регрессия

Согласно варианту Б были загружены данные из файла lab3_poly2.csv как Pandas DataFrame с помощью функции upload_df, представленной на Листинг 1.1.

С помощью метода print_df_data (Листинг 1.2) был вызван у датафрейма метод head, выводящий первые 5 строк датафрейма, и метод describe, выводящий характеристики датафрейма.

Результат вывода представлен на Рисунок 1.

```
============== head ==============
        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

============= describe ===============
                x           y
count  500.000000  500.000000
mean    -0.040319    0.032006
std      0.758688    0.930334
min     -1.284700   -3.163100
25%     -0.719125   -0.500125
50%     -0.074400   -0.062100
75%      0.602725    0.288200
max      1.297200    2.625400
```

Рисунок 15 - вывод первых пяти строк датафрейма и описания датафрейма

С помощью функции train_test_split из библиотеки sklearn данные были разбиты на обучающую и тестовую выборки, и был выведен график, демонстрирующий разбиение данных (Рисунок 16). По графикам видно, что тестовая выборка соответствует обучающей.

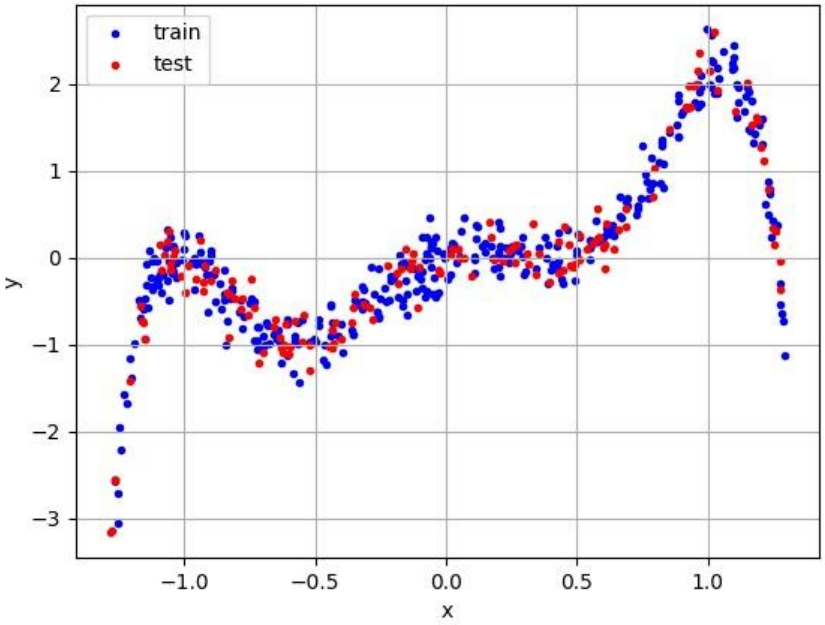

Рисунок 16 – разбиение данных на обучающую и тестовую выборку

Была проверена работа стандартной линейной регрессии на загруженных данных с помощью функции make_regression (Листинг 1.4). На Рисунок 17 показана диаграмма рассеяния данных с выделенной полученной линией регрессии.

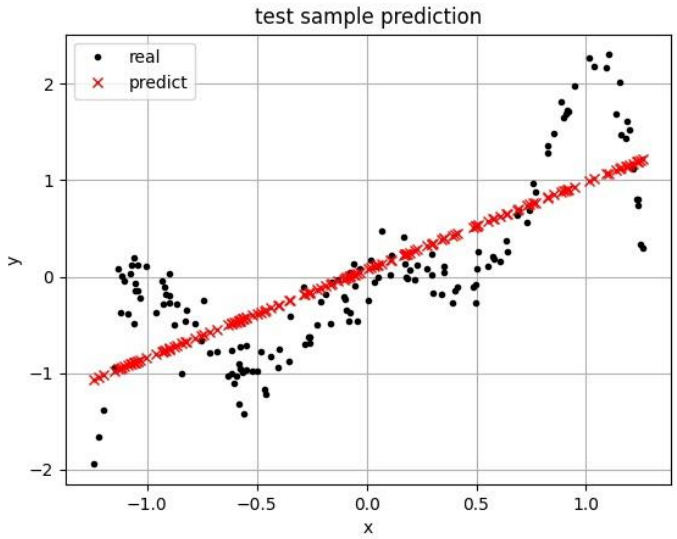

Рисунок 17 – результат линейной регрессии

На Рисунок 17 видно, что предсказанные с помощью решгрессии значения не похожи на тестовую выборку. Такой результат получен из-за того, что данные имеют сложную форму, а метод линейной регрессии плохо работает с такими данными.

Для обучающей и тестовой выборки были рассчитаны коэффициент детерминации, MAPE, MAE. Для вывода метрик была использована функция metrics (Листинг 1.5). По результатам, представленным на Рисунок 18, подтверждается то, что с помощью линейной регрессии не удалось построить хорошую модель: коэффициент детерминации равен 0.5, что говорит о том, что модель плохо обучена.

```
========== linear train metrics ==========
r:  0.5286642189613187
MAPE:  363.56577555737925
MAE:  0.5098527156940852
========== linear test metrics ==========
r:  0.573578975322532
MAPE:  467.1445071066335
MAE:  0.46816303421187155
```

Рисунок 18 – метрики для линейной регрессии

Была написана функция get_predicted_poly_y , представленная на Листинг 2.1, с помощью которой возможно конструировать полиномиальные признаки для разных степеней полинома и рассчитывать предсказанные значения y по заданным x.

Листинг 2.1

```
def get_predicted_poly_y(x, num):
    poly_transform = PolynomialFeatures(num)
    xnl_poly = poly_transform.fit_transform(xp_train)
    lin_nl2 = LinearRegression(fit_intercept=False)
    lin_nl2.fit(xnl_poly, yp_train)
    poly_line = poly_transform.transform(x.reshape(-1, 1))
    y_pred2 = lin_nl2.predict(poly_line)
    return y_pred2
```

С помощью функции show_metrics_plot (Листинг 2.2) был построен график зависимости коэффициента детерминации от степени полинома.

По графику на Рисунок 19 видно, что оптимальные значения степени полинома начинаются со значения 6. Переобучение модели начинается после значения степени полинома, равного 35, когда значения коэффициента детерминации для тестовой выборки начинает резко ухудшаться.

Таким образом, для аппроксимации данных будет использована степень полинома 6.

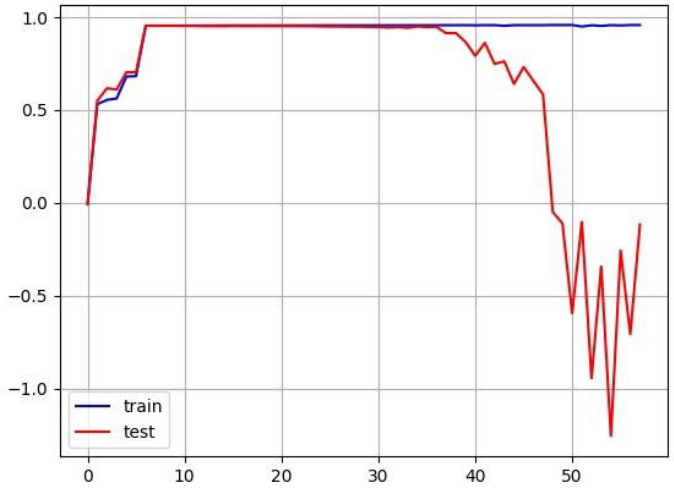

Рисунок 19 – график зависимости коэффициента детерминации от степени полинома

Листинг 2.2

```
def show_metrics_plot(x_train, x_test, y_train, y_test):
    N = 56
    r_train = np.zeros(N)
    r_test = np.zeros(N)
    for num in range(N):
        r_train[num] = r2_score(y_train, get_predicted_poly_y(x_train,
num))
        r_test[num] = r2_score(y_test, get_predicted_poly_y(x_test, num))
        print("\n========== train r2 n=", num, " ==========")
        print(r_train[num])
        print("========== test r2 n=", num, " ==========")
        print(r_test[num])
    plt.plot(range(N), r_train, 'b-')
    plt.plot(range(N), r_test, 'r-')
    plt.grid()
    plt.legend(("train", "test"))

    plt.show()
```

Для степени полинома 6 для обучающей и тестовой выборки были рассчитаны коэффициент детерминации, MAPE, MAE. Для вывода метрик была использована функция metrics (Листинг 1.5).

Результат представлен на Рисунок 20.

```
========== poly train metrics ==========
r:  0.9559809476418263
MAPE:  103.24316273838632
MAE:  0.15892347643189242
========== poly test metrics ==========
r:  0.949014328411333
MAPE:  79.4364763742067
MAE:  0.15707784368868072
```

Рисунок 20 – метрики полиномиальной регрессии

Как видно на Рисунок 20, коэффициент детерминации примерно равен 0.95 для обеих выборок, что довольно близко к 1. Следовательно, модель хорошо обучена.

Был найден аппроксимирующий полином с помощью функции make_nonlinear_poly_regression (Листинг 2.3). Был получен полином $y=-0.18667416*x^6-0.23487423*x^5-0.045141444*x^4-0.5764218*x^3+1.8566804*x^2+0.5853177*x+0.05002964$.

Листинг 2.3

```
def make_nonlinear_poly_regression(n, xp_train, yp_train):
    poly_transform = PolynomialFeatures(n)
    xnl_poly = poly_transform.fit_transform(xp_train)
    linear = LinearRegression(fit_intercept=False)
    linear.fit(xnl_poly, yp_train)
    return linear
```

Была построена диаграмма рассеяния данных с линией, соответствующей полученному полиному. Диаграмма показана на Рисунок 21. Рисунок подтверждает, что полученный полином имеет схожую форму с данными, что говорит о хорошем обучении модели.

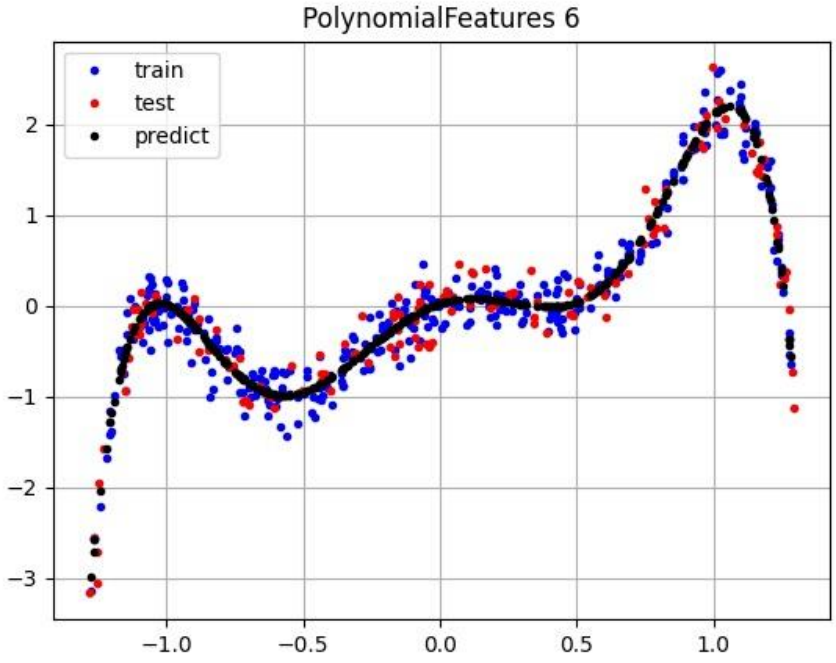

Рисунок 21 - диаграмма рассеяния для полиномиальной регрессии

Для выбранной степени полинома 6 была решена задача нелинейной регрессии без конструирования полиномиальных признаков с использованием библиотеки TensorFlow.

С помощью функции make_nonlinear_tensor_regression (Листинг 2.4) был получен полином $y=-5.110662*x^6-0.06828326*x^5+10.208904*x^4+0.1401791*x^3-4.0896606*x^2+0.9723677*x+0.03127954$.

Листинг 2.4

```
def make_nonlinear_tensor_regression():
    x_tf = tf.constant(np.array(xp_train), dtype=tf.float32)
    y_tf = tf.constant(np.array(yp_train), dtype=tf.float32)
    w = [tf.Variable(np.random.randn()) for _ in range(7)]
    print(*w)
    alpha = tf.constant(0.2, dtype=tf.float32)
    epoch_n = 9000
    for epoch in range(epoch_n):
        with tf.GradientTape() as tape:
            y_pred = w[0] * x_tf ** 6 + w[1] * x_tf ** 5 + w[2] * x_tf ** 4 +
w[3] * x_tf ** 3 + w[4] * x_tf ** 2 + w[
                5] * x_tf + w[6]
            loss = tf.reduce_mean(tf.square(y_tf - y_pred))
        grad = tape.gradient(loss, w)
        for i in range(7):
            w[i].assign_add(-(alpha * grad[i]))
        if (epoch + 1) % 750 == 0:
            print(f"E: {epoch + 1}, L: {loss.numpy()}")
    return w
```

Также для степени полинома 6 для обучающей и тестовой выборки были рассчитаны коэффициент детерминации, MAPE, MAE. Для вывода метрик была использована функция metrics (Листинг 1.5).

Результат представлен на Рисунок 22.

```
========== poly train metrics ==========
r:  0.9543831316548854
MAPE:  108.19330181180943
MAE:  0.15396604702490568
========== poly test metrics ==========
r:  0.9530946858673859
MAPE:  71.6923309280341
MAE:  0.1665831578971247
```

Рисунок 22 – метрики для нелинейной регрессии, сконструированной с помощью TensorFlow

Как видно на Рисунок 22, коэффициент детерминации примерно равен 0.95 для обеих выборок, что довольно близко к 1. Следовательно, модель хорошо обучена.

Была построена диаграмма рассеяния данных с линией, соответствующей полученному полиному. Диаграмма показана на Рисунок 23. Рисунок также

показывает, что полученный полином имеет схожую форму с данными, что говорит о хорошем обучении модели.

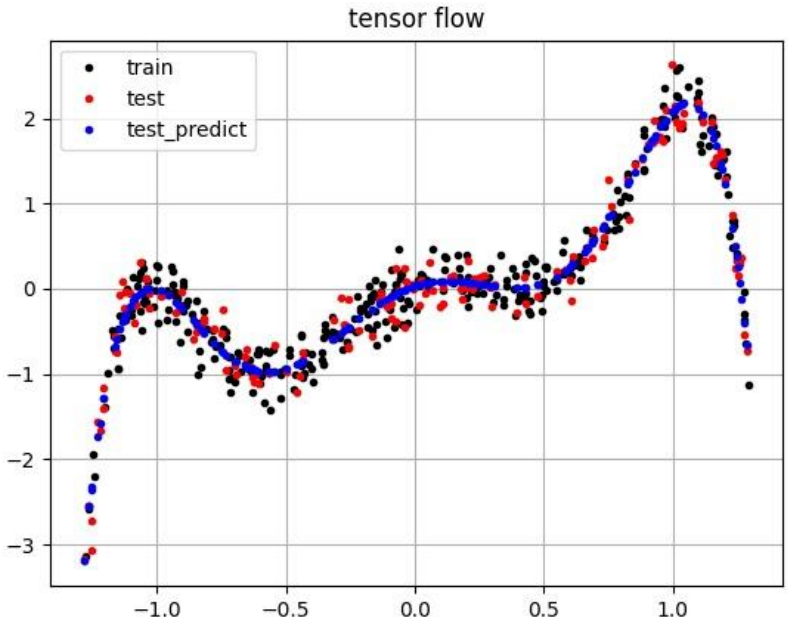

Рисунок 23 - диаграмма рассеяния для полиномиальной регрессии

Таким образом, для данных сложной формы подошли обе модели, полученные с помощью нелинейной регрессии. Для обеих моделей были получены схожие метрики и диаграммы рассеяния, показывающие успешность обучения модели. Однако для данных не подошла модель, построенная методом линейной регрессии. Это подтверждают как метрики, так и диаграмма рассеяния.

3. Оценка модели регрессии

Были загружены данные из файла Student_Performance.csv как Pandas DataFrame с помощью функции upload_df, представленной на Листинг 1.1.

С помощью метода print_df_data (Листинг 1.2) был вызван у датафрейма метод head, выводящий первые 5 строк датафрейма, и метод describe, выводящий характеристики датафрейма.

Результат вывода представлен на Рисунок 24

```
============== head ==============
   Hours Studied  Previous Scores Extracurricular Activities  Sleep Hours  \
0              7               99                        Yes            9
1              4               82                         No            4
2              8               51                        Yes            7
3              5               52                        Yes            5
4              7               75                         No            8

   Sample Question Papers Practiced  Performance Index
0                                 1               91.0
1                                 2               65.0
2                                 2               45.0
3                                 2               36.0
4                                 5               66.0

============= describe ===============
       Hours Studied  Previous Scores   Sleep Hours  \
count   10000.000000     10000.000000  10000.000000
mean        4.992900        69.445700      6.530600
std         2.589309        17.343152      1.695863
min         1.000000        40.000000      4.000000
25%         3.000000        54.000000      5.000000
50%         5.000000        69.000000      7.000000
75%         7.000000        85.000000      8.000000
max         9.000000        99.000000      9.000000

       Sample Question Papers Practiced  Performance Index
count                      10000.000000       10000.000000
mean                           4.583300          55.224800
std                            2.867348          19.212558
min                            0.000000          10.000000
25%                            2.000000          40.000000
50%                            5.000000          55.000000
75%                            7.000000          71.000000
max                            9.000000         100.000000
```

Рисунок 24 - вывод первых пяти строк датафрейма и описания датафрейма

С помощью функции prepearing_df (Листинг 3.1) была проведена предобработка данных - замена текстовых данных, удаление null-значений, удаление дубликатов..

Листинг 3.1

```
def prepearing_df(data):
    result_data = replace_df_column(data, 'Extracurricular Activities',
class_mapping)
    result_data = result_data.dropna()
    result_data = result_data.drop_duplicates()
    return result_data
```

С помощью функции split_and_show, представленной на Листинг 1.3, данные были разбиты на обучающую и тестовую выборки, и были выведены графики, демонстрирующие разбиение данных (Рисунок 25, Рисунок 26, Рисунок 27, Рисунок 28 и Рисунок 29). Так как на графиках точки были очень плотно расположены, и тестовые данные на графиках могут перекрывать обучающие, значение прозрачности для точек было снижено до 0.1. По графикам видно, что тестовая выборка соответствует обучающей.

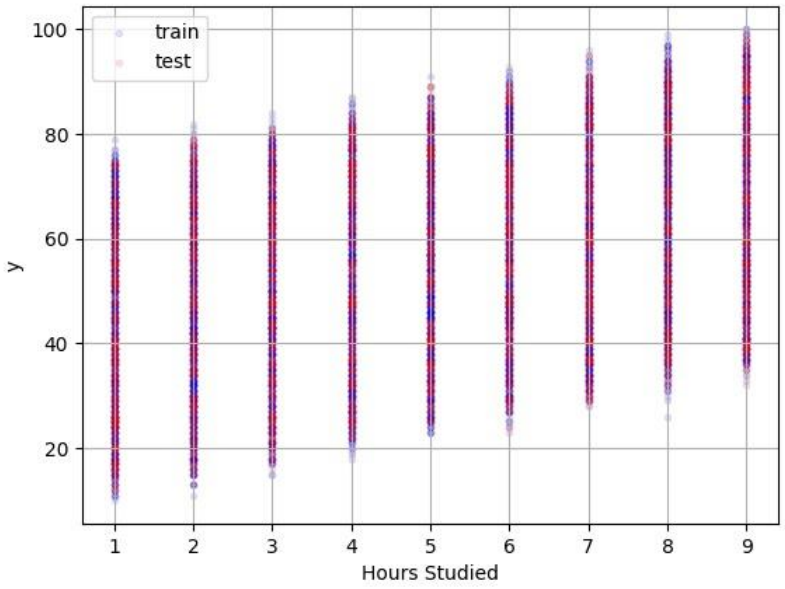

Рисунок 25- зависимость Performance Index от Hours Studied

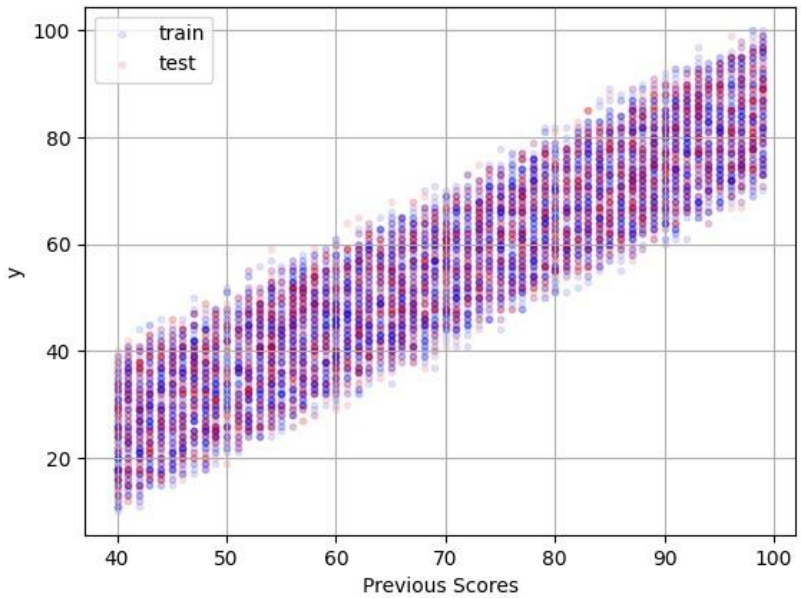

Рисунок 26- зависимость Performance Index от Previous Scores

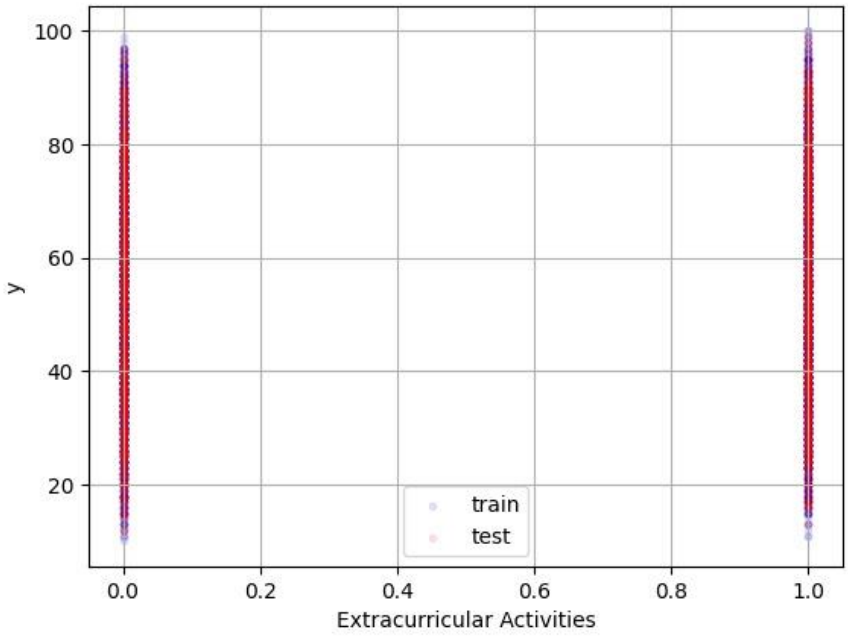

Рисунок 27- зависимость Performance Index от Extracurricular Activities

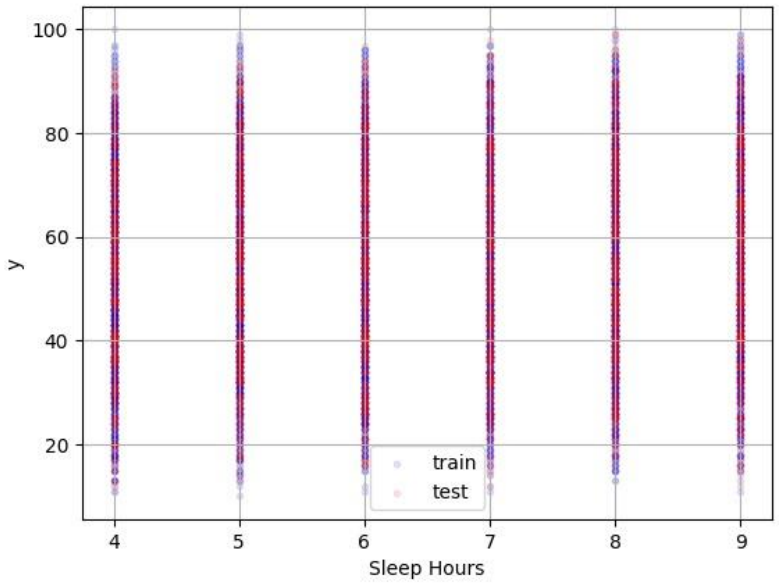

Рисунок 28 - зависимость Performance Index от Sleep Hours

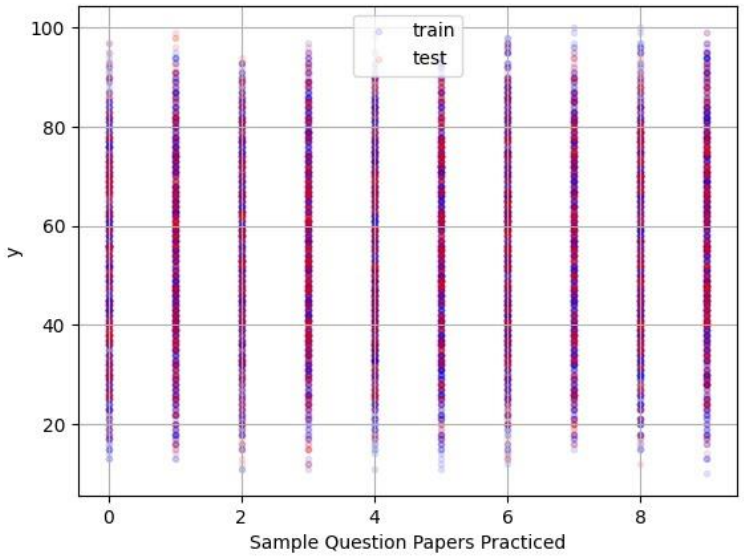

Рисунок 29 - зависимость Performance Index от Sample Question Papers Practiced

Была проведена регрессия с помощью linear regression, lasso regression , ridge regression, elastic net regression с параметром alpha=0.1. Данный параметр был выбран, так как при нём были достигнуты достаточно хорошие метрики для всех моделей. При большом увеличении альфа переставали учитываться последние три предиктора, что ухудшало метрики, при уменьшении альфа значение метрик практически не изменялось. Рассчитанные метрики с помощью функции metrics (Листинг 1.5) показаны на Рисунок 30.

```
С помощью linear получены коэффициенты при x [2.85442904 1.01838731 0.64768793 0.47283788 0.19493335] Полученный свободный член  -34.07110482636021
========== linear train metrics ==========
r:  0.9887625844040151
MAPE:  3.461884465434605 %
MAE:  1.614803852281639
========== linear test metrics ==========
r:  0.9884847172154818
MAPE:  3.4767099183019 %
MAE:  1.6422038622694426
С помощью lasso получены коэффициенты при x [2.85297996 1.01835976 0.60761536 0.46918707 0.19382725] Полученный свободный член  -34.01343524126602
========== lasso train metrics ==========
r:  0.9887613233561099
MAPE:  3.462042452772153 %
MAE:  1.6148684301129994
========== lasso test metrics ==========
r:  0.988488147196386
MAPE:  3.4769451012064696 %
MAE:  1.6421137388870513
С помощью ridge получены коэффициенты при x [2.85442842 1.0183873  0.64768418 0.47283763 0.19493333] Полученный свободный член  -34.07109783253317
========== ridge train metrics ==========
r:  0.988762584403998
MAPE:  3.4618845332623196 %
MAE:  1.6148038610854543
========== ridge test metrics ==========
r:  0.9884847177713031
MAPE:  3.4767099454343127 %
MAE:  1.6422038443416598
С помощью elastic получены коэффициенты при x [2.85156642 1.01835534 0.61534635 0.47014922 0.1943299 ] Полученный свободный член  -34.01846917052853
========== elastic train metrics ==========
r:  0.988761658409437
MAPE:  3.4622489320033374 %
MAE:  1.6148833953524233
========== elastic test metrics ==========
r:  0.9884879699732717
MAPE:  3.4769700009910323 %
MAE:  1.6421190287143286
```

Рисунок 30 – рассчитанные метрики для различных моделей

По Рисунок 30 видно, что все рассмотренные методы дали хорошо обученную модель, о чём говорит коэффициент детерминации, близкий к 0.99 у всех моделей как для тестовой, так и для обучающей выборки. В качестве наилучшей модели было решено взять модель, обученную с помощью lasso regression, так как коэффициент детерминации для тестовой выборки у этой модели самый большой.

Была проведена линейная регрессия с помощью lasso regression с выбранным ранее значением alpha=0.1. Были получены следующие коэффициенты при xn: [0 2.85297996 1.01835976 0.60761536 0.46918707 0.19382725]. Полученный свободный член равен -34.01343524126602.

Таким образом, регрессия дала следующий многочлен (коэффициенты были округлены до сотых): y=2.85*”Hours Studied”+1.02*”Previous Scores”+0.61*” Extracurricular Activities”+0.47*”Sleep Hours”+0.19*”Sample Question Papers Practiced”-34.01. Из уравнения видно, что на Performance Index повлияли все предикторы. Наибольшим образом влияет Hours Studied, среднее влияние имеют Previous Scores, Extracurricular Activities и Sleep Hours (расположены в порядке убывания влияния), и наименьшее влияние производит Sample Question Papers Practiced. Все предикторы имеют прямое влияние на отклик, так как входят в уравнение с положительным знаком.

В получившейся модели спорным является коэффициент при Sample Question Papers Practiced. При бОльших значениях альфа этот коэффициент обнулялся, но было решено оставить этот предиктор значимым, так как его обнуление ухудшало значение метрик в том числе для тестовой выборки.

Заключение.

В ходе лабораторной работы были изучены различные методы регрессии. Были проведены линейная и нелинейная регрессии различных наборов данных.