**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# Лабораторная работа № 2 по дисциплине «Методы машинного обучения»

**Тема** Модель полиномиальной регрессии - Регуляризация

**Студент** Сапожков А.М,

**Группа** ИУ7-23М

**Преподаватель** Солодовников В.И.

Москва, 2025

# Содержание

# ВВЕДЕНИЕ

В современной науке и инженерии методы машинного обучения играют всё более важную роль в решении сложных задач, таких как прогнозирование, классификация и анализ данных. Однако при работе с данными часто возникает проблема переобучения (overfitting), когда модель слишком хорошо подгоняется к конкретной выборке данных и перестаёт работать эффективно на новых, не встречавшихся ранее наблюдениях.

Одним из основных методов борьбы с проблемой переобучения является регуляризация. В рамках этой лабораторной работы мы рассмотрим основы полиномиальной регрессии и различные виды регуляризаций, используемых для предотвращения переобучения: L1-регуляризации (Lasso) и L2-регуляризации (Ridge).

Целью данной лабораторной работы изучение полиномиальной регрессии и регуляризации.

Задачи данной лабораторной работы:

1) для модели полиномиальной регрессии, полученной в ЛР 1 (п. 1) (оптимальный вариант), а также для полиномов больших степеней (+5, +10) вывести значения коэффициентов полинома;

2) к выбранным моделям (полиномам соответствующих степеней) применить метод регуляризации с использованием гребневой регрессии (ридж-регрессии) и Лассо-регрессии.

3) вывести значения коэффициентов полинома для различных значений параметра $\lambda$.

4) рассчитать функционал эмпирического риска (функционал качества) для всех полученных моделей на обучающей и контрольной выборках (вывести графики).

# 1 Аналитическая часть

## 1.1 Модель полиномиальной регрессии

Полиномиальная регрессия является одним из наиболее широко используемых алгоритмов машинного обучения, нацеленных на аппроксимацию нелинейной зависимости между переменными. Основная идея состоит в том, чтобы найти полиномиальное выражение для целевой переменной на основе набора входных переменных.

Пусть $\mathbf{x}$ является набором входных переменных и $y$ — целевой функцией, которую мы хотим предсказать. Цель полиномиальной регрессии — найти коэффициенты $w = (w_0, w_1, ..., w_n)$ в таких, что

$$y \approx w^T \phi(\mathbf{x})$$

где $\phi$ — функция, которая преобразует исходные данные в набор признаков.

## 1.2 Регуляризация

Регуляризация (англ. regularization) в статистике и машинном обучении — метод добавления некоторых дополнительных ограничений к условию с целью предотвратить переобучение. Чаще всего эта информация имеет вид штрафа за сложность модели.

Если выбрана излишне сложная модель при недостаточном объеме данных, то в итоге может быть получена модель, которая хорошо описывает обучающую выборку, но не обобщается на тестовую. Переобучение в большинстве случаев проявляется в том, что итоговые модели имеют слишком большие значения параметров. Одним из способов борьбы с негативным эффектом излишнего подстраивания под данные — использование регуляризации, т.е. добавление некоторого штрафа за большие значения коэффициентов у линейной модели. Тем самым запрещаются слишком «резкие» изгибы, и предотвращается переобучение.

Наиболее часто используемые виды регуляризации — $L_1$ и $L_2$, а также их линейная комбинация — эластичная сеть.

В представленных ниже формулах для эмпирического риска $Q$ приняты следующие обозначения: $L$ — функция потерь, $\beta$ — вектор параметров $g(x_i, \beta)$ из модели алгоритма, $\lambda$ – неотрицательный гиперпараметр (коэффициент регуляризации).

Если в качестве функционал качества используется сумма квадратов остатков (Residual Sum of Squares - RSS), тогда изначально:

$$L(y_i, g(x_i, \beta)) = (g(x_i, \beta) - y_i)^2 \tag{1.1}$$

$$RSS = Q(\beta, X') = \sum_{i=1}^{l} L(y_i, g(x_i, \beta)) = \sum_{i=1}^{l} (g(x_i, \beta) - y_i)^2 \tag{1.2}$$

$L_2$-регуляризация (ridge regularization) или регуляризация Тихонова (Tikhonov regularization):

$$Q(\beta, X') = \sum_{i=1}^{l} L(y_i, g(x_i, \beta)) + \lambda \sum_{j=1}^{n} \beta_j^2 \tag{1.3}$$

Минимизация регуляризованного соотвествующим образом эмпирического риска приводит к выбору такого вектора параметров $\beta$, которое не слишком сильно отклоняется от нуля. В линейных классификаторах это позволяет избежать проблем мультиколлинеарности и переобучения.

$L_1$-регуляризация (lasso regularization) или регуляризация через манхэттенское расстояние:

$$Q(\beta, X') = \sum_{i=1}^{l} L(y_i, g(x_i, \beta)) + \lambda \sum_{j=1}^{n} |\beta_j| \tag{1.4}$$

Данный вид регуляризации также позволяет ограничить значения вектора $\beta$. Однако, к тому же обладает интересными и полезными свойствами — обнуляет значения некоторых параметров, что в случае с линейными моделями приводит к отбору признаков.

# 2 Технологическая часть

## 2.1 Средства реализации

В качестве языка программирования для реализации выбранных алгоритмов был выбран язык программирования Python ввиду наличия библиотек для обучения регрессионных моделей, таких как sklearn и numpy.

## 2.2 Реализация алгоритмов

На листинге 2.1 представлена реализация обучения модели полиномиальной регрессии с использованием метода наименьших квадратов, а также с применением Ридж-регрессии и Лассо-регрессии.

Листинг 2.1 — Обучение модели полиномиальной регрессии с регуляризацией

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 50)
y = 1/2*x + 2*np.sin(x) + 5
y = y + np.random.randn(50)*0.5
plt.scatter(x,y)
plt.show()

degrees = [6, 11, 16, 25]
polyline = np.linspace(np.min(x), np.max(x), 250)

for degree in degrees:
  model = np.poly1d(np.polyfit(x, y, degree))
  plt.plot(polyline, model(polyline), label='Полином степени
     {}:'.format(degree))
  print('Полином степени {}:'.format(degree))
  print(model)
  print()

plt.scatter(x, y, color='black')
plt.legend()
plt.show()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size
   =0.20, random_state=777)
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures

alphas = np.logspace(-3, 2, 12, endpoint=True)
scores = []
for alpha in alphas:
  cur_scores = []
  for degree in degrees:
    poly = PolynomialFeatures(degree=degree)
    xp = poly.fit_transform(x.reshape(-1, 1))
    X_train, X_test, y_train, y_test = train_test_split(xp, y,
       test_size=0.20, random_state=777)
    ridge_regression = Ridge(alpha=alpha)
    ridge_regression.fit(X_train, y_train)
    print('Полином степени {}, alpha {}:'.format(degree, alpha))
    print(ridge_regression.coef_)
    score = ridge_regression.score(X_test, y_test)
    cur_scores.append(score)
    print('Точность предсказания: {}'.format(score))
    print()
  scores.append(cur_scores)

from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
ax = plt.axes(projection='3d')
Alphas, Degrees = np.meshgrid(alphas, degrees)
ax.set_xlabel('alphas')
ax.set_ylabel('degrees')
ax.set_zlabel('scores')
ax.scatter(Alphas, Degrees, np.transpose(scores), c=np.transpose(
   scores))

from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.preprocessing import PolynomialFeatures

alphas = np.logspace(-3, 2, 12, endpoint=True)
scores = []
```

```
for alpha in alphas:
  cur_scores = []
  for degree in degrees:
    poly = PolynomialFeatures(degree=degree)
    xp = poly.fit_transform(x.reshape(-1, 1))
    X_train, X_test, y_train, y_test = train_test_split(xp, y,
       test_size=0.20, random_state=777)
    lasso_regression = Lasso(alpha=alpha)
    lasso_regression.fit(X_train, y_train)
    print('Полином степени {}, alpha {}:'.format(degree, alpha))
    print(lasso_regression.coef_)
    score = lasso_regression.score(X_test, y_test)
    cur_scores.append(score)
    print('Точность предсказания: {}'.format(score))
    print()
  scores.append(cur_scores)

from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
ax = plt.axes(projection='3d')
Alphas, Degrees = np.meshgrid(alphas, degrees)
ax.set_xlabel('alphas')
ax.set_ylabel('degrees')
ax.set_zlabel('scores')
ax.scatter(Alphas, Degrees, np.transpose(scores), c=np.transpose(
   scores))

def compare_models(degree, alpha):
  polyline = np.linspace(np.min(x), np.max(x), 250)
  model = np.poly1d(np.polyfit(x, y, degree))
  plt.plot(polyline, model(polyline), label='Без регуляризации')
  poly = PolynomialFeatures(degree=degree)
  xp = poly.fit_transform(x.reshape(-1, 1))
  ridge_regression = Ridge(alpha=alpha)
  ridge_regression.fit(xp, y)
  plt.plot(polyline, ridge_regression.predict(poly.fit_transform(
     polyline.reshape(-1, 1))), label='Ridge регрессия')
  lasso_regression = Lasso(alpha=alpha)
  lasso_regression.fit(xp, y)
  plt.plot(polyline, lasso_regression.predict(poly.fit_transform(
```

```
        polyline.reshape(-1, 1))), label='Lasso регрессия')
    plt.scatter(x, y, color='red')
    plt.title('Полином степени {}:'.format(degree))
    plt.legend()
    plt.show()

degree = 6
alpha = 1e-3
compare_models(degree, alpha)

degree = 11
alpha = 1e-3
compare_models(degree, alpha)

degree = 16
alpha = 1e-3
compare_models(degree, alpha)

degree = 25
alpha = 1e-3
compare_models(degree, alpha)
```

# 3 Исследовательская часть

## 3.1 Среда для тестирования

Для тестирования разработанного алгоритма применялась облачная платформа Google Colab, не требующая установки ПО на локальный компьютер.

## 3.2 Обучение модели полиномиальной регрессии

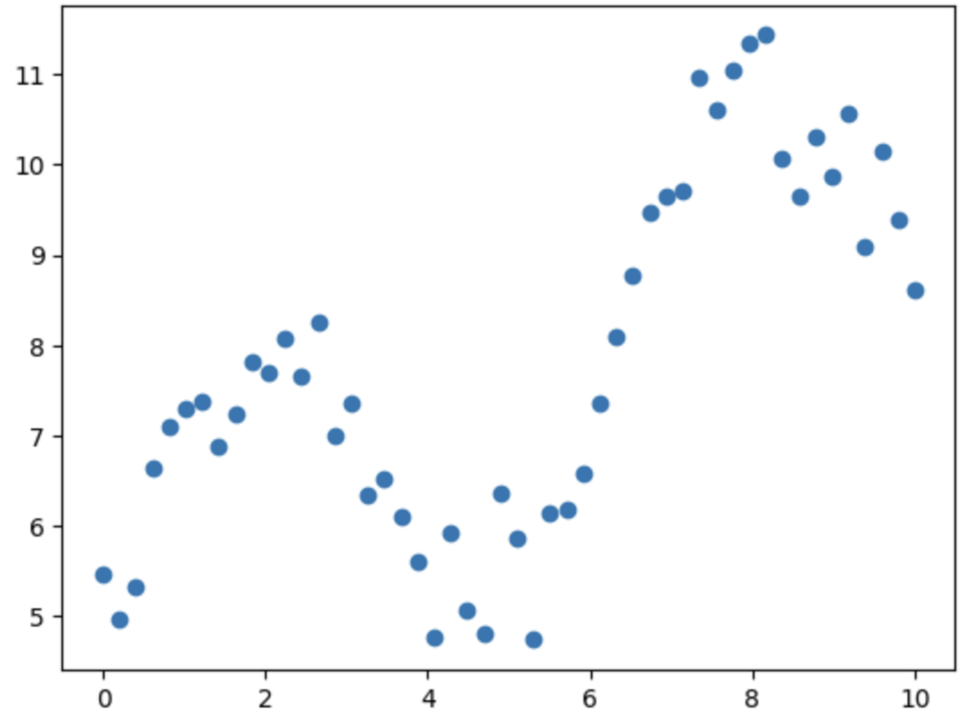


Рисунок 3.1 — Исходные данные

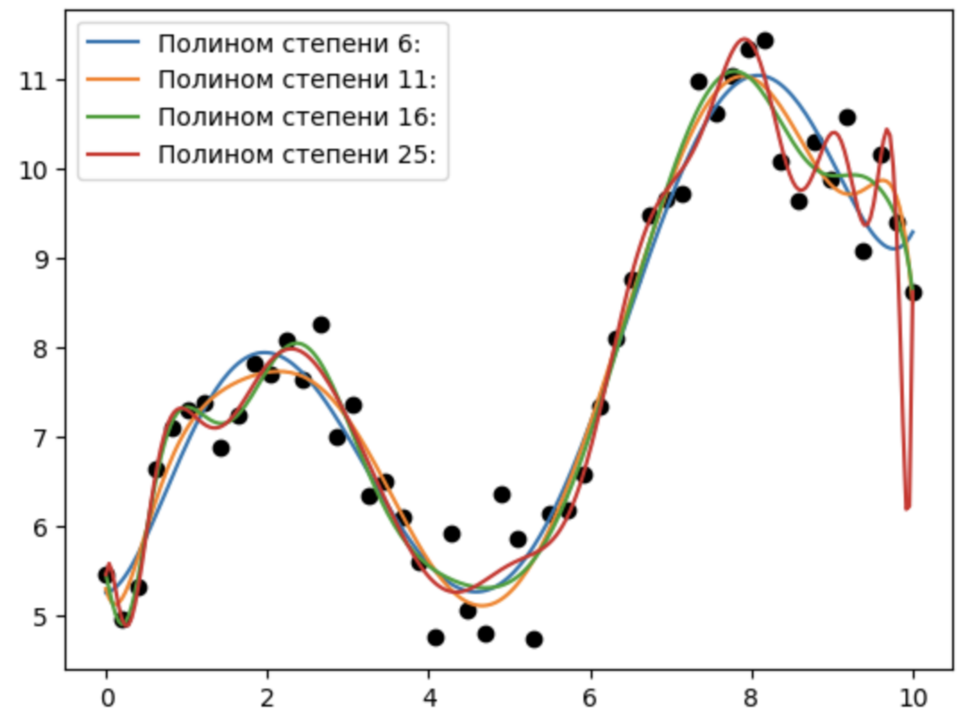


Рисунок 3.2 — Аппроксимация полиномами различных степеней

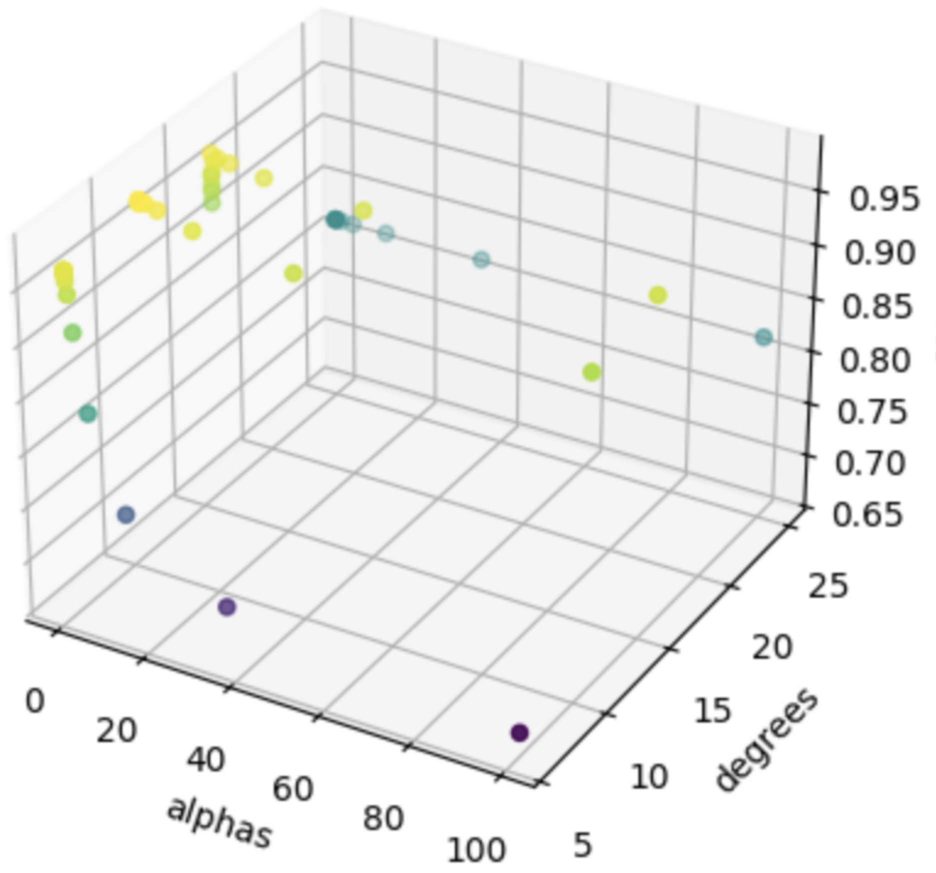


Рисунок 3.3 — Зависимость точности Ридж-регрессии от значения параметра регуляризации и степени полинома

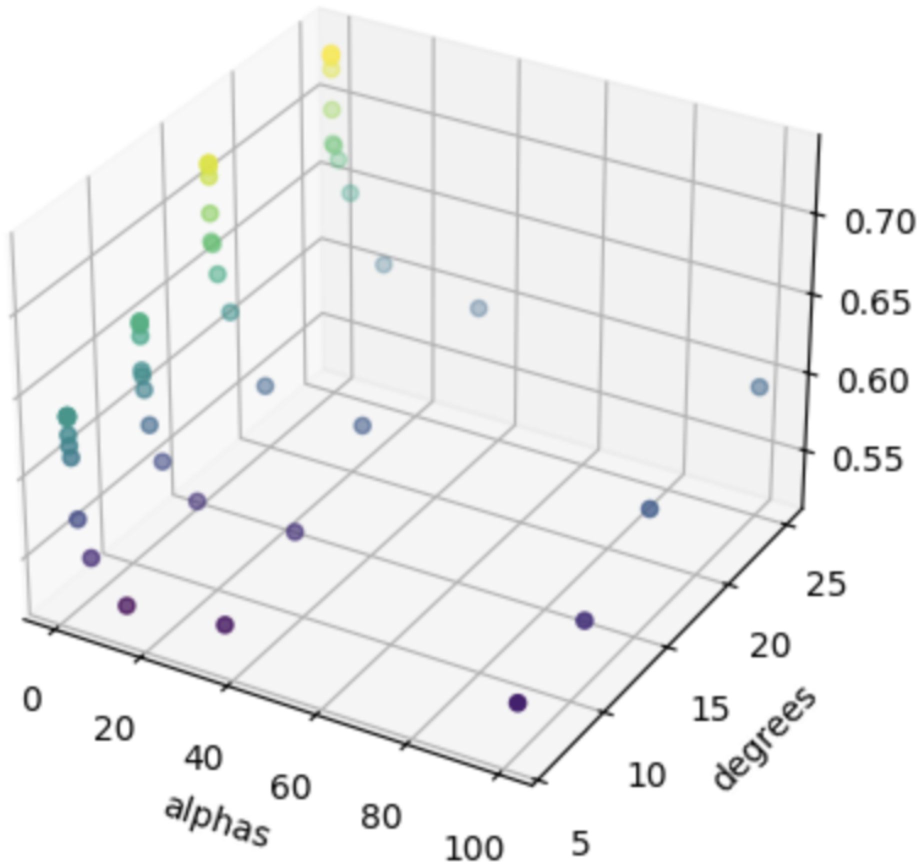


Рисунок 3.4 — Зависимость точности Лассо-регрессии от значения параметра регуляризации и степени полинома

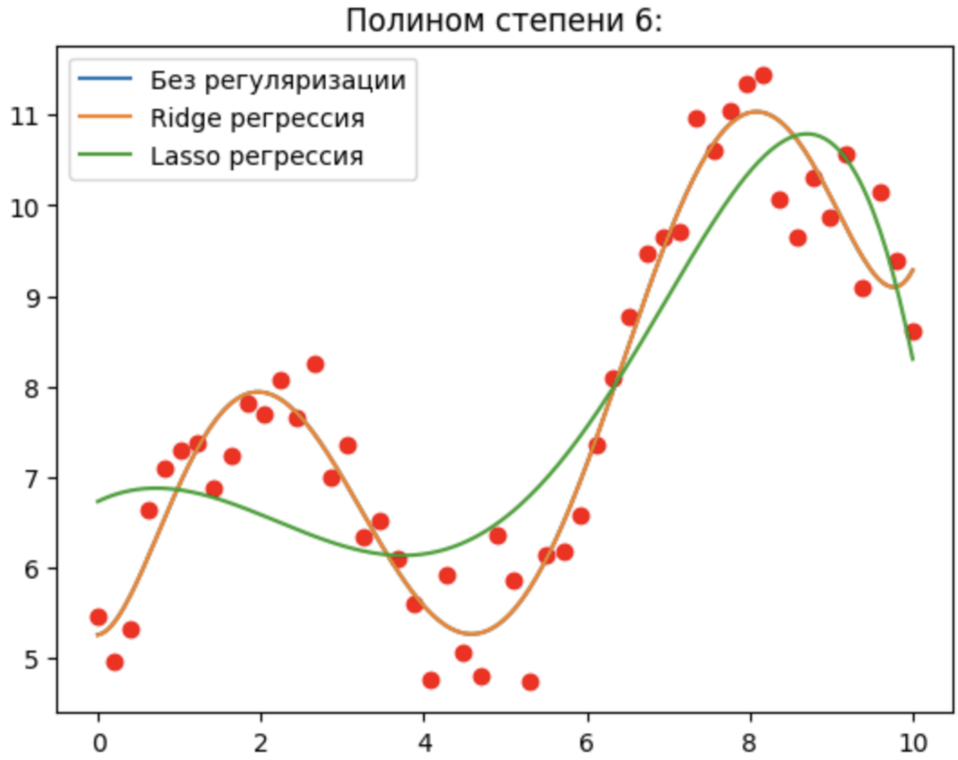


Рисунок 3.5 — Сравнение кривых, полученных без регуляризации, с применением Ридж-регрессии и Лассо-регрессии для полинома степени 6

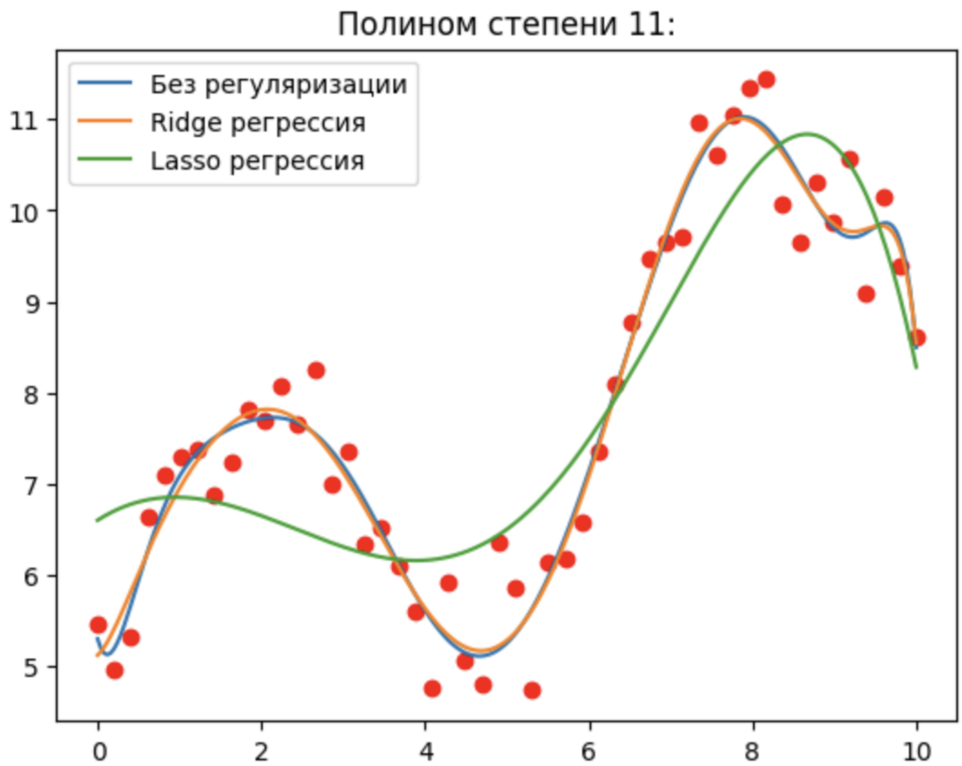


Рисунок 3.6 — Сравнение кривых, полученных без регуляризации, с применением Ридж-регрессии и Лассо-регрессии для полинома степени 11

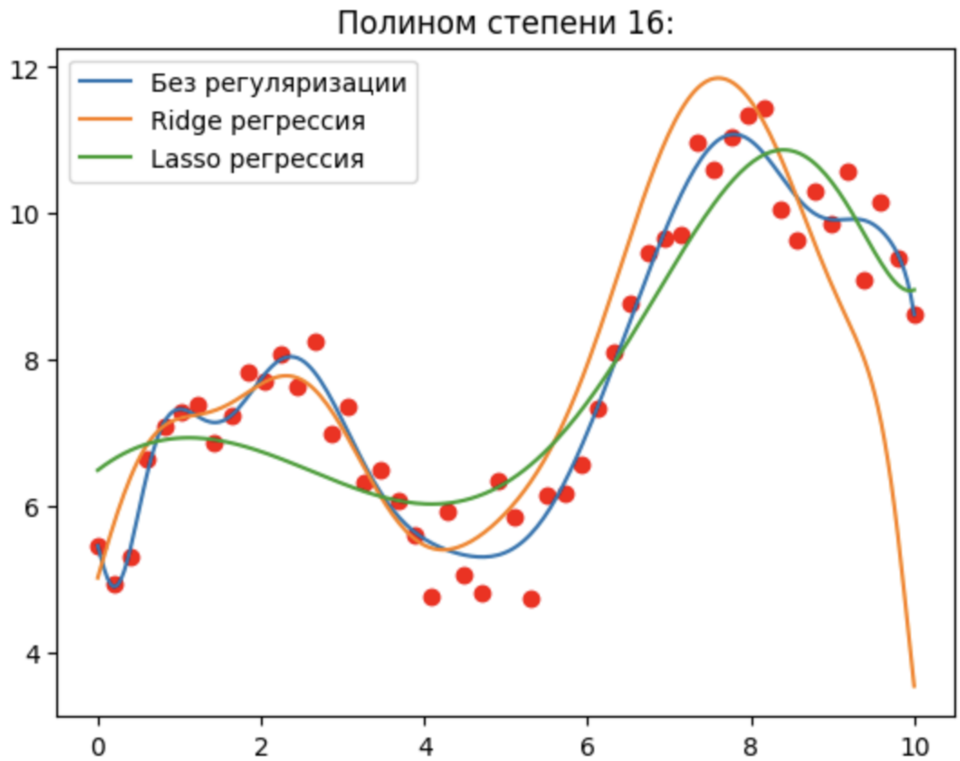


Рисунок 3.7 — Сравнение кривых, полученных без регуляризации, с применением Ридж-регрессии и Лассо-регрессии для полинома степени 16

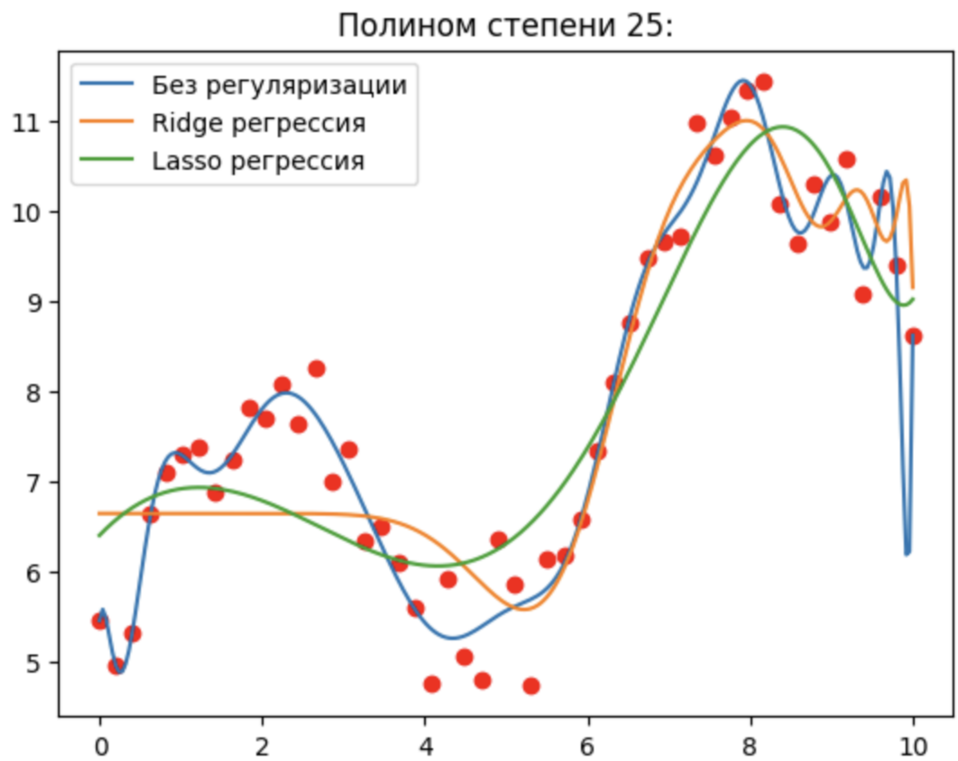


Рисунок 3.8 — Сравнение кривых, полученных без регуляризации, с применением Ридж-регрессии и Лассо-регрессии для полинома степени 25

# ЗАКЛЮЧЕНИЕ

В рамках лабораторной работы была изучена модель полиномиальной регрессии и регуляризация. Все поставленные задачи были выполнены.

1) Для модели полиномиальной регрессии, полученной в ЛР 1 (п. 1) (оптимальный вариант), а также для полиномов больших степеней (+5, +10) выведены значения коэффициентов полинома;

2) К выбранным моделям (полиномам соответствующих степеней) применён метод регуляризации с использованием гребневой регрессии (ридж-регрессии) и Лассо-регрессии.

3) Выведены значения коэффициентов полинома для различных значений параметра $\lambda$.

4) Рассчитан функционал эмпирического риска (функционал качества) для всех полученных моделей на обучающей и контрольной выборках (выведены графики).

Ридж-регрессия позволяет добиться наибольшей точности обучения при степени полинома 11 и параметре регуляризации $\alpha = 1e - 3$. Лассо-регрессия позволяет добиться наибольшей точности обучения при степени полинома 25 и $\alpha = 1e - 3$.