Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ульяновский государственный технический университет» Кафедра «Вычислительная техника»

Дисциплина «Системы искусственного интеллекта»

Лабораторная работа №3

«Линейная регрессия»

Вариант №15

Выполнил студент группы УПАСбд-41:

Пяткин И.А. Проверил: Хайруллин И.Д.

Ульяновск, 2025

# Задание:

1. Написать программу, которая разделяет исходную выборку на обучающую и тестовую (training set, test set). Использовать стандартные функции (train\_test\_split и др. нельзя).
2. С использованием библиотеки scikit-learn обучить модель линейной регрессии по обучающей выборке.
3. Проверить точность модели по тестовой выборке.
4. Построить модель с использованием полиномиальной функции. Построить графики зависимости точности на обучающей и тестовой выборке от степени полиномиальной функции.
5. Построить модель с использованием регуляризации. На основе экспериментов подобрать параметры для регуляризации. Построить графики зависимости точности модели на обучающей и тестовой выборках от коэффициента регуляризации.

Датасет по варианту: ISTANBUL STOCK EXCHANGE.

# Информация по датасету:

Данный датасет содержит информацию о существующих биржах. В данном случае нас интересует, как будет меняться курс турецкой лиры на бирже, в зависимости от изменения курса других мировых бирж.

# Описание реализации:

Первым делом, необходимо читать данные из csv файла.

Необходимо разделить выборку на тренировочную и тестовую(80% и 20%).

**Обучение линейное регрессии:**

**Линейная регрессия** — это метод статистического анализа и машинного обучения, который моделирует связь между зависимой (предсказываемой) переменной и одной или несколькими независимыми переменными с помощью линейного уравнения **y = kx + b.**

Для данной лабораторной работы была использована функция LinearRegression() - модель, которая обучается с помощью линейной регрессии. После обучения, были выполнены предсказание и нахождение точности между обучающими и тестовыми данными.

**Обучение модели по PolynomialFeatures:**

**Полиномиальная регрессия** — это метод, который позволяет моделировать нелинейные зависимости, расширяя возможности линейной регрессии путём добавления новых признаков в виде степеней исходных. Вместо прямой линии (как в линейной регрессии) она создаёт кривую (параболу, кубическую кривую и т.д.), что позволяет лучше аппроксимировать данные с криволинейными тенденциями.

В лабораторной работе используется функция **PolynomialFeatures**() - модель, которая обучается с помощью полиномиальной регрессии. После обучения, были выполнены предсказание и нахождение точности между обучающими и тестовыми данными. Был построен графики точности зависимости между степенями полиномиальной регрессии.

**Регуляризация L1 и L2 (Ridge и Lasso):**

**Лассо-регрессия (Lasso — Least Absolute Shrinkage and Selection Operator)** — это метод линейной регрессии, который добавляет штраф за сумму абсолютных значений коэффициентов (L1-регуляризацию) к стандартной задаче минимизации суммы квадратов ошибок. Это приводит к обнулению коэффициентов наименее важных признаков, что позволяет проводить отбор признаков и делать модель более интерпретируемой.

**Гребневая (или ridge) регрессия** — это метод линейной регрессии, который используется для уменьшения переобучения и стабилизации модели, когда наблюдается мультиколлинеарность (сильная корреляция) между независимыми переменными. Она достигается путем добавления штрафного слагаемого (L2-регуляризации) к функции потерь, которое наказывает за большие значения коэффициентов модели.  
Использовались функции Ridge и Lasso. Были подобраны параметры регуляризации. И построены графики зависимости точности модели на обучающей и тестовой выборках.

# Вывод:

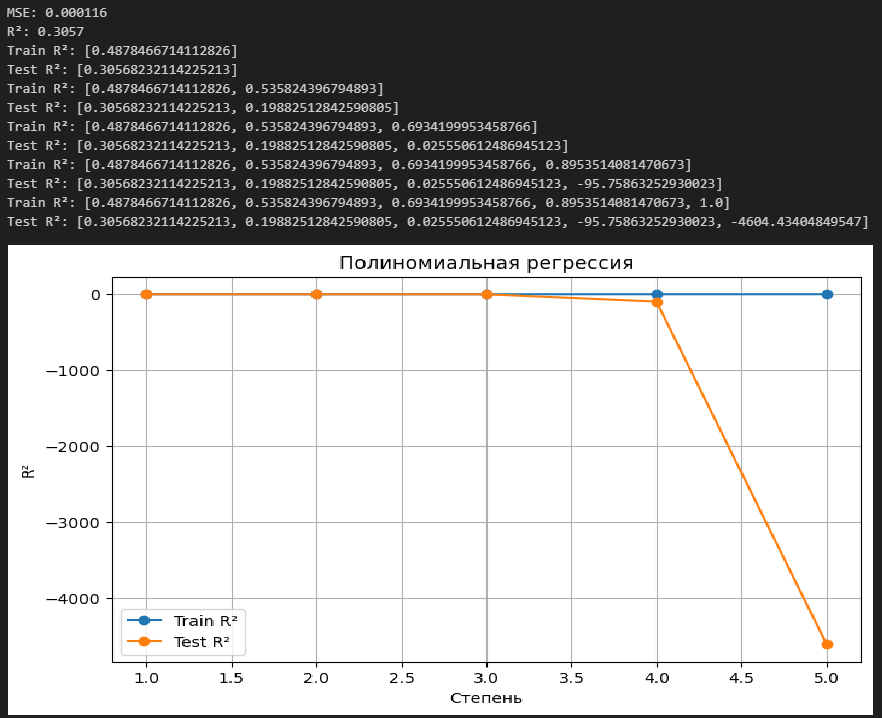


Рисунок 1 - Результаты точности линейной регрессии и график зависимости точности полиномиальной степени обучающей и тестовой выборки.

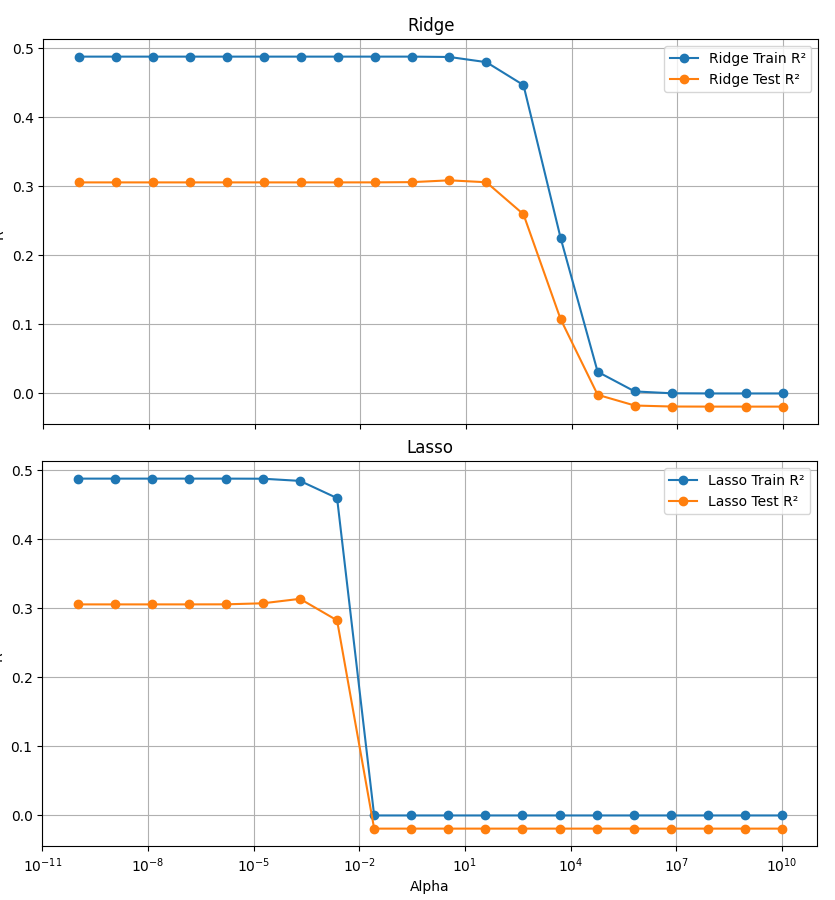


Рисунок 2 - Графики зависимости точности параметров регуляризации тестовой и обучающей выборки.

Вывод: лучшая точность получилась 30,57 %. Это показывает, что 30 % объясняют изменение курса турецкой лиры на бирже из-за других мировых бирж. Т.к. мы работает с экономическими данными, то это значит, что изменение курса турецкой лиры зависит и от других признаков, которые не учли в датасете поэтому получили 30,57%. После использования регуляризации, нам удалось повысить точность до 31,37%. Значит регуляризация немного помогла нам в улучшении обучения нашей модели.

# Приложение:

import pandas as pd

import numpy as np

from sklearn.linear\_model import LinearRegression # type: ignore

from sklearn.metrics import mean\_squared\_error, r2\_score # type: ignore

from sklearn.preprocessing import PolynomialFeatures # type: ignore

from sklearn.pipeline import make\_pipeline # type: ignore

from sklearn.linear\_model import Ridge, Lasso # type: ignore

from sklearn.preprocessing import StandardScaler # type: ignore

import matplotlib.pyplot as plt

df = pd.read\_excel('data\_akbilgic.xlsx', sheet\_name='orjinal data', skiprows=2)

df.columns = ['date', 'ISE\_TL', 'ISE\_USD', 'SP', 'DAX', 'FTSE', 'NIKKEI', 'BOVESPA', 'EU', 'EM']

df = df.dropna()

X = df[['SP', 'DAX', 'FTSE', 'NIKKEI', 'BOVESPA', 'EU', 'EM']].values

y = df['ISE\_TL'].values

split\_ratio = 0.8

split\_index = int(len(X) \* split\_ratio)

X\_train, X\_test = X[:split\_index], X[split\_index:]

y\_train, y\_test = y[:split\_index], y[split\_index:]

model = LinearRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"MSE: {mse:.6f}")

print(f"R²: {r2:.4f}")

train\_scores = []

test\_scores = []

degrees = range(1, 6)

for degree in degrees:

    model = make\_pipeline(PolynomialFeatures(degree), LinearRegression())

    model.fit(X\_train, y\_train)

    train\_pred = model.predict(X\_train)

    test\_pred = model.predict(X\_test)

    train\_scores.append(r2\_score(y\_train, train\_pred))

    test\_scores.append(r2\_score(y\_test, test\_pred))

    print(f"Train R²: {train\_scores}")

    print(f"Test R²: {test\_scores}")

plt.figure(figsize=(8, 5))

plt.plot(degrees, train\_scores, marker='o', label="Train R²")

plt.plot(degrees, test\_scores, marker='o', label="Test R²")

plt.xlabel("Степень")

plt.ylabel("R²")

plt.title("Полиномиальная регрессия")

plt.legend()

plt.grid(True)

plt.show()

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

alphas = np.logspace(-10, 10, 20)

ridge\_train\_scores = []

ridge\_test\_scores = []

lasso\_train\_scores = []

lasso\_test\_scores = []

for alpha in alphas:

    ridge = Ridge(alpha=alpha)

    ridge.fit(X\_train\_scaled, y\_train)

    ridge\_train\_scores.append(ridge.score(X\_train\_scaled, y\_train))

    ridge\_test\_scores.append(ridge.score(X\_test\_scaled, y\_test))

    lasso = Lasso(alpha=alpha, max\_iter=10000)

    lasso.fit(X\_train\_scaled, y\_train)

    lasso\_train\_scores.append(lasso.score(X\_train\_scaled, y\_train))

    lasso\_test\_scores.append(lasso.score(X\_test\_scaled, y\_test))

print(f"Ridge Train R²: {ridge\_train\_scores}")

print(f"Ridge Test R²: {ridge\_test\_scores}")

plt.figure(figsize=(10, 5))

plt.semilogx(alphas, ridge\_train\_scores, marker='o', label="Ridge Train R²")

plt.semilogx(alphas, ridge\_test\_scores, marker='o', label="Ridge Test R²")

plt.xlabel("Alpha")

plt.ylabel("R²")

plt.title("Ridge")

plt.legend()

plt.grid(True)

plt.show()

print(f"Lasso Train R²: {lasso\_train\_scores}")

print(f"Lasso Test R²: {lasso\_test\_scores}")

plt.figure(figsize=(10, 5))

plt.semilogx(alphas, lasso\_train\_scores, marker='o', label="Lasso Train R²")

plt.semilogx(alphas, lasso\_test\_scores, marker='o', label="Lasso Test R²")

plt.xlabel("Alpha")

plt.ylabel("R²")

plt.title("Lasso")

plt.legend()

plt.grid(True)

plt.show()