МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Системы искусственного интеллекта**

**Отчет по выполнению лабораторной работы №3**

Выполнил

студент группы ИВТАПбд-31

Гаязов Т. Р.

Проверил:

преподаватель

Хайруллин И. Д.

Ульяновск

2024

**Цель работы**

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.

1. 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.

**Ход работы**

1. Импорт и подготовка данных:

Импортированы библиотеки NumPy, Matplotlib и модули Scikit-Learn. Данные из файла data.txt были прочитаны, преобразованы в числовой формат и разделены на признаки и целевую переменную.

1. Разделение данных:

Выборка данных разделена на обучающую (80%) и тестовую (20%) части. Обучающая и тестовая выборки сохранены в отдельные файлы.

1. Обучение и оценка линейной модели:

Модель линейной регрессии обучена на обучающей выборке. Предсказанные значения сравнивались с истинными значениями на обучающей и тестовой выборках. Рассчитаны метрики MSE и R-squared.

1. Полиномиальная регрессия:

Построены модели полиномиальной регрессии различных степеней. Для каждой модели проведена кросс-валидация и построены графики зависимости точности от степени полинома.

1. Регуляризация моделей:

Построены модели Ridge-регрессии для различных значений коэффициента регуляризации. Оценена точность моделей на обучающей и тестовой выборках, построены соответствующие графики.

**Тестирование**

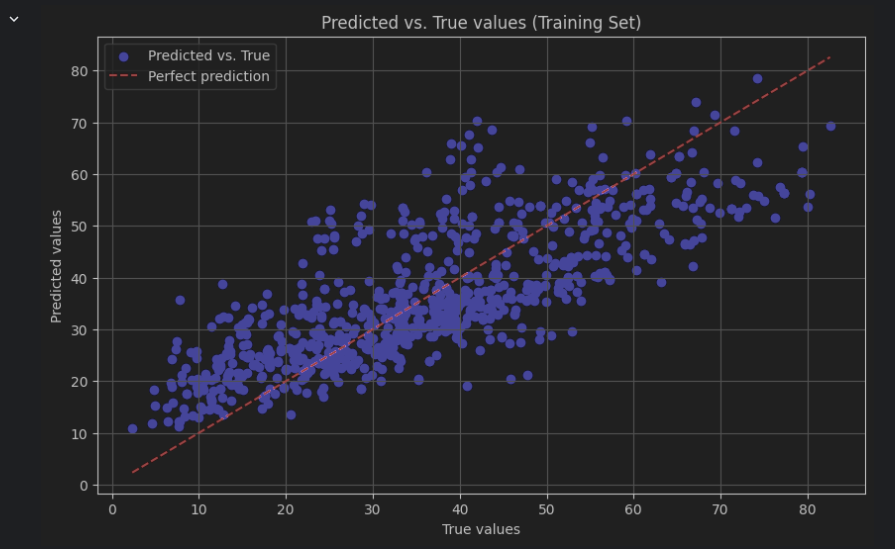


Рис.1. График предсказанных против истинных значений

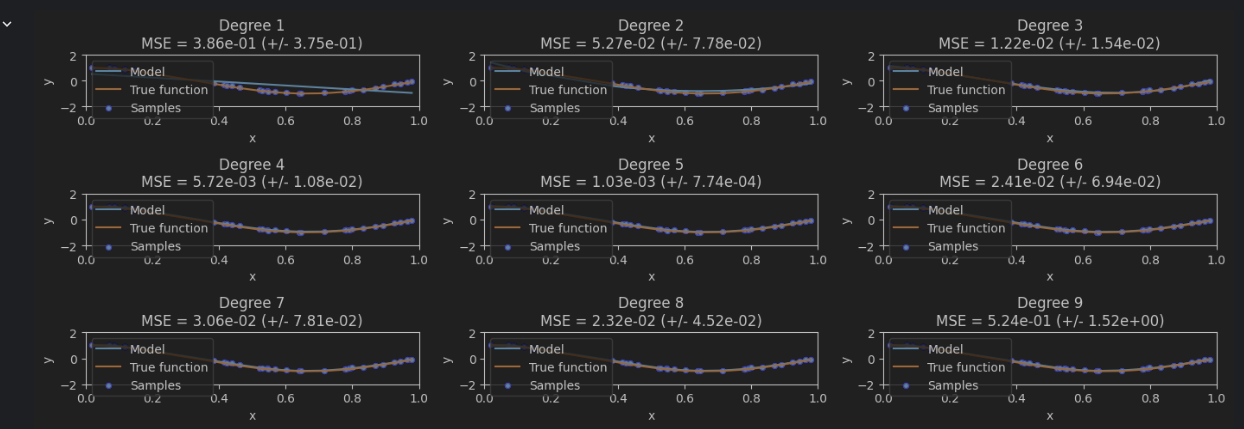


Рис.2. Графики зависимости точности модели от степени полиномиальной функции со степенями 1-9

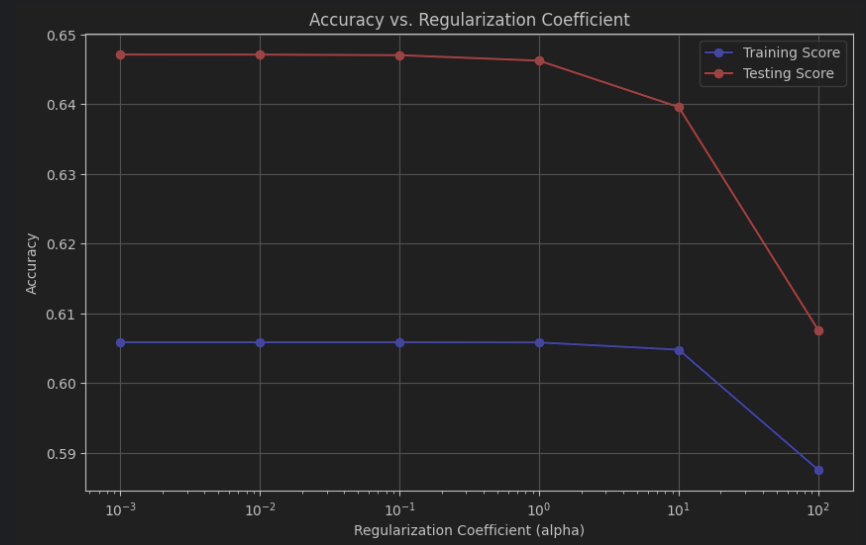


Рис.3. Графики зависимости точности на обучающей и тестовой выборках от коэффициента регуляризации

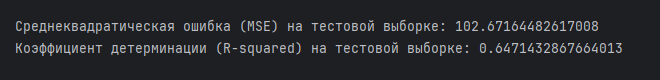


Рис.4. Вывод точностей модели

**Заключение**

В процессе выполнения лабораторной работы проводилось обучение моделей с различными методами на обучающей выборке, а их точность проверялась на тестовой выборке. Была построена модель линейной регрессии, затем применена полиномиальная функция для улучшения модели, и были исследованы графики зависимости точности моделей от степени полиномиальной функции. Также была использована регуляризация для построения модели с оптимальными параметрами, которые были подобраны после экспериментов. Затем были построены графики зависимости точности моделей на обучающей и тестовой выборках от коэффициента регуляризации.**Список литературы**

1. <https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py>
2. <https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html#sphx-glr-auto-examples-model-selection-plot-underfitting-overfitting-py>
3. <https://scikit-learn.org/stable/modules/linear_model.html>

**Приложение**

|  |
| --- |
| import pandas as pd import numpy as np from sklearn.linear\_model import LinearRegression from sklearn.metrics import mean\_squared\_error, r2\_score from sklearn.preprocessing import PolynomialFeatures from sklearn.pipeline import Pipeline from sklearn.model\_selection import cross\_val\_score import matplotlib.pyplot as plt from sklearn.linear\_model import Ridge from sklearn.preprocessing import StandardScaler  # Чтение данных из файла data = pd.read\_excel("Concrete\_Data.xls")  # Размер обучающей выборки (например, 80% данных) train\_size = int(len(data) \* 0.8)  # Перемешиваем индексы indices = np.random.permutation(data.shape[0])  # Разделение на обучающую и тестовую выборку train\_indices, test\_indices = indices[:train\_size], indices[train\_size:]  # Создание обучающей и тестовой выборки train\_data, test\_data = data.iloc[train\_indices], data.iloc[test\_indices]  # Сохранение обучающей и тестовой выборки в файлы CSV train\_data.to\_csv("train\_data.csv", index=False) test\_data.to\_csv("test\_data.csv", index=False) print(f'{train\_data.columns}')  X\_train = train\_data.drop(columns=['Concrete compressive strength(MPa, megapascals) ']) y\_train = train\_data['Concrete compressive strength(MPa, megapascals) '] model = LinearRegression() model.fit(X\_train, y\_train)  y\_train\_pred = model.predict(X\_train)  # Построение графика plt.figure(figsize=(10, 6)) plt.scatter(y\_train, y\_train\_pred, color='blue', label='Predicted vs. True') plt.plot([min(y\_train), max(y\_train)], [min(y\_train), max(y\_train)], color='red', linestyle='--', label='Perfect prediction') plt.title('Predicted vs. True values (Training Set)') plt.xlabel('True values') plt.ylabel('Predicted values') plt.legend() plt.grid(True) plt.show()  # Получение матрицы признаков и вектора целевой переменной для тестовой выборки X\_test = test\_data.drop(columns=['Concrete compressive strength(MPa, megapascals) ']) y\_test = test\_data['Concrete compressive strength(MPa, megapascals) ']  # Предсказание значений целевой переменной на тестовой выборке y\_test\_pred = model.predict(X\_test)  # Оценка точности модели на тестовой выборке mse\_test = mean\_squared\_error(y\_test, y\_test\_pred) r2\_test = r2\_score(y\_test, y\_test\_pred)  print(f"Среднеквадратическая ошибка (MSE) на тестовой выборке: {mse\_test}") print(f"Коэффициент детерминации (R-squared) на тестовой выборке: {r2\_test}")  # Функция, представляющая истинную зависимость def true\_fun(X):  return np.cos(1.5 \* np.pi \* X)  # Генерация случайных данных np.random.seed(0) n\_samples = 30 degrees = range(1, 10)  X = np.sort(np.random.rand(n\_samples)) y = true\_fun(X) + np.random.rand(n\_samples) \* 0.1  plt.figure(figsize=(14, 5))  # Проход по разным степеням полинома for i, degree in enumerate(degrees, 1):  # Создание полиномиальных признаков и линейной регрессии  polynomial\_features = PolynomialFeatures(degree=degree, include\_bias=False)  linear\_regression = LinearRegression()  pipeline = Pipeline([  ("polynomial\_features", polynomial\_features),  ("linear\_regression", linear\_regression),  ])   # Обучение модели и оценка с использованием кросс-валидации  scores = cross\_val\_score(pipeline, X[:, np.newaxis], y, scoring="neg\_mean\_squared\_error", cv=10)    # Оценка на обучающем наборе данных  pipeline.fit(X[:, np.newaxis], y)  y\_pred = pipeline.predict(X[:, np.newaxis])    # Построение графика  plt.subplot(3, 3, i)  plt.plot(X, y\_pred, label="Model")  plt.plot(X, true\_fun(X), label="True function")  plt.scatter(X, y, edgecolor='b', s=20, label="Samples")  plt.xlabel("x")  plt.ylabel("y")  plt.xlim((0, 1))  plt.ylim((-2, 2))  plt.legend(loc="best")  plt.title(f"Degree {degree}\nMSE = {-scores.mean():.2e} (+/- {scores.std():.2e})")  # Подготовка данных scaler = StandardScaler() X\_train\_scaled = scaler.fit\_transform(X\_train) X\_test\_scaled = scaler.transform(X\_test)  # Создание списка значений параметра регуляризации alphas = [0.001, 0.01, 0.1, 1, 10, 100]  # Списки для сохранения точности моделей на обучающей и тестовой выборках train\_scores = [] test\_scores = []  # Обучение моделей с разными значениями параметра регуляризации for alpha in alphas:  # Создание и обучение модели Ridge  model = Ridge(alpha=alpha)  model.fit(X\_train\_scaled, y\_train)    # Оценка точности на обучающей выборке  train\_score = model.score(X\_train\_scaled, y\_train)  train\_scores.append(train\_score)    # Оценка точности на тестовой выборке  test\_score = model.score(X\_test\_scaled, y\_test)  test\_scores.append(test\_score)  # Построение графиков зависимости точности от коэффициента регуляризации plt.figure(figsize=(10, 6)) plt.plot(alphas, train\_scores, label='Training Score', marker='o', color='blue') plt.plot(alphas, test\_scores, label='Testing Score', marker='o', color='red') plt.xlabel('Regularization Coefficient (alpha)') plt.ylabel('Accuracy') plt.title('Accuracy vs. Regularization Coefficient') plt.xscale('log') plt.legend() plt.grid(True) plt.show() |