МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №3
"Регрессионная модель изменения цен на дома в Бостоне"
по дисциплине «Искусственные нейронные сети»

Студент гр. 8382 _____ Облизанов А.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель.

Реализовать предсказание медианной цены на дома в пригороде Бостона в середине 1970-х по таким данным, как уровень преступности, ставка местного имущественного налога и т. д.

Данный набор содержит относительно немного образцов данных: всего 506, разбитых на 404 обучающих и 102 контрольных образца. И каждый признак во входных данных (например, уровень преступности) имеет свой масштаб. Например, некоторые признаки являются пропорциями и имеют значения между 0 и 1, другие — между 1 и 12 и т. д.

Задание.

- Ознакомиться с задачей регрессии
- Изучить отличие задачи регрессии от задачи классификации
- Создать модель
- Настроить параметры обучения
- Обучить и оценить модели
- Ознакомиться с перекрестной проверкой

Требуется:

- Объяснить различия задач классификации и регрессии
- Изучить влияние кол-ва эпох на результат обучения модели
- Выявить точку переобучения
- Применить перекрестную проверку по K блокам при различных K
- Построить графики ошибки и точности во время обучения для моделей, а также усредненные графики по всем моделям

Выполнение работы.

Работа выполнялась на базе операционной системы Windows 10 в среде разработки PyCharm.

1. Задача классификации, отличие от регрессии

Чтобы поставить задачу классификации, необходимо определить входные данные. Имеется множество объектов, которые разделены на классы. Для конечного множества объектов известно, к каким классам они относятся – это обучающая выборка. Для других объектов классовая принадлежность неизвестна. Задача классификации – построить алгоритм, который сможет классифицировать произвольный объект, то есть указать класс, к которому он относится.

Регрессия – это задача, в которой по заданному набору признаков объекта необходимо спрогнозировать некоторую целевую переменную. Тут важно отметить, что прогнозируется не дискретная переменная (как например номер класса в задаче классификации), множество возможный значений непрерывно. Например, предсказание скорости, температуры, координат.

Таким образом ключевое отличие регрессии и классификации как задач состоит в том, что при классификации необходимо отнести объект к одному из конечно заданного числа классов, а при регрессии необходимо спрогнозировать некоторое не дискретное значение (параметр).

2. Построение модели, подготовка данных

Так как входные данные имеют различные диапазоны, была выполнена нормализация значений. Для каждого признака во входных данных из каждого значения вычитается среднее по этому признаку, а далее результат делится на стандартное отклонение. Таким образом получаем данные со средним значением 0 и стандартным отклонением 1. Листинг приведен ниже:

```
(train_data, train_targets), (test_data, test_targets) = boston_housing.load_data()
mean = train_data.mean(axis=0)
std = train_data.std(axis=0)
train_data -= mean
train_data /= std
test_data -= mean
test_data /= std
```

Построение и компиляция модели происходят в функции build_model(). Листинг приведен ниже:

```
def build_model():
    model = Sequential()
    model.add(Dense(64, activation='relu', input_shape=(train_data.shape[1],)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model
```

Также реализована перекрестная проверка по K блокам. Реализацию можно увидеть в коде программы в Приложении А.

3. Тестирование модели

Были выбраны следующие параметры обучения и перекрестной проверки:

- Число эпох: 100
- Число блоков: 4

Графики средних абсолютных ошибок и средних квадратичных ошибок (функция потерь) при обучении и проверке для каждой модели приведены на рис. 1-8.

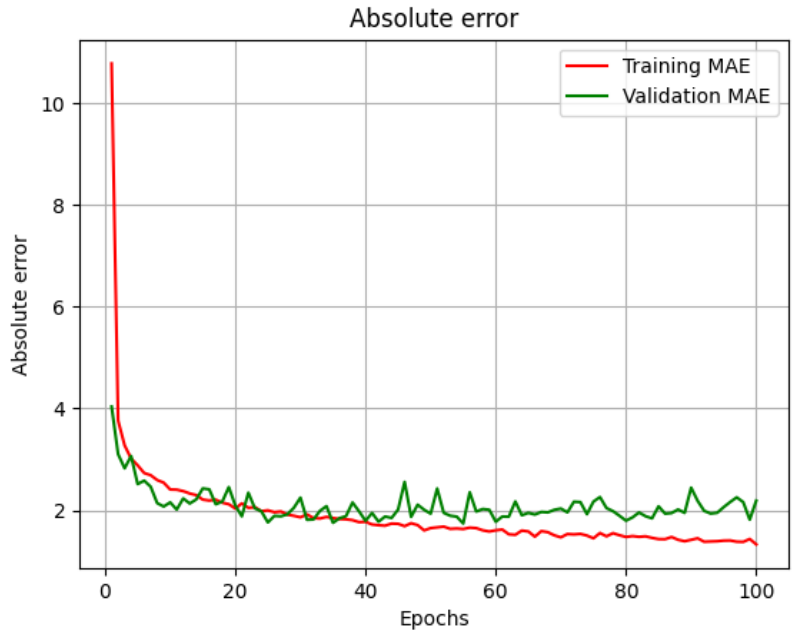

Рисунок 1 – График средней абсолютной ошибки, 1 модель

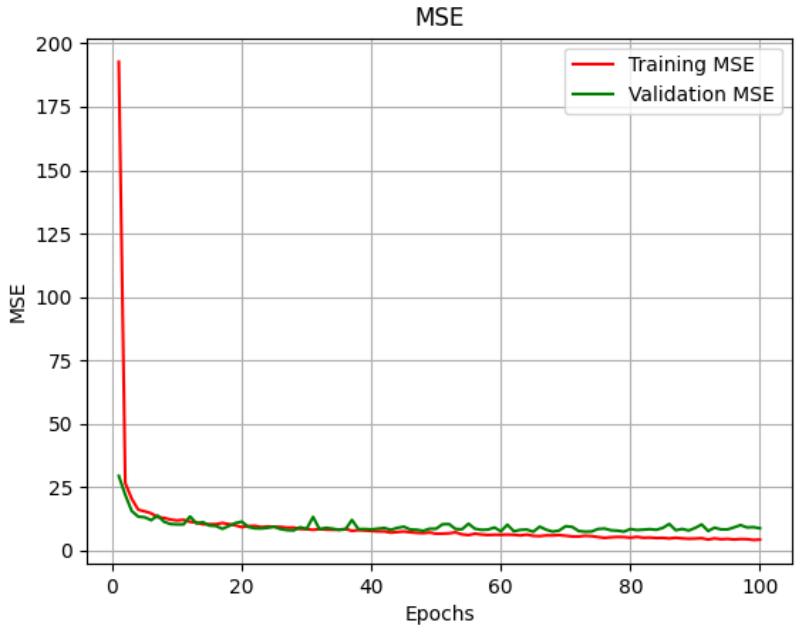

Рисунок 2 – График средней квадратичной ошибки для 1 модели

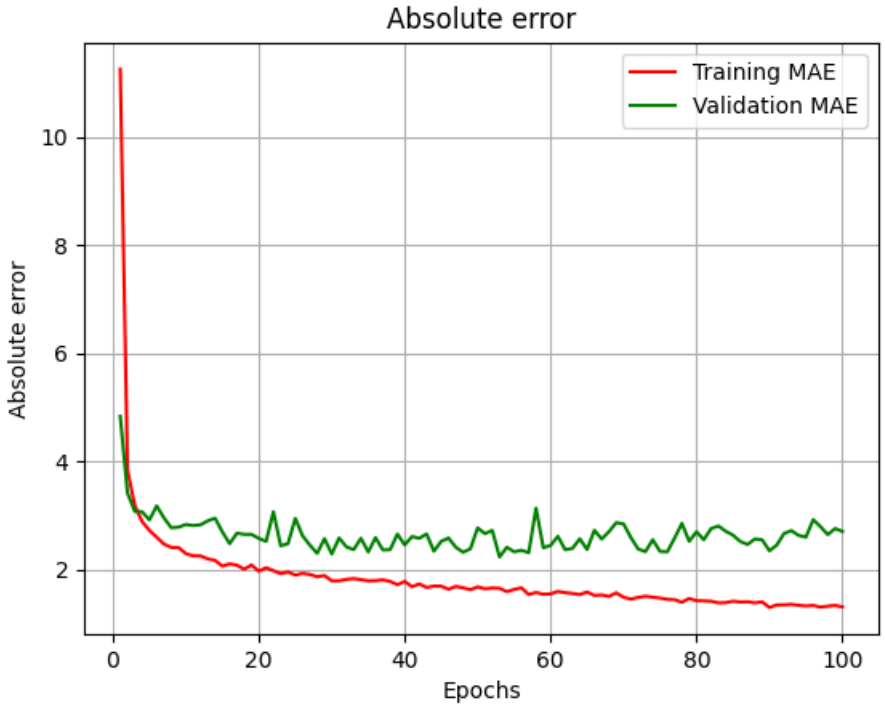

Рисунок 3 – График средней абсолютной ошибки для 2 модели

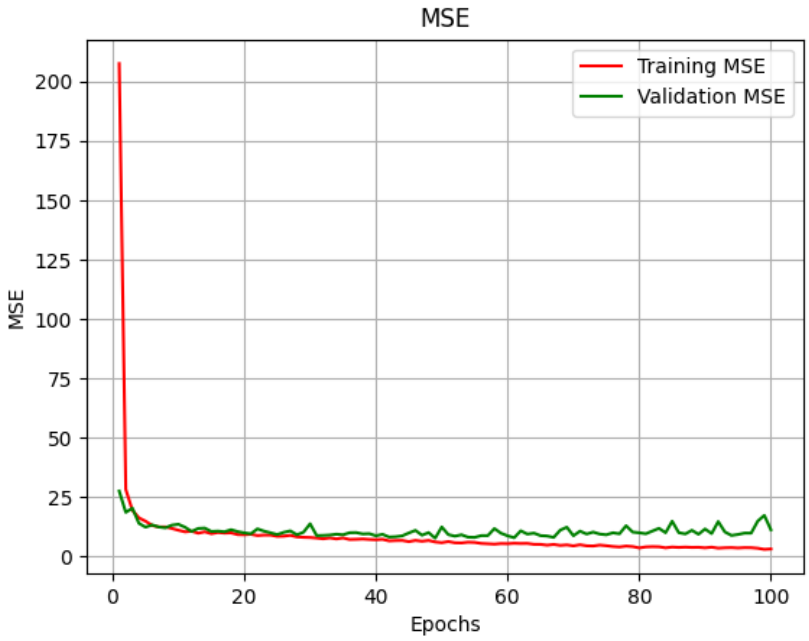

Рисунок 4 – График средней квадратичной ошибки для 2 модели

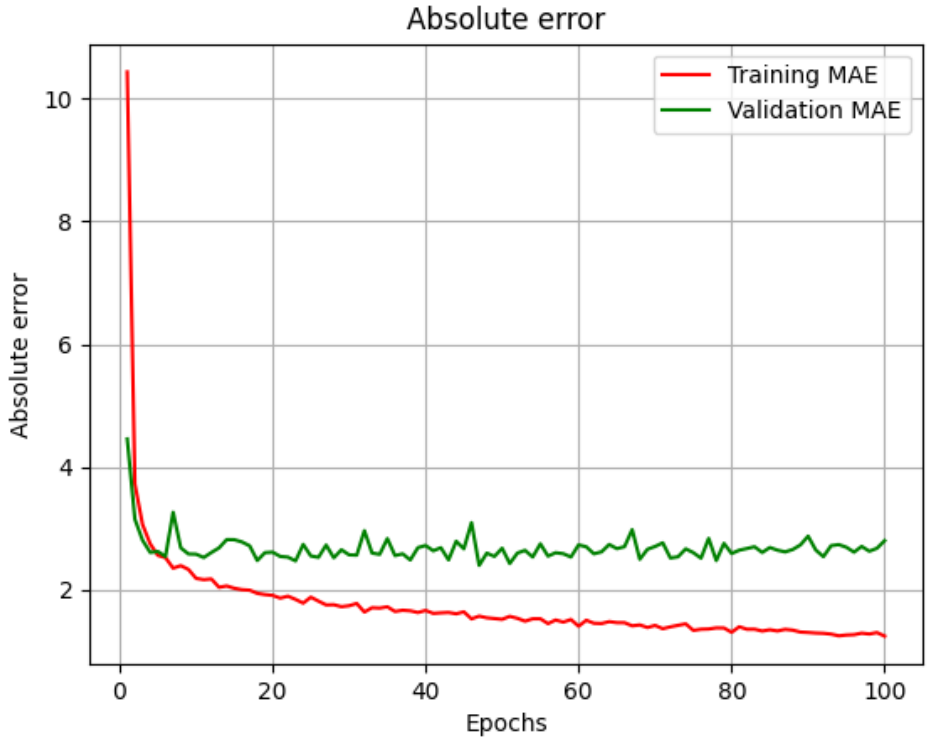

Рисунок 5 – График средней абсолютной ошибки для 3 модели

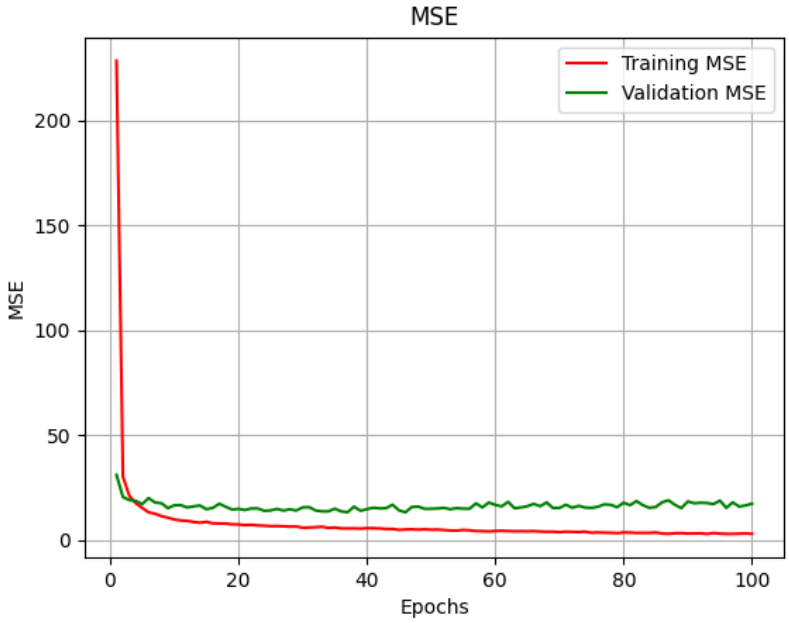

Рисунок 6 – График средней квадратичной ошибки для 3 модели

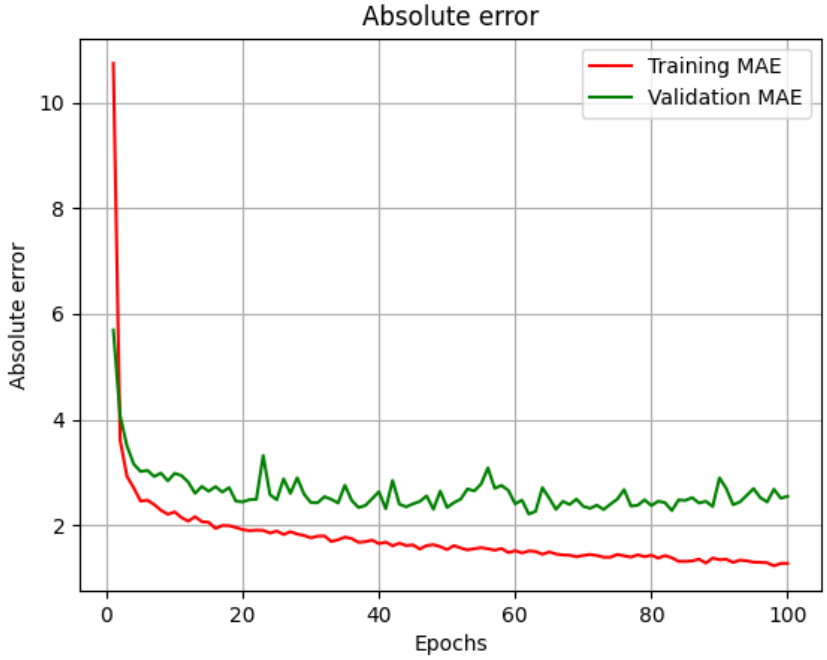

Рисунок 7 – График средней абсолютной ошибки для 4 модели

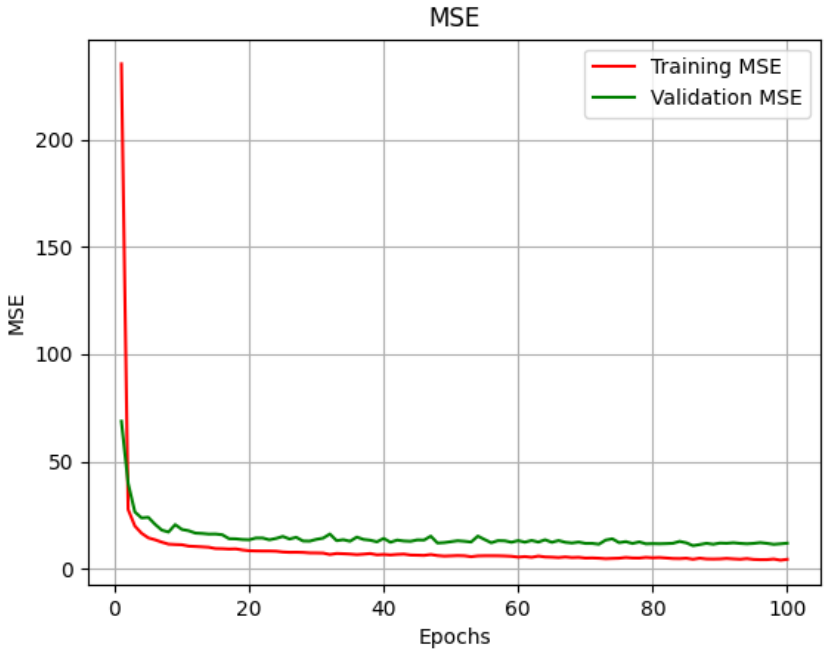

Рисунок 8 – График средней квадратичной ошибки для 4 модели

График среднего значения по четырем моделям оценки средней абсолютной ошибки приведен на рис. 9.

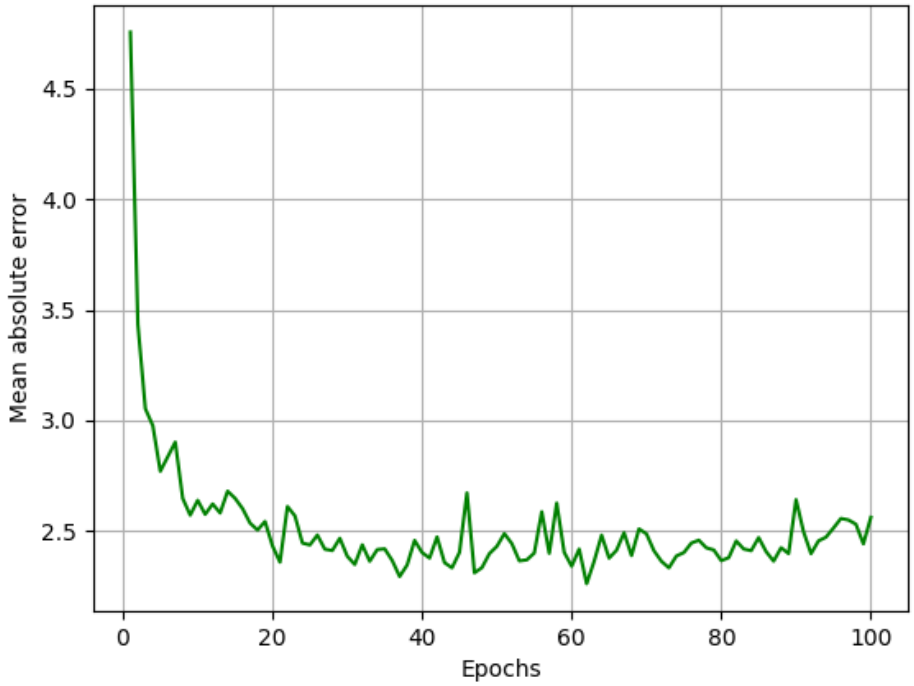

Рисунок 9 – График среднего значения оценки средней абсолютной ошибки

Из графика среднего значения оценки средней абсолютной ошибки (рис. 5) можно заметить, что значение ошибки перестает падать около 40-й эпохи, а примерно с 70-й эпохи начинает расти. При этом на рис. 1-4 видно, что на протяжении всех эпох ошибки на данных для обучения падают. Из этого можно сделать вывод, что примерно после 40-й эпохи модель склонна к переобучению: улучшение показателей на данных для обучения не приводит к лучшим результатам при проверке, а напротив ухудшает их.

Число эпох было уменьшено до 40. Число блоков для перекрестной проверки – 4. Графики оценок средних абсолютных ошибок при обучении и проверке для каждой модели приведены на рис. 10-13.

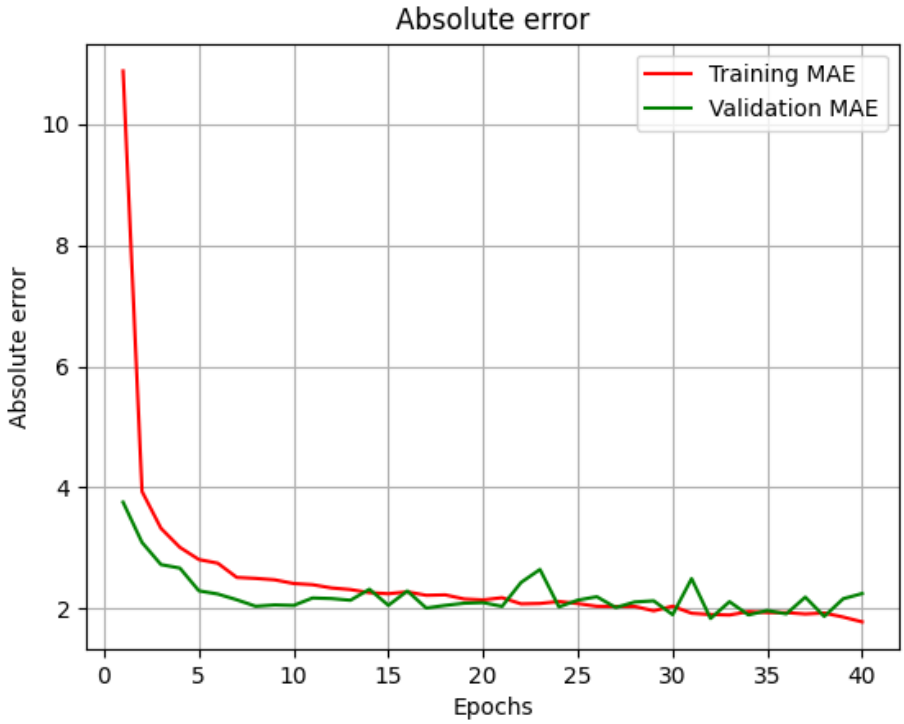

Рисунок 10 – График средней абсолютной ошибки для 1 модели

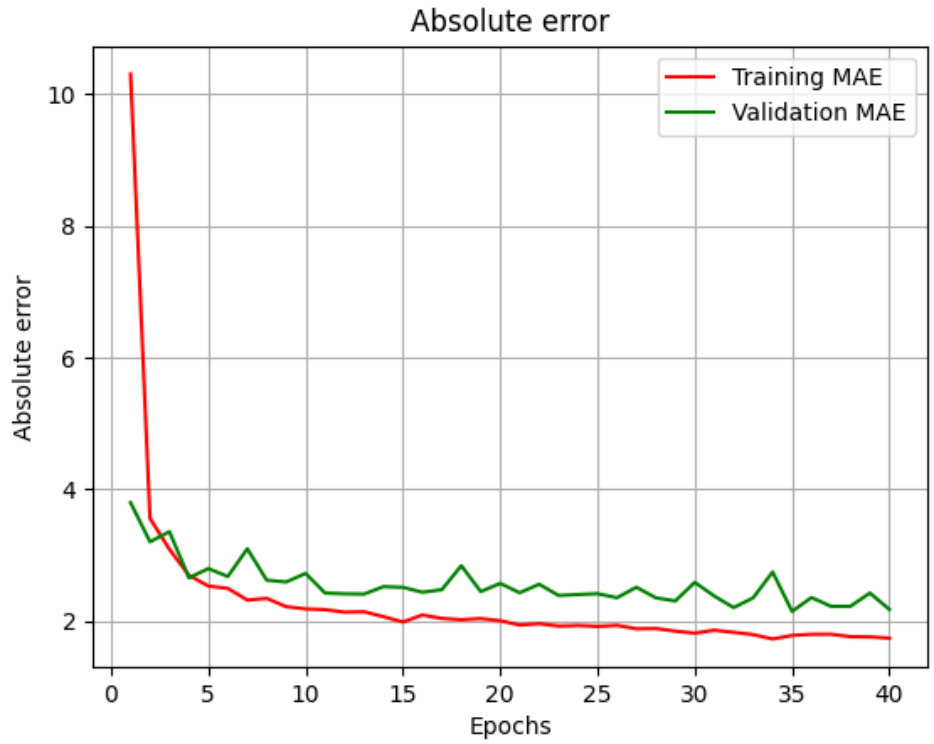

Рисунок 11 – График средней абсолютной ошибки для 2 модели

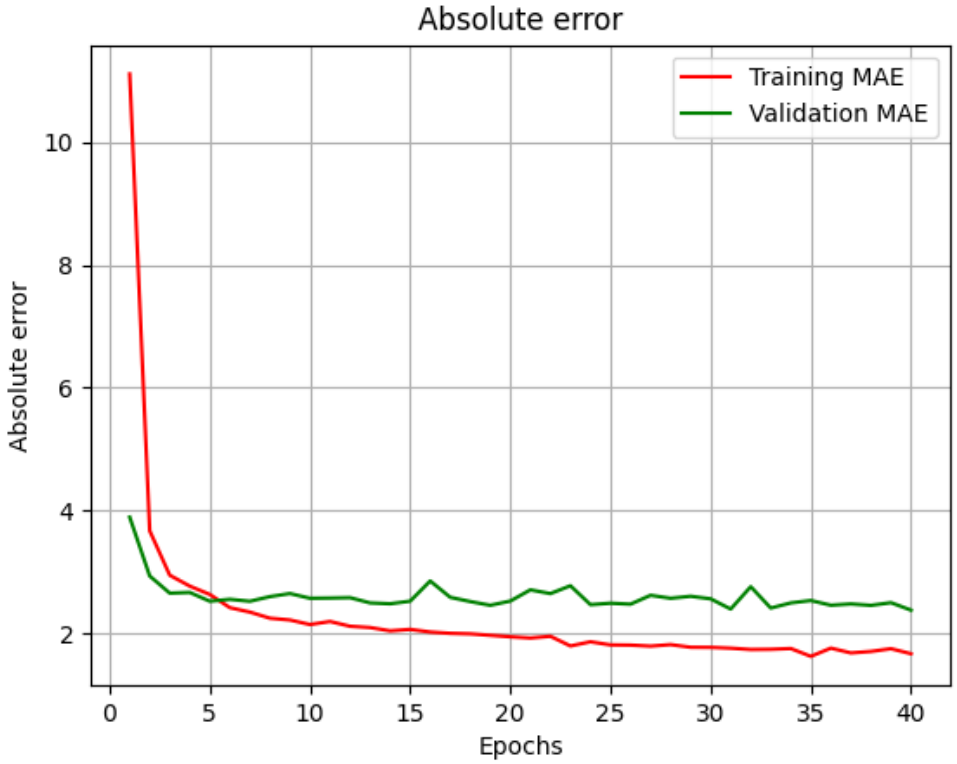

Рисунок 12 – График средней абсолютной ошибки для 3 модели

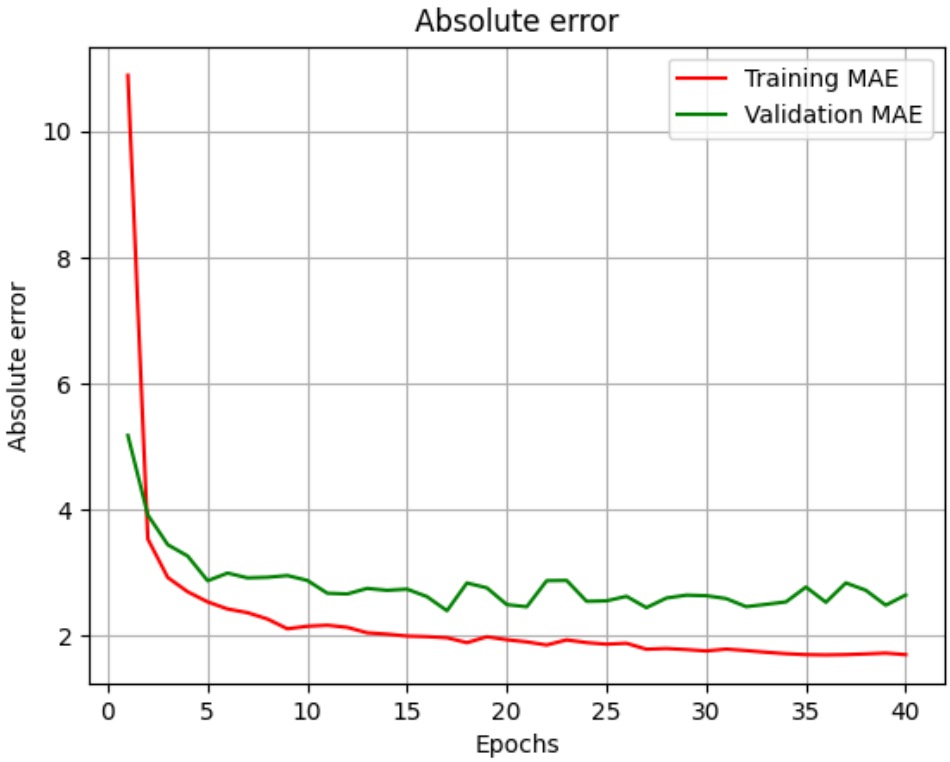

Рисунок 13 – График средней абсолютной ошибки для 4 модели

График среднего значения по четырем блокам оценки средней абсолютной ошибки приведен на рис. 14.

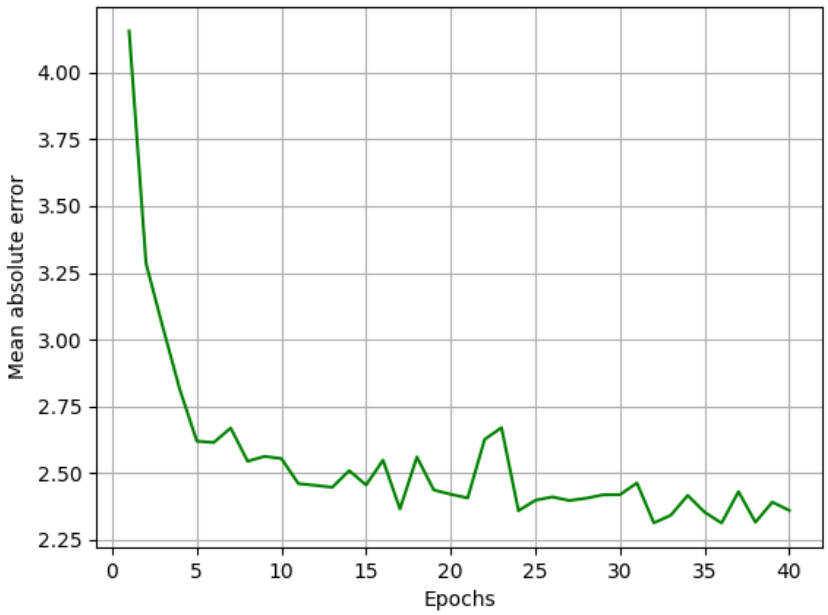

Рисунок 14 – График среднего значения оценки средней абсолютной ошибки

Значение среднего значения оценки абсолютной ошибки после обучения за 40 эпох уменьшилось, если при обучении за 100 эпох оно было > 2.5, то теперь < 2.4. Также на графиках 6-10 не видно признаков переобучения – значение средних абсолютных ошибок уменьшается на протяжении всего обучения во всех 4-х блоках. Можно сказать, что уменьшение числа эпох улучшило показатели модели.

График среднего значения оценки средней абсолютной ошибки при обучении за 40 эпох и числом блоков для перекрестной проверки $k = 5$ представлен на рис. 15.

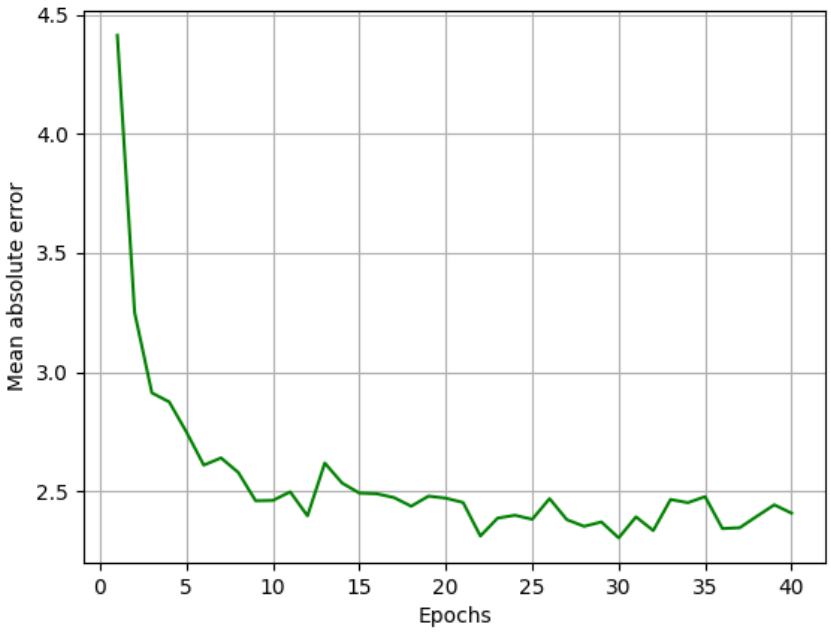

Рисунок 15 – Перекрестная проверка, 5 блоков

График среднего значения оценки средней абсолютной ошибки при обучении за 40 эпох и числом блоков для перекрестной проверки $k = 6$ представлен на рис. 16.

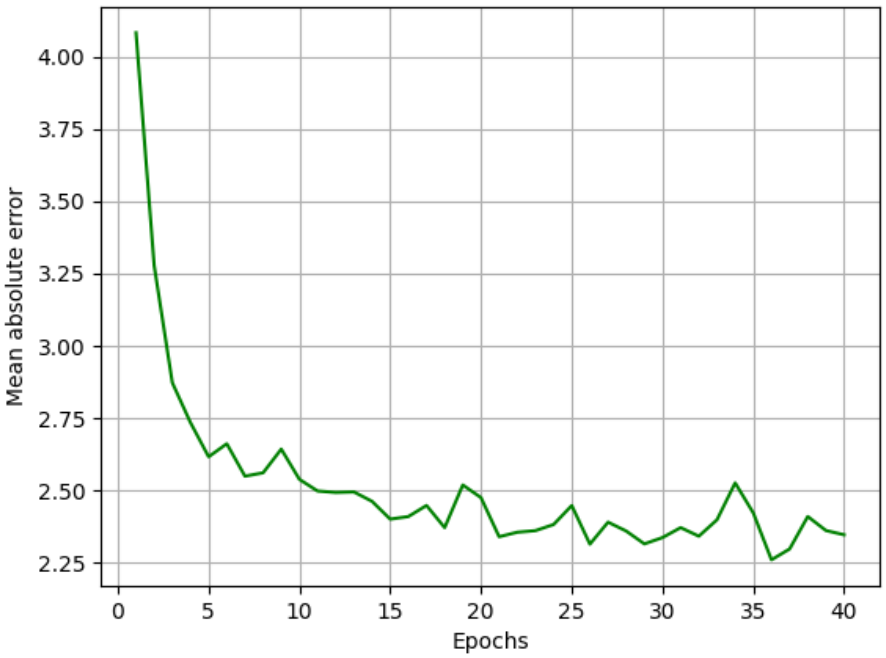

Рисунок 16 – Перекрестная проверка, 6 блоков

График среднего значения оценки средней абсолютной ошибки при обучении за 40 эпох и числом блоков для перекрестной проверки $k = 3$ представлен на рис. 17.

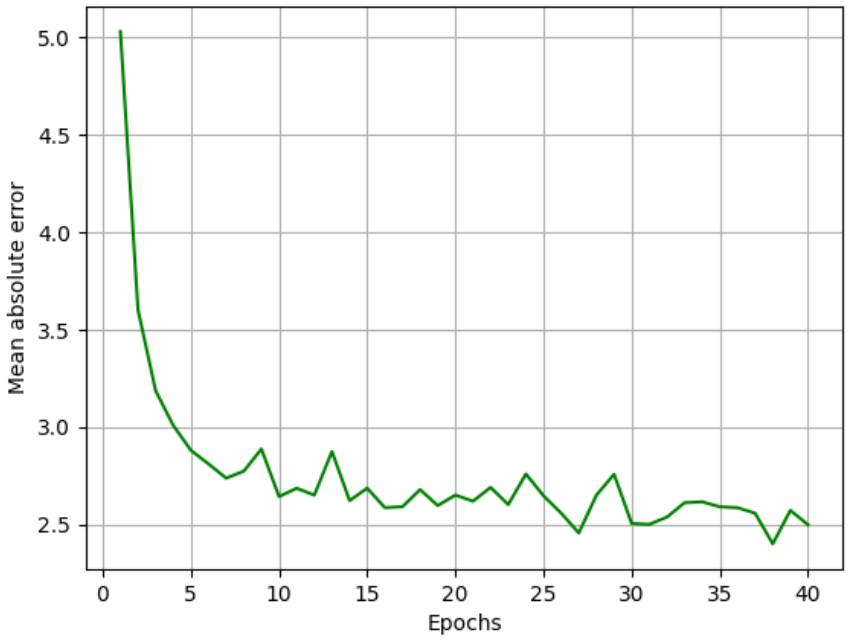

Рисунок 17 – Перекрестная проверка, 3 блока

Из рис. 10-13 видно, что наименьшее значение средней абсолютной ошибки (в среднем по всем блокам) модель показала при 6 блоках перекрестной проверки, наибольшее – при 3 блоках. Это можно объяснить тем, что при разделении на 3 блока данных для обучения не хватает.

Итак, наилучшие результаты модель показывает при использовании 6 блоков для перекрестной проверки.

Выводы.

В ходе выполнения лабораторной работы была изучена реализация регрессии в машинном обучении для решения задачи предсказания медианной цены на дома по различным показателям. Было изучено влияние числа эпох и количество блоков для перекрестной проверки на результат обучения искусственной нейронной сети.

ПРИЛОЖЕНИЕ А

Исходный код программы. Файл lr3.py

```
import numpy as np
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

import matplotlib.pyplot as plt
from tensorflow.keras.datasets import boston_housing

(train_data, train_targets), (test_data, test_targets) =
boston_housing.load_data()

mean = train_data.mean(axis=0)
std = train_data.std(axis=0)

train_data -= mean
train_data /= std

test_data -= mean
test_data /= std


def build_model():
    model = Sequential()
    model.add(Dense(64, activation='relu', input_shape=(train_data.shape[1],)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model


k = 3
num_val_samples = len(train_data) // k
num_epochs = 40
all_scores = []

for i in range(k):
    print('processing fold #', i)
    val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = train_targets[i * num_val_samples: (i + 1) * num_val_samples]

    partial_train_data = np.concatenate([train_data[:i * num_val_samples],
                                         train_data[(i + 1) *
num_val_samples:]], axis=0)
    partial_train_target = np.concatenate([train_targets[: i * num_val_samples],
```

```
                                            train_targets[(i      +      1)      *
num_val_samples:]], axis=0)

    model = build_model()
    history      =      model.fit(partial_train_data,      partial_train_target,
epochs=num_epochs, batch_size=1,
                        validation_data=(val_data, val_targets), verbose=2)
    mae = history.history['mae']
    val_mae = history.history['val_mae']
    x = range(1, num_epochs + 1)
    all_scores.append(val_mae)
    plt.figure(i)
    plt.plot(x, mae, 'r', label='Training MAE')
    plt.plot(x, val_mae, 'g', label='Validation MAE')
    plt.xlabel('Epochs')
    plt.ylabel('Absolute error')
    plt.title('Absolute error')
    plt.legend()
    plt.grid()

avg_mae = [np.mean([x[i] for x in all_scores]) for i in range(num_epochs)]

plt.figure(k)
plt.plot(range(1, num_epochs + 1), avg_mae, 'g')
plt.xlabel('Epochs')
plt.ylabel("Mean absolute error")
plt.grid()
figs = [plt.figure(n) for n in plt.get_fignums()]
for i in range(len(figs)):
    figs[i].savefig("./f%d.png" % i, format='png')

print(np.mean(all_scores))
```