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

ОТЧЕТ

по лабораторной работе № 3

по дисциплине «Искусственные нейронные сети»

ТЕМА: Регрессионная модель изменения цен на дома в Бостоне

Студент гр. 8383 _____ Мололкин К.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

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.

Задачи

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

Выполнение работы

Регрессия — это процесс поиска модели или функции для разделения данных на непрерывные реальные значения вместо использования классов. Математически, с проблемой регрессии, каждый пытается найти приближение функции с минимальным отклонением ошибки. В регрессии предсказывается, что числовая зависимость данных будет отличать ее.

Классификация — это процесс поиска или обнаружения модели (функции), которая помогает разделить данные на несколько категориальных классов. При классификации определяется членство группы в проблеме, что означает, что данные классифицируются под разными метками в соответствии с некоторыми параметрами, а затем метки прогнозируются для данных.

Данные подходы различаются тем, что процесс классификации моделирует функцию, с помощью которой данные прогнозируются в метках дискретных классов, а регрессия — это процесс создания модели, которая предсказывает непрерывное количество.

Была создана модель, состоящая из трех слоев первый и второй слой, каждый из которых имеют по 64 нейрона с функцией активации relu. Данная модель была запущена со 100 эпохами, данные были разделены на 4 блока. На рис. 1-8 представлены графики средней абсолютной ошибки и среднего квадратичного отклонения для каждого блока.

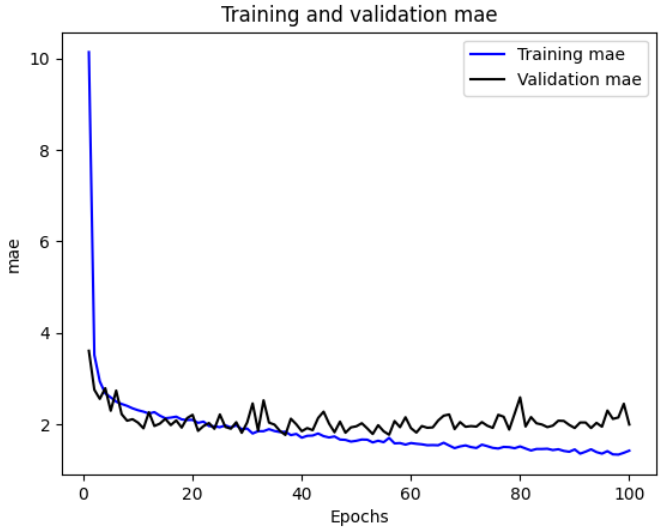

Рисунок 1 – график средней абсолютной ошибки для первого блока

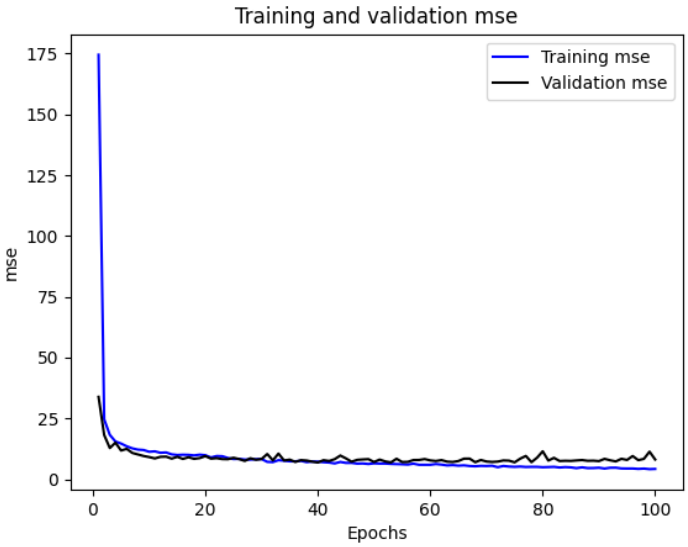

Рисунок 2 – график среднего квадратичного отклонения для первого блока

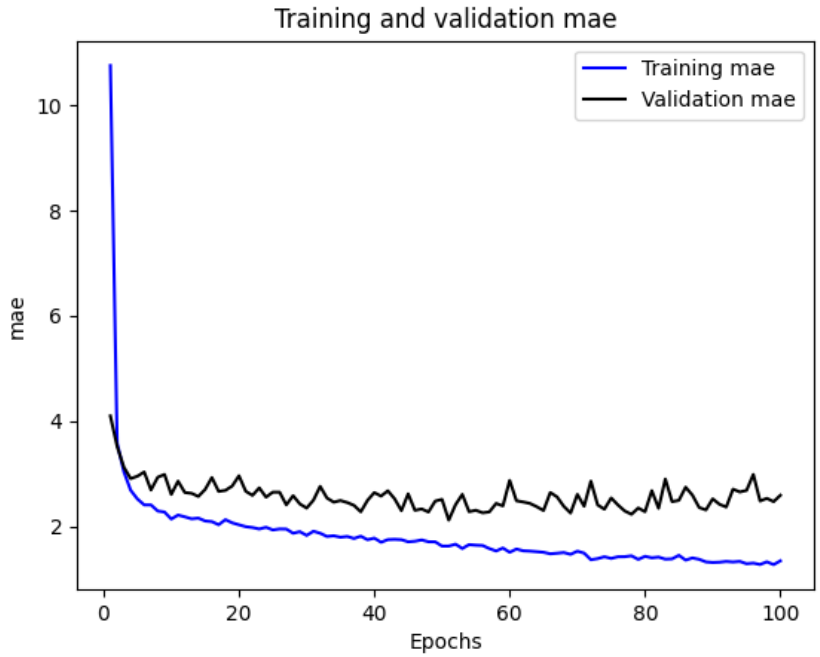

Рисунок 3 – график средней абсолютной ошибки для второго блока

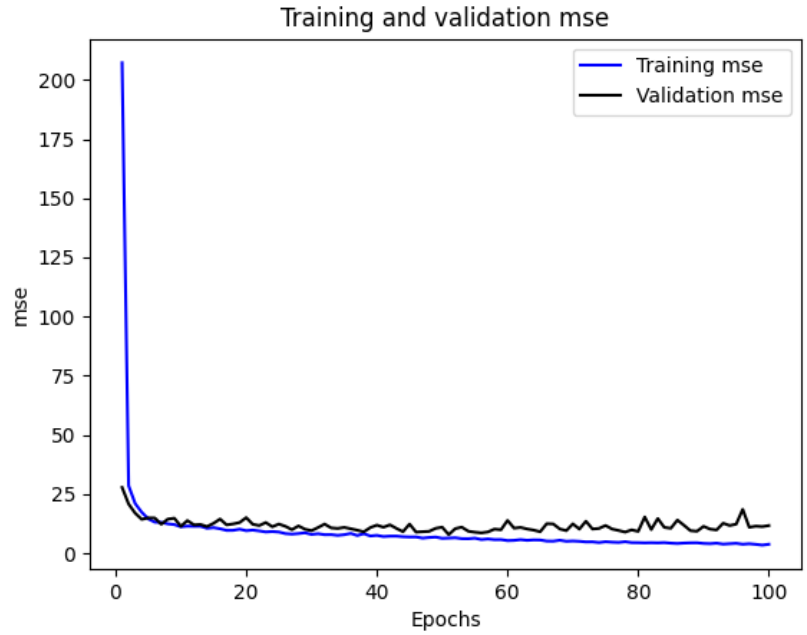

Рисунок 4 – график среднего квадратичного отклонения для второго блока

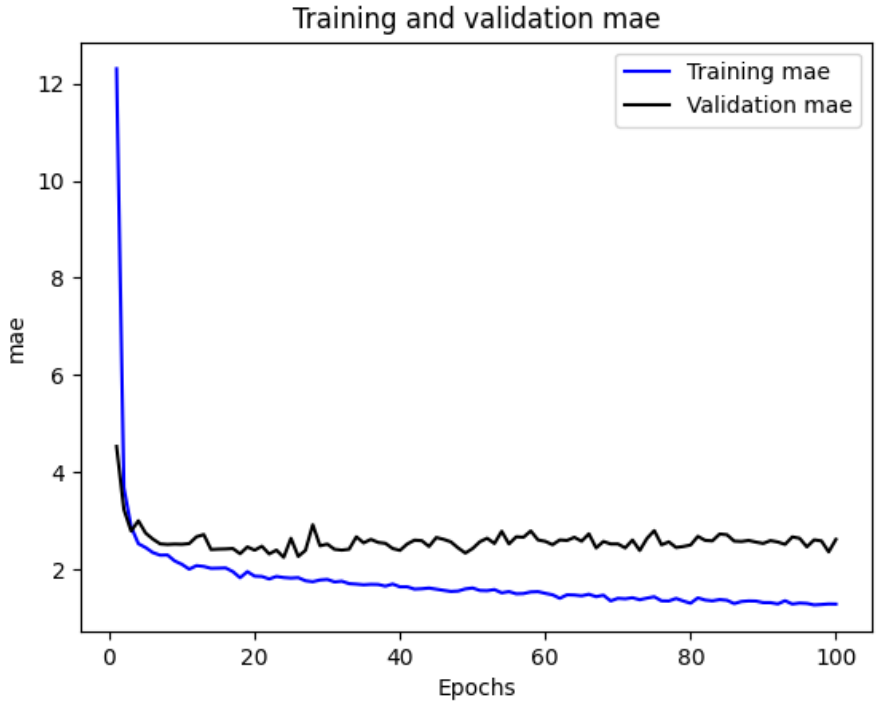

Рисунок 5 – график средней абсолютной ошибки для третьего блока

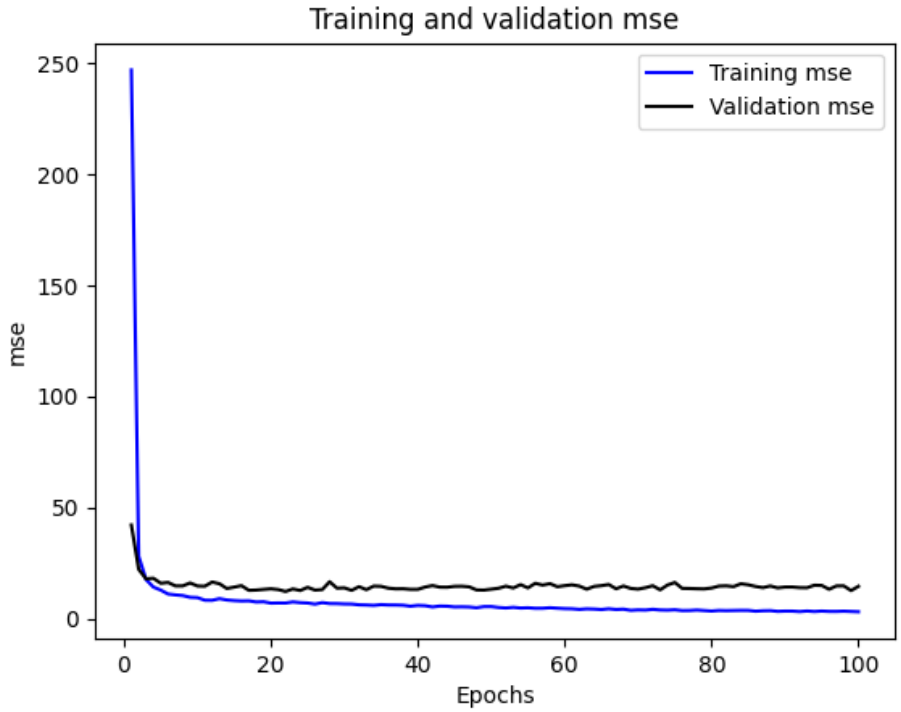

Рисунок 6 – график среднего квадратичного отклонения для третьего блока

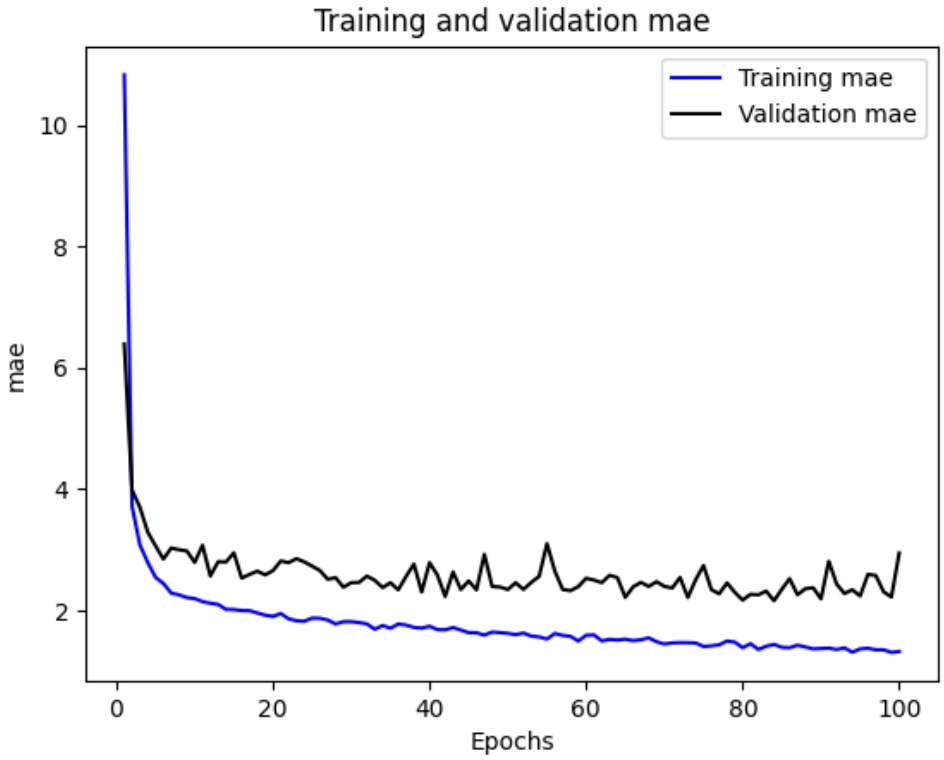

Рисунок 7 – график средней абсолютной ошибки для четвертого блока

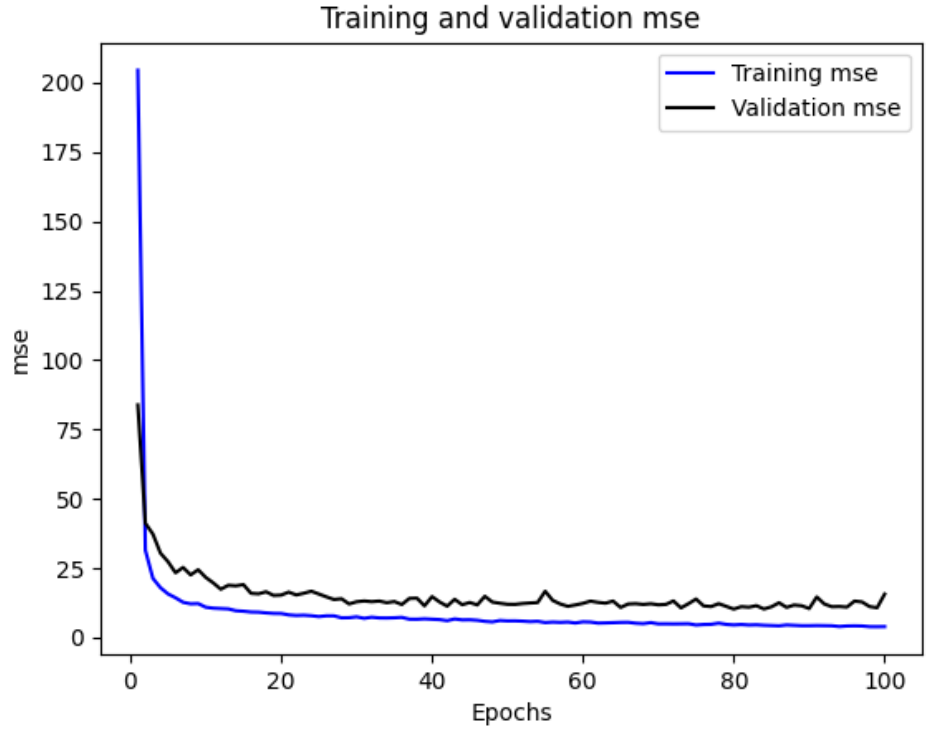

Рисунок 8 – график среднего квадратичного отклонения для четвертого блока

На рис. 9 представлен график средней абсолютной ошибки по всем блокам. Оценка модели равна 2.4542797207832336.

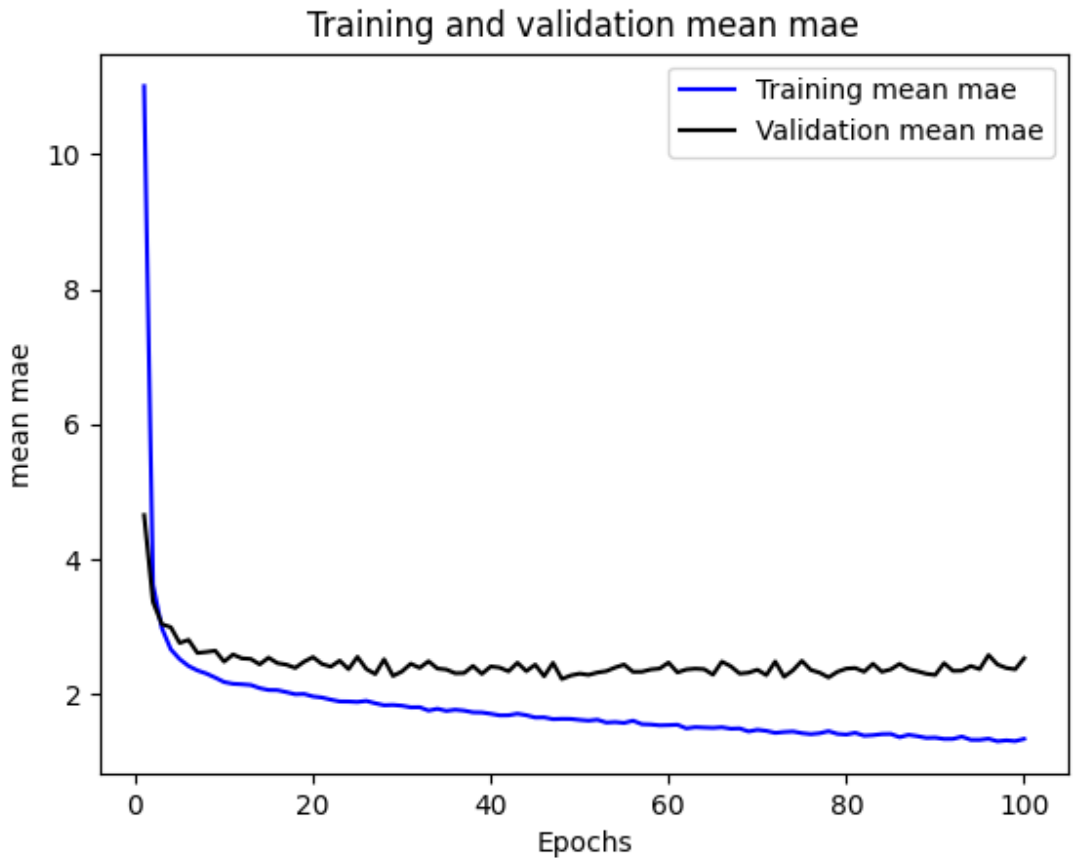

Рисунок 10 - график средней абсолютной ошибки по всем блокам

По данному графику можно проследить, что абсолютна ошибки на контрольных данных после 40 эпохи начинает расти, таким образом можно сделать вывод, что модель начинает переобучение.

Таким образом запустим модель с 40 эпохами и так же с 4 блоками. На рис. 11 представлен график средней абсолютной ошибки по всем блокам. Оценка модели равна 2.288475751876831.

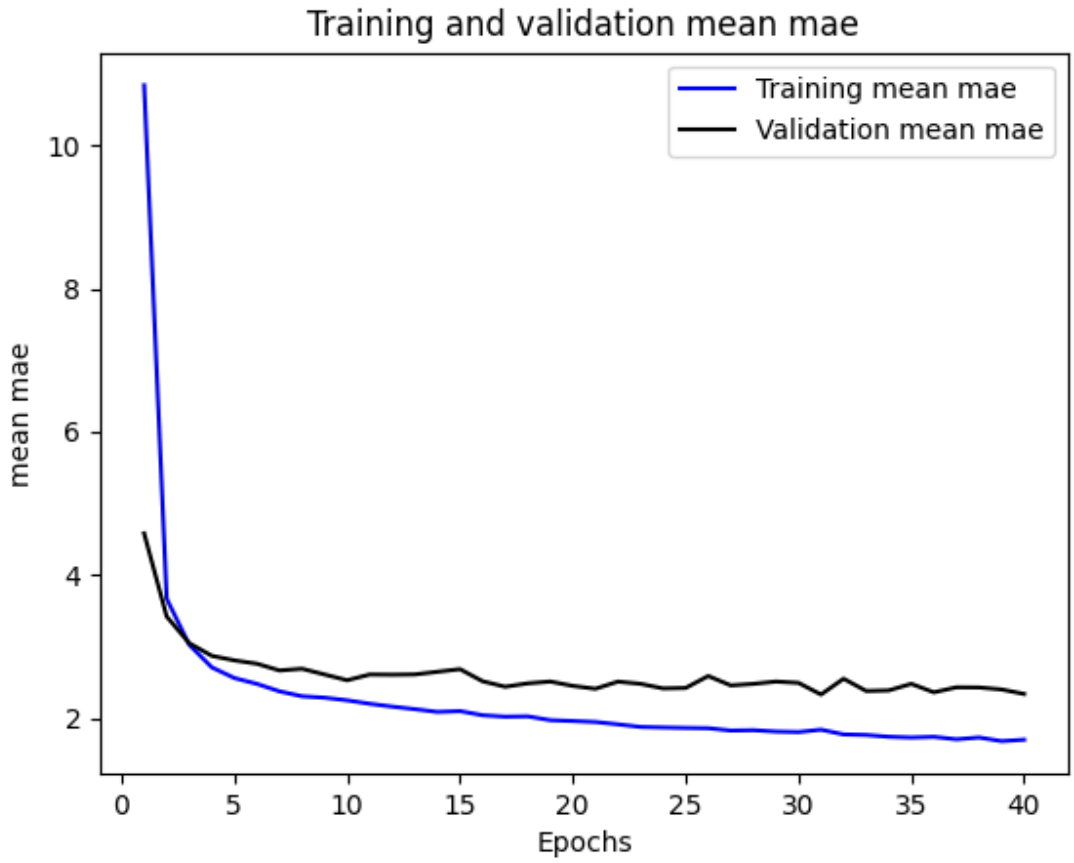

Рисунок 11 - график средней абсолютной ошибки по всем блокам при 40 эпохах

Следующим шагом изучим влияние количества блоков на значение ошибки. На рис. 12-16 представлены графики средней абсолютной ошибки по всем блокам при разных количествах блоков и 40 эпохах обучения модели. 2

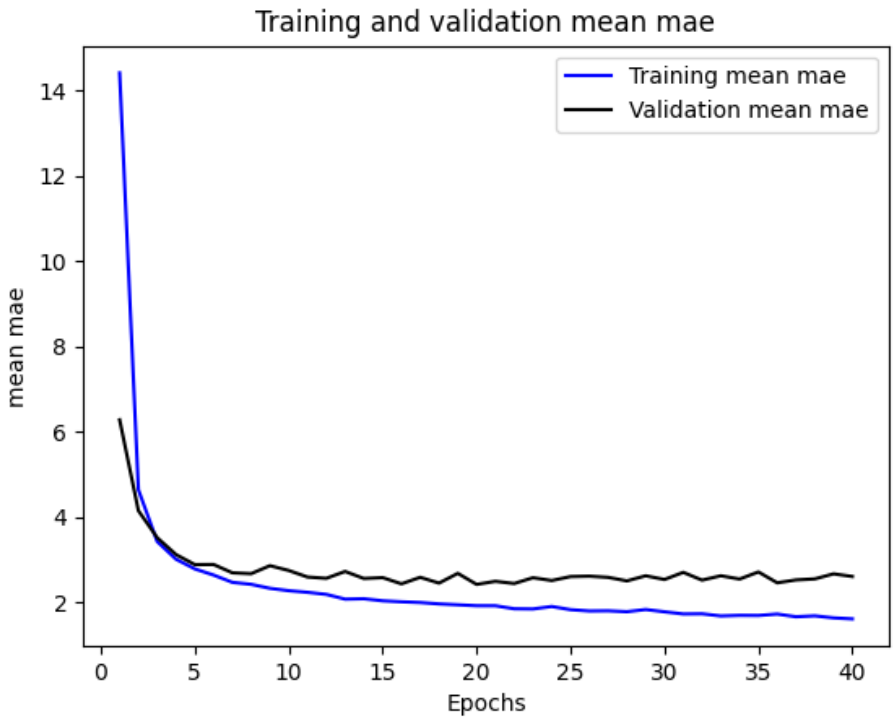

Рисунок 12 - график средней абсолютной ошибки по всем блокам при 3 блоках

Оценка модели с 3 блоками и 40 эпохах равна 2.656266450881958.

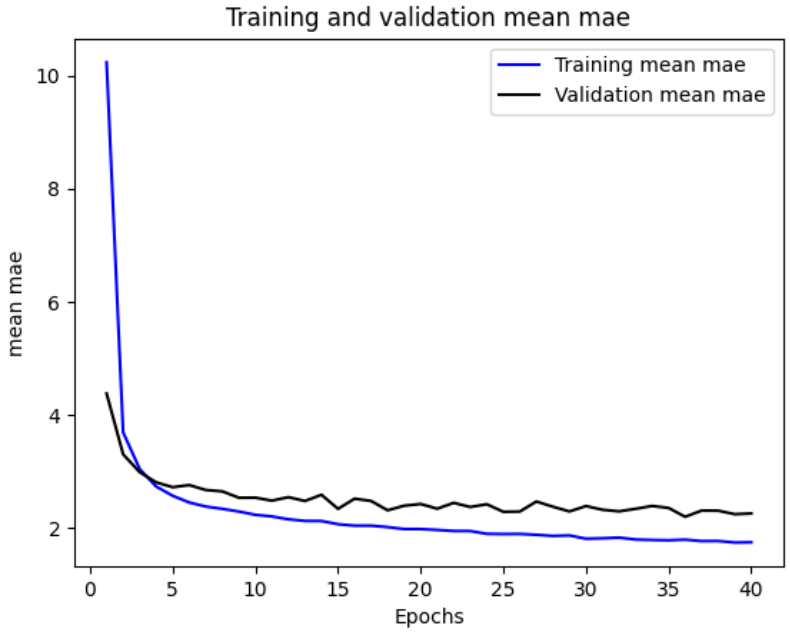

Рисунок 13 - график средней абсолютной ошибки по всем блокам при 6 блоках

Оценка модели с 6 блоками и 40 эпохах равна 2.26108185450236

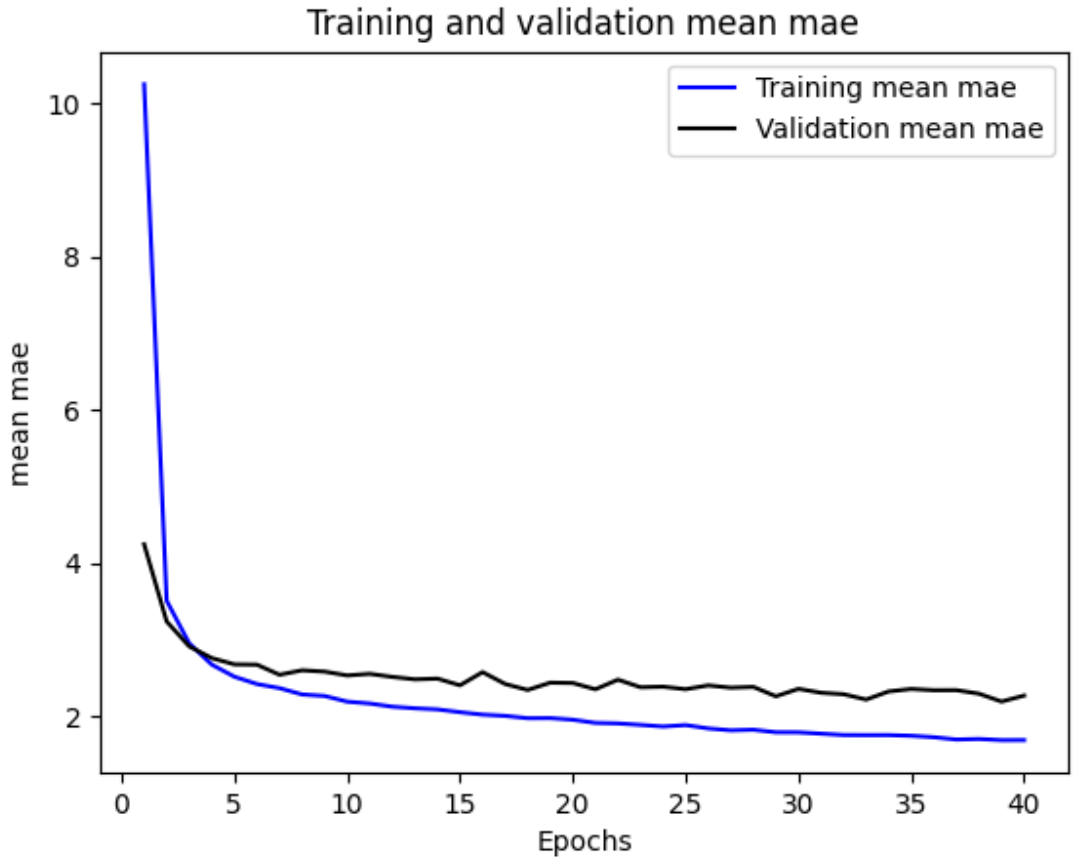

Рисунок 14 - график средней абсолютной ошибки по всем блокам при 8 блоках

Оценка модели с 8 блоками и 40 эпохах равна 2.2780626863241196

Вывод

Во время выполнения лабораторной работы была создана ИНС, которая строит регрессионную предсказание медианной цены на дома в пригороде Бостона, так же можно сделать вывод, что оптимальной является модель с 6 блоками перекрестной проверки и 40 эпохами обучения.

Приложение А

Листинг программы

```
import numpy as np
from tensorflow.keras.datasets import boston_housing
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt


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


def paint_plots(epochs, mae, val_mae, label):
    plt.plot(epochs, mae, 'b', label='Training ' + label)
    plt.plot(epochs, val_mae, 'k', label='Validation ' + label)
    plt.title('Training and validation ' + label)
    plt.xlabel('Epochs')
    plt.ylabel(label)
    plt.legend()
    plt.show()


(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

k = 4
num_val_samples = len(train_data) // k
num_epochs = 100
all_scores = []
all_mae = np.zeros(num_epochs)
all_val_mae = np.zeros(num_epochs)
epochs = range(1, num_epochs + 1)
```

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

    model = build_model()
    history_dict = model.fit(partial_train_data, partial_train_targets, epochs=num_epochs,
            batch_size=1, verbose=0,
                    validation_data=(val_data, val_targets)).history

    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    all_scores.append(val_mae)

    all_mae += np.array(history_dict['mae'])
    all_val_mae += np.array(history_dict['val_mae'])

    paint_plots(epochs, history_dict['mae'], history_dict['val_mae'], 'mae')
    plt.clf()
    paint_plots(epochs, history_dict['loss'], history_dict['val_loss'], 'mse')


paint_plots(epochs, all_mae / k, all_val_mae / k, 'mean mae')

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