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»

Студентка гр. 8382 _____ Ивлева О.А.

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

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 некоторый параметр.

2. Построение модели, подготовка данных

Построение и компиляция модели происходят в функции build_model().

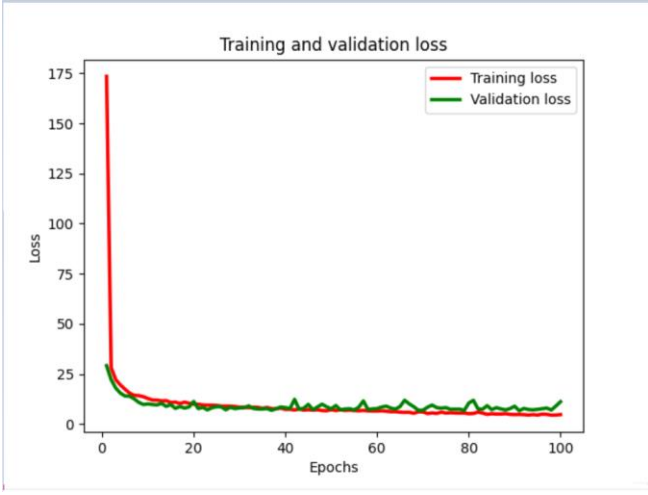
```
def build_model():
    model = Sequential()
    model.add(Dense(64, activation='relu', input_shape=(train_data.shape[1],)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model
```

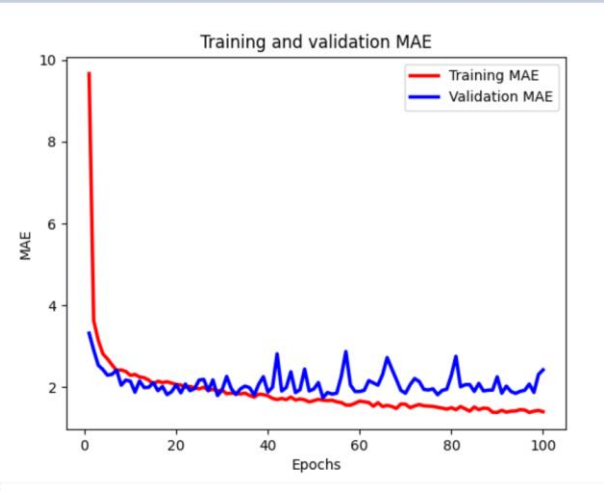
3. Тестирование модели

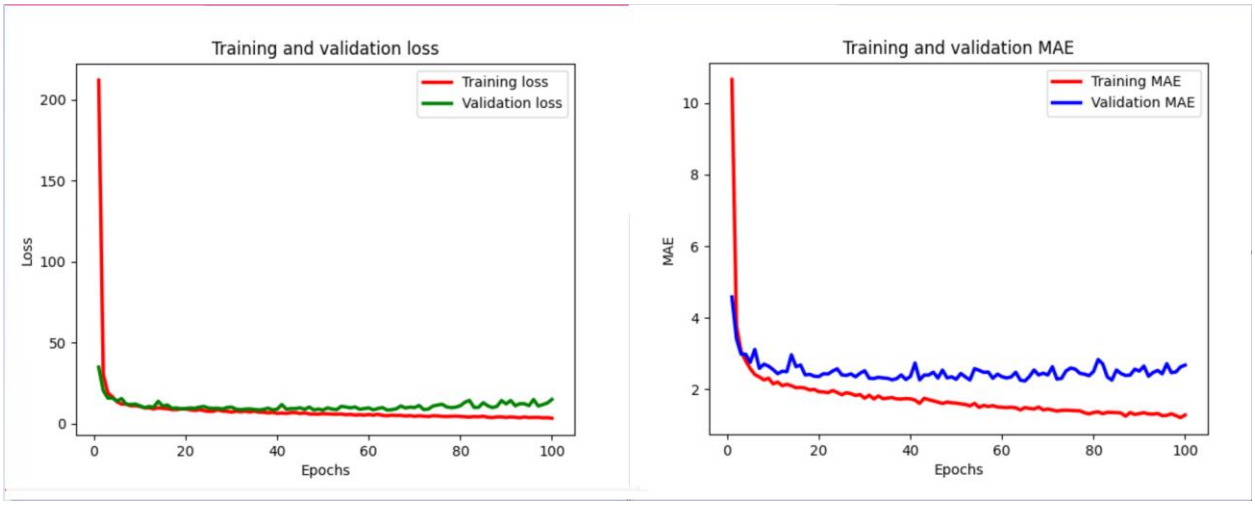
Были выбраны следующие параметры обучения и перекрестной проверки:

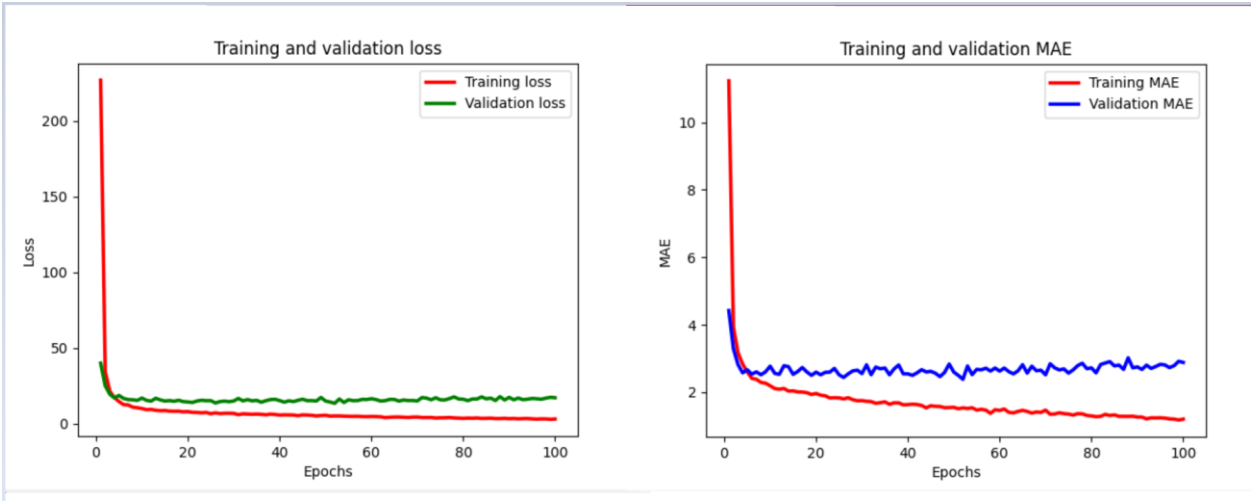
Число эпох: 100

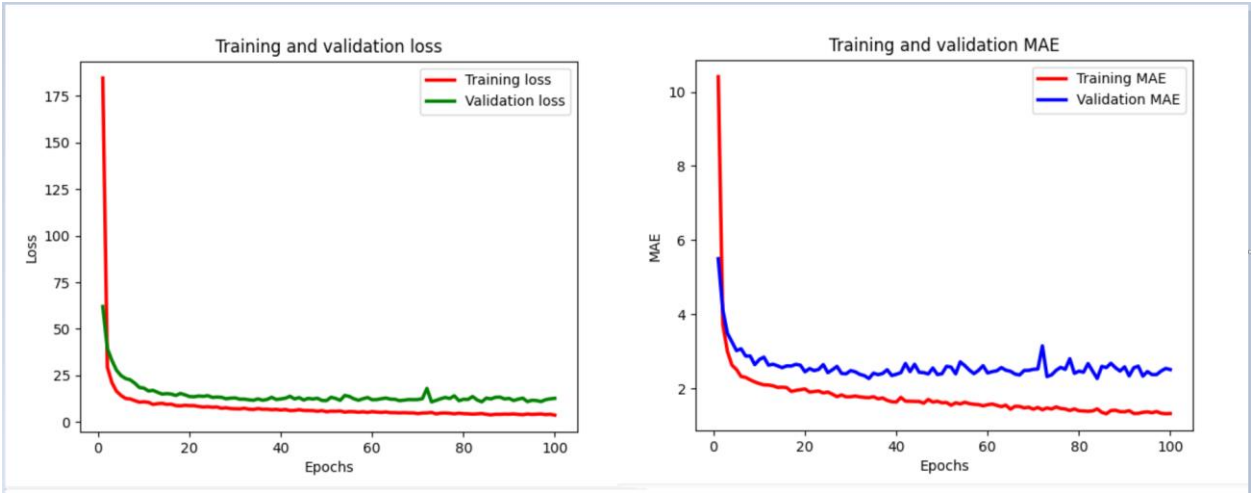
Число блоков: 4

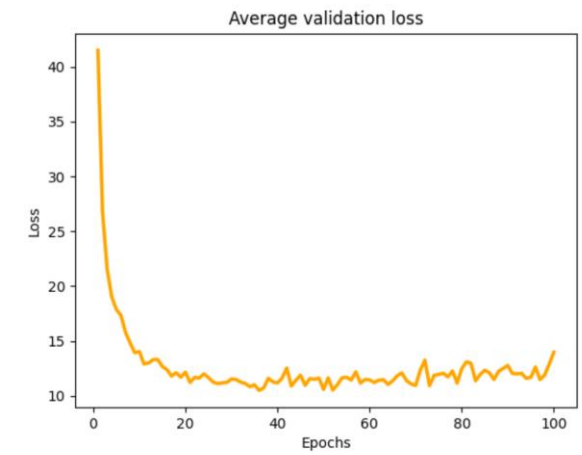


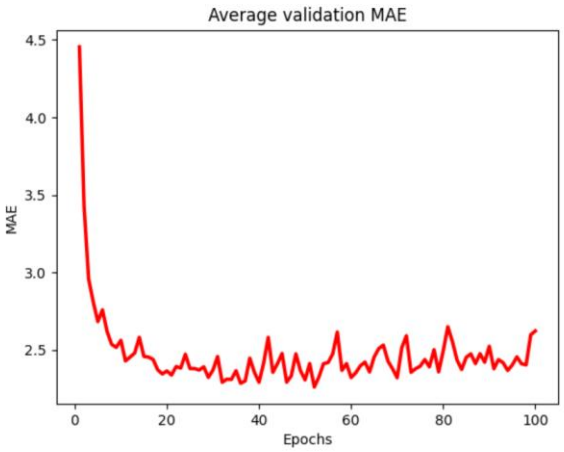



Training and validation loss
Training loss
Validation loss
Loss
Epochs
Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs


Training and validation loss
Training loss
Validation loss
Loss
Epochs
Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs


Training and validation loss
Training loss
Validation loss
Loss
Epochs
Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs



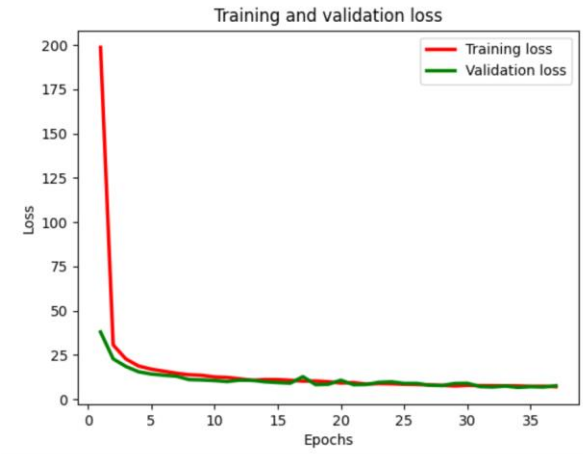


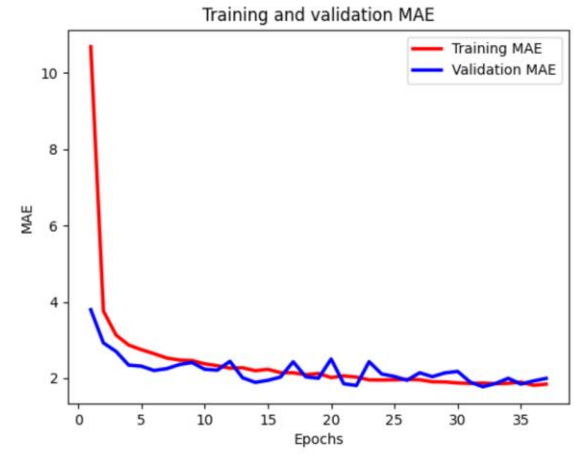
Показатели первой модели

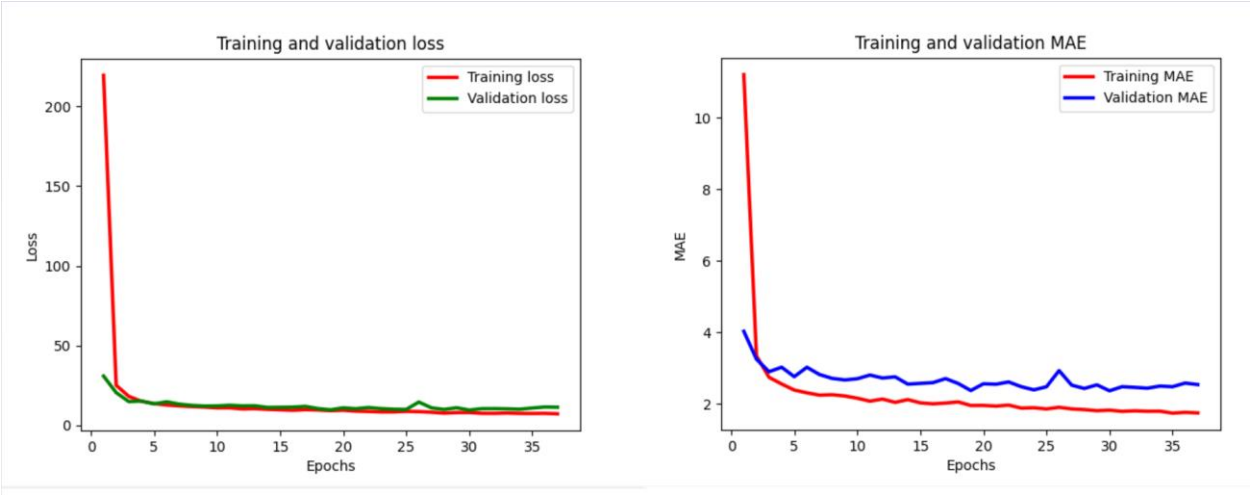
Среднее MAE обученных моделей - 2.468187366724014

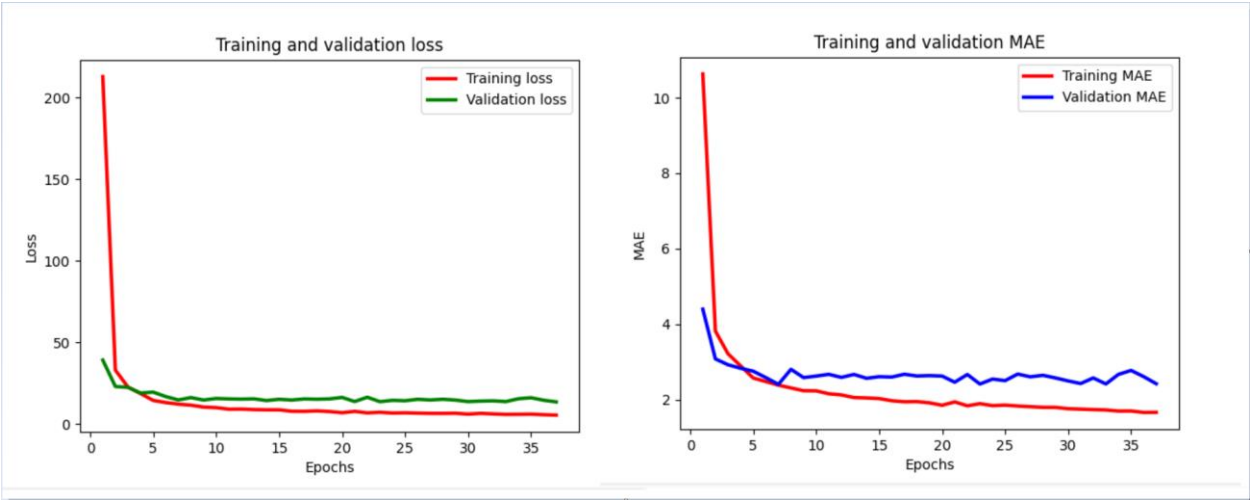
Можно сделать вывод, что примерно после 37-й эпохи модель склонна к переобучению: улучшение показателей на данных для обучения не приводит к лучшим результатам при проверке, а напротив ухудшает их.

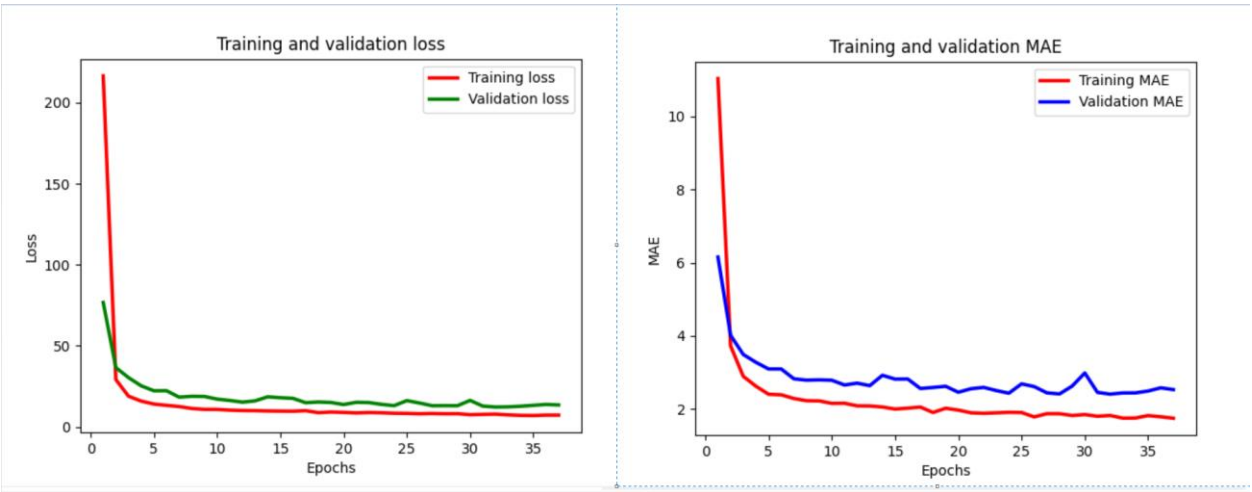
Заменим число эпох на 37.

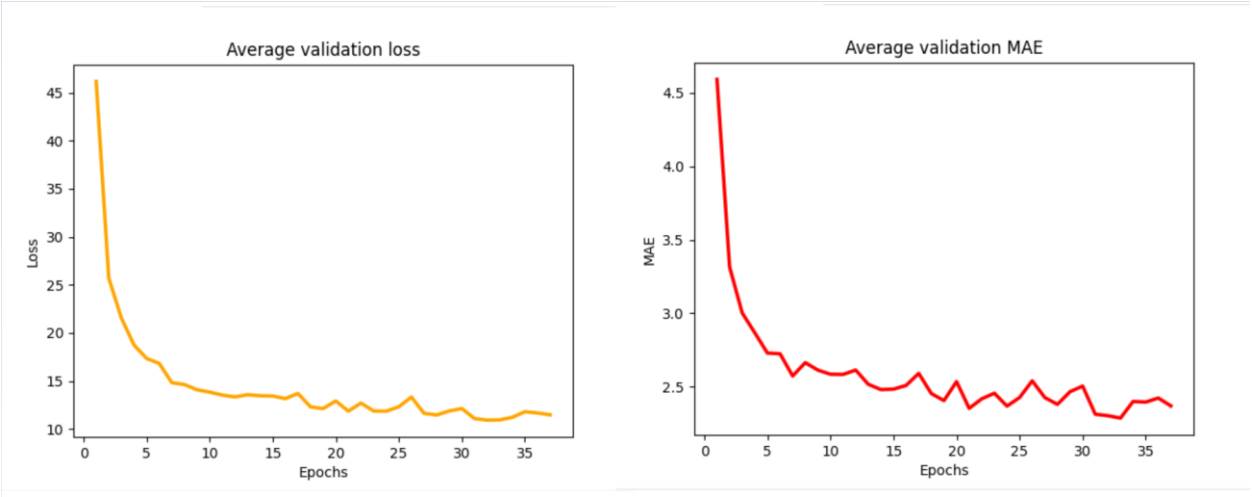





Training and validation loss
Training loss
Validation loss
Loss
Epochs
Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs


Training and validation loss
Training loss
Validation loss
Loss
Epochs
Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs

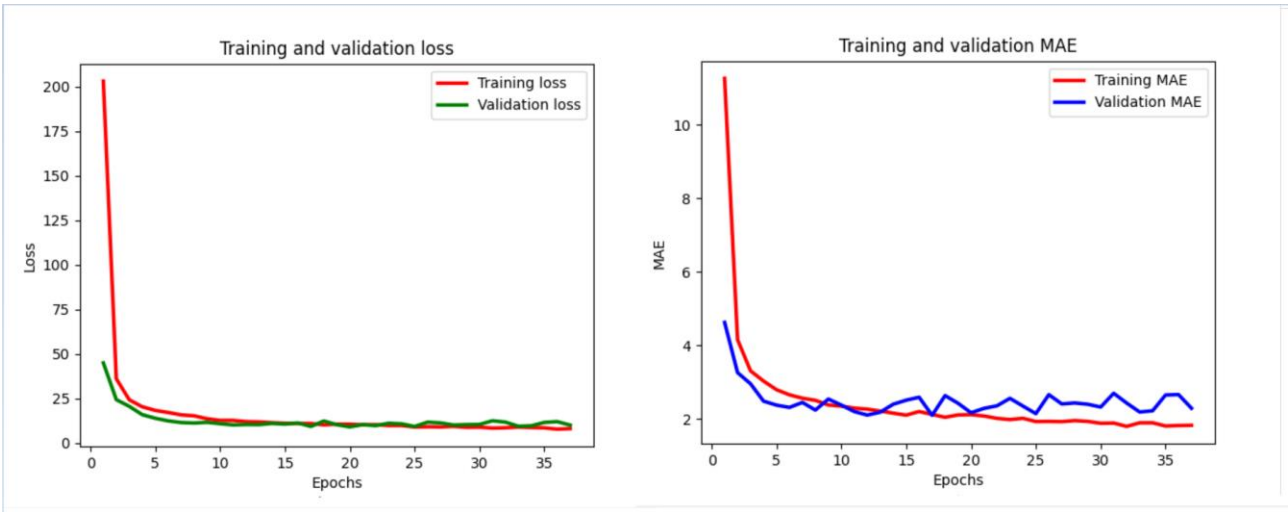

Training and validation loss
Training loss
Validation loss
Loss
Epochs
Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs

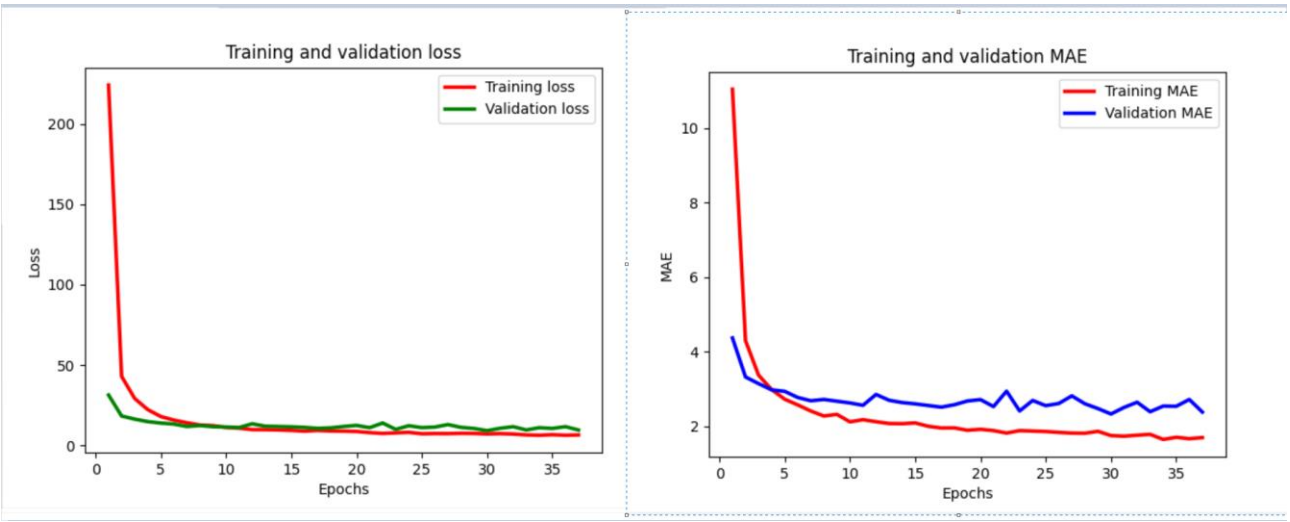


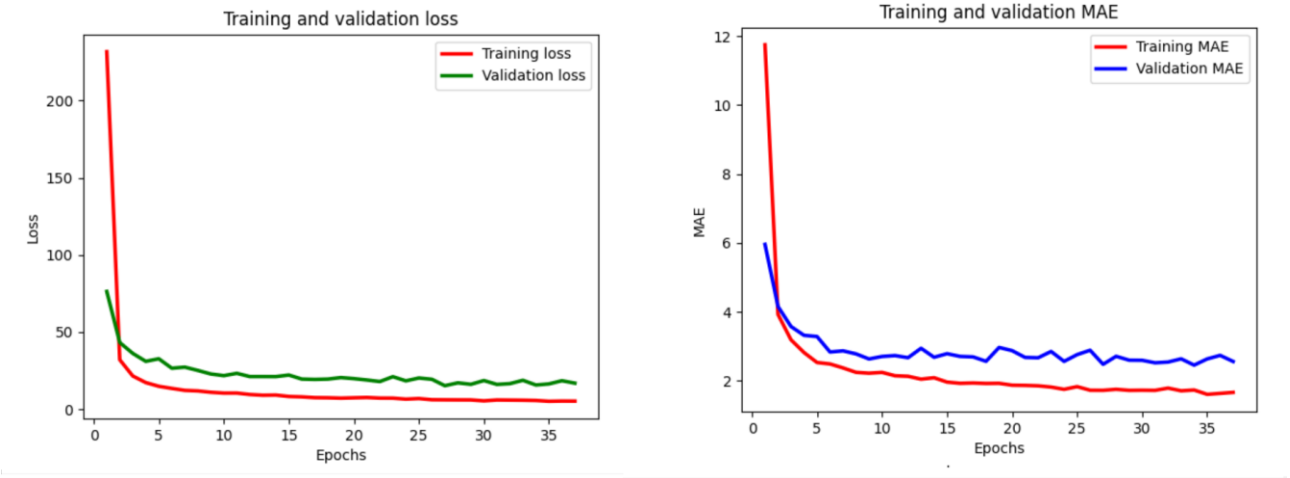
Показатели второй модели

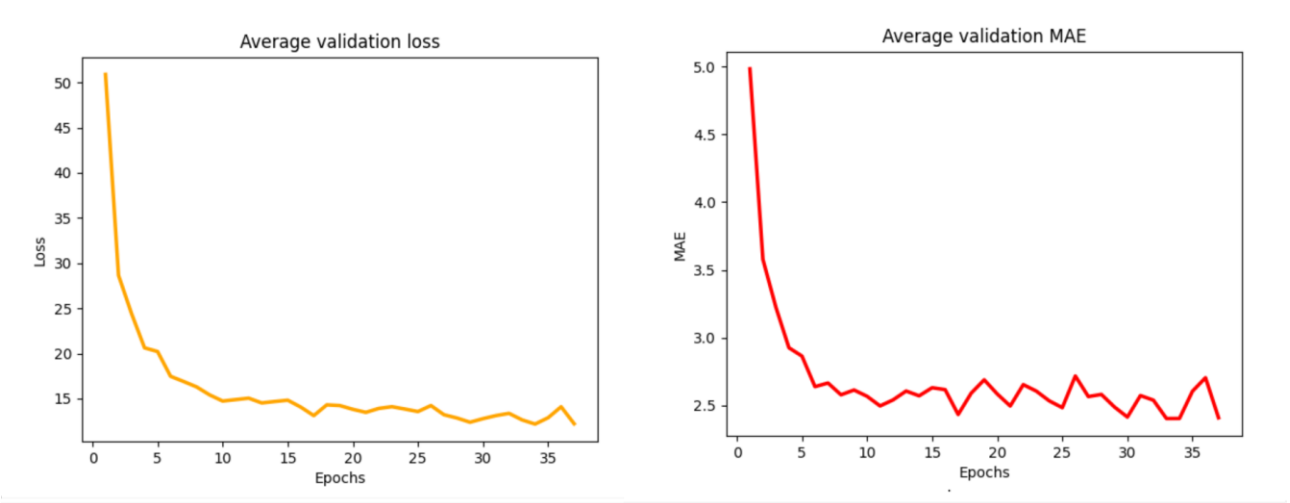
Среднее MAE обученных моделей - 2.3853103310675234, то есть значение стало чуть лучше.

Заменим кол-во блоков на 3:



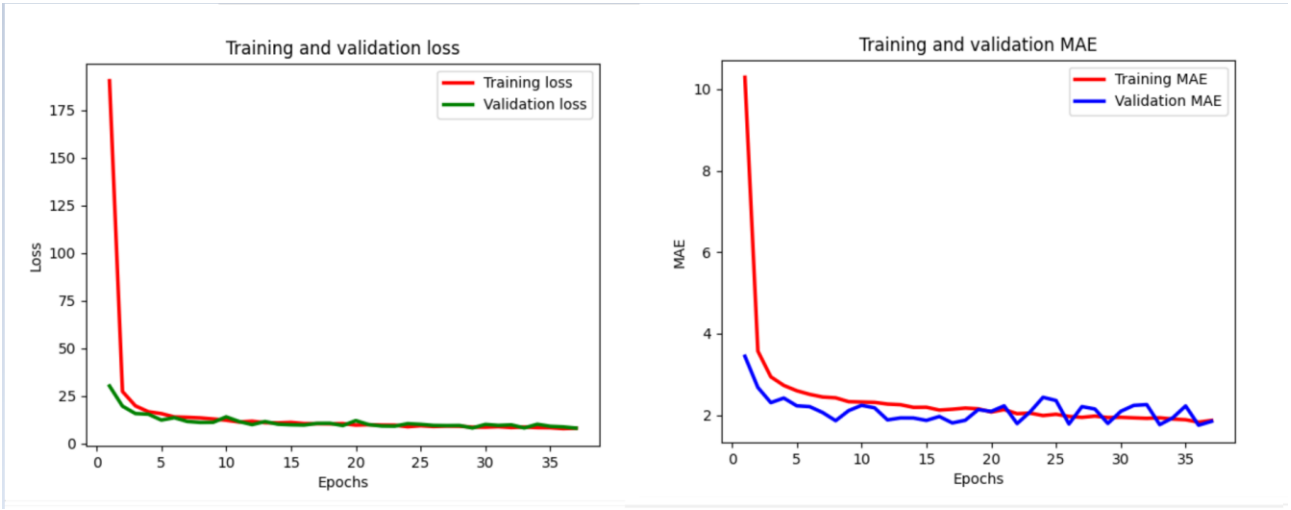

Training and validation loss
Training loss
Validation loss
Loss
Epochs
Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs

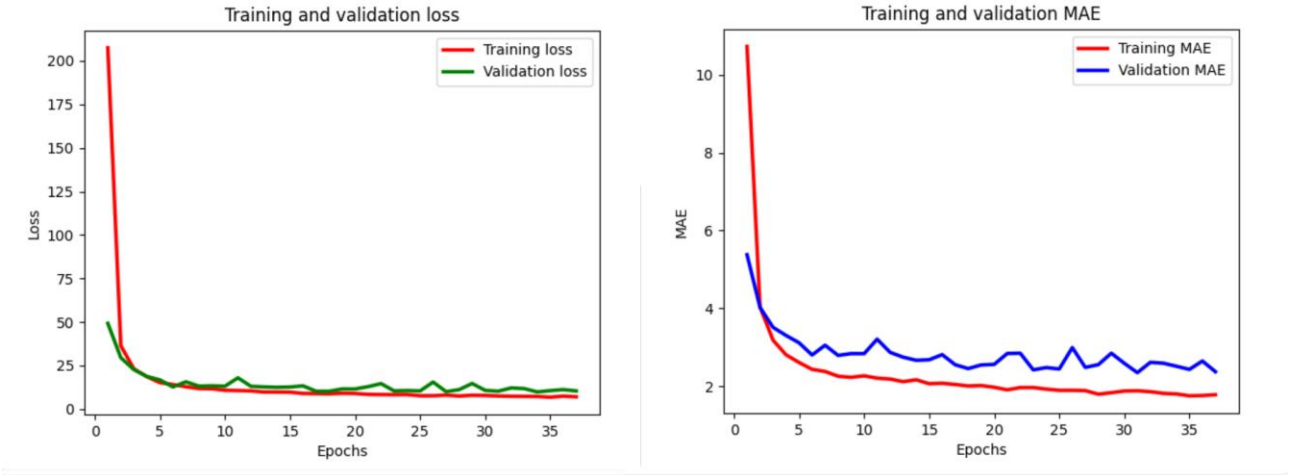

Training and validation loss
Training loss
Validation loss
Loss
Epochs
Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs

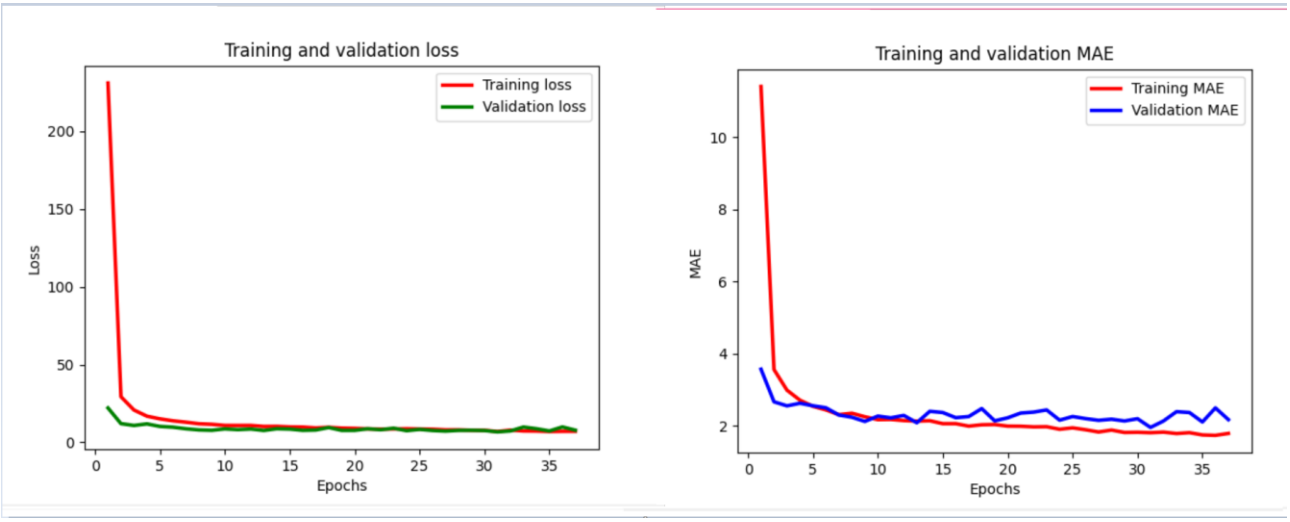

Average validation loss
Loss
Epochs
Average validation MAE
MAE
Epochs

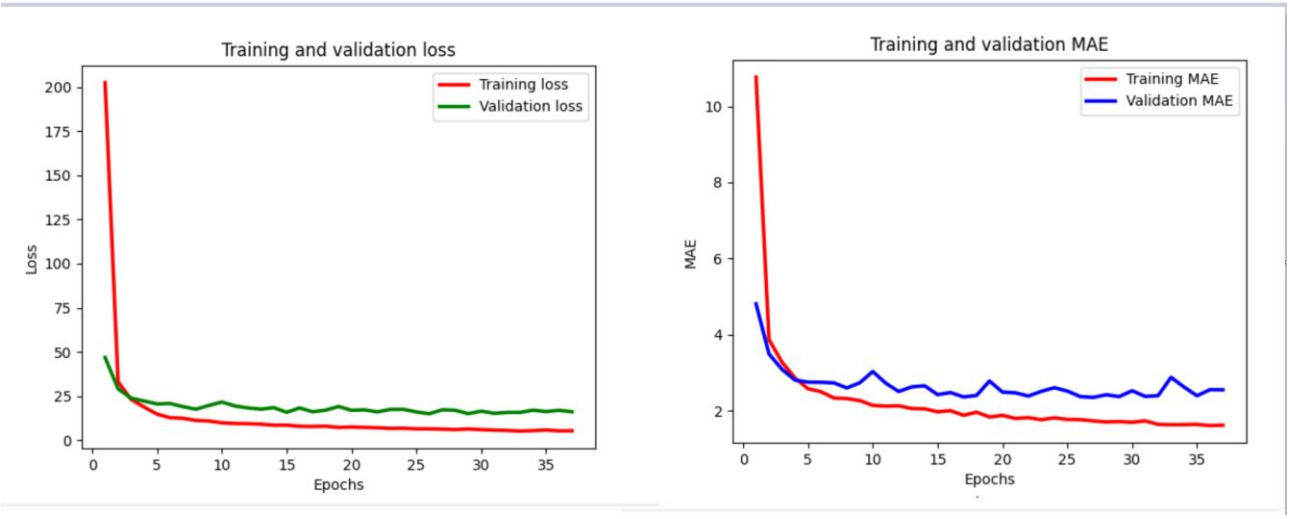
Среднее MAE обученных моделей - 2.5853103310675234, то есть значение стало хуже, чем в случае 4 блоков. Уменьшение кол-ва блоков повышает негативное воздействие случая неудачного обучения. Это можно объяснить тем, что при разделении на 3 блока данных для обучения не хватает.

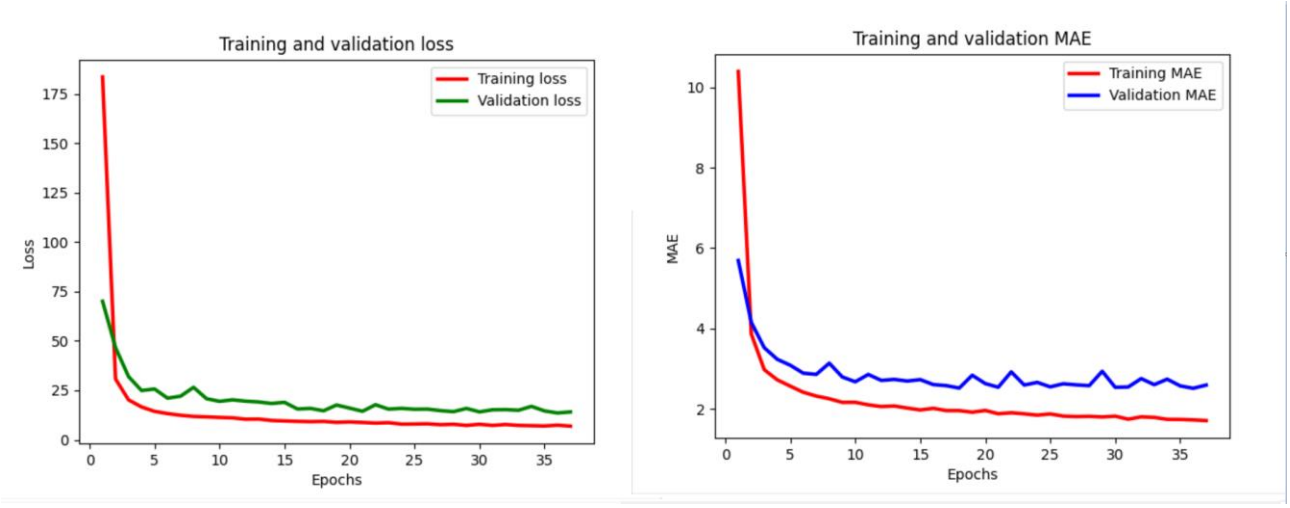
Заменим кол-во блоков на 5:

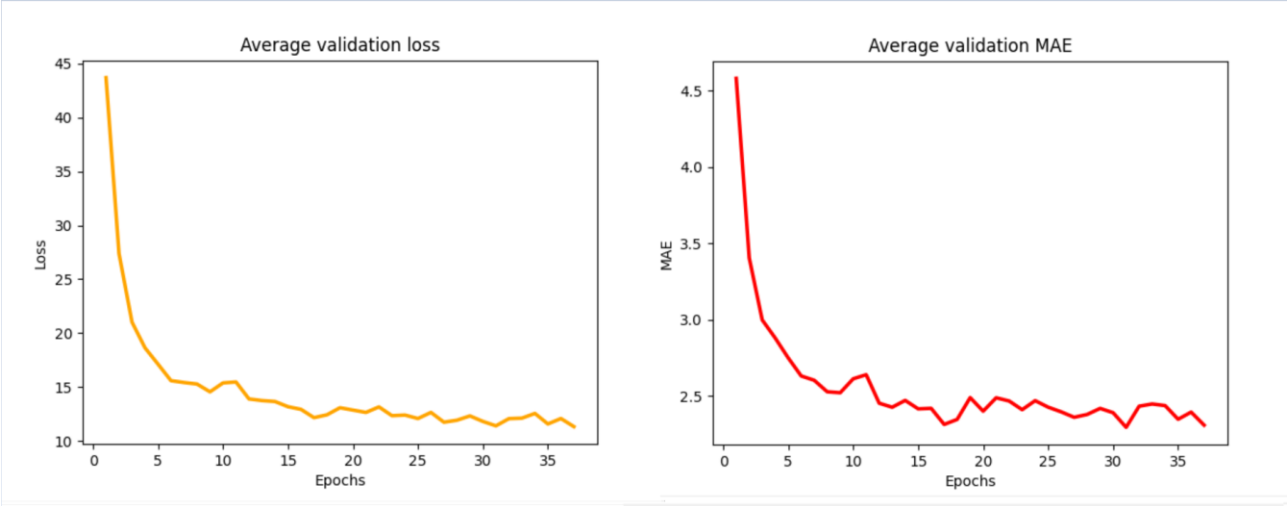





Training and validation loss
Training loss
Validation loss
Loss
Epochs
Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs


Training and validation loss
Training loss
Validation loss
Loss
Epochs
Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs


Training and validation loss
Training loss
Validation loss
Loss
Epochs
Training and validation MAE
Training MAE
Validation MAE
MAE
Epochs



Среднее MAE обученных моделей - 2.359690972276636, то есть значение стало лучше.

Наилучшие результаты модель показывает при использовании 5 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.

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

(train_data, train_targets), (test_data, test_targets) =
boston_housing.load_data()
print(train_data.shape)
print(test_data.shape)
print(test_targets)
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
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 = 5
num_val_samples = len(train_data) // k
num_epochs = 37
all_loss = []
all_mae = []
histories = []

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()
    H = model.fit(partial_train_data, partial_train_targets, epochs=num_epochs,
batch_size=1, verbose=0,
                          validation_data=(val_data, val_targets))
    # val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    all_loss.append(H.history['val_loss'])
    all_mae.append(H.history['val_mae'])
    histories.append(H)


print(np.mean(all_mae))

for history in histories:
    history_dict = history.history

    mse_values = history_dict['loss']
    val_mse_values = history_dict['val_loss']
    epochs = range(1, len(mse_values) + 1)
    plt.plot(epochs, mse_values, 'red', label='Training loss', linewidth=2.5)
    plt.plot(epochs,   val_mse_values,   'green',   label='Validation   loss',
linewidth=2.5)
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    plt.clf()
    mae_values = history_dict['mae']
    val_mae_values = history_dict['val_mae']
    plt.plot(epochs, mae_values, 'red', label='Training MAE', linewidth=2.5)
    plt.plot(epochs,    val_mae_values,    'blue',    label='Validation    MAE',
linewidth=2.5)
    plt.title('Training and validation MAE')
    plt.xlabel('Epochs')
    plt.ylabel('MAE')
    plt.legend()
    plt.show()


avg_mae = np.asarray(all_mae).mean(axis=0)
avg_loss = np.asarray(all_loss).mean(axis=0)

epochs = range(1, len(avg_mae) + 1)
plt.plot(epochs, avg_loss, 'orange', linewidth=2.5)
plt.title('Average validation loss')
plt.xlabel('Epochs')
```

```
plt.ylabel('Loss')
plt.show()

plt.clf()
plt.plot(epochs, avg_mae, 'red', linewidth=2.5)
plt.title('Average validation MAE')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.show()
```