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 домов в Бостоне

Студент гр. 8383 _____ Дейнега В. Е.

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.

Задание.

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

Выполнение работы

1) Задача классификации – получение категориального ответа на основе набора признаков. Имеет конечное количество ответов. Например задача бинарной классификации состоит в определении принадлежности объекта к одному из двух классов.

Задача регрессии – прогноз на основе выборки объектов с различными признаками. На выходе должно получиться вещественное число. Количество видов прогнозов не ограничено чем-либо.

2) Создадим модель ИНС.

Исходные данные имеют самые разные дипазоны, поэтому было принято решение применить нормализацию. Из каждого значения вычитается среднее по этому признаку, и разность делится на стандартное отклонение, в результате признак центрируется по нулевому значению и имеет стандартное отклонение, равное единице.

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

```
train_data /= std
test_data -= mean
test_data /= std
```

Построим и скомпилируем модель с помощью функции `def 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
```

Так как количество данных невелико, применим перекрестную проверку по K блокам. Т.е. разобьем тестовые данные на K блоков, создадим K моделей обучения, 1 блок оставляется для тестирования модели, а K-1 для тренировки, получается K оценок, средняя величина будет оценкой модели.

3) Протестируем модель для количество эпох 150 и K = 4, графики СКО и средней абсолютной ошибки приведены на рис. 1-4, график среднего значения средней абсолютной на рис. 5.

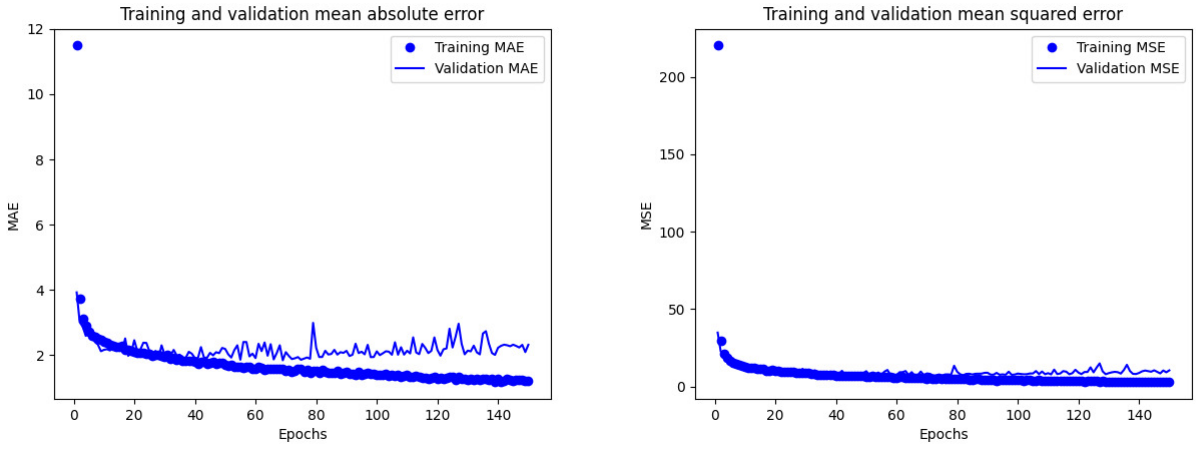


Рисунок 1 – Графики средней абсолютной ошибки и СКО для 1 итерации

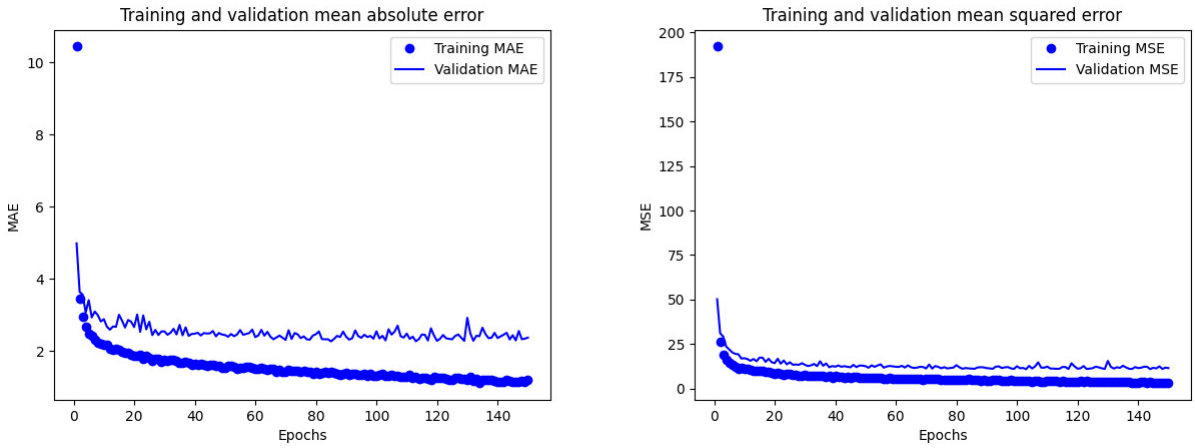


Рисунок 2 – Графики средней абсолютной ошибки и СКО для 2 итерации

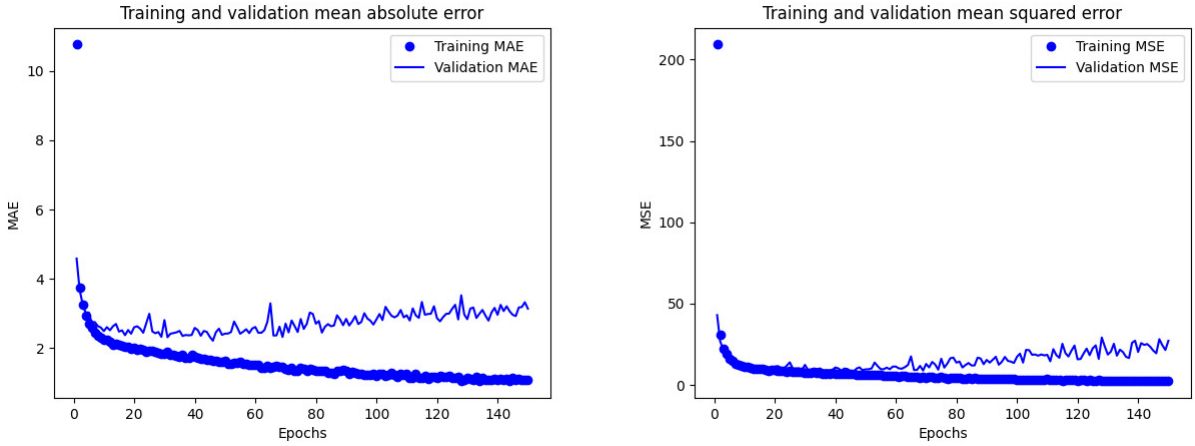


Рисунок 3 – Графики средней абсолютной ошибки и СКО для 3 итерации

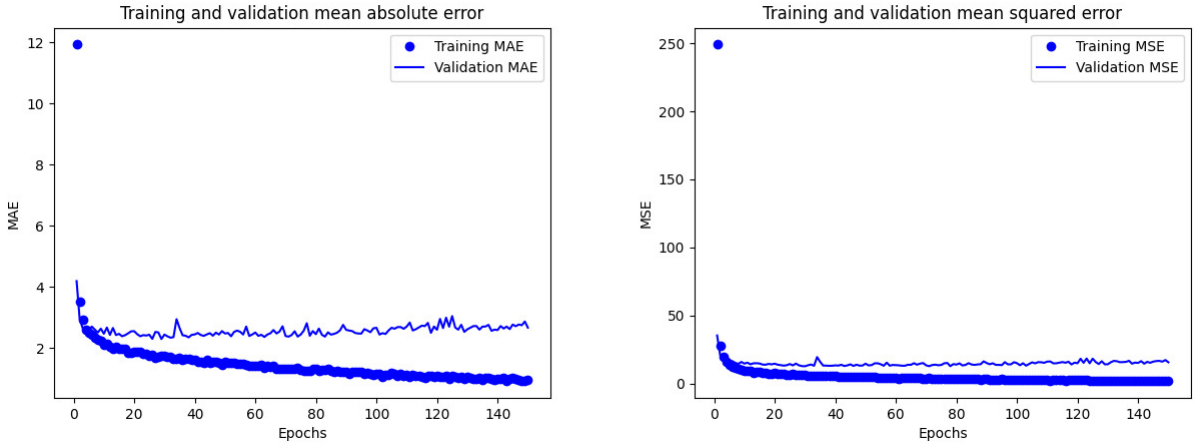


Рисунок 4 – Графики средней абсолютной ошибки и СКО для 4 итерации

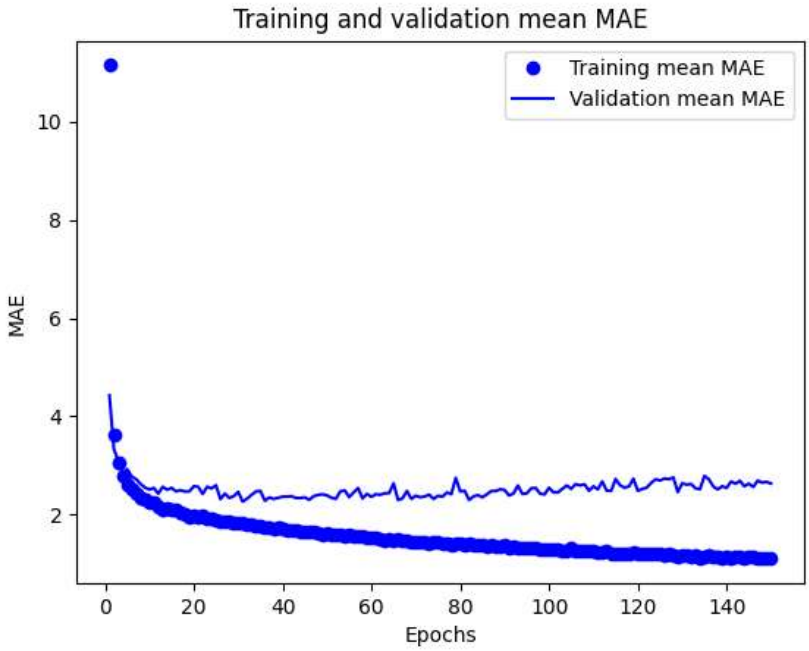


Рисунок 5 - График среднего значения средней абсолютной ошибки модели.

На графике среднего значения средней абсолютной ошибки видно, что значение ошибки для тестовых данных не уменьшается примерно с 20 эпохи, при этом ошибки на 140 эпохе значительно больше ошибки на 20 эпохе, что свидетельствует о переобучении модели. Предположим, что точка переобучения – 35 эпоха. Наиболее ярко переобучение заметно на рис. 3.

4) Уменьшим количество эпох до 35, графики СКО и средней абсолютной ошибки приведены на рис. 6-9, график среднего значения средней абсолютной на рис. 10.

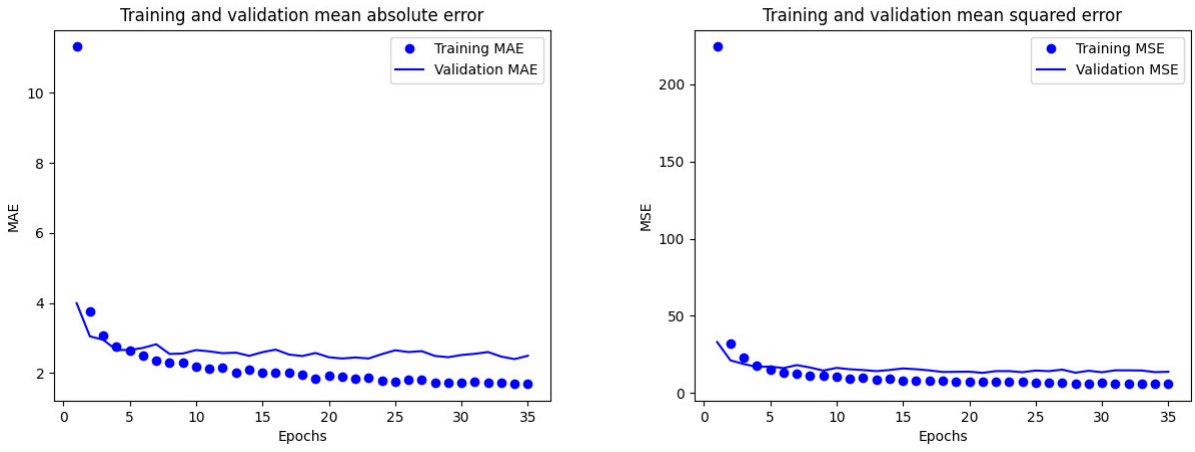
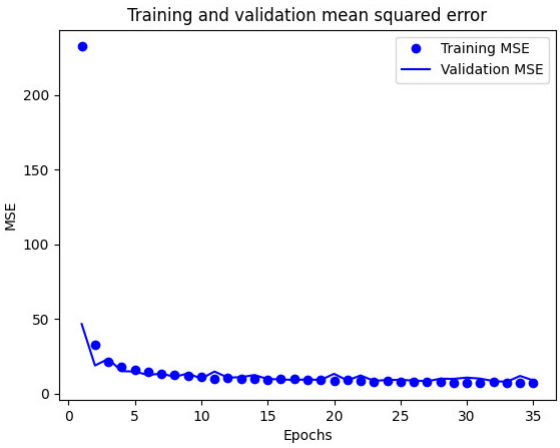


Рисунок 6 – Графики средней абсолютной ошибки и СКО для 1 итерации



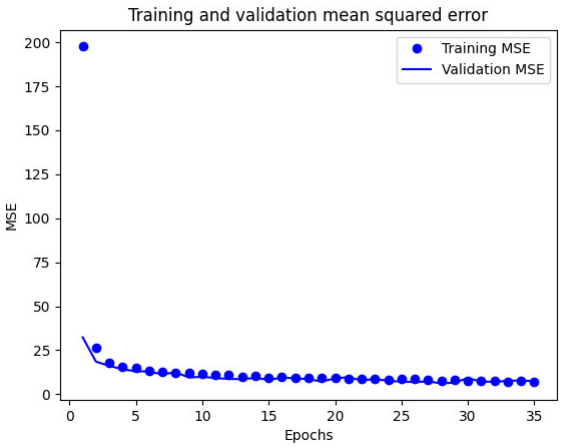
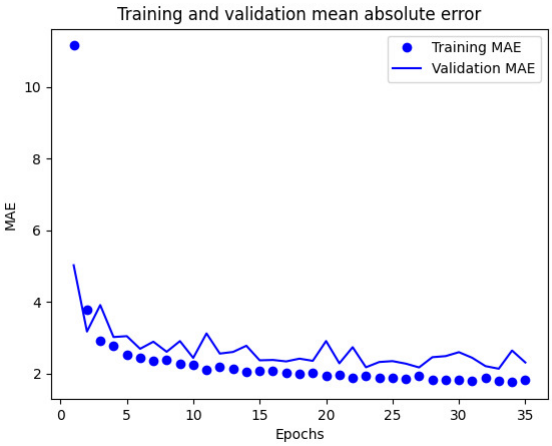


Рисунок 7 – Графики средней абсолютной ошибки и СКО для 2 итерации



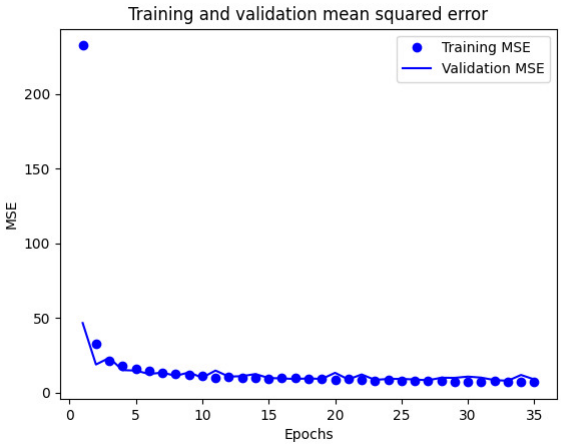
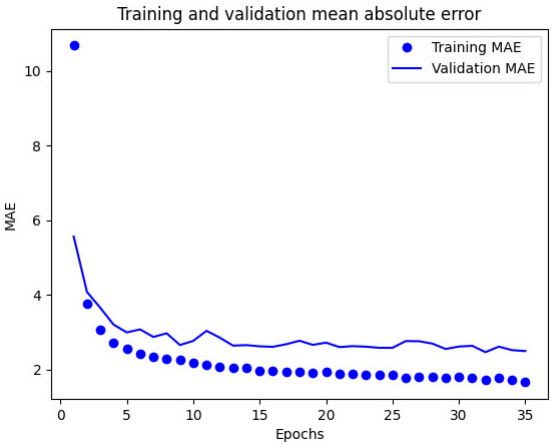


Рисунок 8 – Графики средней абсолютной ошибки и СКО для 3 итерации



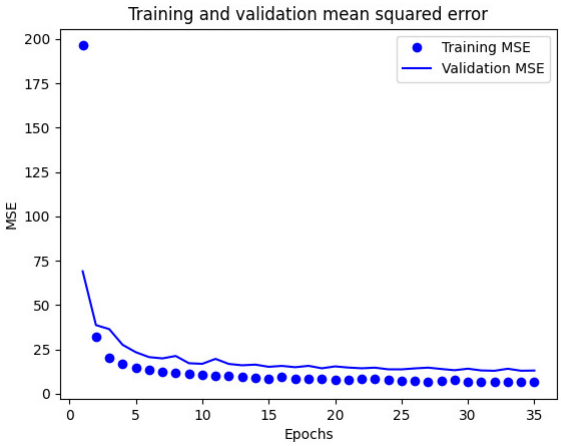


Рисунок 9 – Графики средней абсолютной ошибки и СКО для 4 итерации

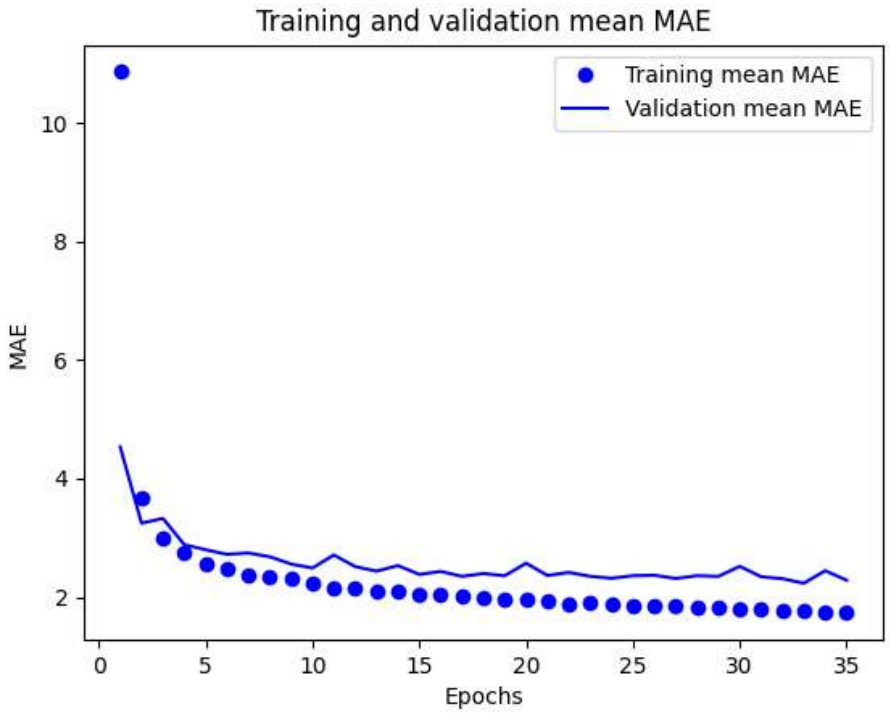


Рисунок 10 - График среднего значения средней абсолютной ошибки модели 2.

Теперь на графике среднего значения средней абсолютной ошибки не видно признаков переобучения, также ошибка уменьшилась. На рис. 7 – 9 также нет признаков переобучения, однако на рисунке 6 можно заметить, что средняя абсолютная ошибка значительно не уменьшается и не растет после 15 эпохи, полагаю, что это можно списать на неудачный выбор набора данных.

Для предложенных данных можно сделать вывод, что уменьшение количества эпох до 35 улучшило работу нейросети.

5) Изменим количество блоков разбиения, графики среднего значения средней абсолютной ошибки буду приведены на рис. 11, 12, 13.

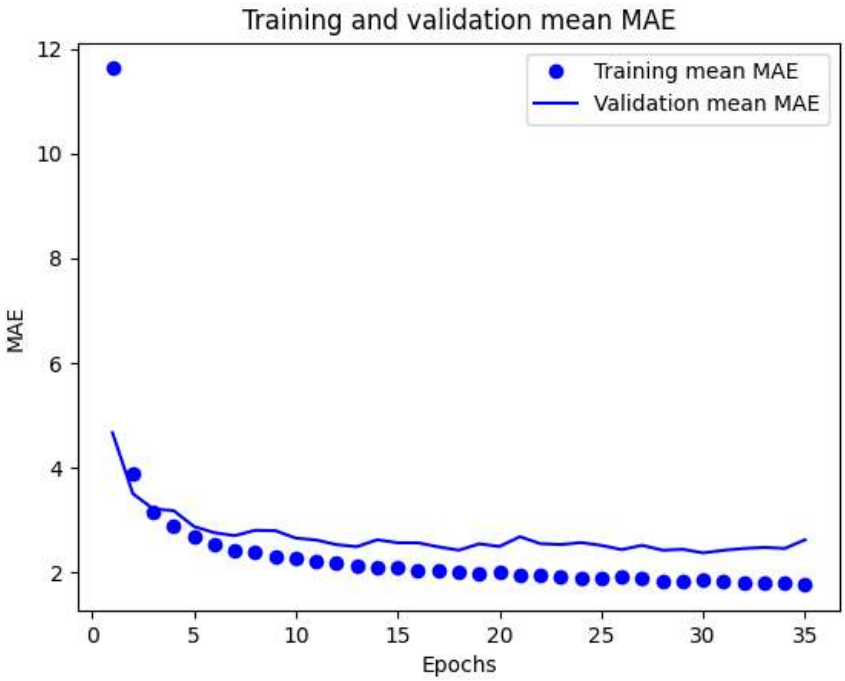


Рисунок 11 - График среднего значения MAE при k = 3.

При k = 3 заметного падения ошибки добиться не удалось, оценка модели 2.6292686462402344, увеличим k до 6.

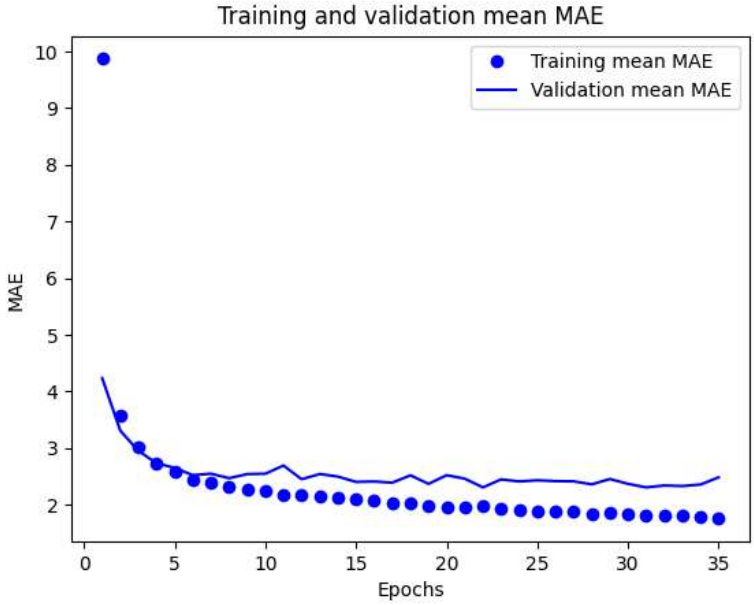


Рисунок 12 - График среднего значения MAE при k = 6.

При k = 6 график ошибки на тестовых данных стал более пологим, ошибка уменьшилась по сравнению с k = 3, однако заметного улучшения по сравнению

с k = 4 добиться не удалось, оценка модели 2.4848204056421914 увеличим k до 10.

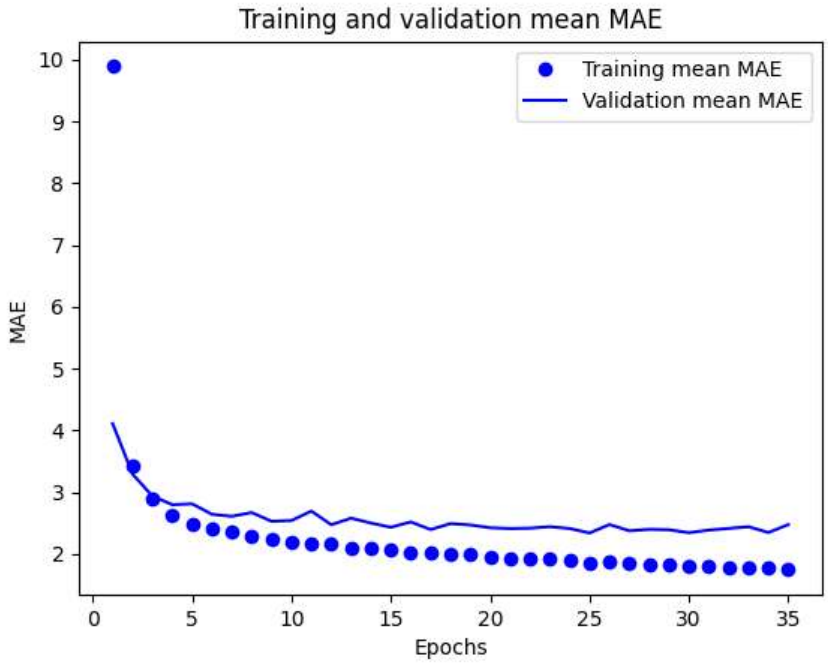


Рисунок 13 - График среднего значения MAE при k = 10.

При k = 10 заметного улучшения результатов ИНС добиться не удалось, оценка модели 2.475683164596558. Код представлен в приложении А.

Выводы.

В ходе лабораторной работы было реализовано предсказание медианной цены на дома в пригороде Бостона в середине 1970-х. Из полученных данных можно сделать вывод, что оптимальная конфигурация модели – это 35 эпох и 4 блока перекрестной проверки.

Приложение А

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

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

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

mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /= std
test_data -= mean
test_data /= std


k = 4

num_val_samples = len(train_data) // k
num_epochs = 35
all_scores = []
all_train_mae = np.zeros(num_epochs)
all_val_mae = np.zeros(num_epochs)

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()
    hist = 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_scores.append(val_mae)
    epochs = range(1, num_epochs + 1)
    all_train_mae += np.array(hist.history['mae'])
    all_val_mae += np.array(hist.history['val_mae'])
    plt.plot(epochs, hist.history['mae'], 'bo', label='Training MAE')
    plt.plot(epochs, hist.history['val_mae'], 'b', label='Validation MAE')
    plt.title('Training and validation mean absolute error')
    plt.xlabel('Epochs')
```

```
    plt.ylabel('MAE')
    plt.legend()
    plt.show()

    plt.clf()
    mse_values = hist.history['loss']
    val_mse_values = hist.history['val_loss']
    plt.plot(epochs, mse_values, 'bo', label='Training MSE')
    plt.plot(epochs, val_mse_values, 'b', label='Validation MSE')
    plt.title('Training and validation mean squared error')
    plt.xlabel('Epochs')
    plt.ylabel('MSE')
    plt.legend()
    plt.show()



print(np.mean(all_scores))

plt.clf()
plt.plot(epochs, all_train_mae/k, 'bo', label='Training mean MAE')
plt.plot(epochs, all_val_mae/k, 'b', label='Validation mean MAE')
plt.title('Training and validation mean MAE')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.legend()
plt.show()
```