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

Студентка гр. 8382 _____ Кузина А.М.

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

Требования:

- 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

Ход работы

Данные для анализа в программу загружаются из набор данных, который присутствует в составе Keras. Затем входные данные нормализуются – признаки центрируются по нулевому значению со стандартным отклонением 1. Это нужно, потому что входные значениях имеют разные диапазоны, сеть хоть и может адаптироваться к этому, но это значительно усложнит ее обучение.

```
(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
```

Затем данные разделяются на k блоков – будет рассмотрено k моделей сети, которые будут отличаться между собой лишь тем, какой из блоков будет использован как валидационные данные, а какие оставшиеся как обучающие данные.

Модели имеют три слоя – входной и скрытый по 64 нейрона с функцией активации relu и выходной слой с 1м нейроном без функции активации. На выходном слое не нужна функция активации так как перед нами стоит задача регресии, а не классификации. Если в задаче классификации мы хотим определить класс (вероятность попадания в класс), к которому относятся входные параметры, при конечном числе классов, то в задаче регресии мы хотим по входным данным получить выходное значение, которое по сути может быть любым. Функция потерь - mse — mean squared error (среднеквадратичная ошибка), вычисляющей квадрат разности между предсказанными и целевыми значениями. Эта функция широко используется в задачах регрессии. Также добавлен новый параметр на этапе обучения: mae — mean absolute error (средняя абсолютная ошибка). Это абсолютное значение разности между предсказанными и целевыми значениями.

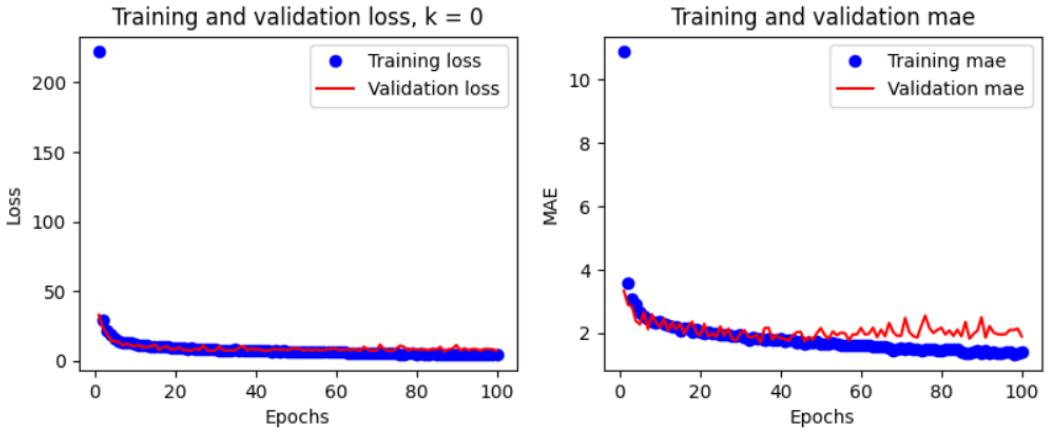
```
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(train_data.shape[1],)))
model.add(Dense(64, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
```

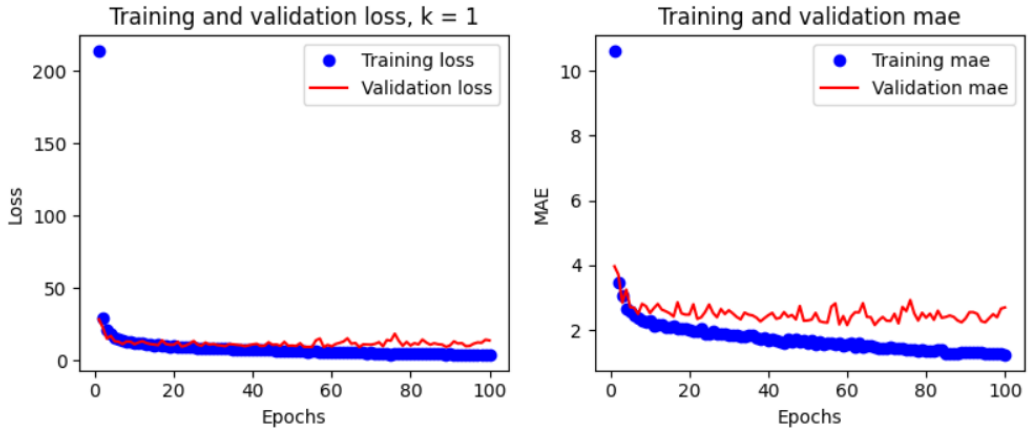
Полученная сеть проходит обучение в течение num_epochs эпох, и с размером батча – 1.

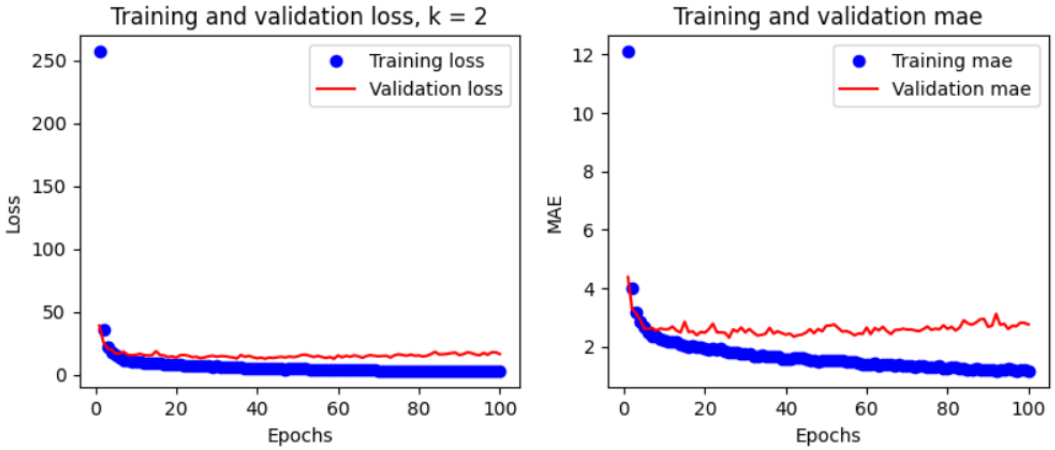
```
history = model.fit(partial_train_data, partial_train_targets,
validation_data=(val_data, val_targets),
                    epochs=num_epochs, batch_size=1, verbose=0)
```

Вначале был рассмотрен вариант деления входных данных на 4 блока и обучение в течение 100 эпох.

Графики ошибок и точности изначальной сети представлены на рисунке 1


Training and validation loss, k = 0
Training loss
Validation loss
Loss
Epochs
Training and validation mae
Training mae
Validation mae
MAE
Epochs


Training and validation loss, k = 1
Training loss
Validation loss
Loss
Epochs
Training and validation mae
Training mae
Validation mae
MAE
Epochs


Training and validation loss, k = 2
Training loss
Validation loss
Loss
Epochs
Training and validation mae
Training mae
Validation mae
MAE
Epochs

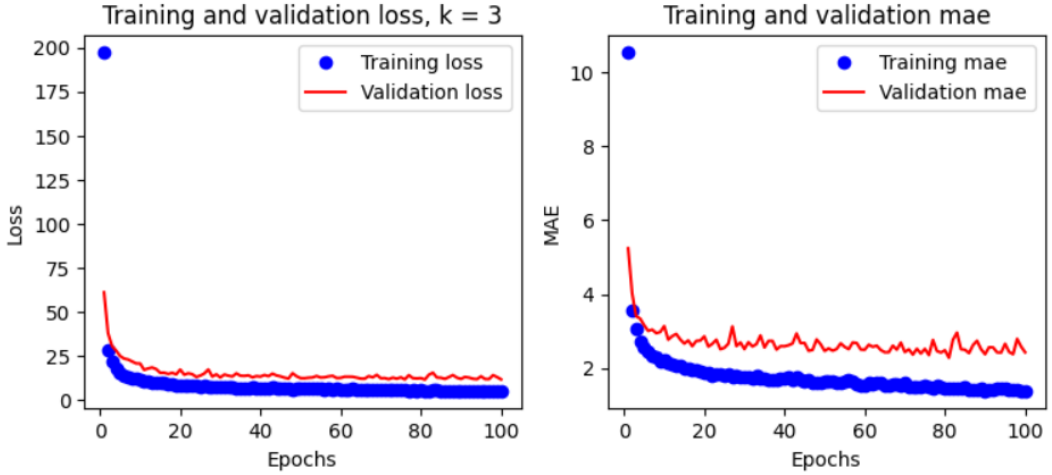

Training and validation loss, k = 3
Training loss
Validation loss
Loss
Epochs
Training and validation mae
Training mae
Validation mae
MAE
Epochs

Рисунок 1 – Ошибки изначальной сети

Усредненные графики потерь и средней абсолютной ошибки представлены на рисунке 2.

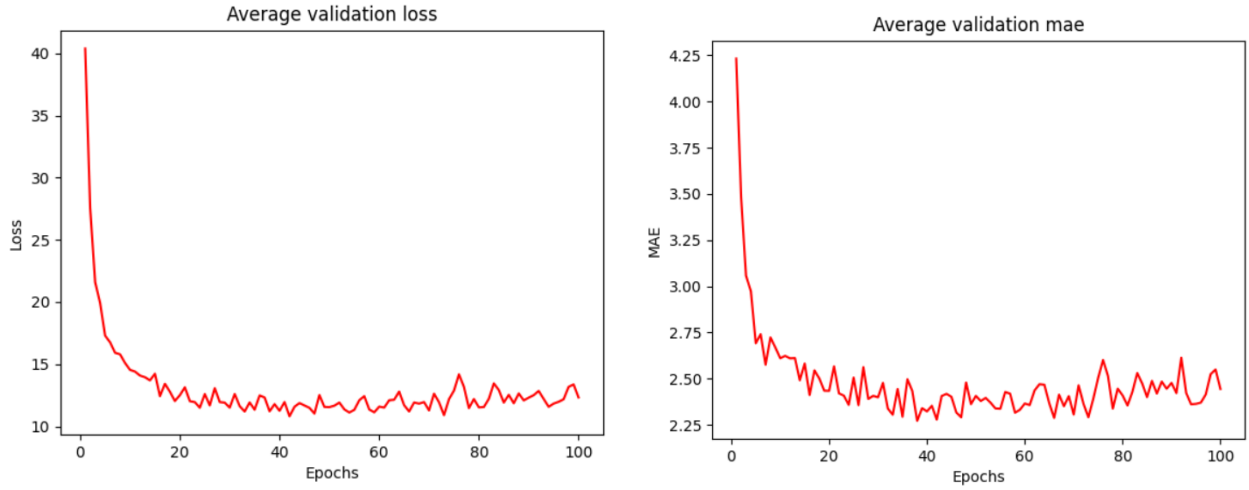
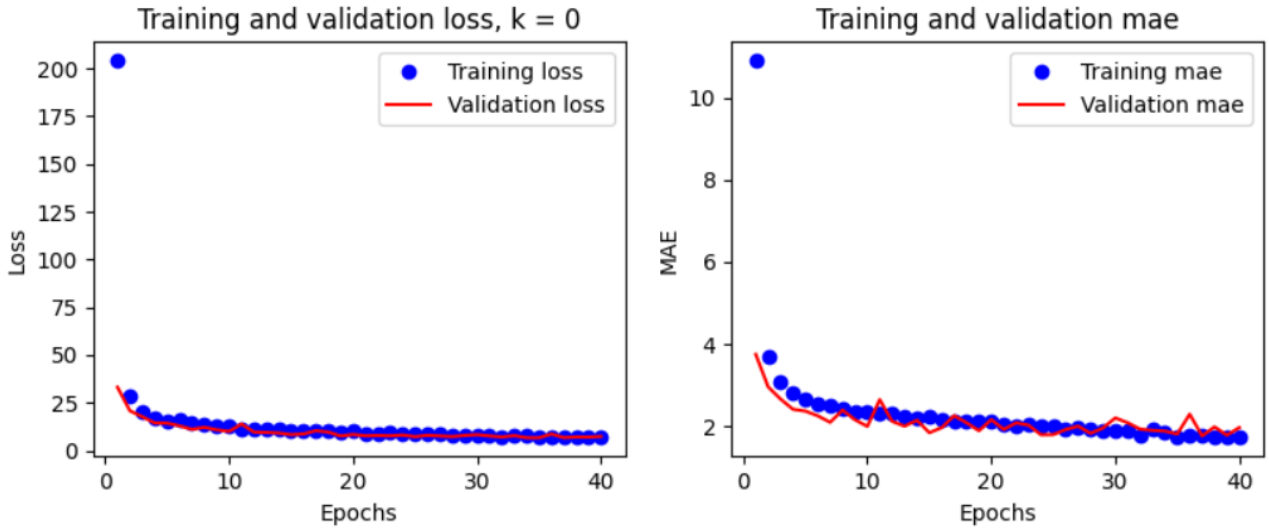


Рисунок 2 – усредненные графики первой конфигурации сети

Полученная оценка модели – 2.44 т.е. средняя разница между реальной и предзаказанной ценами составила 2440 долларов, что весьма немало в заданном диапазоне цен.

На представленных графиках можно заметить, что в среднем после 40 эпохи модель начинает переобучаться, поэтому количество эпох было уменьшено до 40. Была проведена оценка данной конфигурации сети, ее результаты отображены на рисунке 3.



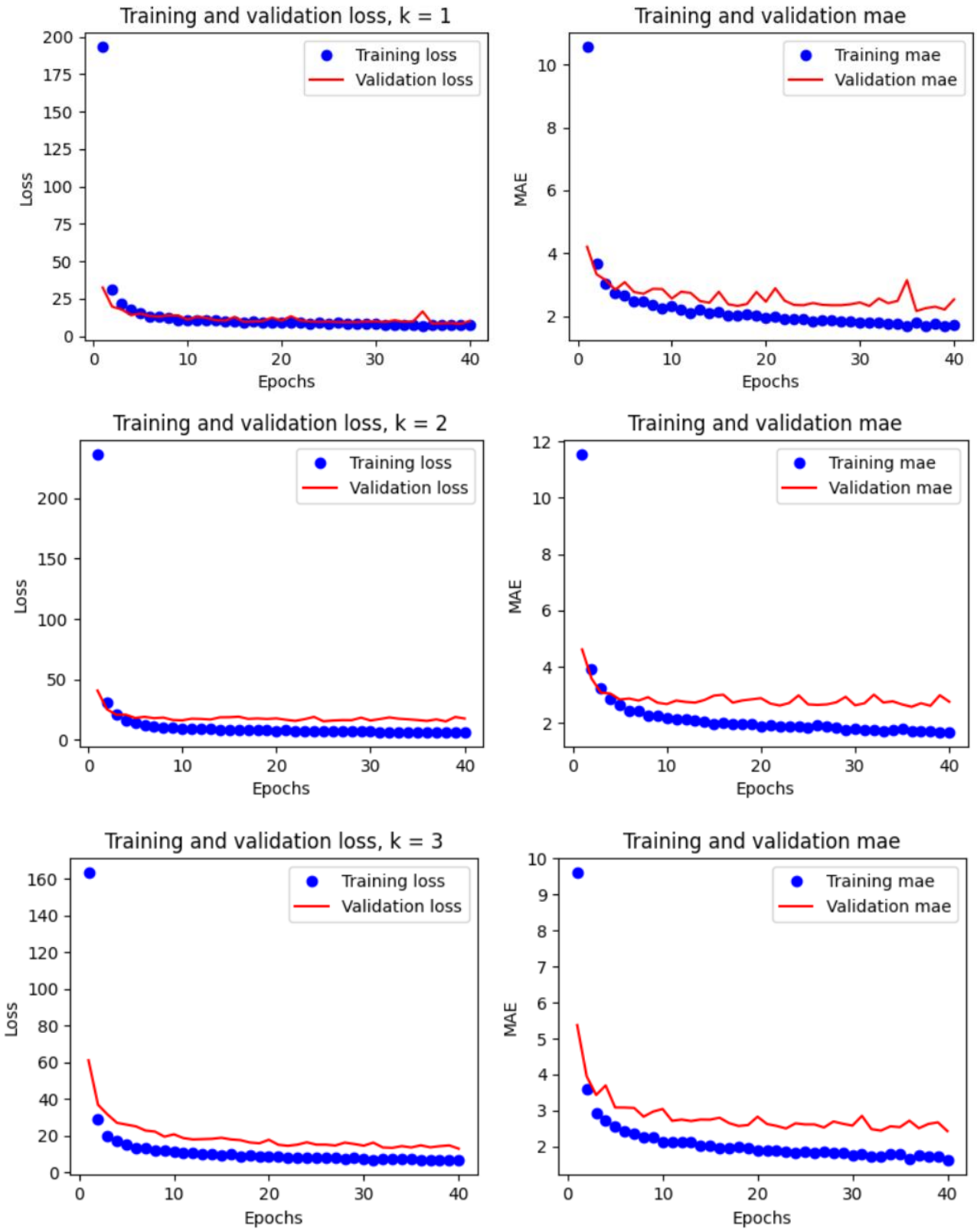


Рисунок 3 – Ошибки модифицированной сети

На рисунке 4 представлены усредненные графики потерь и средней абсолютной ошибки данной сети.

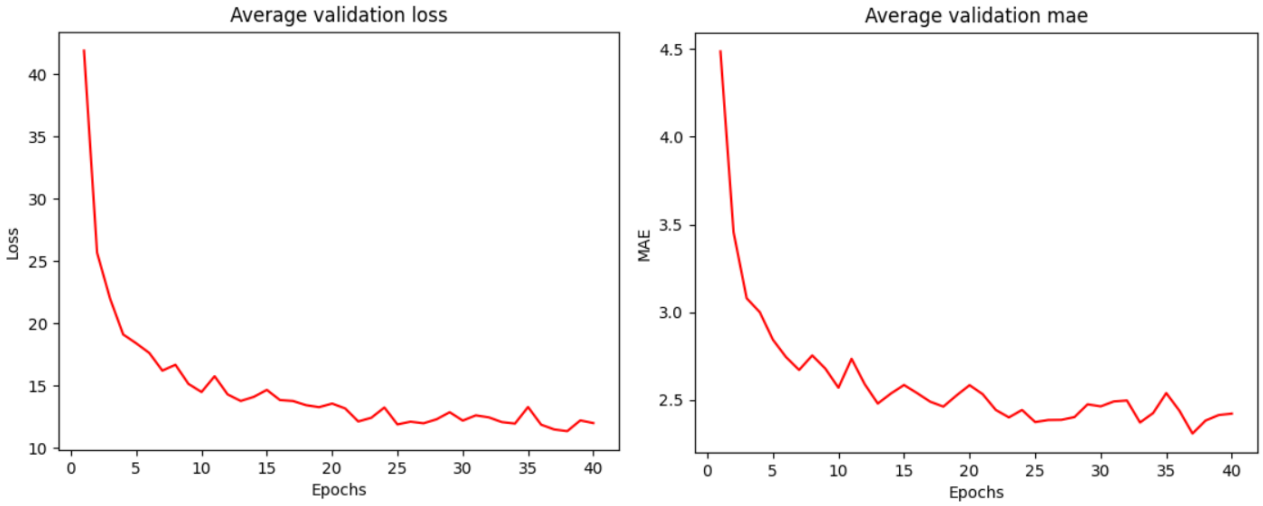


Рисунок 4 – усредненные графики модифицированной сети

Полученная оценка модели – 2.40, что лучше чем в предыдущем рассмотренном варианте, однако результат еще можно улучшить рассмотрев другое количество блоков, на которые делятся данные.

Были рассмотрены варианты конфигураций от 3х до 8ми блоков, полученные результаты: оценка модели с 5ю блоками – 2.3, оценка модели с 6ю блоками – 2.26, оценка модели с 7ю блоками – 2.41, оценка модели с 8ю блоками – 2.36. Однако, при выборе финальной конфигурации стоит учитывать, что чем больше блоков, тем дольше работает сеть, поэтому если бы обучение проходило бы на большем количестве эпох, был бы смысл использовать чуть менее точные, но более быстрые варианты. В данном случае был выбран вариант с 6ю блоками – он дает наивысшую точность при данных условиях.

Полный код программы, реализующий данную нейронную сеть представлен в приложении А.

Выводы

В ходе лабораторной работы была изучена задача регрессии и ее отличие от задачи классификации. Были рассмотрены разные конфигурации сети на одном наборе данных и было изучено влияние количества эпох и блоков в перекрестной проверке.

Была создана нейронная сеть, предсказывающая цены на дома. Итоговая ошибка предсказаний оказалась равна примерно 2260 долларов и была получена сетью с 6ю блоками и обучающаяся в течение 40 эпох, после этого происходит переобучение.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import numpy as np
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.datasets import boston_housing
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 draw_plots(history, i):
    m = 'Training and validation loss, k = ' + str(i)

    loss = history.history['loss']
    val_loss = history.history['val_loss']
    mae = history.history['mae']
    val_mae = history.history['val_mae']
    epochs = range(1, len(mae) + 1)
    fig, ax = plt.subplots(2, 2)

    ax[0][0].plot(epochs, loss, 'bo', label='Training loss')
    ax[0][0].plot(epochs, val_loss, 'r', label='Validation loss')
    ax[0][0].set_title(m)
    ax[0][0].set_xlabel('Epochs')
    ax[0][0].set_ylabel('Loss')
    ax[0][0].legend()

    ax[0][1].plot(epochs, mae, 'bo', label='Training mae')
    ax[0][1].plot(epochs, val_mae, 'r', label='Validation mae')
    ax[0][1].set_title('Training and validation mae')
    ax[0][1].set_xlabel('Epochs')
    ax[0][1].set_ylabel('MAE')
    ax[0][1].legend()
    plt.show()


def draw_last(epochs, avg_loss):
    # потери
    plt.plot(epochs, avg_loss, 'r')
    plt.title('Average validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.show()

    # MAE
    plt.clf()
    plt.plot(epochs, avg_mae, 'r')
    plt.title('Average validation mae')
    plt.xlabel('Epochs')
    plt.ylabel('MAE')
    plt.show()
```

```
(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 = 6
num_val_samples = len(train_data) // k
num_epochs = 40
all_val_loss = []
all_val_mae = []

for i in range(k):
    print('Рассматриваем блок №', 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]

    # Выбрали блок, на оставшихся обучение
    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)

    # Модель
    model = build_model()
    history = model.fit(partial_train_data, partial_train_targets,
validation_data=(val_data, val_targets),
                        epochs=num_epochs, batch_size=1, verbose=0)
    # Графики для текущей модели
    #draw_plots(history, i)
    # Сохранение данные о MAE и ошибке для среднего
    val_loss = history.history["val_loss"]
    val_mae = history.history["val_mae"]
    all_val_mae.append(val_mae)
    all_val_loss.append(val_loss)

# Графики по средним и оценка модели
all_val_mae = np.asarray(all_val_mae)
avg_mae = all_val_mae.mean(axis=0)
all_val_loss = np.asarray(all_val_loss)
avg_loss = all_val_loss.mean(axis=0)
epochs = range(1, len(avg_mae) + 1)

draw_last(epochs, avg_loss)

print("Срдений MAE: ", np.mean(all_val_mae[:, -1]))
```