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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Искусственные нейронные сети»

Тема: «Регрессионная модель изменения цен на дома в Бостоне»

Студент гр. 7382 _____ Находько А.Ю.

Преподаватель _____ Жукова Н.В.

Санкт-Петербург

2020

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 работы.

•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 работы.

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

Основные теоретические положения.

Искусственные нейронные сети - совокупность моделей, которые представляют собой сеть элементов - искусственных нейронов, связанных между собой синаптическими соединениями.

Нейронные сети используются как среда, в которой осуществляется адаптивная настройка параметров дискриминантных функций. Настройка происходит при последовательном предъявлении обучающих выборок образов из разных классов. Обучение - такой выбор параметров нейронной сети, при котором сеть лучше всего справляется с поставленной проблемой.

Нейрон — элемент, преобразующий входной сигнал по функции.

Сумматор — элемент, осуществляющий суммирование сигналов поступающих на его вход.

Синапс — элемент, осуществляющий линейную передачу сигнала.

Экспериментальные результаты.

Задача классификации — определение отношений объектов из заданного множества к классам из некоторого конечного множества классов. Т.о. перед нами стоит задача классификации объекта, т. е. Отнести его к соответствующему классу.

Задача регрессии — определеить значение какой-либо характеристики объекта, значением параметра является не конечное множество классов, а множество действительных чисел.

В ходе проведения лабораторной работы была построена модель нейронной сети, код которой представлен в приложении А.

Была организована работа с моделью с перекрёстной проверкой по K блокам, при различных значениях K, также в отчёте представлены графики для каждого блока и график средних значений mae и график оценки mae для каждого из блоков.

Изучим влияние кол-ва эпох на результат обучения модели, первоначальное значение эпох будет равно 100, K=4.

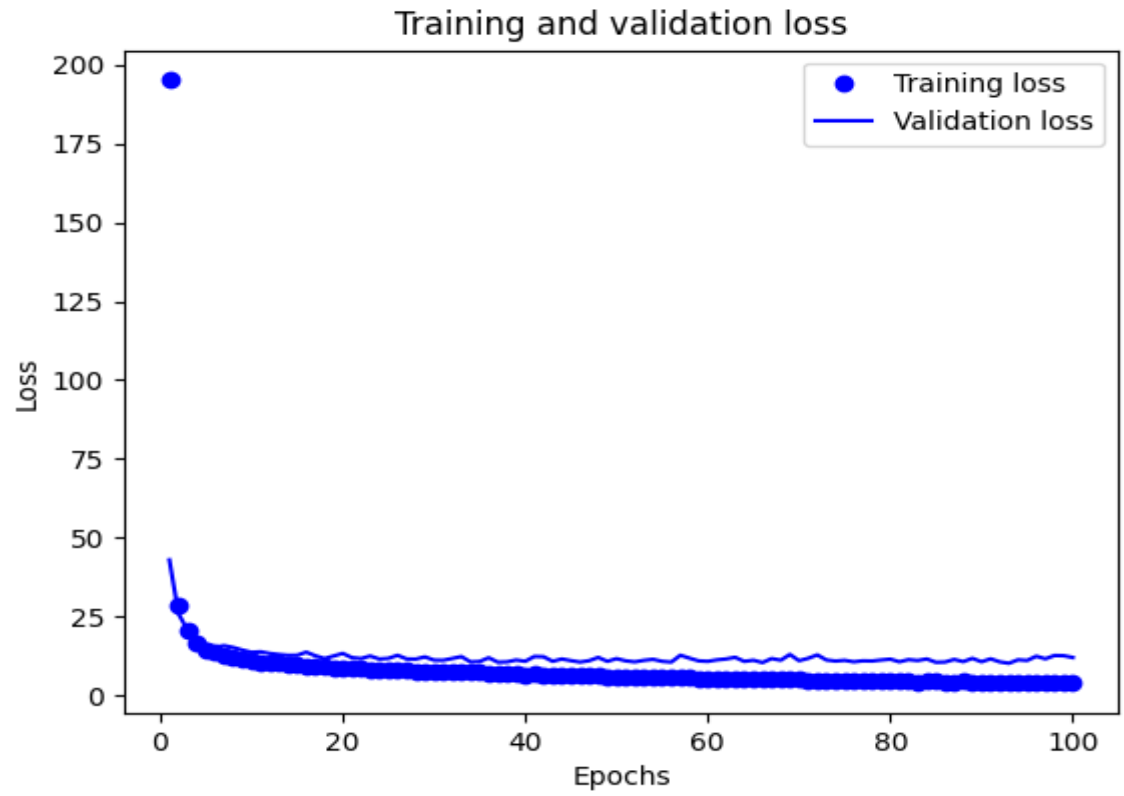

Рисунок 1 — Средний график ошибки модели при 100 эпохах

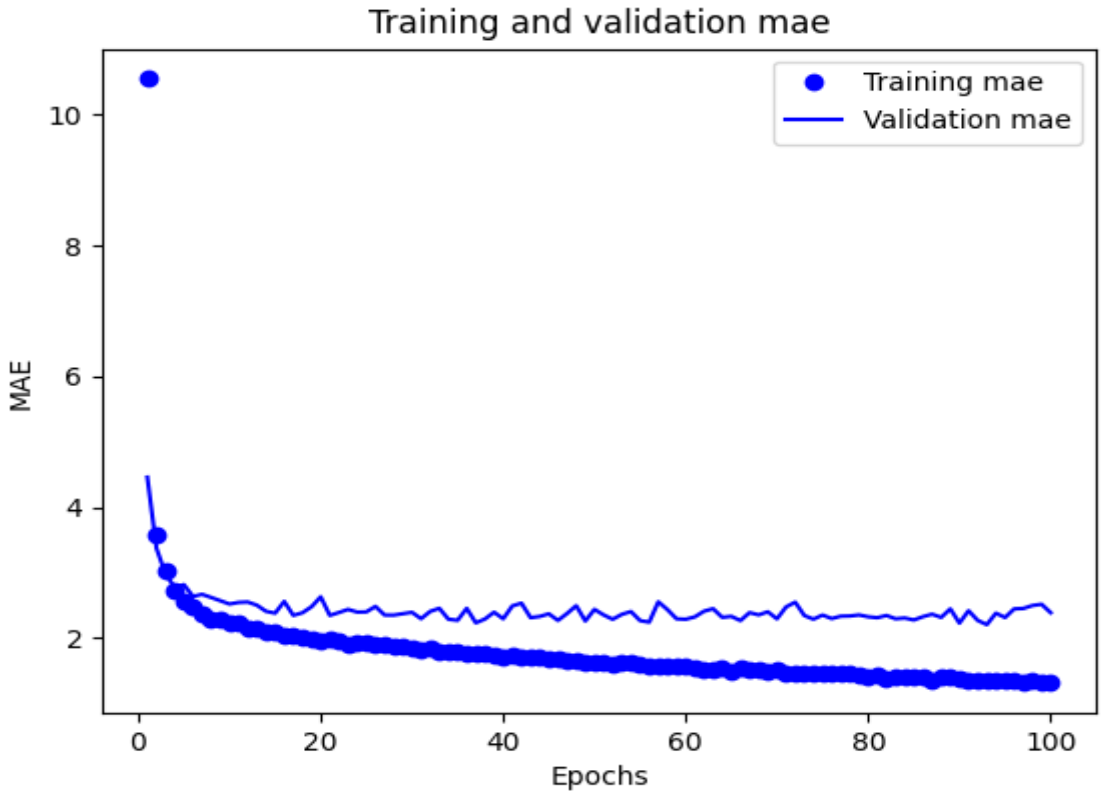

Рисунок 2 — Средний график точности модели при 100 эпохах

Проанализировав графики заметим, что примерно на 40 эпохе происходит разрыв между графиками, придём к выводу что это переобучение НС. Поэтому уменьшим количество эпох до 40.

Для применения перекрестной проверки по K блокам, будем изменять значения K.

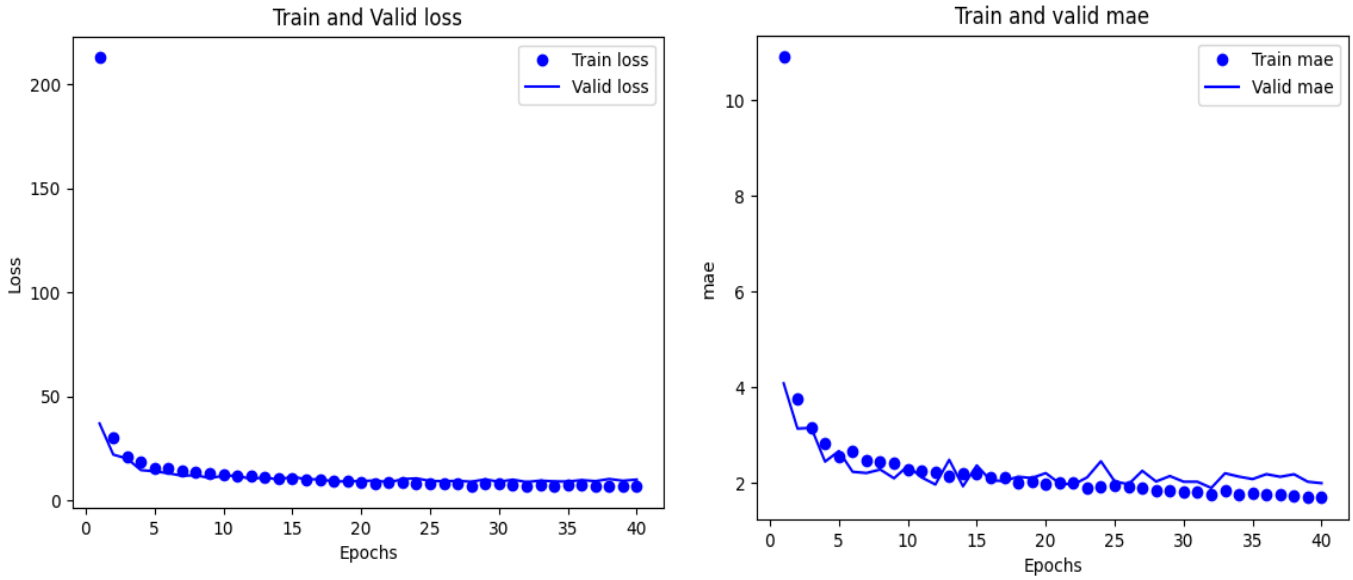

Рисунок 3 — графики для первого блока

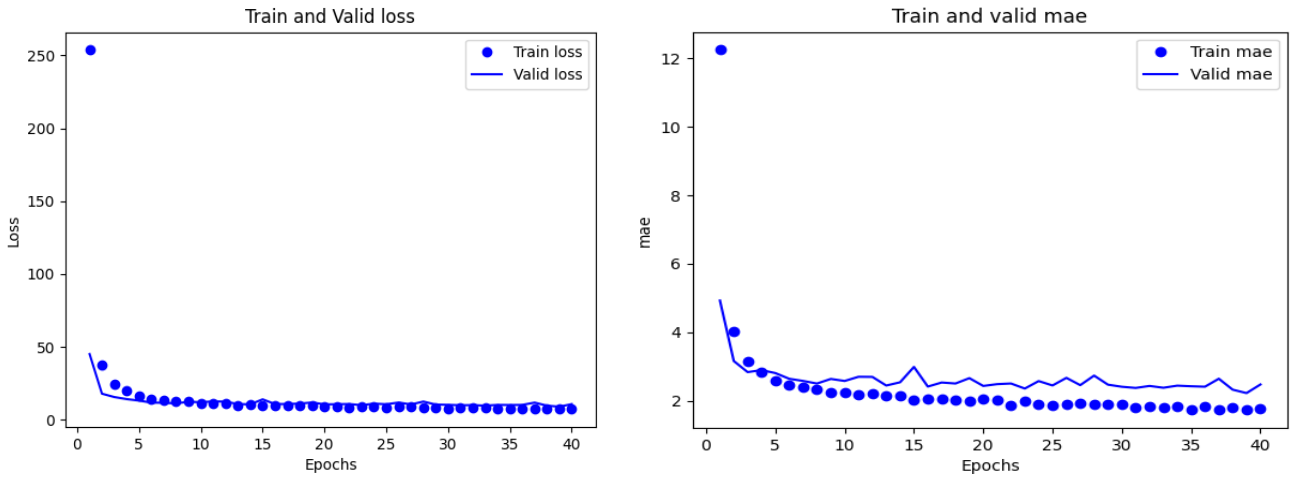

Рисунок 4 — графики для второго блока

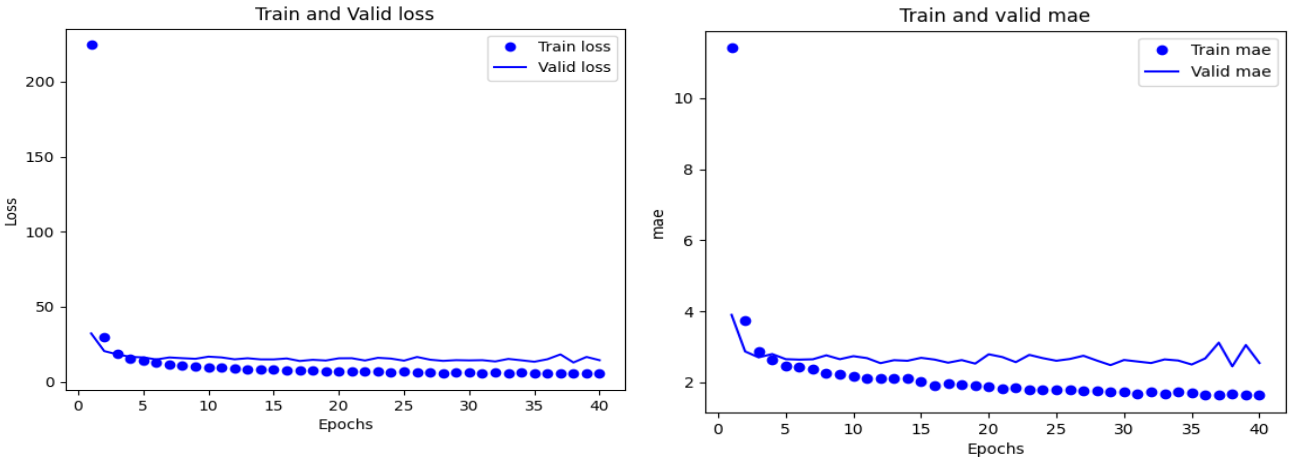

Рисунок 5 — графики для третьего блока

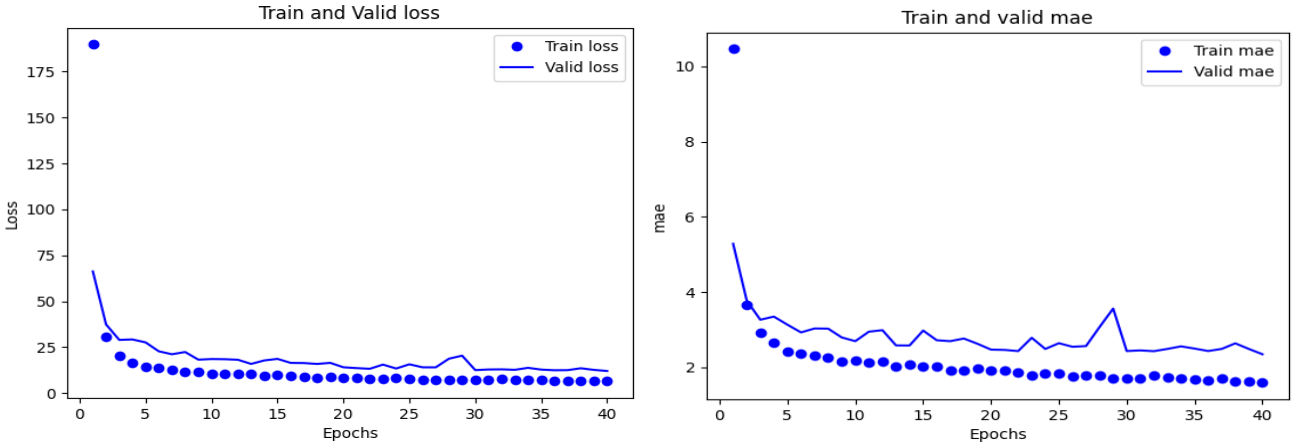

Рисунок 6 — графики для четвертого блока

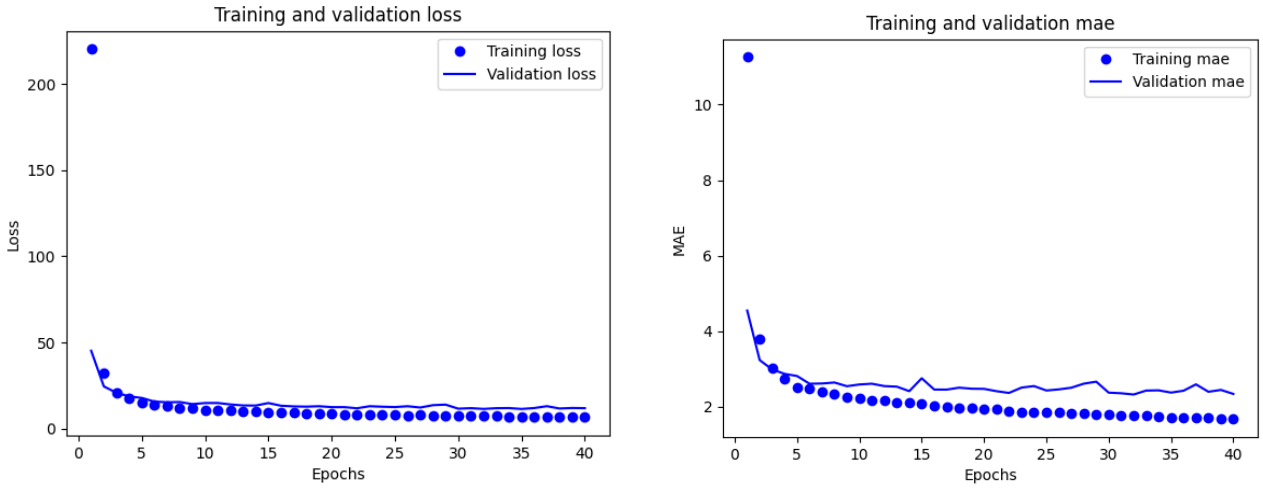

Рисунок 7 — графики средних значений для блоков при 40 эпохах

Для применения перекрестной проверки по K блокам, будем изменять значения K.

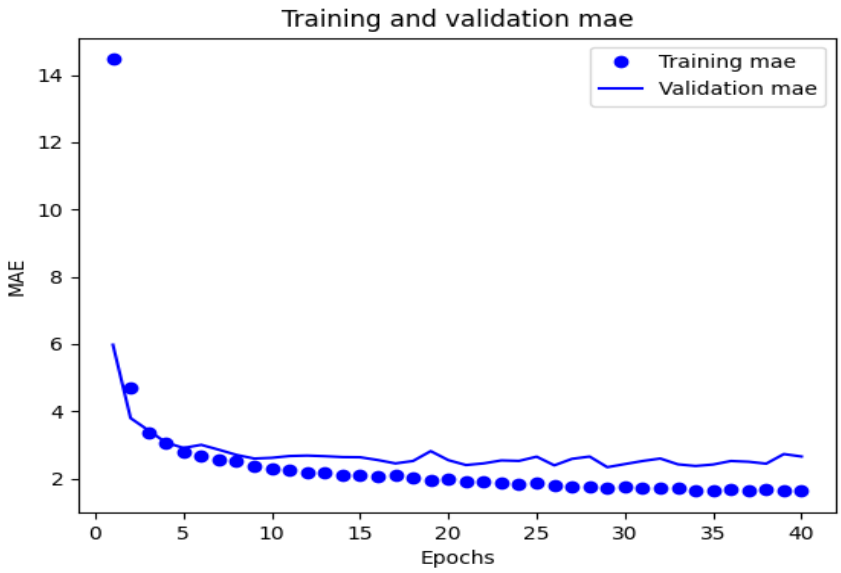

Рисунок 8 — усредненный график среднеквадратичной ошибки при K=2

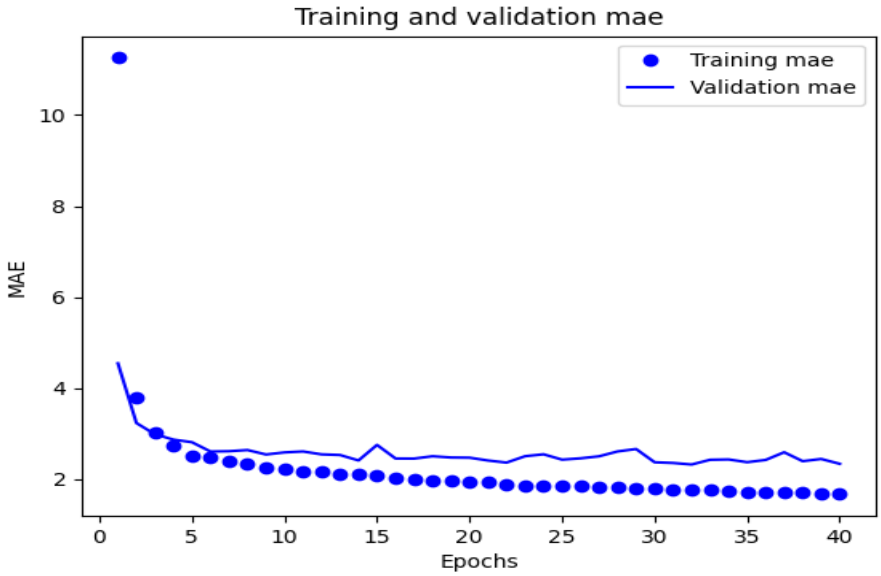

Рисунок 9 — усредненный график среднеквадратичной ошибки при K=4

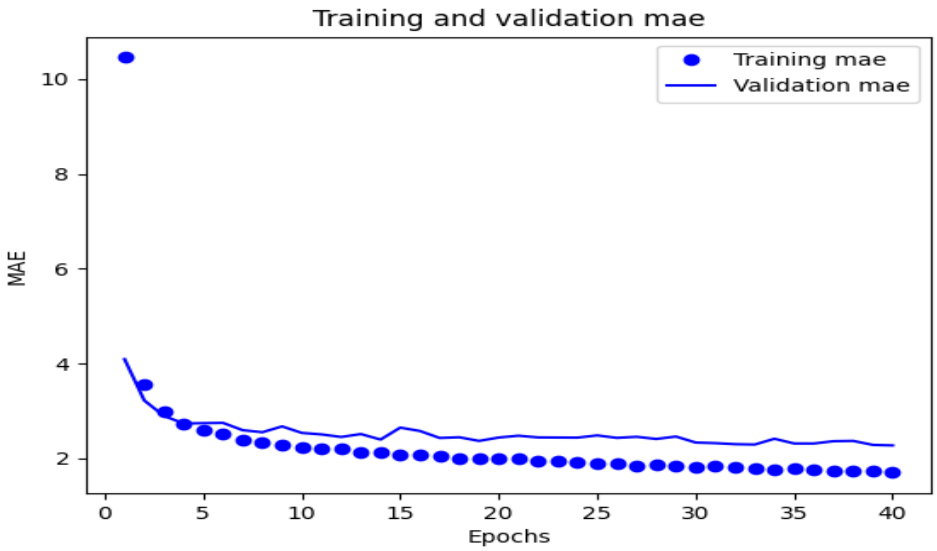

Рисунок 10 — усредненный график среднеквадратичной ошибки при K=6

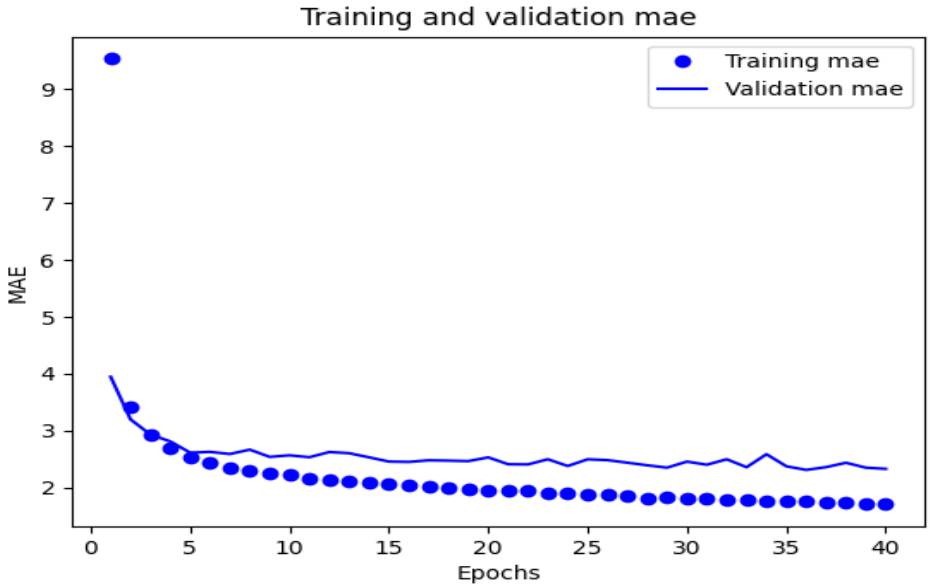

Рисунок 10 — усредненный график среднеквадратичной ошибки при K=8

Проанализировав полученные значения, установим что наименьшее среднеквадратичное отклонение получается при K=6.

Выводы.

В ходе выполнения лабораторной работы было реализовано предсказание медианной цены на дома в пригороде Бостона. Также ознакомился с задачей регрессии, изучил отличие задачи регрессии от задачи классификации,

ознакомился с перекрёстной проверкой. Модель была создана и обучена корректна, в ходе работы были настроены параметры обучения.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
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)
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 = 6
num_val_samples = len(train_data) // k
num_epochs = 40
all_scores = []
mae_histories = []
loss_array = []
```

```
val_loss_array = []
mae_array = []
val_mae_array = []

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))
mae_array.append(H.history['mae'])
val_mae_array.append(H.history['val_mae'])
loss_array.append(H.history['loss'])
val_loss_array.append(H.history['val_loss'])

print(H.history.keys())
loss = H.history['loss']
val_loss = H.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'bo', label='Train loss')
plt.plot(epochs, val_loss, 'b', label='Valid loss')
plt.title('Train and Valid loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.clf()
```

```
mae = H.history['mae']
val_mae_graph = H.history['val_mae']
plt.plot(epochs, mae, 'bo', label='Train mae')
plt.plot(epochs, val_mae_graph, 'b', label='Valid mae')
plt.title('Train and valid mae')
plt.xlabel('Epochs')
plt.ylabel('mae')
plt.legend()
plt.show()

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

epochs = range(1, num_epochs + 1)
plt.plot(epochs, np.mean(loss_array, axis=0), 'bo', label='Training
loss')
plt.plot(epochs, np.mean(val_loss_array, axis=0), 'b', label='Validation
loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.clf()
plt.plot(epochs, np.mean(mae_array, axis=0), 'bo', label='Training mae')
plt.plot(epochs, np.mean(val_mae_array, axis=0), 'b', label='Validation
mae')
plt.title('Training and validation mae')
plt.xlabel('Epochs')
plt.ylabel('MAE')
plt.legend()
plt.show()

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