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

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине «Искусственные нейронные сети»

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

Студент гр. 7381 _____ Павлов А.П.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург
2020

Цели.

Реализовать предсказание медианной цены на дома в пригороде бостона в середине 1970-ых по таким данным, как уровень преступности, ставка местного имущественного налога и т.д.

Задачи.

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

Выполнение работы.

Задача классификации определяет принадлежность объекта, описанного входными данными, к одному из заданных классов, а задача регрессии определяет значение какой-либо характеристики объекта, в зависимости от характеристик объекта, подаваемых на вход. В задаче классификации результатом будет значение из конечного множества значений, а результатом задачи регрессии может быть любое число.

1. Была создана и обучена модель искусственной нейронной сети для нахождения оптимального числа эпох (код представлен в приложении А). Количество блоков было выбрано равным 4, а количество эпох – 150.

   Результат обучения нейронной сети представлен на графике на рис. 1.

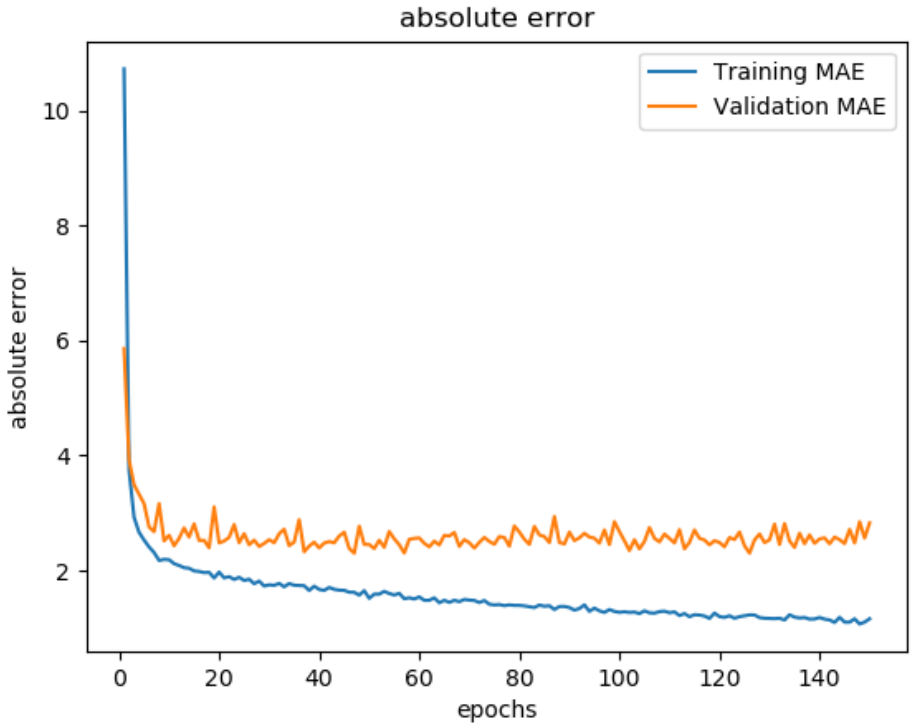

Рисунок 1 – нахождение оптимального числа эпох

По рис. 1 видно, что ошибка на проверочных данных уменьшается до 40-50 эпох обучения, после она либо не меняется, либо становится больше при уменьшении ошибки на тестовых данных. Это говорит о переобучении модели, поэтому оптимальным значением числа эпох будет 45.

2. Было проведено тестирование обучения модели на изменяющемся числе блоков, на которые делятся данные. Значения числа блоков были взятыми 4, 6 и 8. Промежуточные результаты для 4 блоков представлены на рис. 2-5.

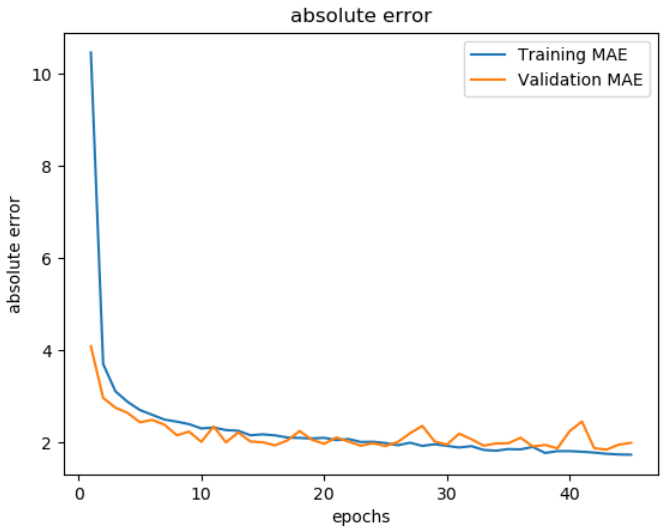

Рисунок 2 – ошибка для блока 1

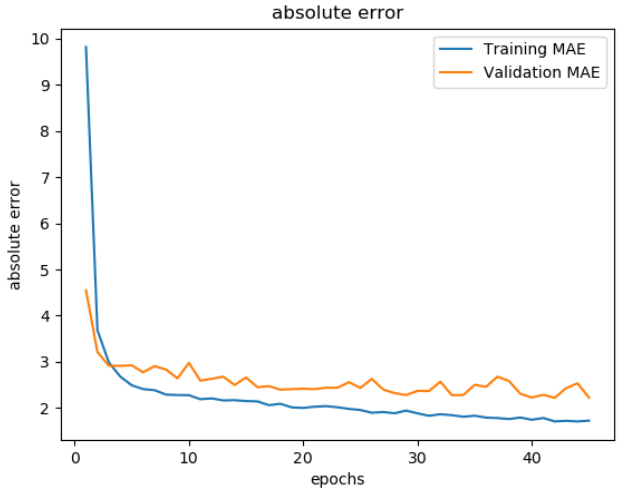

Рисунок 3 – ошибка для блока 2

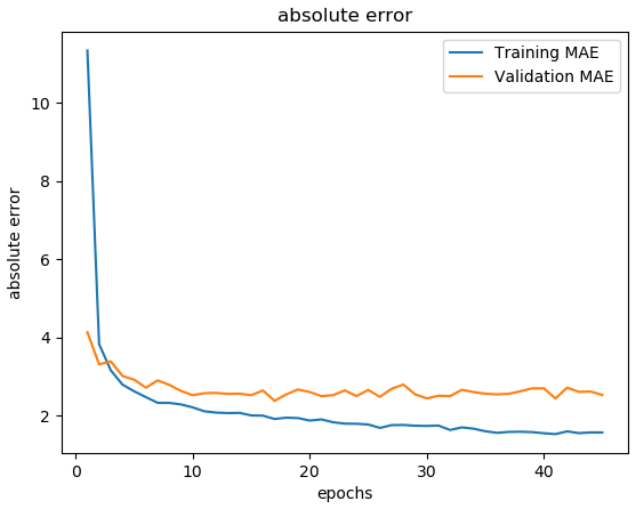

Рисунок 4 – ошибка для блока 3

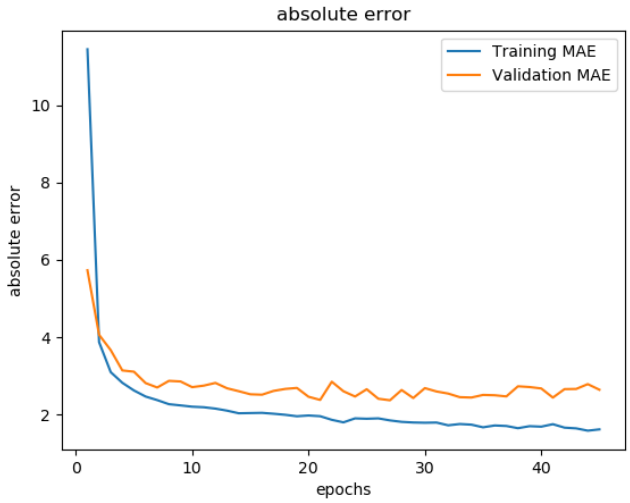

Рисунок 5 – ошибка для блока 4

График среднего значения ошибки для 4 блоков представлен на рис. 6.

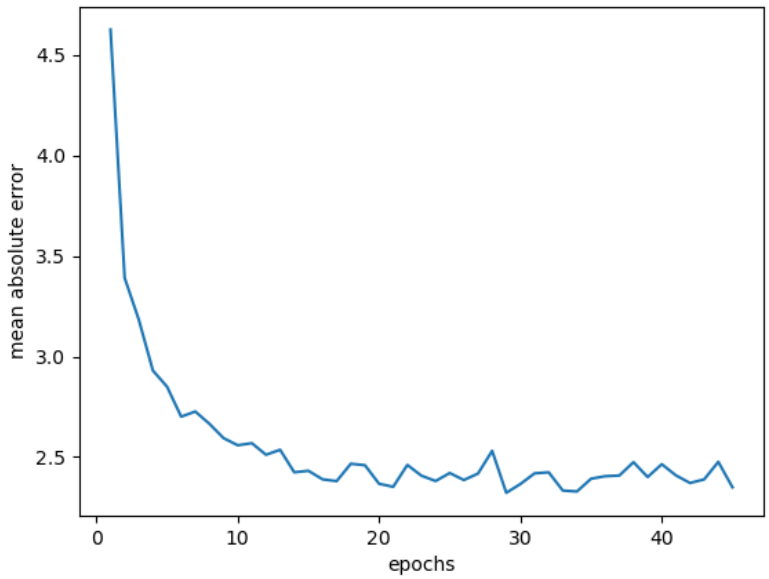

Рисунок 6 – средняя ошибка для 4 блоков

Промежуточные результаты для 6 блоков представлены на рис. 7-12.

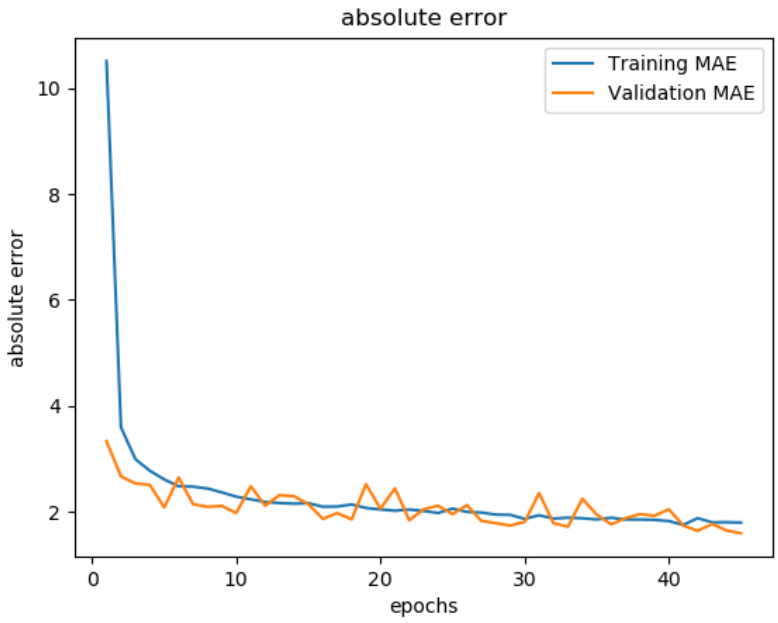

Рисунок 7 – ошибка для блока 1

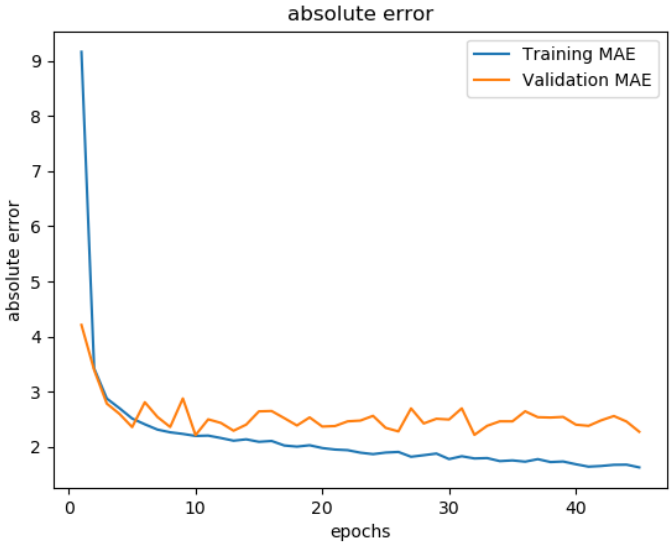

Рисунок 8 – ошибка для блока 2

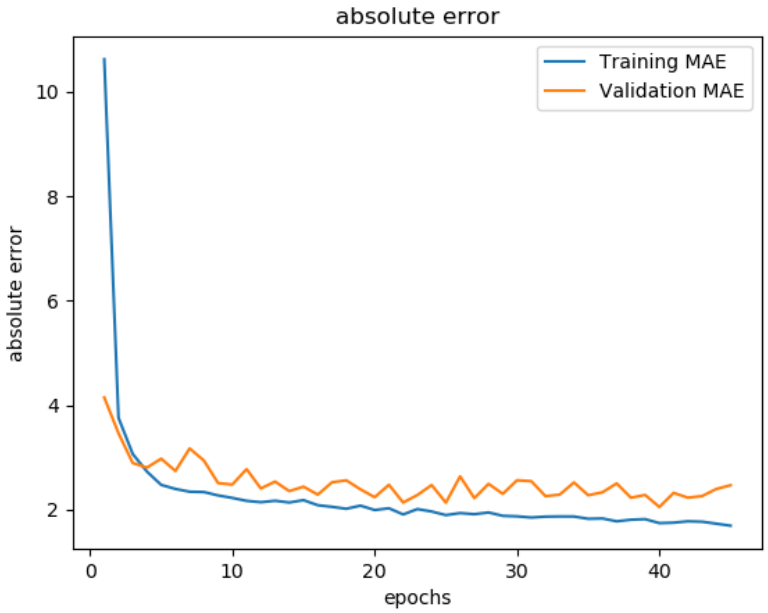

Рисунок 9 – ошибка для блока 3

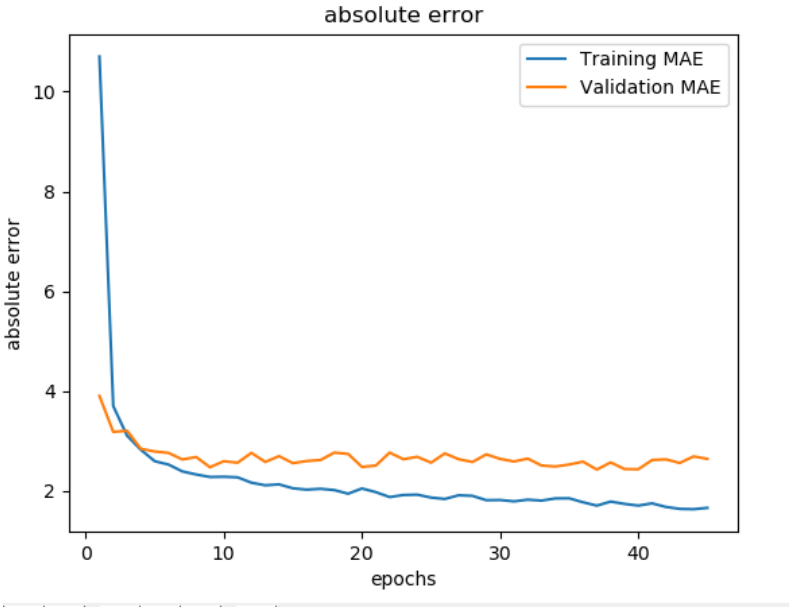

Рисунок 10 – ошибка для блока 4

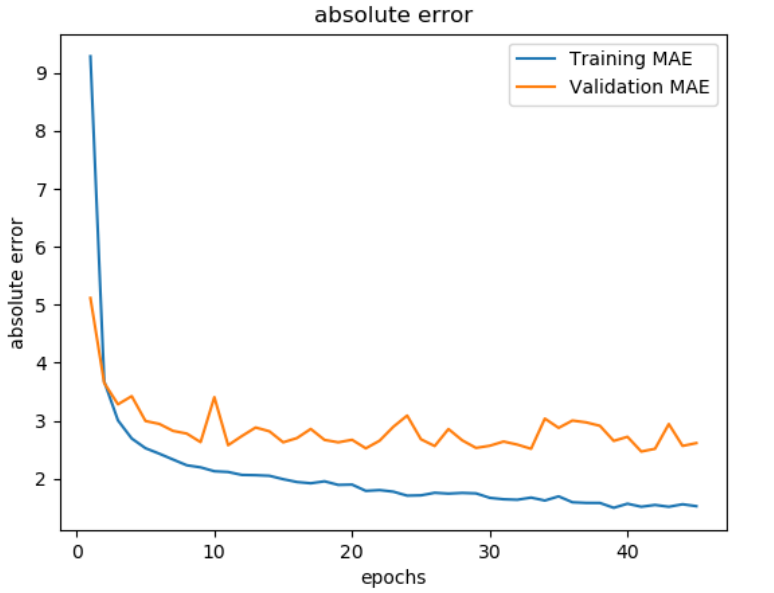

Рисунок 11 – ошибка для блока 5

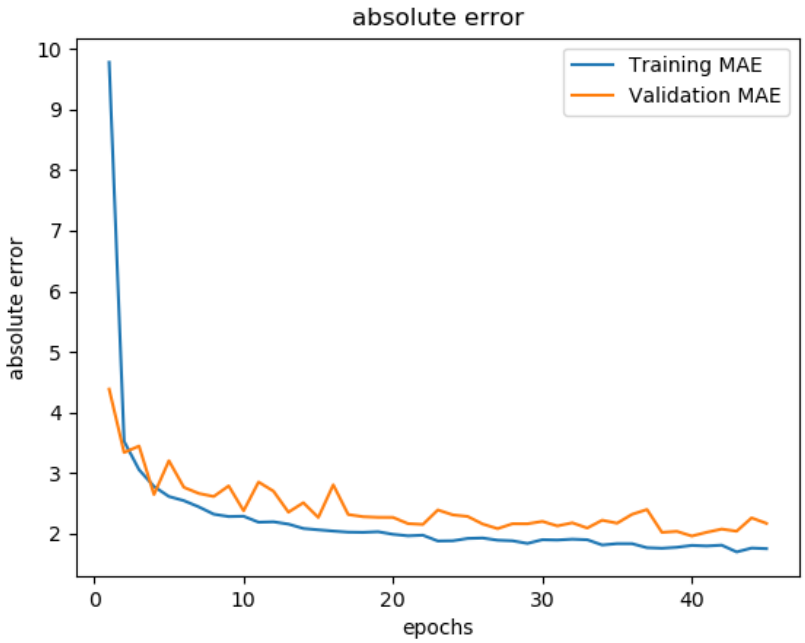

Рисунок 12 – ошибка для блока 6

График среднего значения ошибки для 6 блоков представлен на рис. 13.

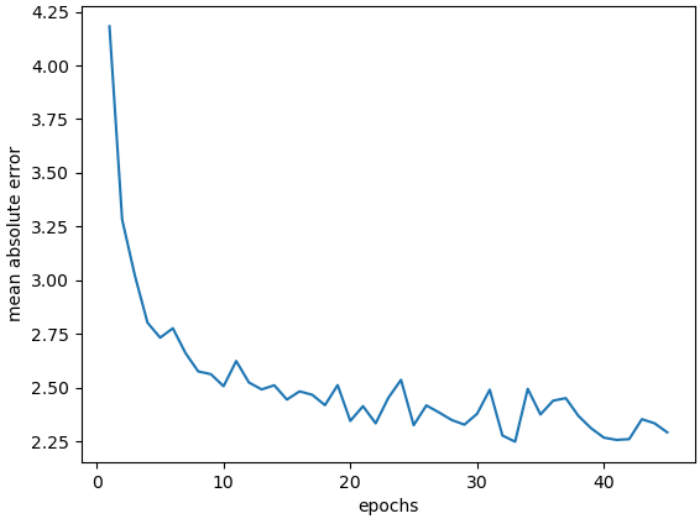

Рисунок 13 – средняя ошибка для 6 блоков

Промежуточные результаты для 8 блоков представлены на рис. 14-21

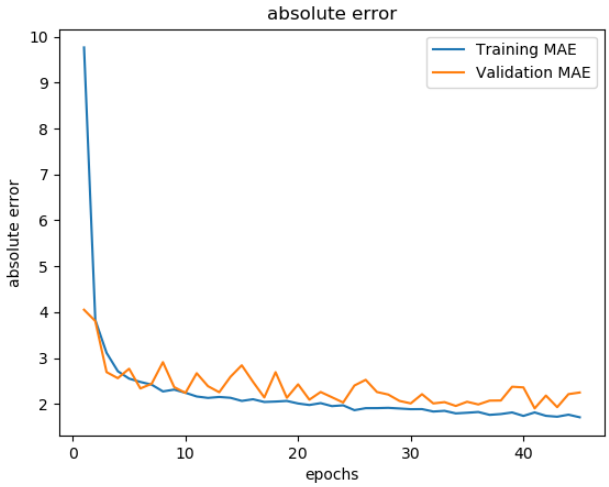

Рисунок 14 – ошибка для блока 1

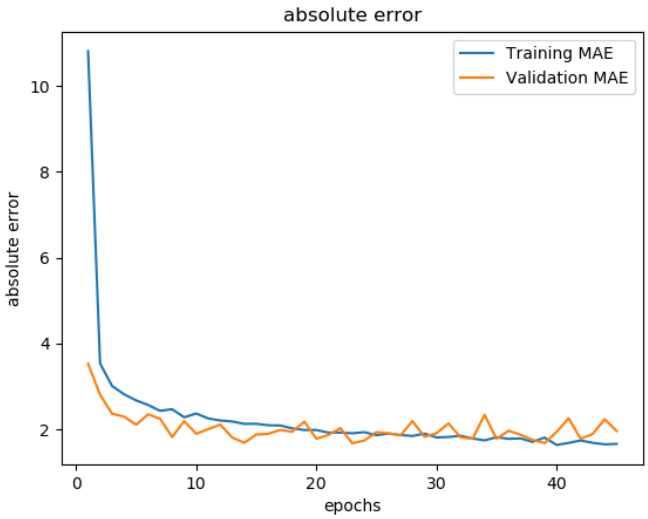

Рисунок 15 –ошибка для блока 2

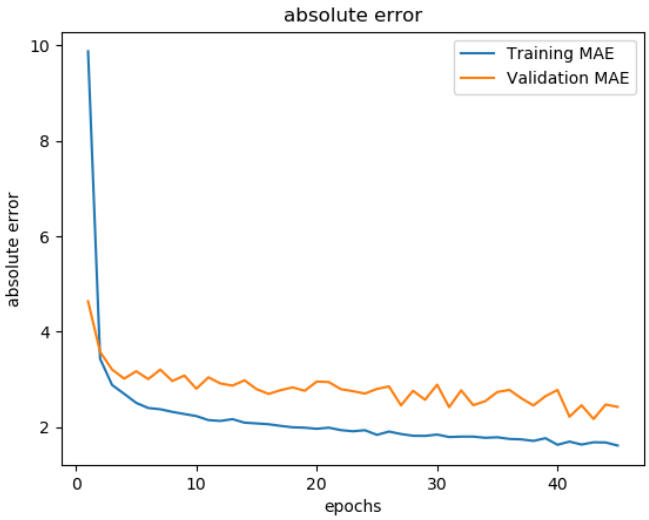

Рисунок 16 –ошибка для блока 3

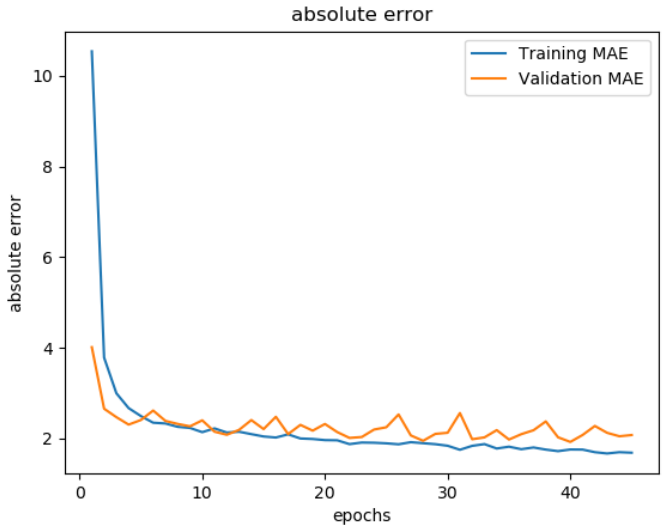

Рисунок 17 –ошибка для блока 4

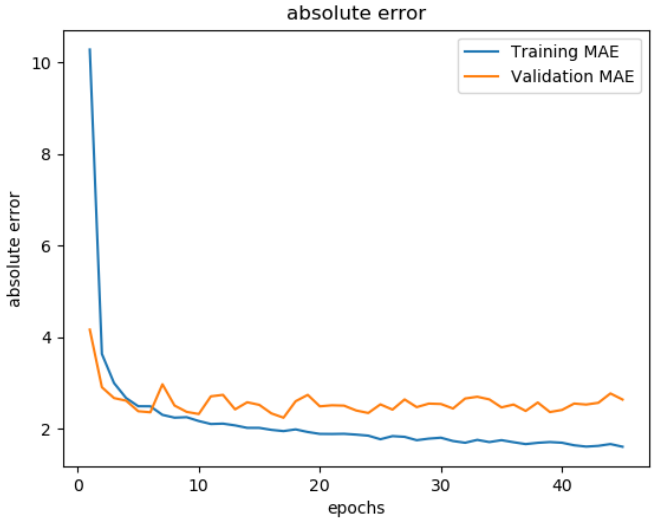

Рисунок 18 –ошибка для блока 5

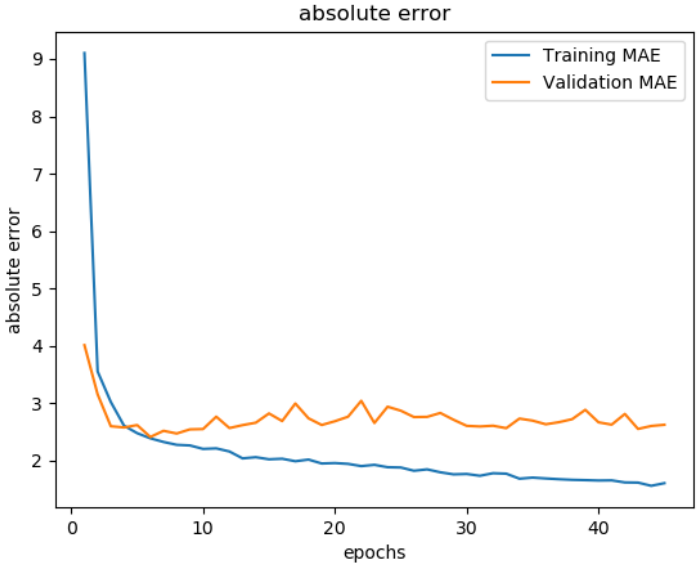

Рисунок 19 –ошибка для блока 6

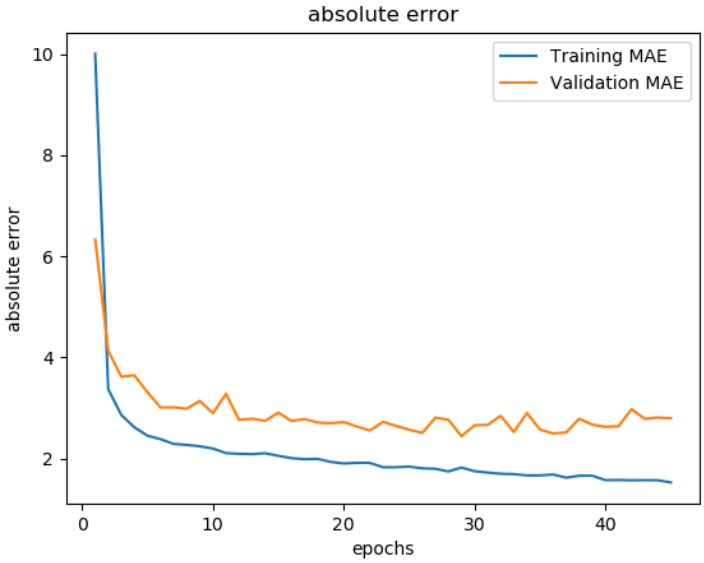

Рисунок 20 –ошибка для блока 7

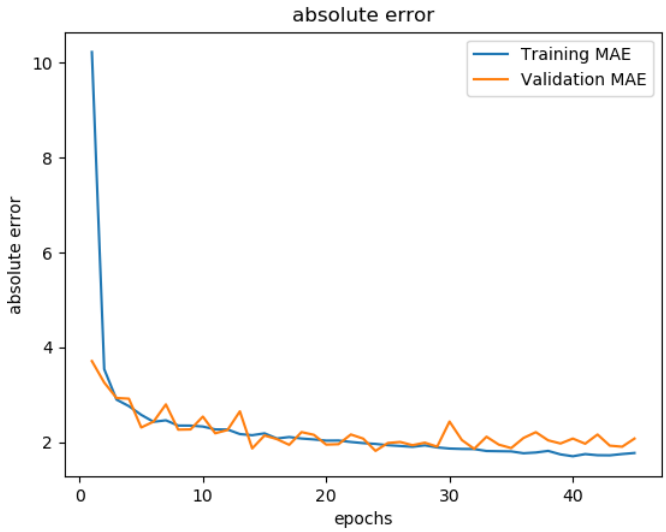

Рисунок 21 –ошибка для блока 8

График среднего значение ошибки для 8 блоков представлен на рис. 22.

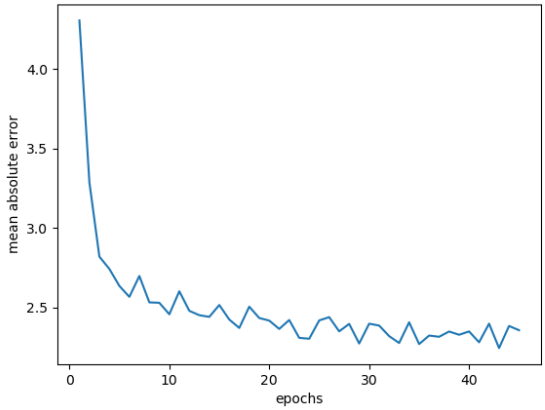

Рисунок 22 – средняя ошибка для 8 блоков

По вышеприведенным графикам видно, что наименьшая ошибка наблюдается в модели, использующей 6 блоков.

Вывод.

В ходе выполнения данной работы было изучено влияние количества эпох и количества блоков в перекрестной проверке на результат обучения модели искусственной нейронной сети, решающей задачу регрессии.

Приложения

Приложение А

```
from keras.layers import Dense
from keras.models import Sequential
from keras.datasets import boston_housing
import numpy as np
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


(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 = 6
num_val_samples = len(train_data) // k
num_epochs = 45
mae_histories = []
for i in range(k):
    print(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_target = np.concatenate([train_targets[: i *
num_val_samples],
```

```
                                                train_targets[(i +
1) * num_val_samples:]], axis=0)
    model = build_model()
    history = model.fit(partial_train_data,
partial_train_target, epochs=num_epochs, batch_size=1,
                        validation_data=(val_data,
val_targets), verbose=0)
    mae = history.history['mae']
    v_mae = history.history['val_mae']
    x = range(1, num_epochs + 1)
    mae_histories.append(v_mae)
    plt.figure(i + 1)
    plt.plot(x, mae, label='Training MAE')
    plt.plot(x, v_mae, label='Validation MAE')
    plt.title('absolute error')
    plt.ylabel('absolute error')
    plt.xlabel('epochs')
    plt.legend()
    plt.show()

average_mae_history = [np.mean([x[i] for x in mae_histories])
for i in range(num_epochs)]
plt.figure(0)
plt.plot(range(1, num_epochs + 1), average_mae_history)
plt.xlabel('epochs')
plt.ylabel("mean absolute error")
plt.show()
```