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 _____ Дорох С.В.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цели.

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.

Задачи.

- Ознакомиться с задачей регрессии
- Изучить отличие задачи регрессии от задачи классификации
- Создать модель
- 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.

Была создана и обучена модель искусственной нейронной сети в соответствии с условиями(весь код представлен в приложении А). Задача регрессии отличается от задач классификации тем, что результатом предсказания является не одно из дискретных значений, а значение из некоторого промежутка, числовой прямой.

Для проверки влияния количества эпох на результат обучения модели был выбран диапазон от 50 до 150 с шагом 25. Для выбора K наиболее подходящего для обучения модели были выбраны диапазоны K

от 4 до 7. На ниже представленных рисунках можно увидеть влияние количества эпох на обучения при различных K.

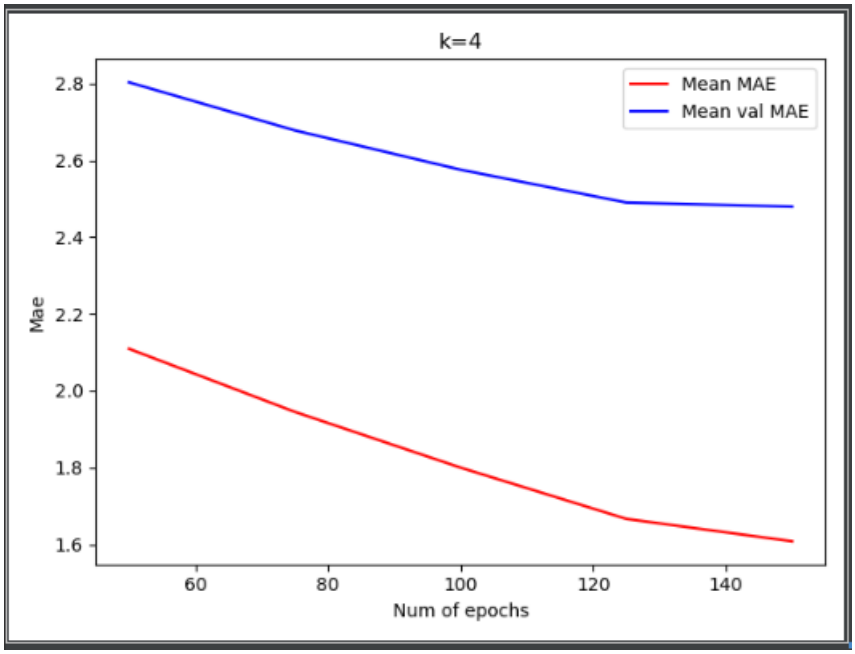


рисунок 1 – график зависимости средней абсолютной ошибки от кол-ва эпох обучения при K = 4

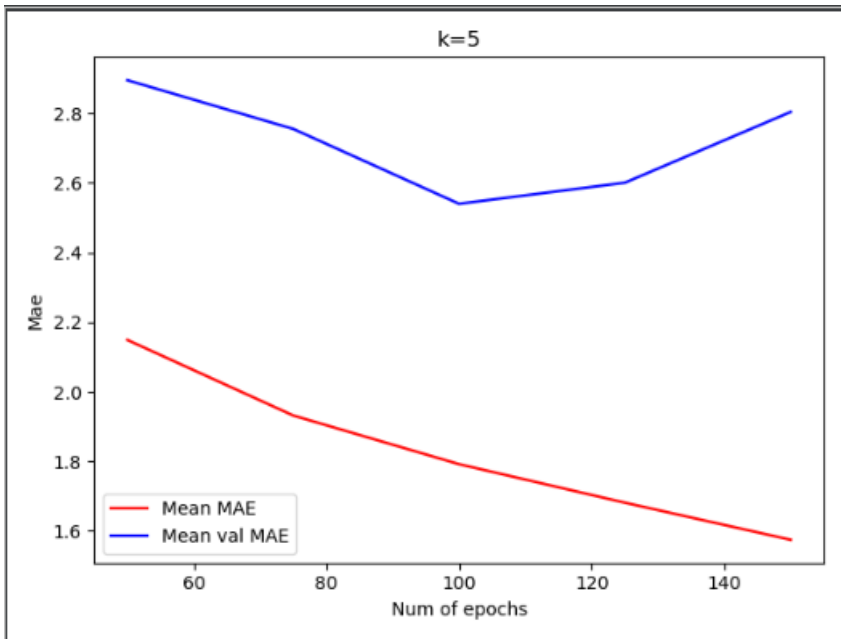


рисунок 2 – график зависимости средней абсолютной ошибки от кол-ва эпох обучения при K = 5

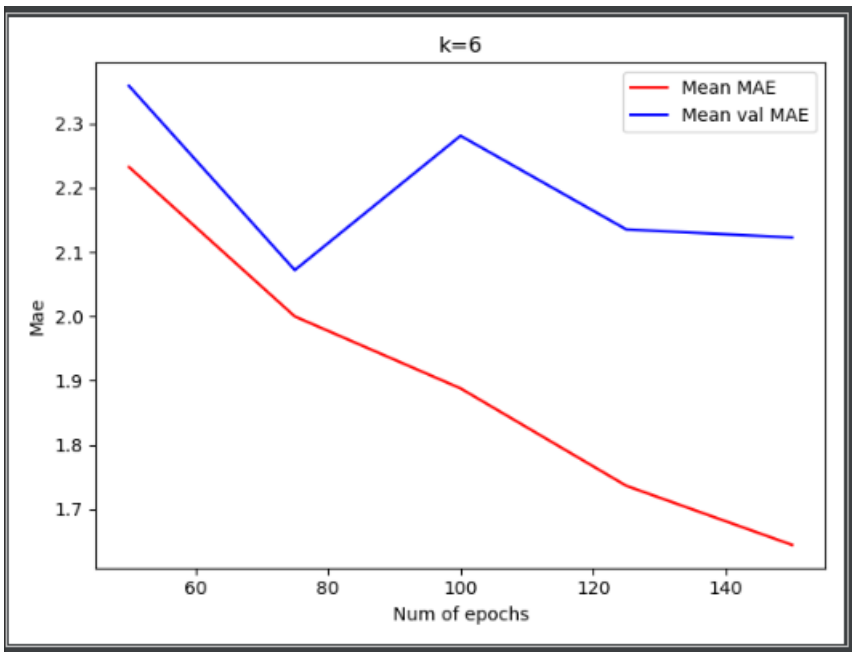


рисунок 3 – график зависимости средней абсолютной ошибки от кол-ва эпох обучения при К = 6

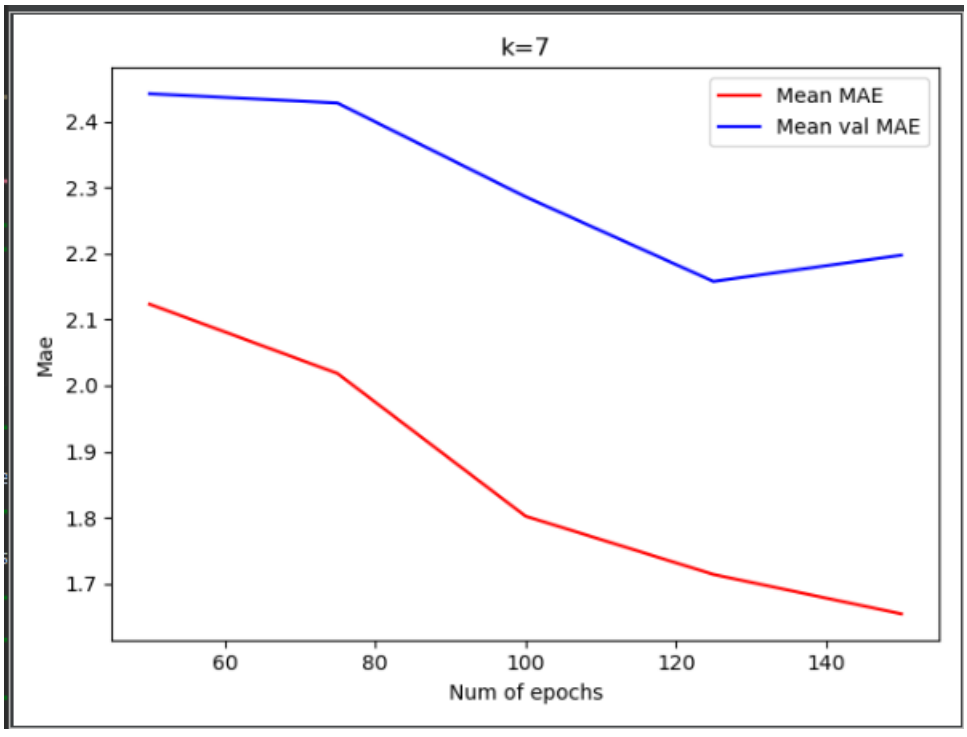
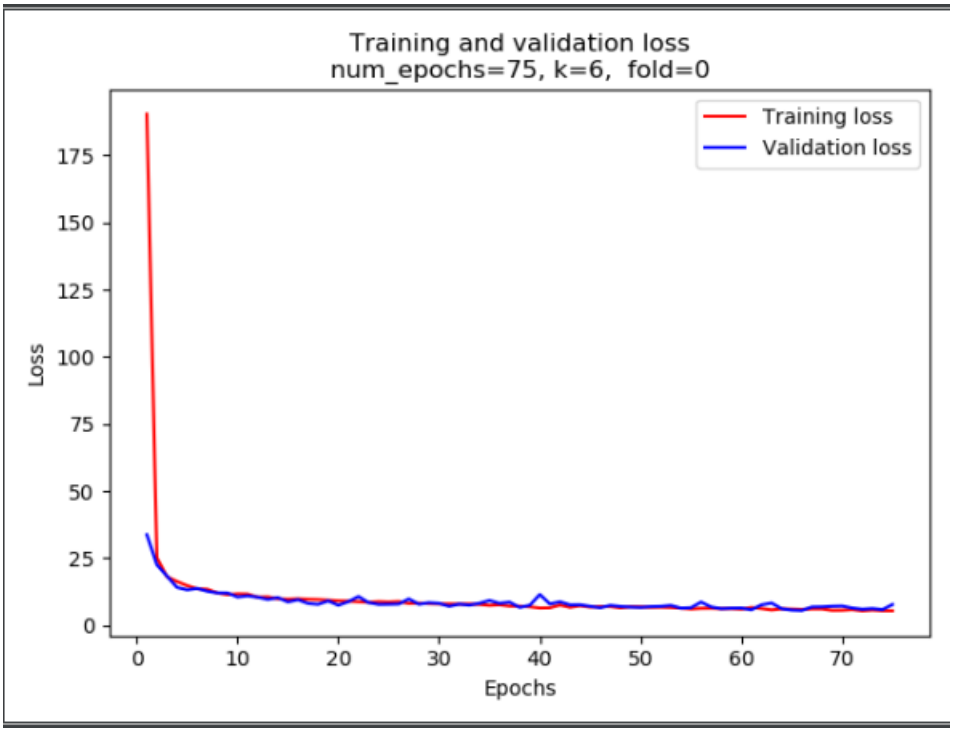


рисунок 4 – график зависимости средней абсолютной ошибки от кол-ва эпох обучения при К = 7

Как видно из графиков, оптимальным числом эпох является число 75, а К = 6.

Были построены графики точности и ошибок обучения модели с параметрами: количество эпох обучения - 75, количество блоков – 6.



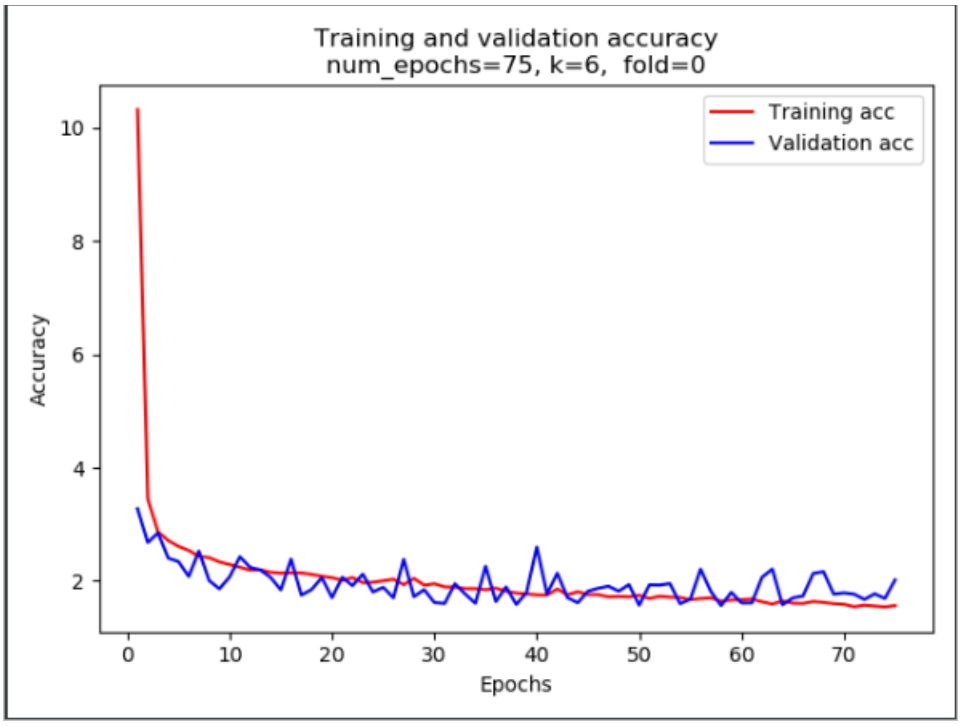
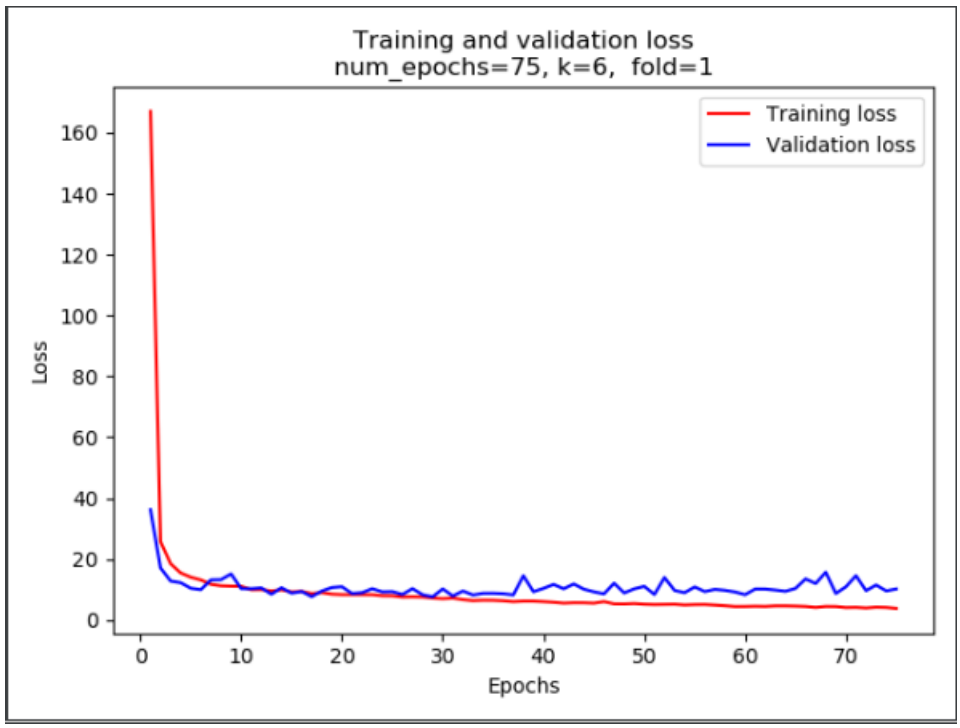


рисунок 5,6 – графики точности и ошибки модели на 1-ом блоке



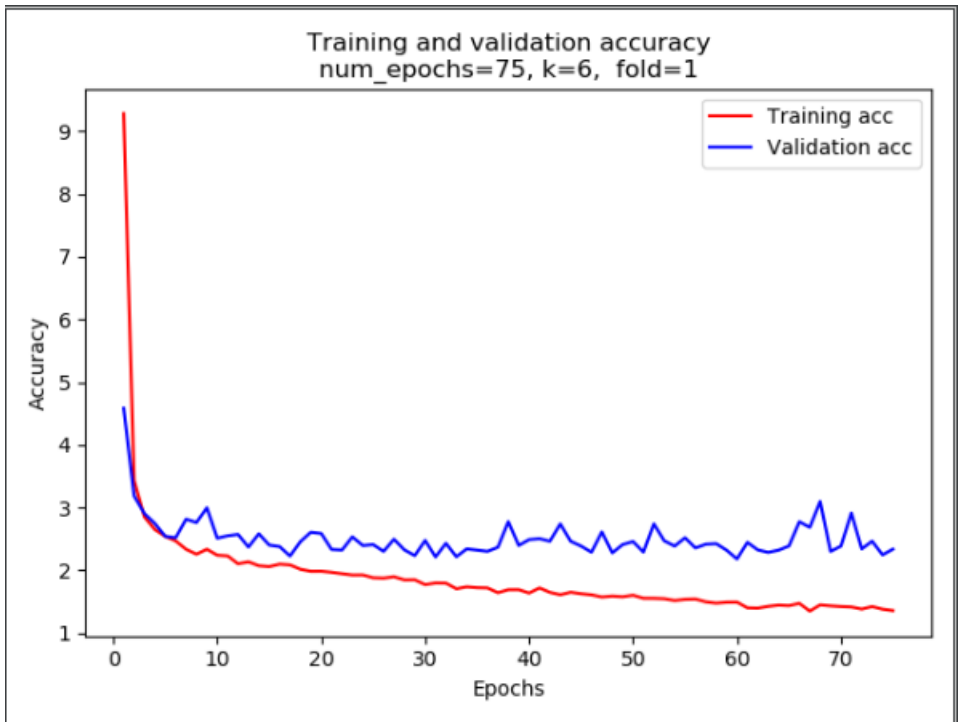
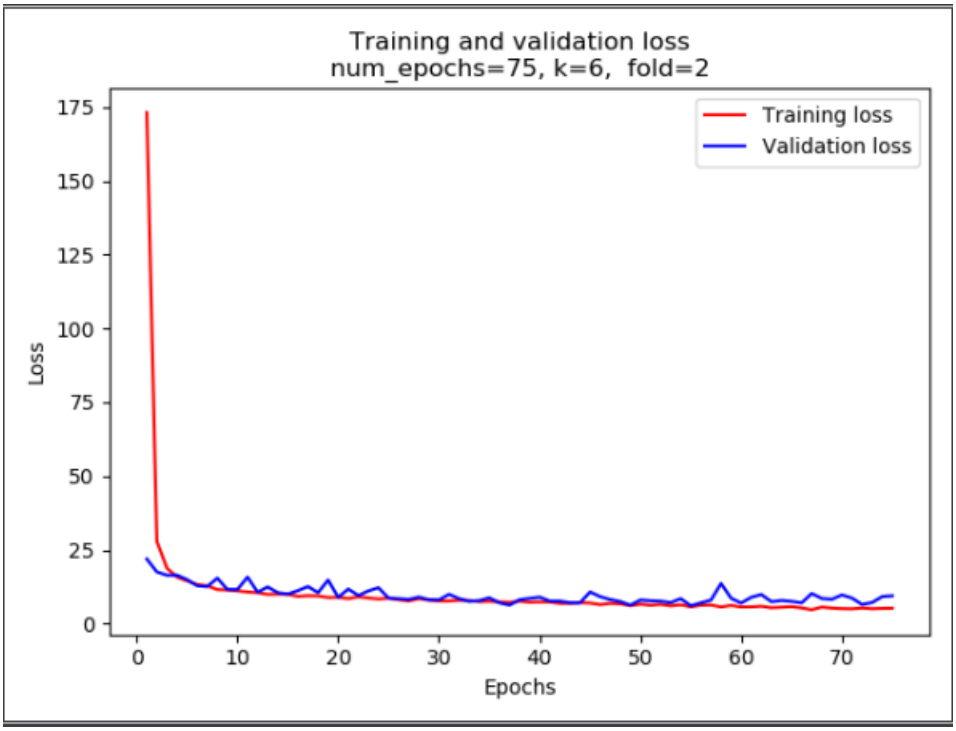


рисунок 7,8 – графики точности и ошибки модели на 2-ом блоке



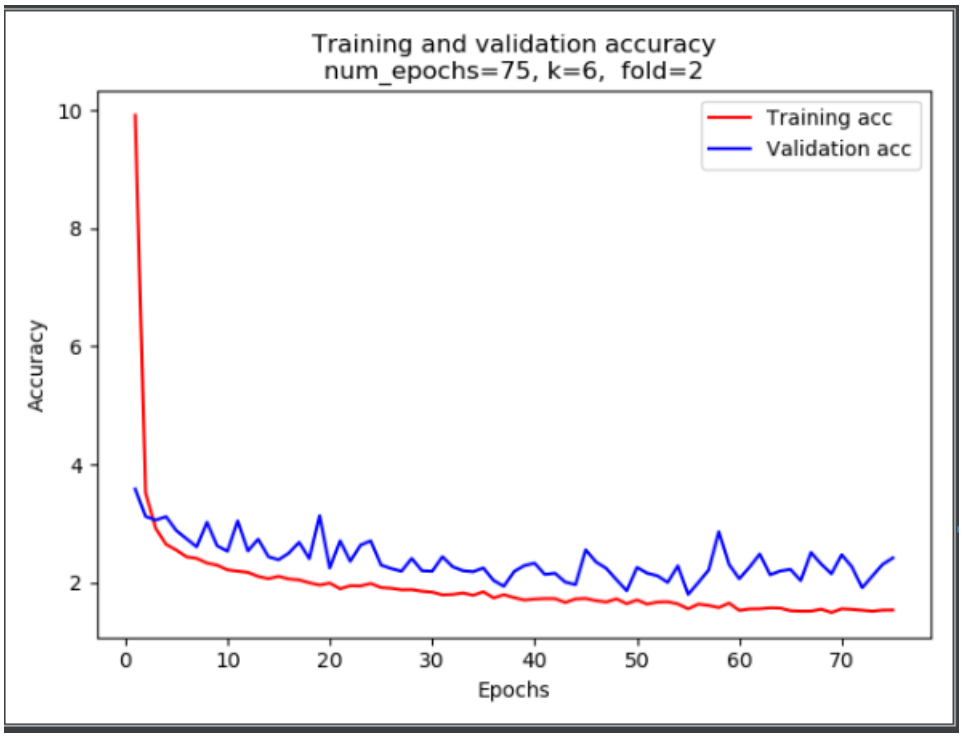
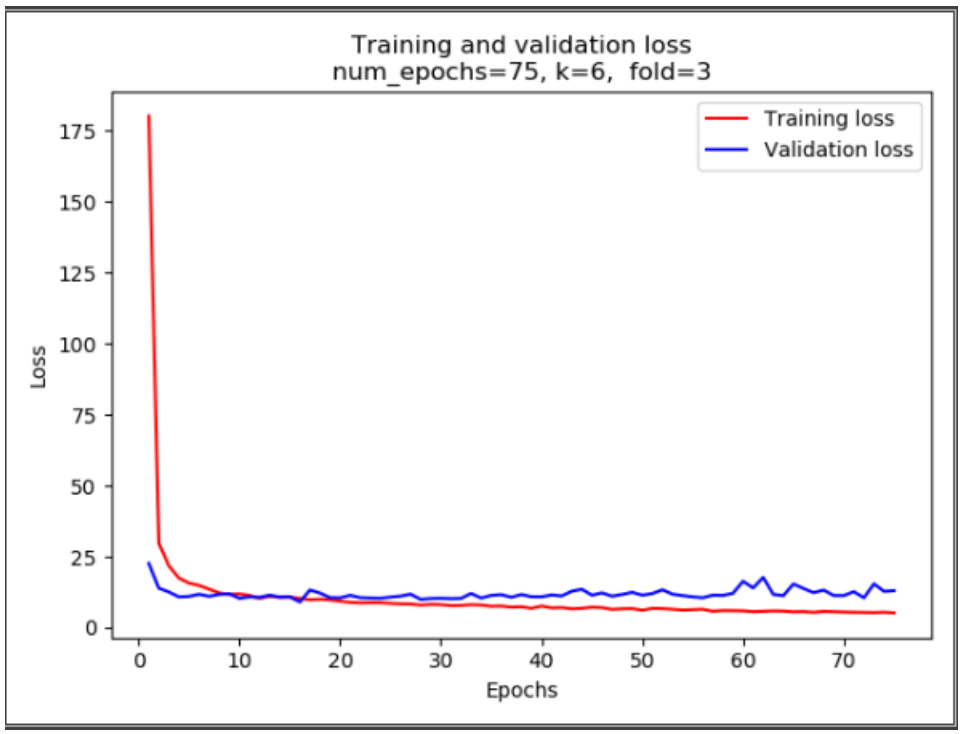


рисунок 9,10 – графики точности и ошибки модели на 3-ом блоке



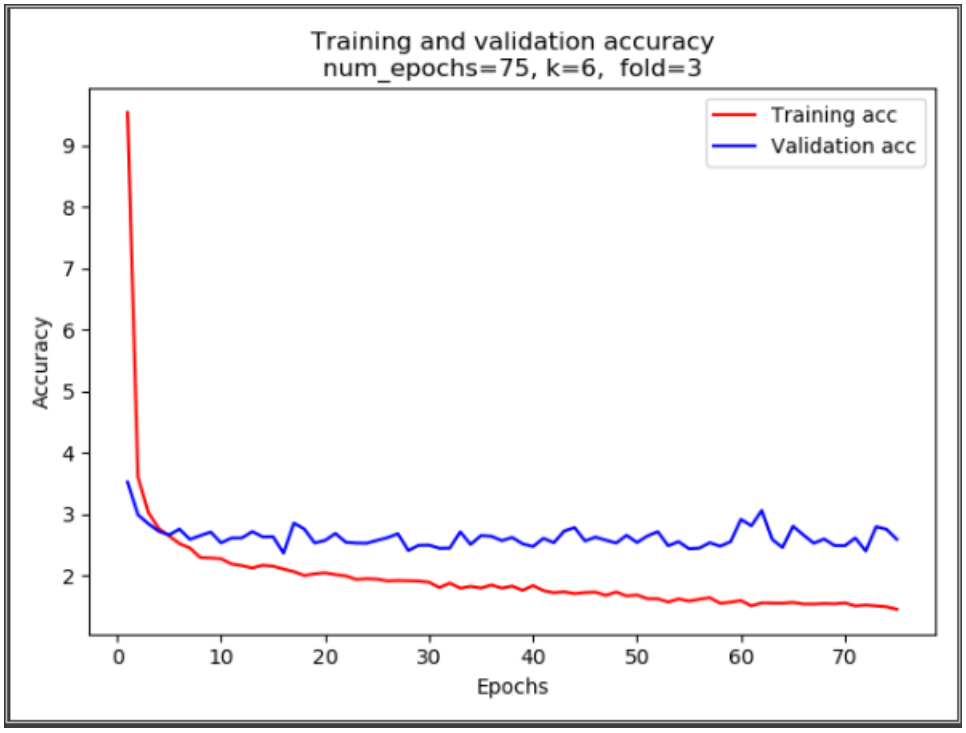
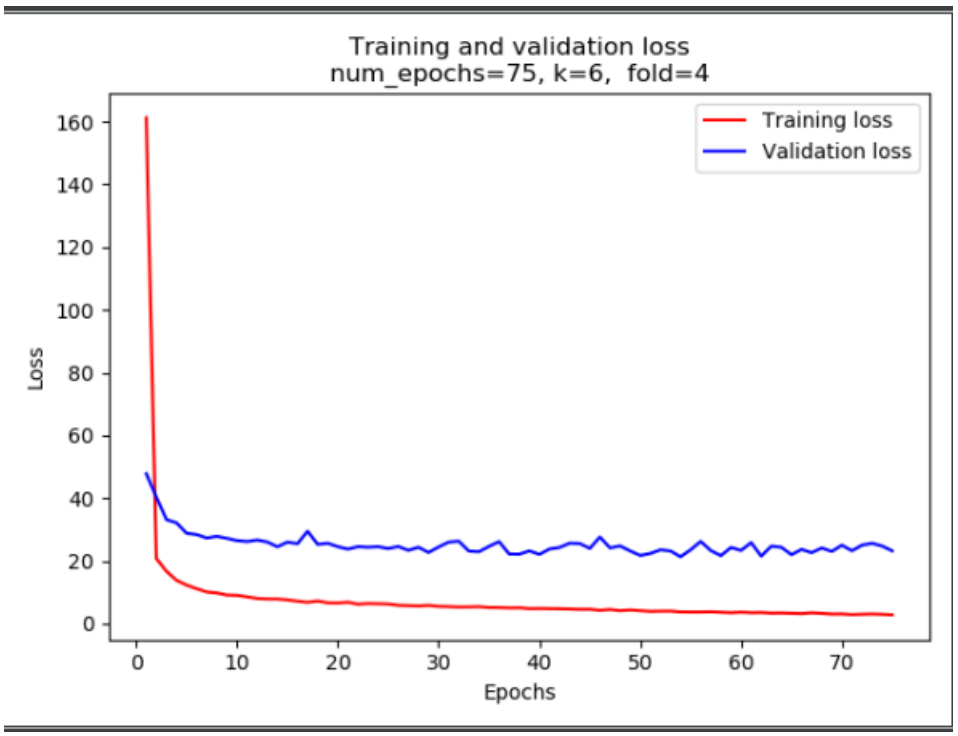


рисунок 11,12 – графики точности и ошибки модели на 4-ом блоке



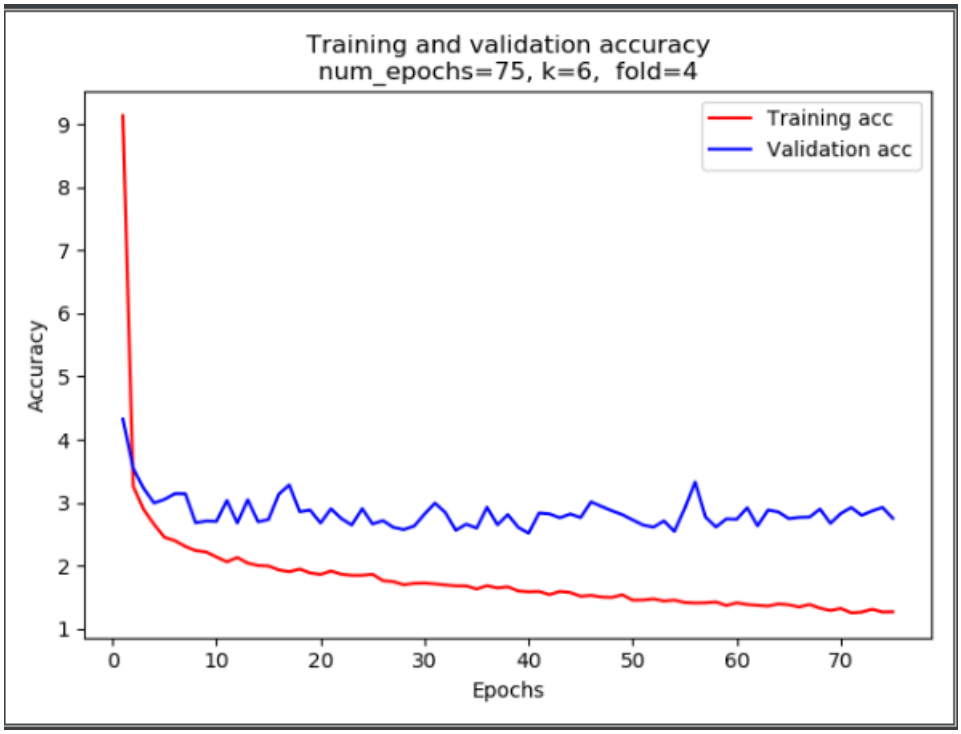
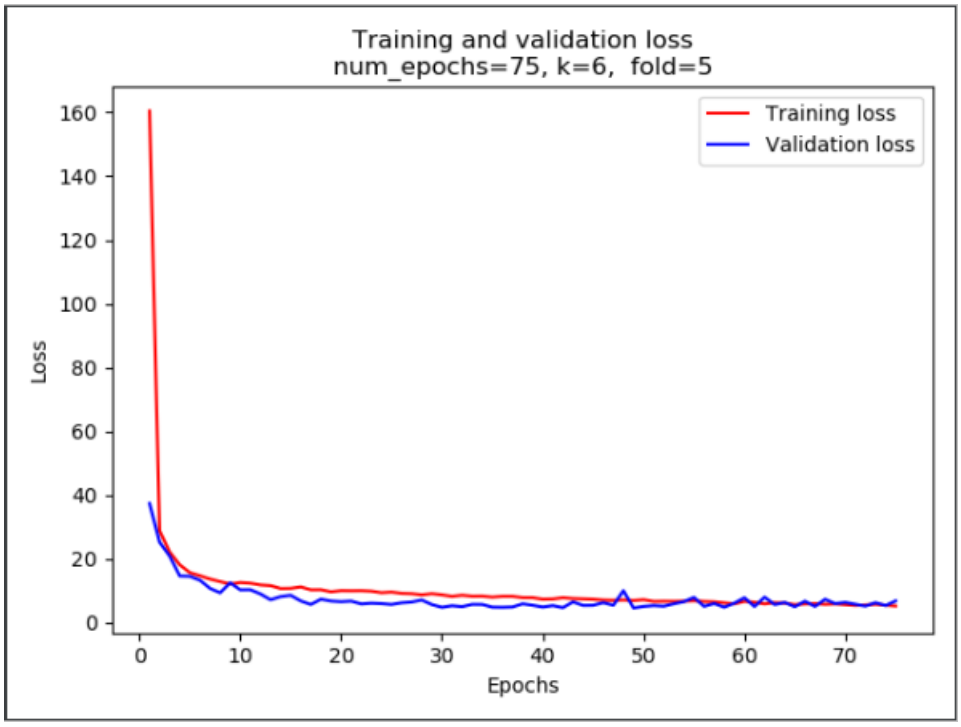


рисунок 13,14 – графики точности и ошибки модели на 5-ом блоке



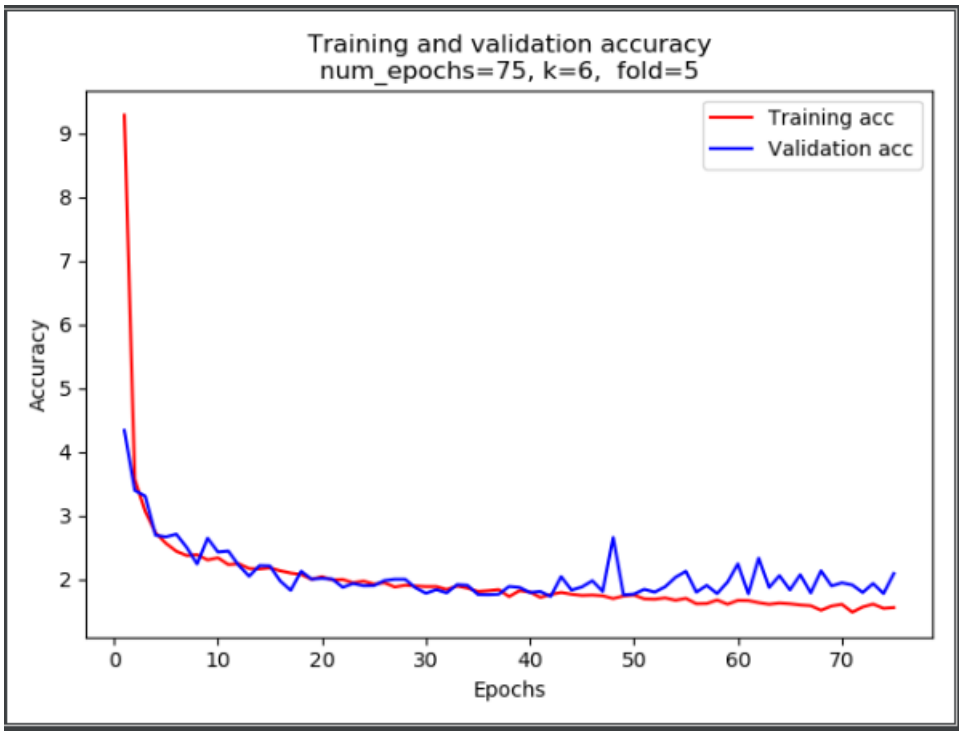


рисунок 15,16 – графики точности и ошибки модели на 6-ом блоке

Вывод.

В ходе выполнения данной работы была изучена задача регрессии с помощью библиотеки keras и ее отличие от задачи классификации.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
import numpy as np

import matplotlib.pyplot as plt
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

(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


best_mae = [1, 0, 0]
best_val_mae = [1, 0, 0]
for k in range(4, 9):
    epochs_scores = []
    mae_scores = []
    val_mae_scores = []
    for num_epochs in range(50, 175, 25):
        num_val_samples = len(train_data) // k
        mae_values = []
        val_mae_values = []
        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()
            history = model.fit(partial_train_data,
partial_train_targets, epochs=num_epochs, batch_size=1,
                                validation_data=(val_data,
val_targets), verbose=0)
            history_dict = history.history
            loss_values = history_dict['loss']
            val_loss_values = history_dict['val_loss']
            mae_values = history_dict['mean_absolute_error']
            val_mae_values = history_dict['val_mean_absolute_error']
            epochs = range(1, len(mae_values) + 1)
            plt.plot(epochs, loss_values, 'r', label='Training
loss')
            plt.plot(epochs, val_loss_values, 'b', label='Validation
loss')
            plt.title(f'Training and validation loss\n'
                      f'num_epochs={num_epochs}, k={k},  fold={i}')
            plt.xlabel('Epochs')
            plt.ylabel('Loss')
            plt.legend()
            plt.show()

            plt.clf()
            plt.plot(epochs, mae_values, 'r', label='Training acc')
            plt.plot(epochs, val_mae_values, 'b', label='Validation
acc')
            plt.title('Training and validation accuracy\n'
                      f'num_epochs={num_epochs}, k={k},  fold={i}')
            plt.xlabel('Epochs')
            plt.ylabel('Accuracy')
            plt.legend()
            plt.show()
        mean_mae = np.mean(mae_values)
        mean_val_mae = np.mean(val_mae_values)
        if mean_mae <= best_mae[0]:
            best_mae[0] = mean_mae
            best_mae[1] = num_epochs
            best_mae[2] = k
            print(best_mae)
        if mean_val_mae <= best_val_mae[0]:
```

```
            best_val_mae[0] = mean_val_mae
            best_val_mae[1] = num_epochs
            best_val_mae[2] = k
            print(best_val_mae)
        mae_scores.append(mean_mae)
        val_mae_scores.append(mean_val_mae)
        epochs_scores.append(num_epochs)
    plt.plot(epochs_scores, mae_scores, 'r', label='Mean MAE')
    plt.plot(epochs_scores, val_mae_scores, 'b', label='Mean val
MAE')
    plt.title(f'k={k}')
    plt.xlabel('Num of epochs')
    plt.ylabel('Mae')
    plt.legend()
    plt.show()
```