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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Искусственные нейронные сети»

Тема: Регрессионная модель изменения цен на дома в Бостоне

Студент гр. 7383 Сычевский Р.А.

Преподаватель Жукова Н.А.

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.

Порядок выполнения работы.

1. Ознакомиться с задачей регрессии
2. Изучить отличие задачи регрессии от задачи классификации
3. Загрузить данные
4. Создать модель
5. Настроить параметры обучения
6. Обучить и оценить модель
7. Ознакомиться с перекрестной проверкой

Ход работы.

Для изучения регрессионной модели ИНС была разработана и использована программа. Код программы приведен в приложении А.

Рассмотрим зависимость результата обучения от количества эпох. Изначально была рассмотрена модель с перекрестной проверкой на 4 блоках и 100 эпохами. Графики оценки MAE для каждого блока приведены на рис. 1-4. График средних значений MAE, приведен на рис. 5. Из графиков видно, что точность перестает расти после примерно 80 эпохи, тогда как на тестовых данных оно продолжает уменьшаться — это символизирует о том, что начинается переобучение нейронной сети.

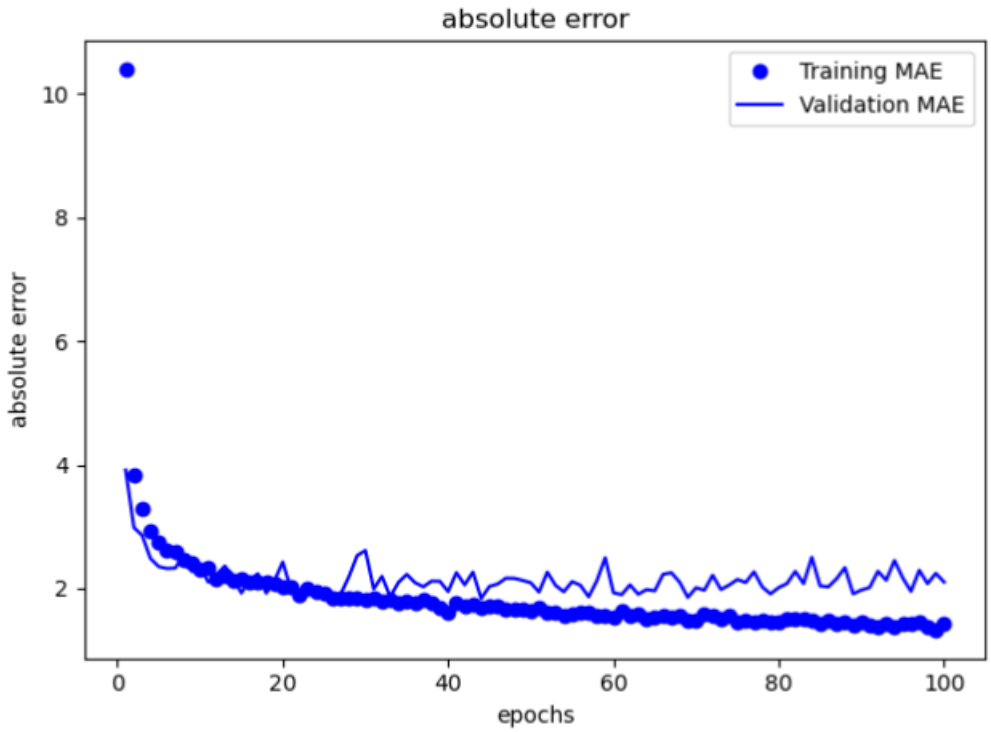

Рисунок 1 – График оценки MAE для блока 1

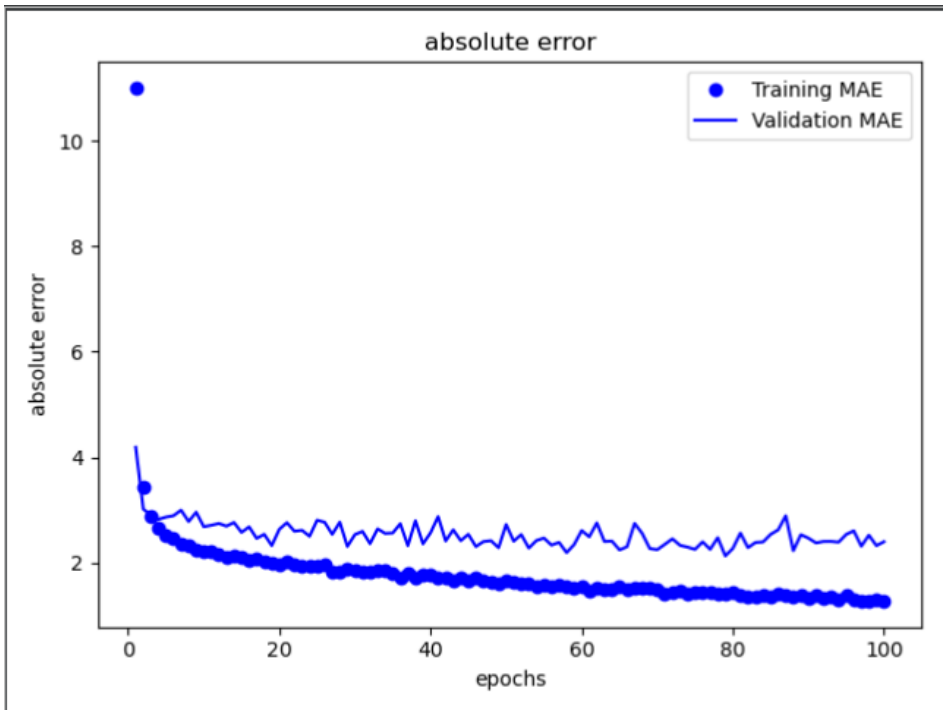

Рисунок 2 – График оценки MAE для блока 2

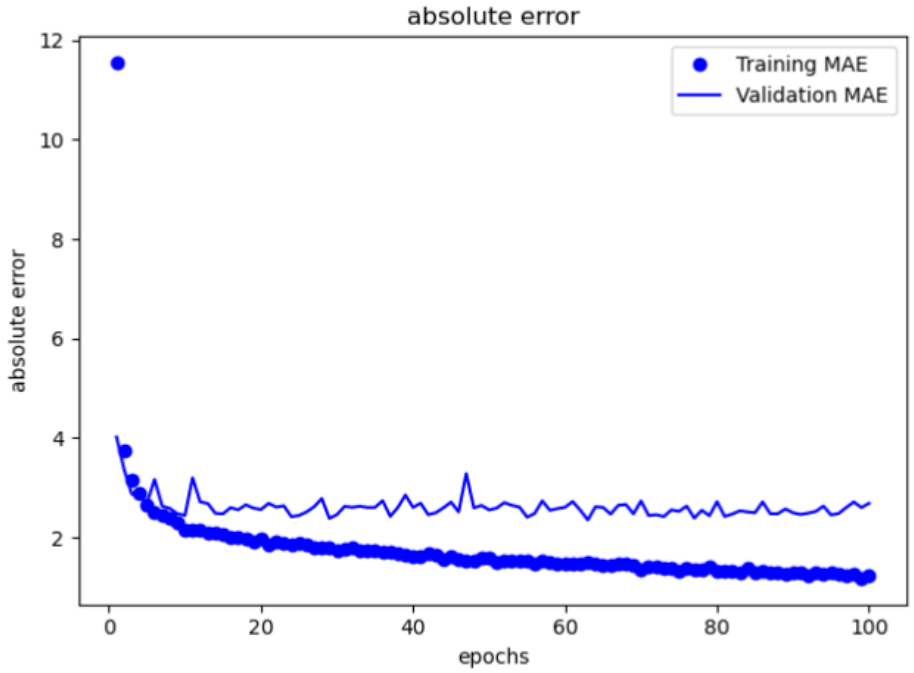

Рисунок 3 – График оценки MAE для блока 3

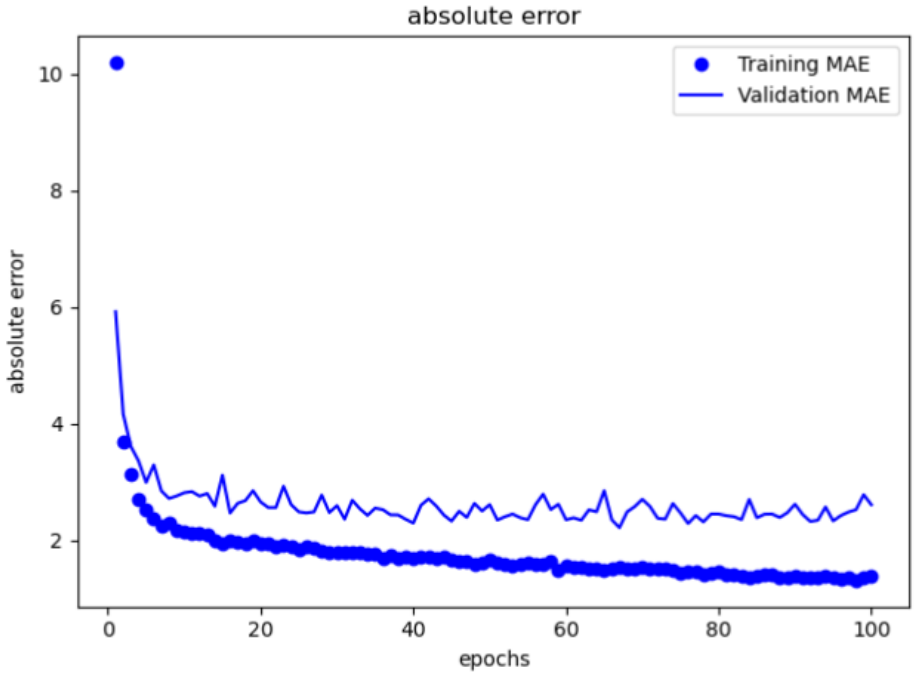

Рисунок 4 – График оценки MAE для блока 4

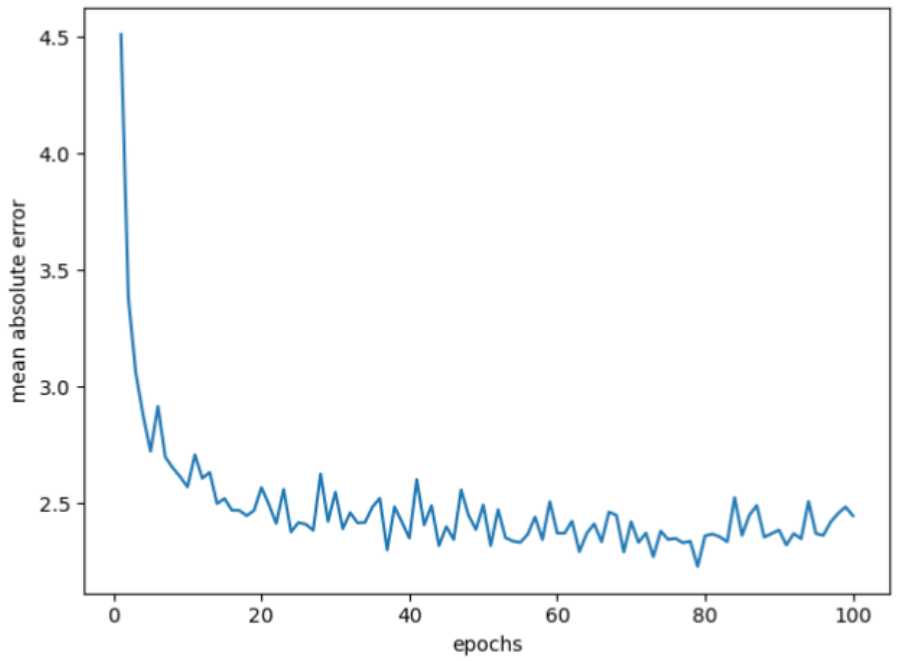

Рисунок 5 — График среднего значения MAE для модели с 100 эпохами и 4 блоками для перекрестной проверки

Далее рассмотрим модели с 80 эпохами и 4, 6 и 8 блоками для перекрестной проверки. Графики среднего значения оценки MAE для этих моделей приведены на рис. 6-8. Из графиков видно, что наилучшие значения средней оценки MAE достигается в модели с 6 блоками перекрестной проверки.

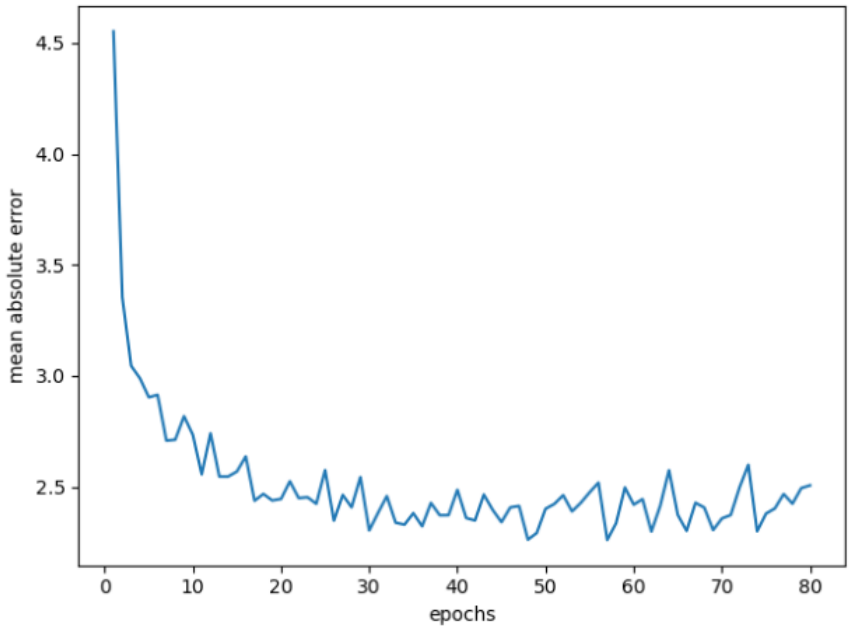

Рисунок 6 – График среднего значения MAE для 4 блоков

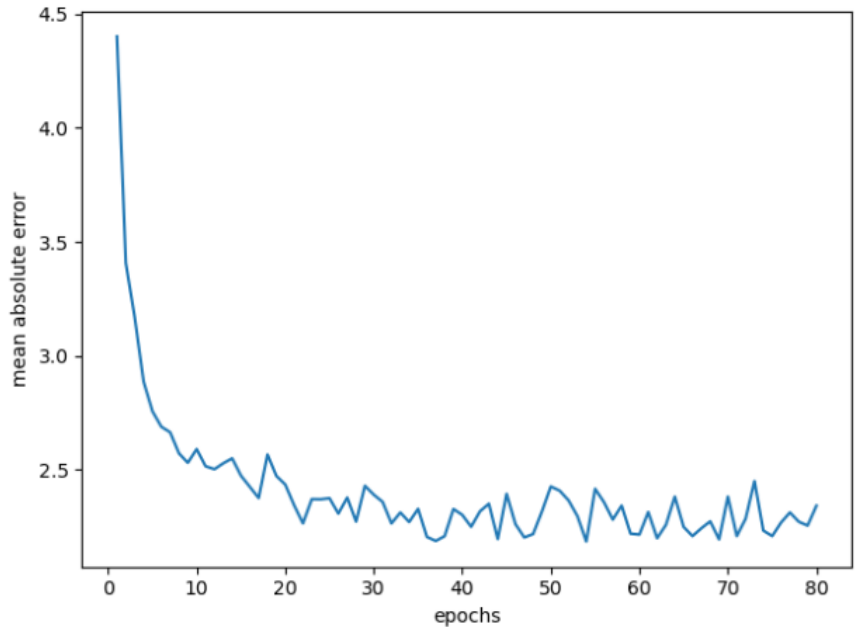

Рисунок 7 – График среднего значения MAE для 6 блоков

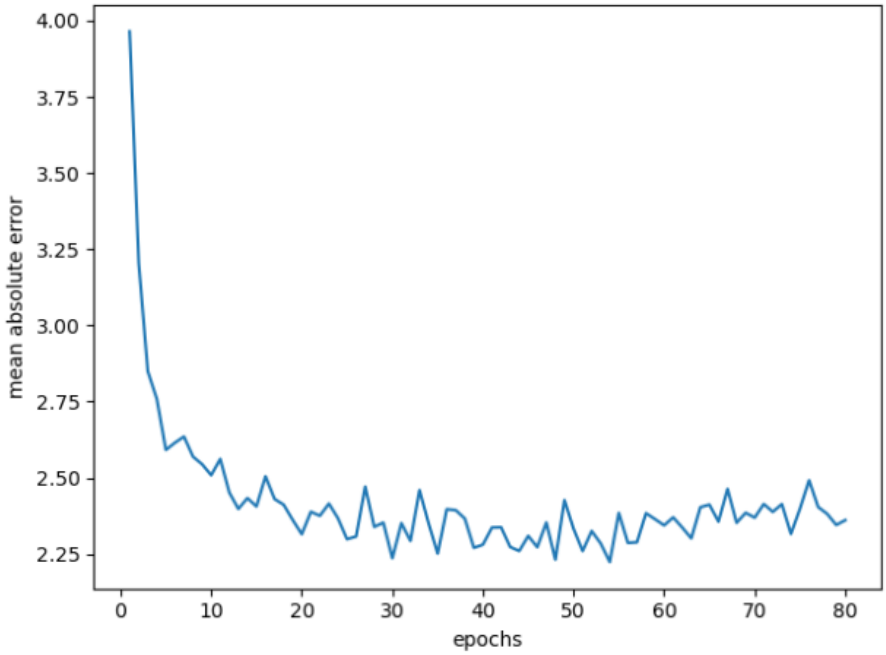

Рисунок 8 – График среднего значения MAE для 8 блоков

Выводы.

В ходе выполнения данной работы была изучена задача регрессии и ее отличие от задачи классификации. Была изучена и проведена перекрестная проверка модели.

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 plot

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()

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

k = 8
num_val_samples = len(train_data) // k
num_epochs = 80
all_scores = []
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_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))
    mae = history.history['mae']
    v_mae = history.history['val_mae']
    x = range(1, num_epochs + 1)
    all_scores.append(v_mae)
    plot.figure(i + 1)
    plot.plot(x, mae, 'bo', label='Training MAE')
    plot.plot(x, v_mae, 'b', label='Validation MAE')
    plot.title('absolute error')
    plot.ylabel('absolute error')
    plot.xlabel('epochs')
    plot.legend()

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