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»

Студентка гр. 7383 _____ Иолшина В.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цель работы.

Реализовать предсказать 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.

Задача классификации – построение алгоритма, определяющего принадлежность объекта к одному из заданных классов. Задача регрессии –

определение значения некоторой характеристики объекта, в зависимости от других характеристик, подаваемых на вход. Другими словами, регрессия позволяет предсказывать не дискретную метку, а значения на непрерывной числовой прямой.

Была создана и обучена модель искусственной нейронной сети с перекрестной проверкой по K блокам (K-fold cross-validation) (код программы представлен в приложении А).

Первоначальное количество блоков было равным 4, а количество эпох 300. Графики среднего абсолютного отклонения модели mae для каждого блока приведены на рис. 1 – 4. График средних значений mae, приведен на рис. 5.

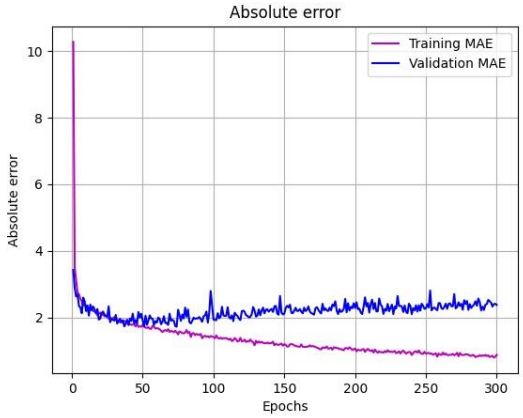

Рисунок 1 – График оценки mae для 1 блока.

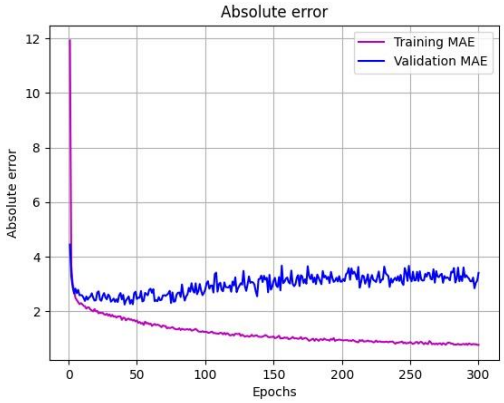

Рисунок 2 – График оценки mae для 2 блока.

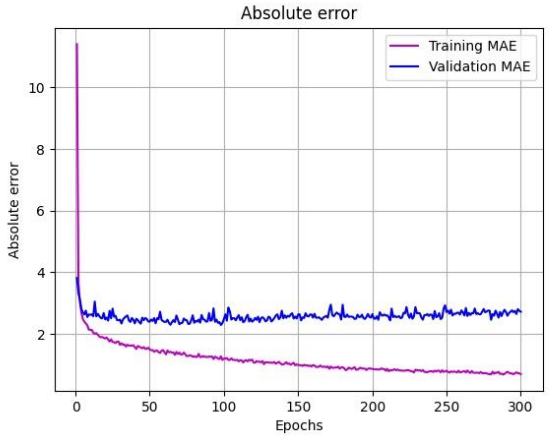

Рисунок 3 – График оценки mae для 3 блока.

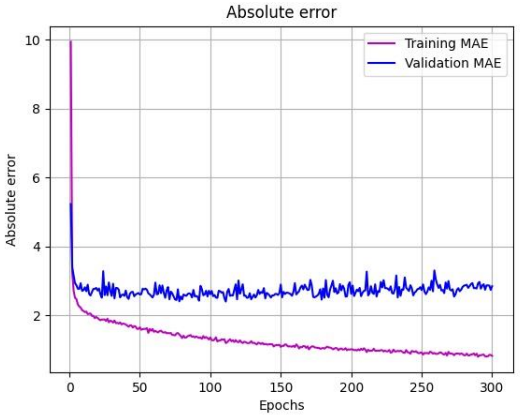

Рисунок 4 – График оценки mae для 4 блока.

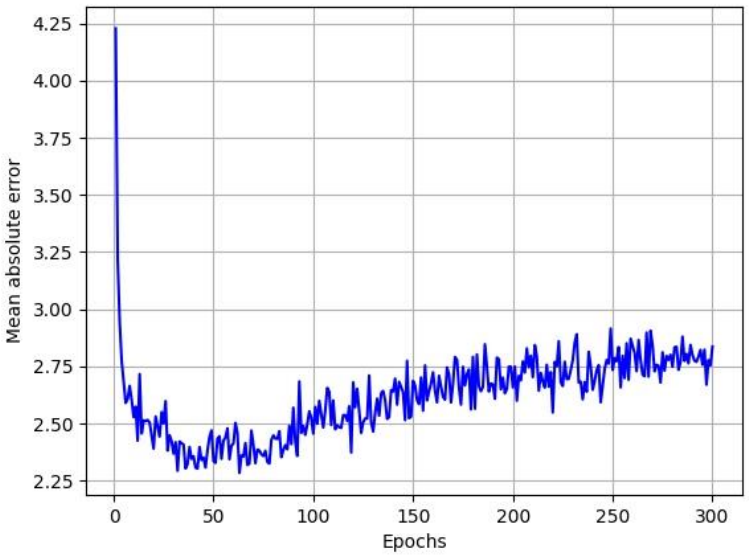

Рисунок 5 – График среднего значения mae (модель: k = 4 и epochs = 500).

По графику (рис. 5) видно, что mae на проверочных данных уменьшается до ≈ 60 эпох обучения, после mae только увеличивается, тогда как на тестовых данных продолжает уменьшаться.

Это связано с переобучением нейронной сети, поэтому за оптимальное число эпох берётся 60.

При заданном числе эпох было рассмотрено среднее значение оценки mae при k = 4, 6 и 8 блокам для перекрестной проверки. Результаты приведены на графиках, показанных на рис. 6 – 9.

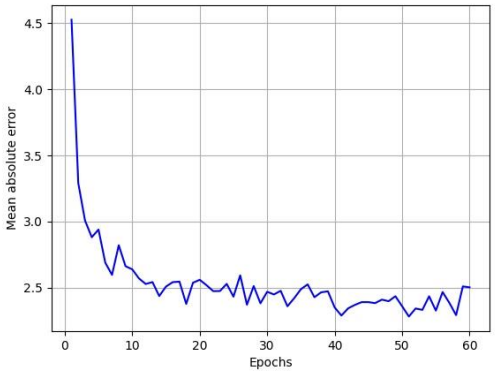

Рисунок 8 – График среднего значения mae для k = 4.

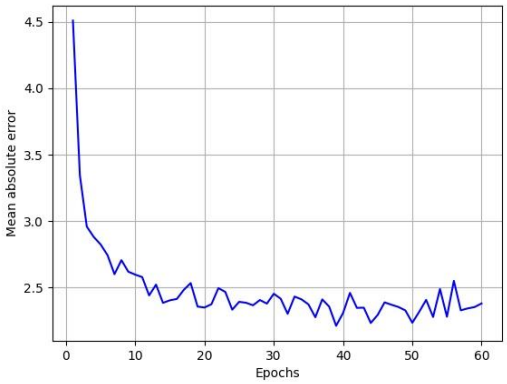

Рисунок 8 – График среднего значения mae для k = 6.

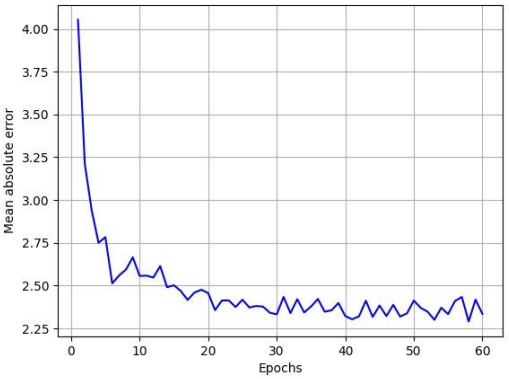

Рисунок 9 – График среднего значения mae для k = 8.

По графикам (рис. 6 – 9) видно, что наилучшее значение средней оценки mae достигается в модели, использующей 8 блоков (рис. 8).

Вывод.

В ходе выполнения лабораторной работы было изучено влияние количества эпох и количества блоков в перекрестной проверке по K блокам (K-fold cross-validation) на результат обучения модели искусственной нейронной сети, решающей задачу регрессии. Оптимальной моделью была выбрана следующая - при k = 8 и epochs = 60.

ПРИЛОЖЕНИЕ А

```
import numpy as np
from tensorflow.keras.layers import  Dense
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt
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)
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-fold cross-validation
k = 8
```

```
    num_val_samples = len(train_data) // k
    num_epochs = 60
    all_scores = []
    mae_histories = []

    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['mean_absolute_error']
        v_mae = history.history['val_mean_absolute_error']
        x = range(1, num_epochs + 1)
        mae_histories.append(v_mae)
        plt.figure(i + 1)
        plt.plot(x, mae, 'm', label='Training MAE')
        plt.plot(x, v_mae, 'b', label='Validation MAE')
        plt.title('Absolute error')
        plt.ylabel('Absolute error')
        plt.xlabel('Epochs')
        plt.legend()
        plt.grid()

    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, 'b')
plt.xlabel('Epochs')
plt.ylabel("Mean absolute error")
plt.grid()
figs = [plt.figure(n) for n in plt.get_fignums()]
for i in range(len(figs)):
        figs[i].savefig("./%d.png" %(i), format='png')
```