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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Искусственные нейронные сети»

Тема: Регрессионная модель изменения цен на дома в Бостоне

Студент гр. 7383 _____ Ласковенко Е.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.

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 ИНС в tf.Keras
- Настроить параметры обучения
- Обучить и оценить модель
- Ознакомиться с перекрестной проверкой

Требования.

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.

Ход работы.

Задачей классификации является определение отношений объектов из заданного множества к классам из некоторого конечного множества классов. Иначе говоря, необходимо классифицировать объекты, то есть отнести каждый объект к соответствующему классу. Задача регрессии – определить значение какой-либо характеристики объекта, значение которой может быть любым числом.

Была создана и обучена модель искусственной нейронной сети в соответствии с условиями (весь код представлен в приложении А).

При исследовании разных архитектур и обучения при различных параметрах обучения ИНС необходимо было:

- изменить количество эпох обучения
- применить перекрестную проверку по K блокам при различных K

Для определения точки переобучения была рассмотрена изначальная модель с перекрестной проверкой по K блокам при K = 4 и количеством эпох обучения, равным 100. На рис. 1 - 4 представлены графики оценки средней абсолютной ошибки ИНС в ходе обучения для каждого блока. График средних значений средней абсолютной ошибки ИНС приведен на рис. 5.

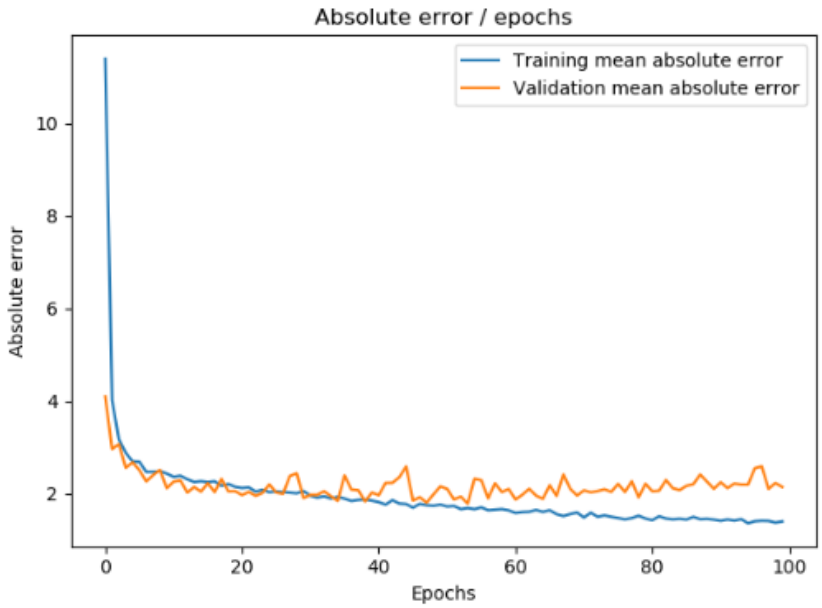

Рисунок 1 – График оценки mae для 1 блока

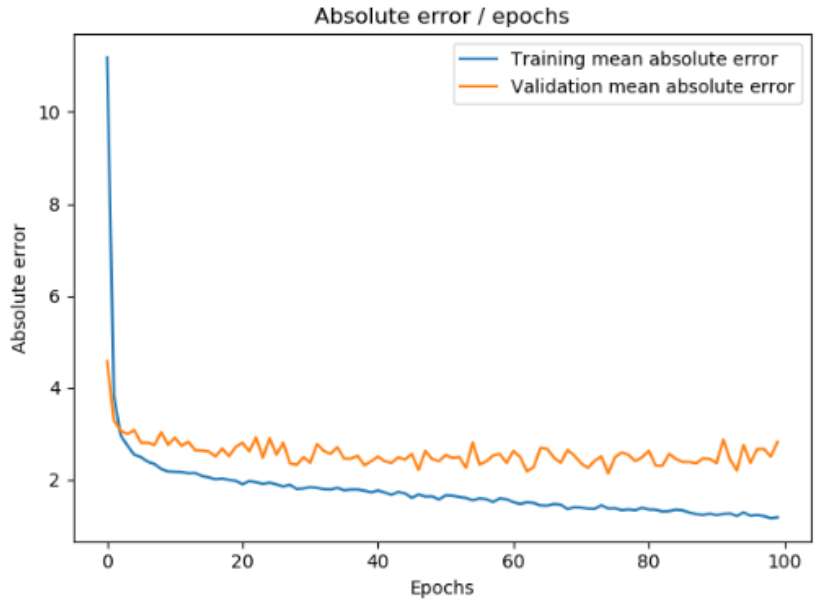

Рисунок 2 – График оценки mae для 2 блока

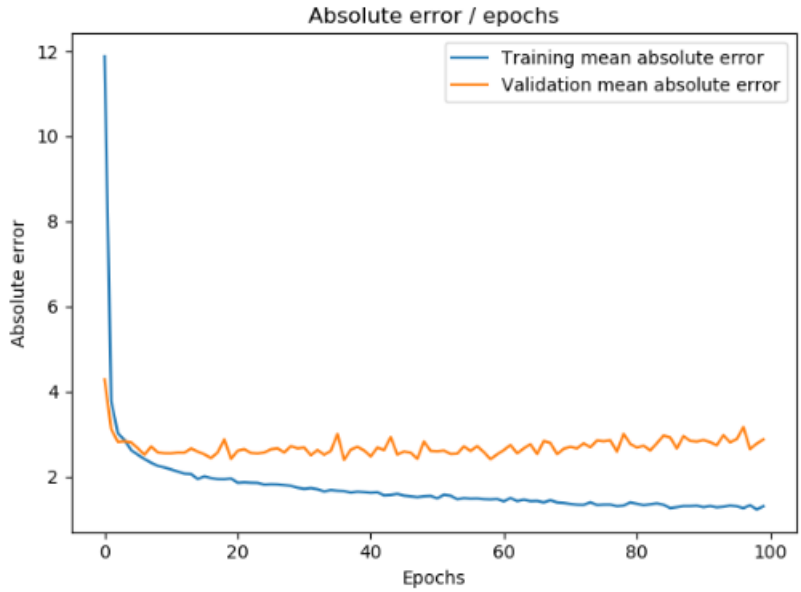

Рисунок 3 – График оценки mae для 3 блока

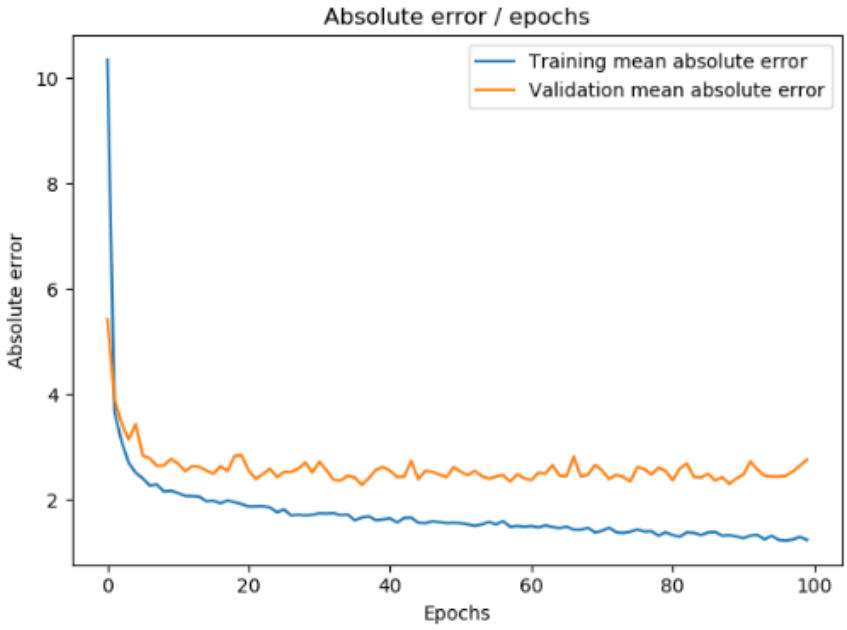

Рисунок 4 – График оценки mae для 4 блока

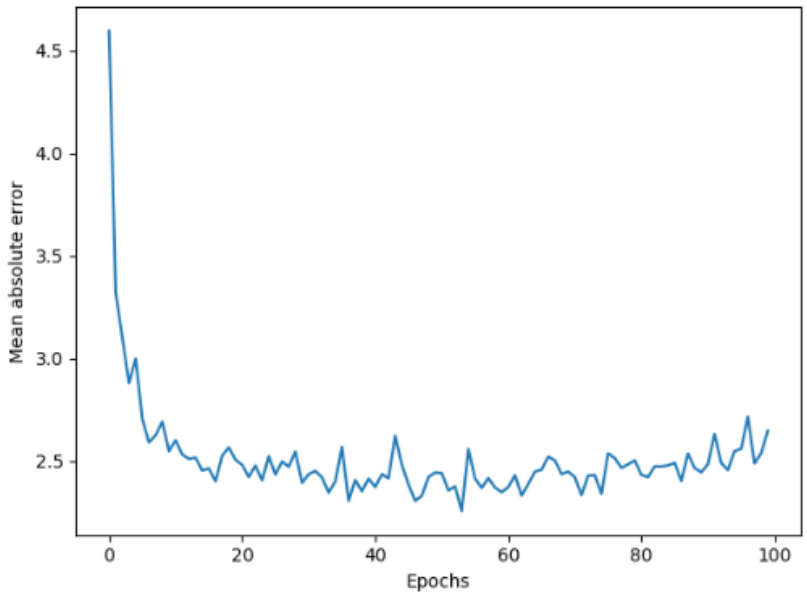

Рисунок 5 – График среднего значения mae

Исходя из полученного графика среднего значения mae видно, что значение средней абсолютной ошибки уменьшается примерно до 40 эпохи обучения, после оно увеличивается, что свидетельствует о дальнейшем переобучении сети. Следовательно, наиболее оптимальное число эпох обучения – 40.

Для рассмотрения перекрестной проверки по К блокам было рассмотрено среднее значение оценки mae при К = 2, 4, 6 и 8. Графики средних значений mae представлены на рис. 6 – 9.

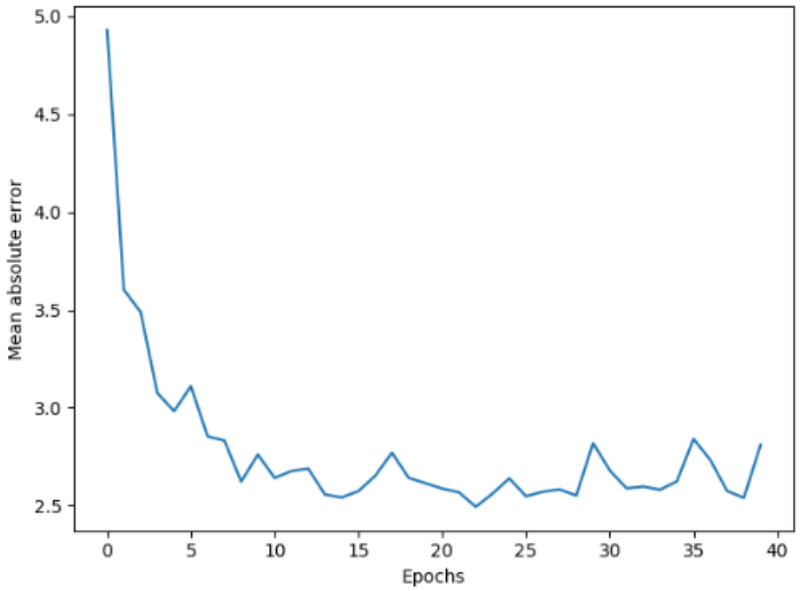

Рисунок 6 – График среднего значения mae для K = 2

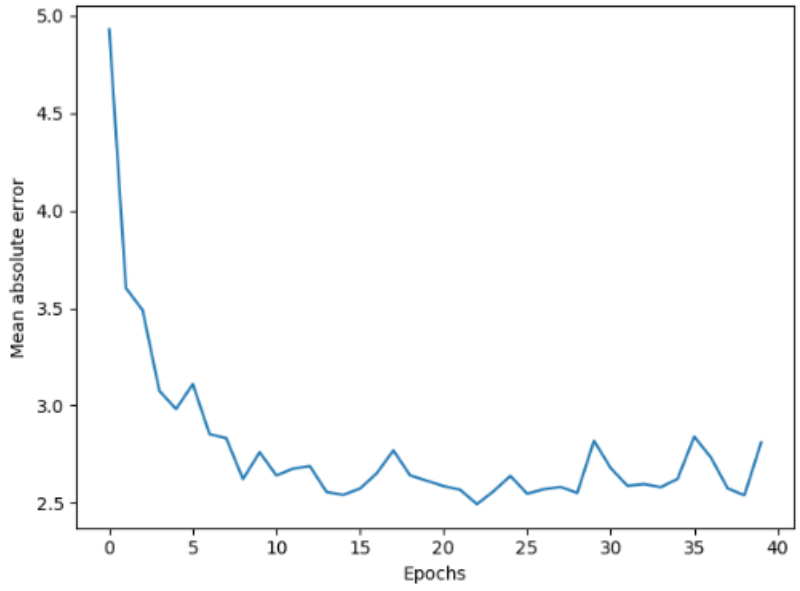

Рисунок 7 – График среднего значения mae для K = 4

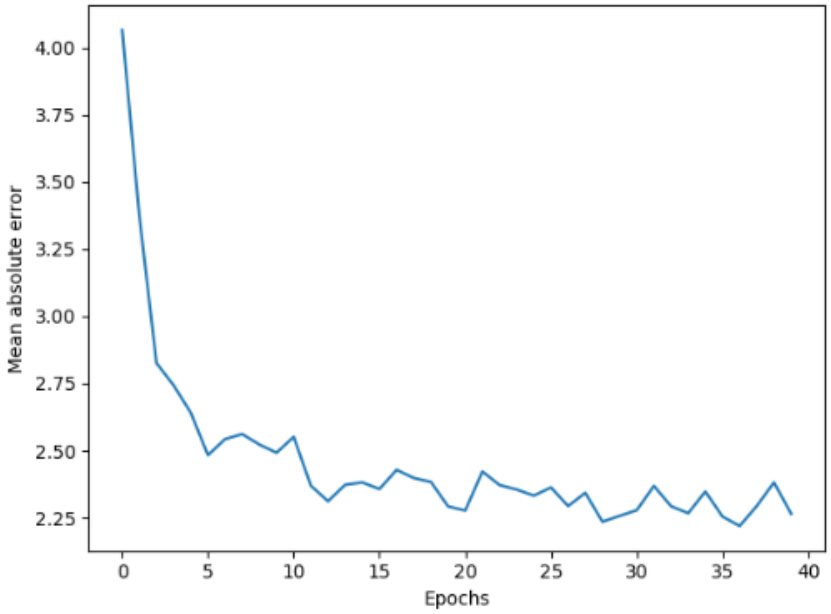

Рисунок 8 – График среднего значения mae для K = 6

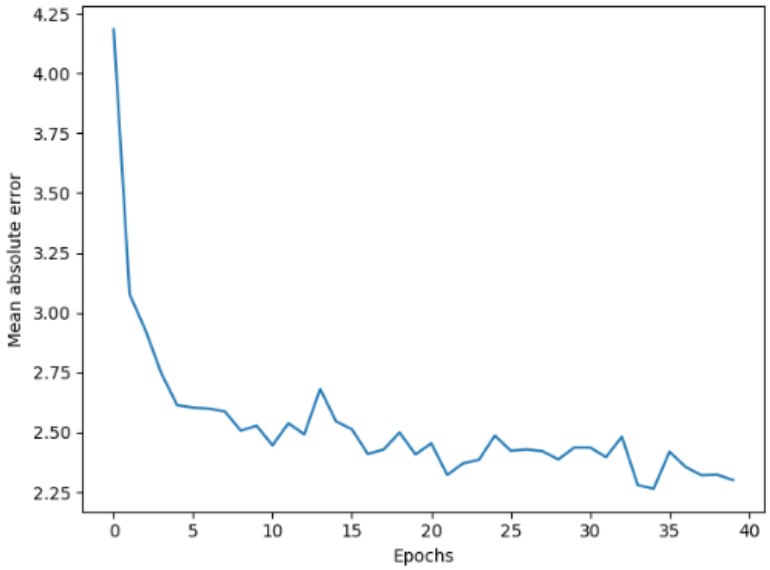

Рисунок 9 – График среднего значения mae для K = 8

По графикам видно, что наименьшее значение средней оценки mae за наименьшее число эпох обучения достигается в модели, использующей 6 блоков при перекрестной проверке.

Вывод.

В ходе выполнения лабораторной работы было изучено влияние количества эпох и количества блоков в перекрестной проверке по K блокам на результат обучения ИНС, решающей задачу регрессии.

Приложения

Приложение А

```
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.datasets import boston_housing
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential


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


def set_plot(figure_num, epoch_num, train_mae, validation_mae):
    x = range(0, epoch_num)
    plt.figure(figure_num)
    plt.plot(x, train_mae, label='Training mean absolute error')
    plt.plot(x, validation_mae, label='Validation mean absolute
error')
    plt.title('Absolute error / epochs')
    plt.xlabel('Epochs')
    plt.ylabel('Absolute error')
    plt.legend()
    plt.show()


# Data loading:
(train_data, train_targets), (test_data, test_targets) =
boston_housing.load_data()
print(train_data.shape)
print(test_data.shape)
print(test_targets)

# Data normalization:
mean = train_data.mean(axis=0)
std = train_data.std(axis=0)
train_data -= mean
train_data /= std
test_data -= mean
test_data /= std

# Model building and fitting with K-fold cross-validation:
k = 8
```

```
num_val_samples = len(train_data) // k
num_epochs = 40
all_scores = []
val_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_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, verbose=0,
                        validation_data=(val_data, val_targets))
    mae_history = history.history["mean_absolute_error"]
    val_mae_history = history.history["val_mean_absolute_error"]
    val_mae_histories.append(val_mae_history)
    set_plot(i, num_epochs, mae_history, val_mae_history)
    val_mse, val_mae = model.evaluate(val_data, val_targets,
verbose=0)
    all_scores.append(val_mae)

print(np.mean(all_scores))

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