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»

Студентка гр. 7381 _____ Машина Ю.Д.

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.

1) Различия задач классификации и регрессии.

Задача регрессии – прогноз на основе выборки объектов с различными признаками. На выходе должно получиться вещественное число.

Задача классификации – получение категориального ответа на основе набора признаков. Имеет конечное количество ответов (как правило, в формате «да» или «нет»).

2) Изучение влияния кол-ва эпох на результат обучения модели.

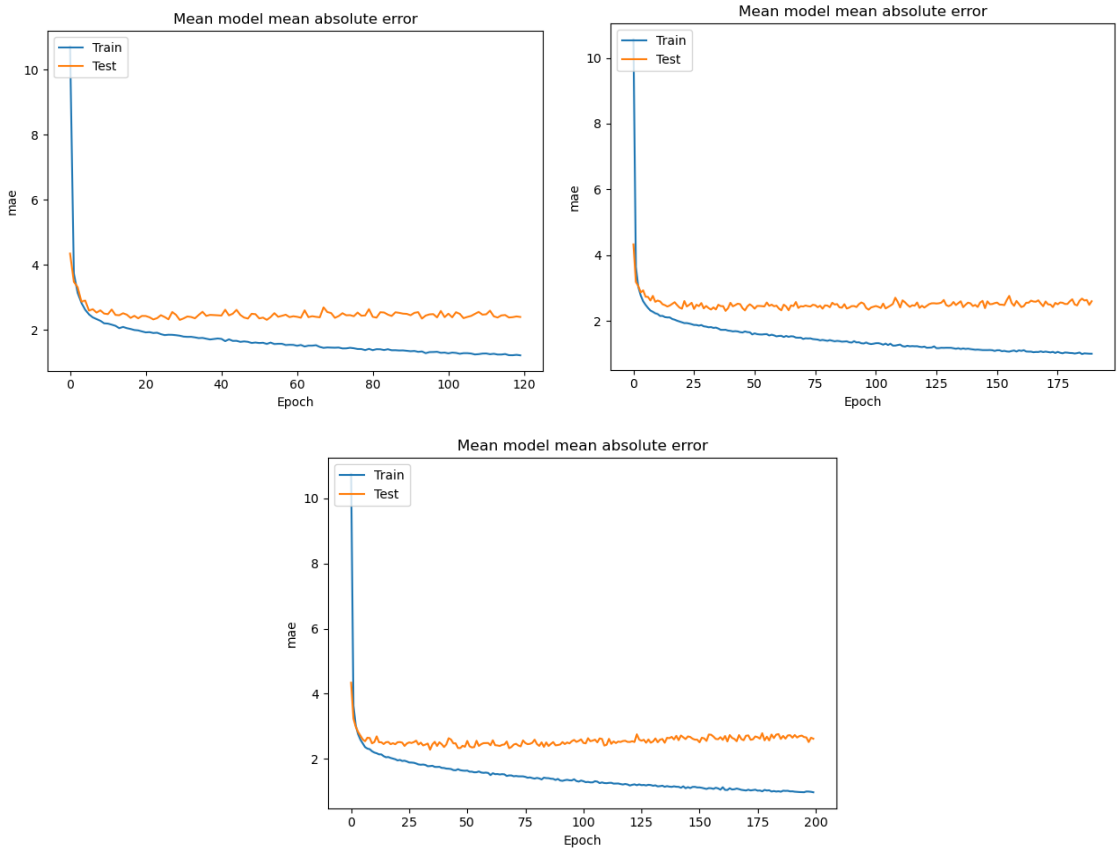

Рисунок 1 – График зависимости средней абсолютной ошибки от кол-ва эпох обучения выглядел примерно одинаково для k = 4, 5 и 6, было протестировано кол-во эпох от 70 до 200 и график mae почти не изменялся, разве что на 70 mae получился даже меньше, чем при бОльших кол-вах эпох, что скорее похоже на случайность. Из-за всего этого оказалось трудно эмпирическим путем заключить, как влияет кол-во эпох на результат обучения модели.

3) Выявить точку переобучения.

Выявить точку переобучения (то есть переобучить сеть так, чтобы потери только стали возрастать), увеличивая кол-во блоков К и или эпох, не получилось, поэтому я стала добавлять промежуточные слои с функцией активации relu и 64 нейронами, но сдалась на 63-м добавленном. Результаты тестов становились хаотичнее, но по графикам я не смогла увидеть даже тенденцию к приближению к переобучению. Выявить точку переобучения не удалось.

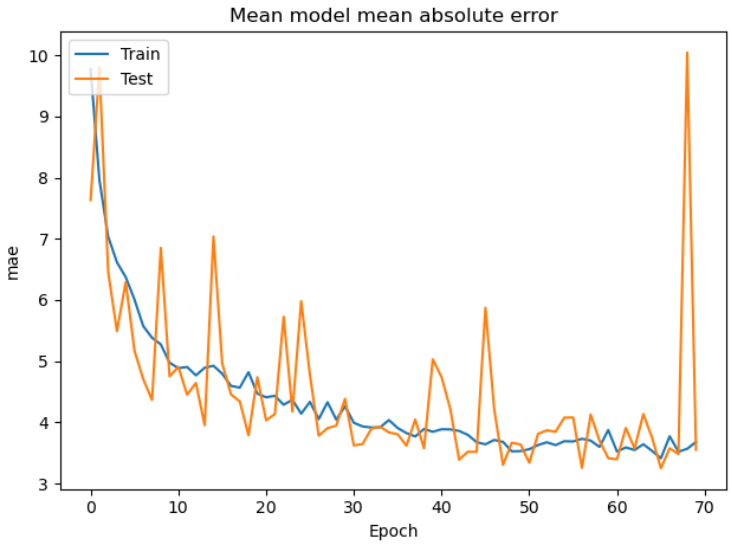

Рисунок 2 – Так выглядел mae.

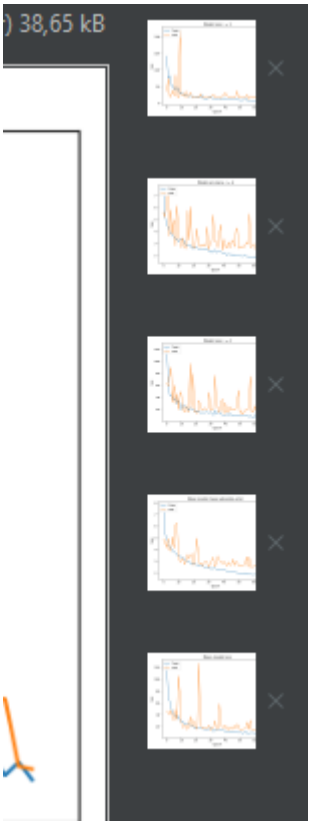

Рисунок 1.5 – Так выглядела средняя картина с графиками. А если бы получилось переобучение, то на графиках model loss показатели у Test (оранжевый) бы гиперболично увеличивались.

4) Применить перекрестную проверку по K блокам при различных K. Для этого были перебраны значения от 4 до 6. Особой разницы не было замечено.

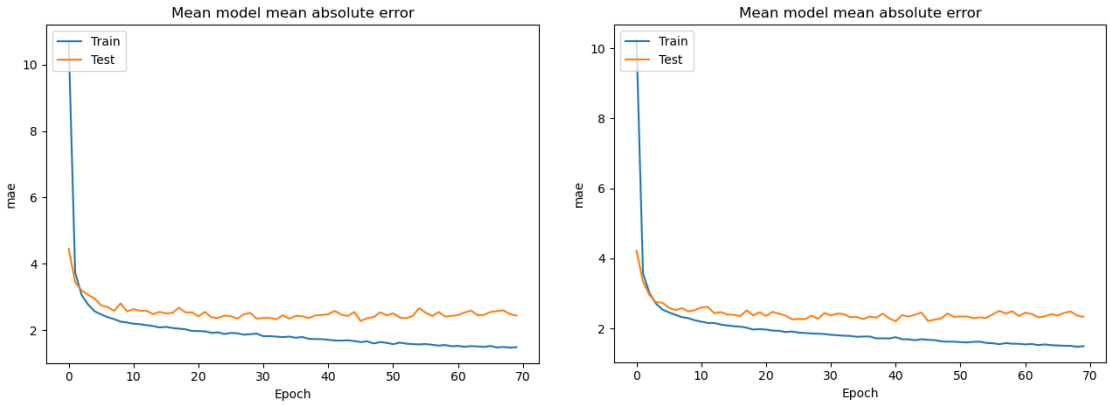

Рисунок 3 – Графики mae при k = 4 и 6.

Графики точности и ошибок обучения модели с параметрами: количество эпох обучения - 70, количество блоков – 4.

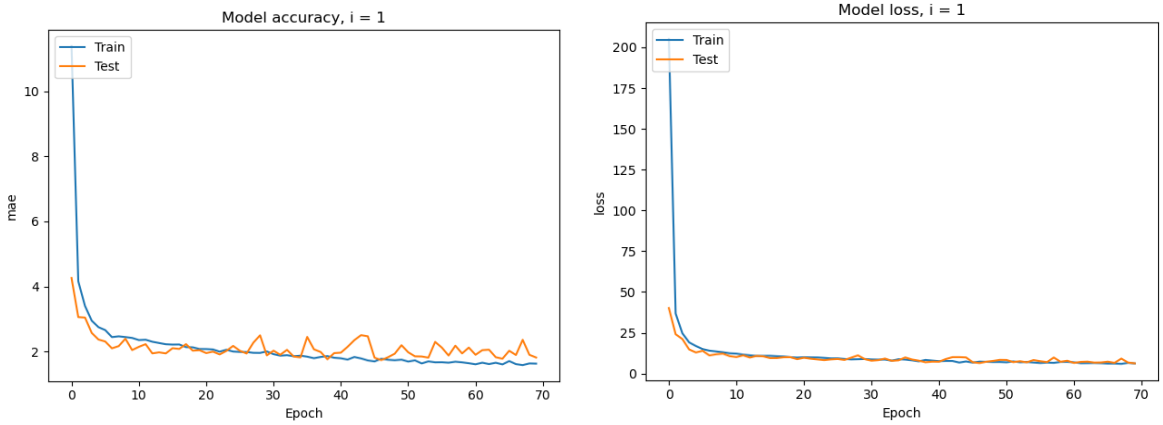

Рисунок 4 – график точности и обучения модели на 1-ом блоке.

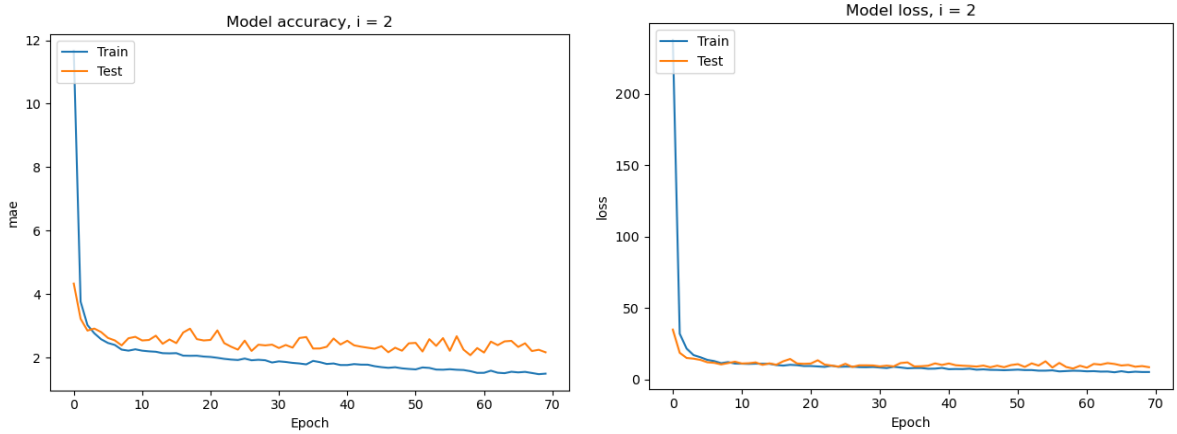

Рисунок 5 – график точности и обучения модели на 2-ом блоке.

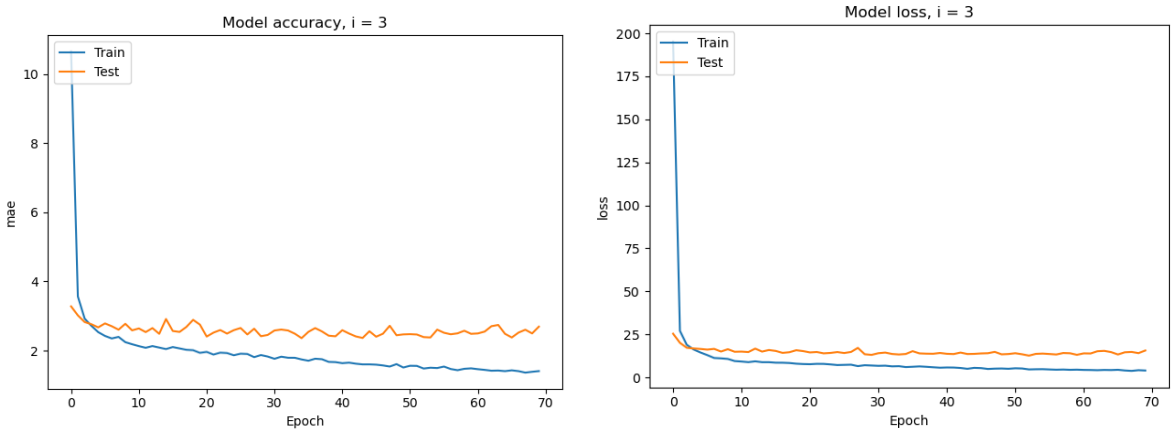

Рисунок 6 – график точности и обучения модели на 3-ом блоке.

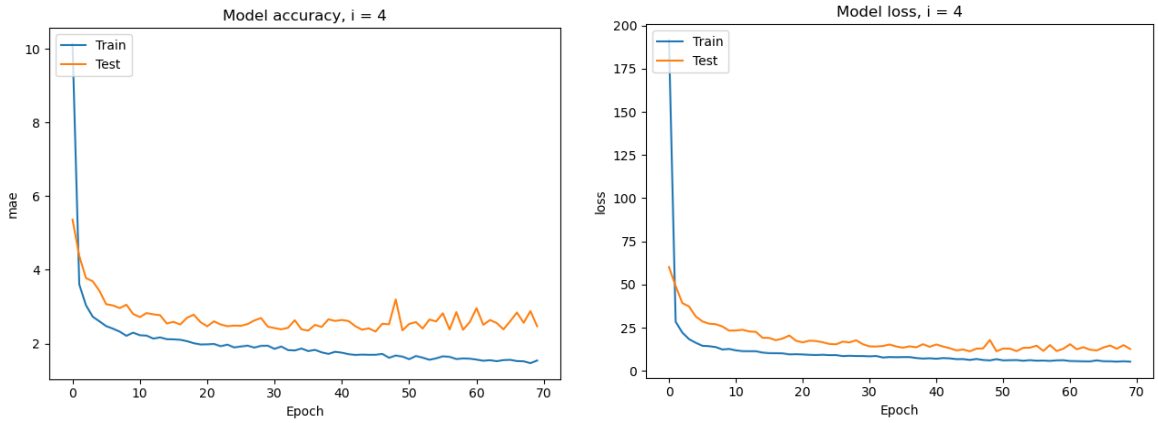

Рисунок 7 – график точности и обучения модели на 4-ом блоке.

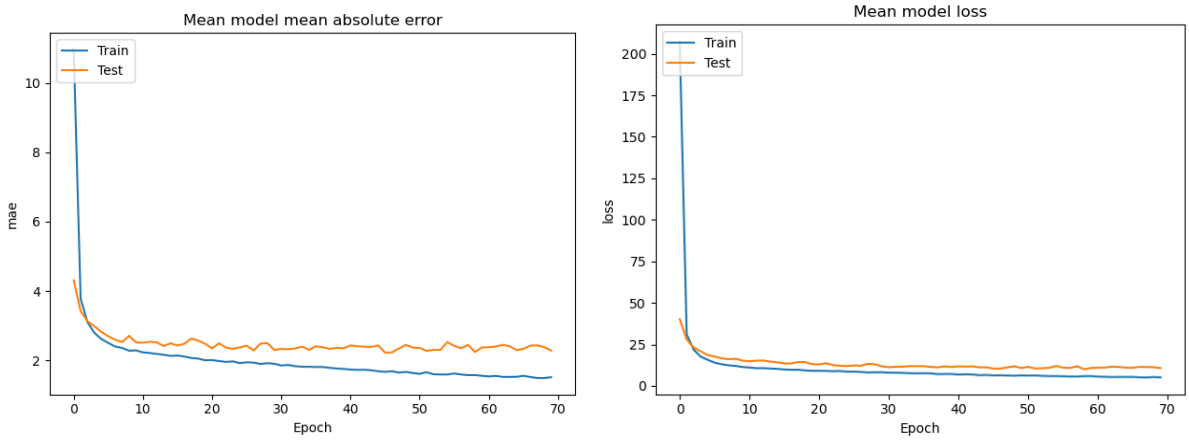

Рисунок 8 – график точности и обучения усредненной модели.

Вывод.

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

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
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

import numpy as np
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


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


if __name__ == '__main__':
    (train_data, train_targets), (test_data, test_targets) =
boston_housing.load_data()

    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 = 4
    num_val_samples = len(train_data) // k

    num_epochs = 70
    #all_scores = []
    mean_loss = []
    mean_mae = []
    mean_val_loss = []
    mean_val_mae = []
```

```
    for i in range(k):
        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)


mean_val_mae.append(history.history['val_mean_absolute_error'])
        mean_mae.append(history.history['mean_absolute_error'])

        plt.plot(history.history['mean_absolute_error'])
        plt.plot(history.history['val_mean_absolute_error'])
        title = 'Model accuracy' + ', i = ' + str(i+1)
        plt.title(title)
        plt.ylabel('mae')
        plt.xlabel('Epoch')
        plt.legend(['Train', 'Test'], loc='upper left')
        plt.show()

        mean_val_loss.append(history.history['val_loss'])
        mean_loss.append(history.history['loss'])

        plt.plot(history.history['loss'])
        plt.plot(history.history['val_loss'])
        title = 'Model loss' + ', i = ' + str(i+1)
        plt.title(title)
        plt.ylabel('loss')
        plt.xlabel('Epoch')
        plt.legend(['Train', 'Test'], loc='upper left')
        plt.show()

    plt.plot(np.mean(mean_mae, axis=0))
    plt.plot(np.mean(mean_val_mae, axis=0))
    title = 'Mean model mean absolute error'
    plt.title(title)
    plt.ylabel('mae')
```

```
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

plt.plot(np.mean(mean_loss, axis=0))
plt.plot(np.mean(mean_val_loss, axis=0))
title = 'Mean model loss'
plt.title(title)
plt.ylabel('loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```