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. 7382 Лящевская А.П.

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) Была создана и обучена модель искусственной нейронной сети в соответствии с условиями(весь код представлен в приложении А). 2) Для проверки влияния количества эпох на результат обучения модели был выбран диапазон от 10 до 23. Ниже на рис. 1 представлен график среднего абсолютного отклонения модели при всех значениях в зависимости от количества эпох.

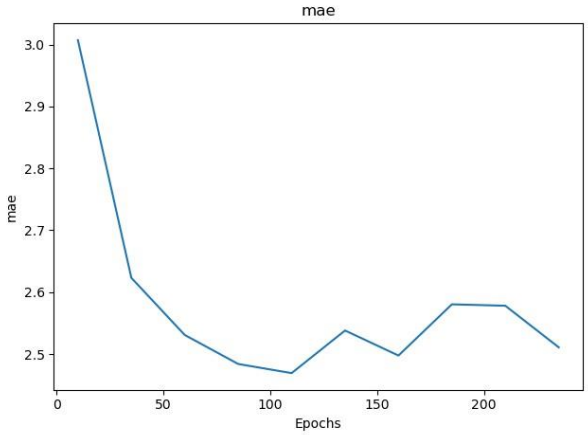

рисунок 1 – график зависимости средней абсолютной ошибки от кол-ва эпох обучения

3) Как видно из графика, оптимальным числом эпох является число 110. Далее нужно определить оптимальное число К. Для этого были перебраны значения от 3 до 8. Ниже представлен график среднего абсолютного отклонения модели при всех значениях в зависимости от значения К.

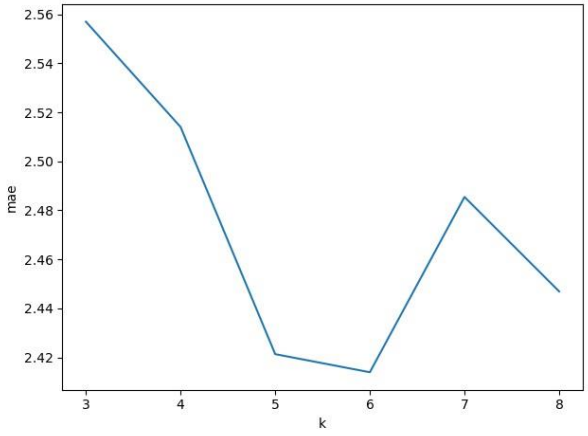

рисунок 2 – график зависимости средней абсолютной ошибки от числа k

По графику видно, что оптимальным К является 6.

4) Были построены графики точности и ошибок обучения модели с параметрами: количество эпох обучения - 110, количество блоков – 6.

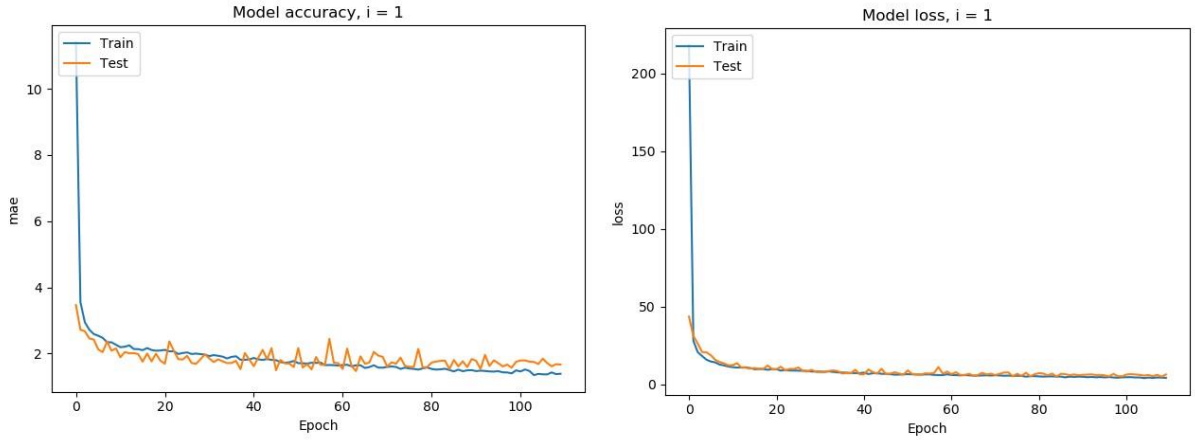

рисунок 3 – график точности и обучения модели на 1-ом блоке

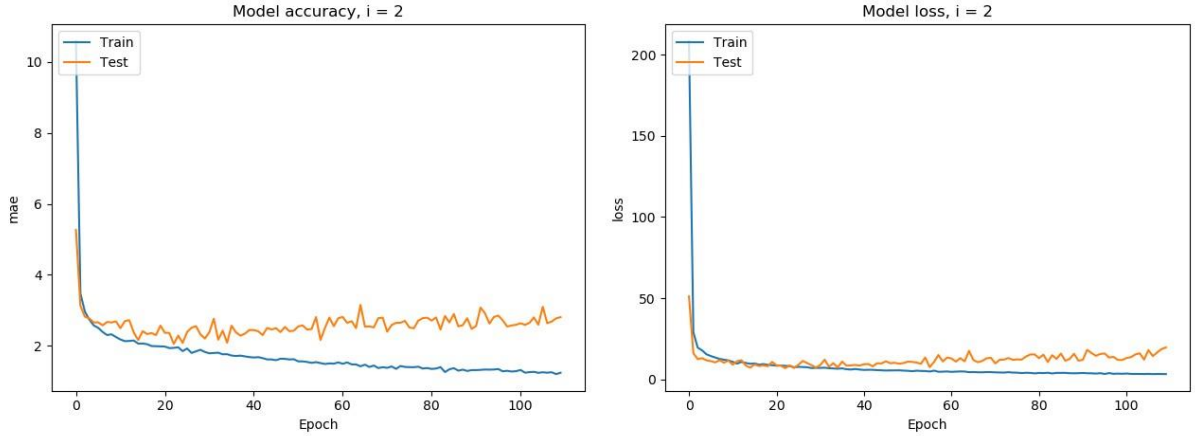

рисунок 4 – график точности и обучения модели на 2-ом блоке

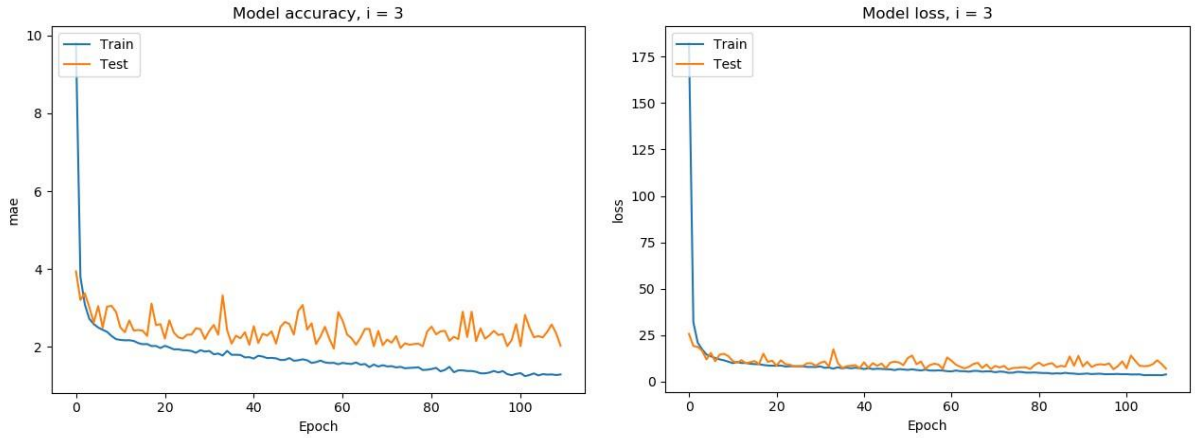

рисунок 5 – график точности и обучения модели на 3-ом блоке

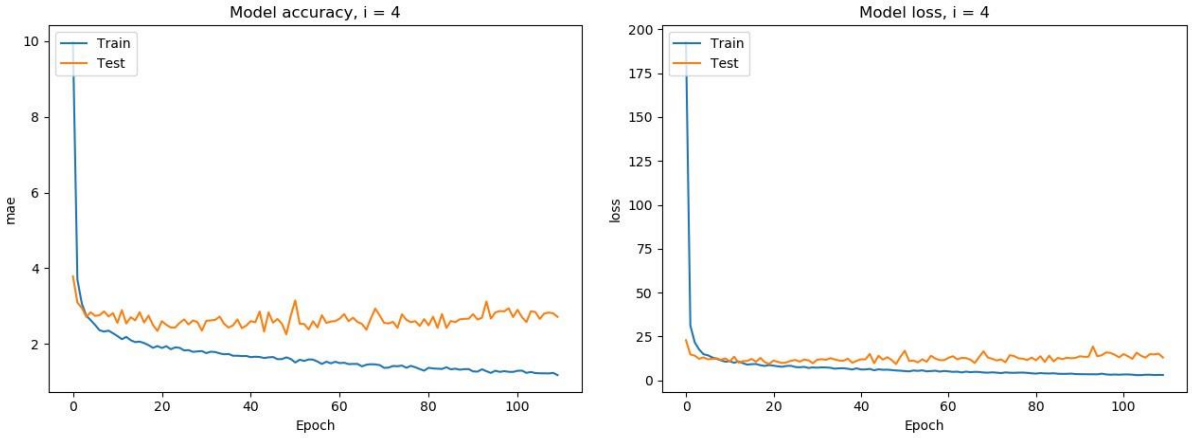

рисунок 6 – график точности и обучения модели на 4-ом блоке

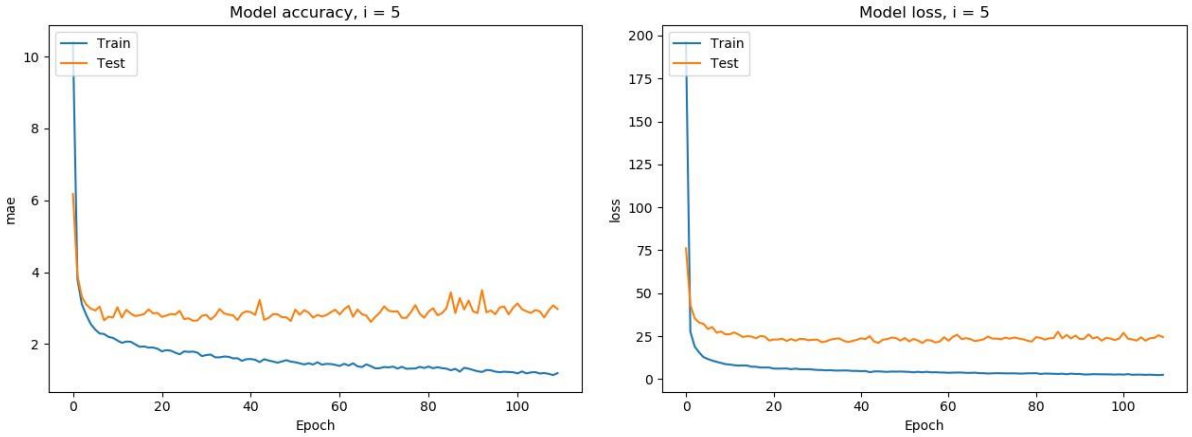

рисунок 7 – график точности и обучения модели на 5-ом блоке

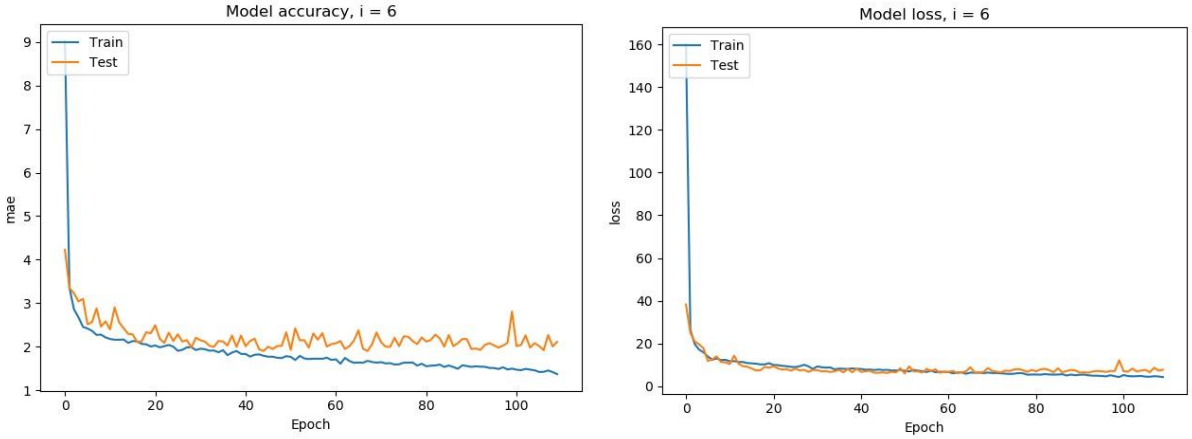

рисунок 8 – график точности и обучения модели на 6-ом блоке

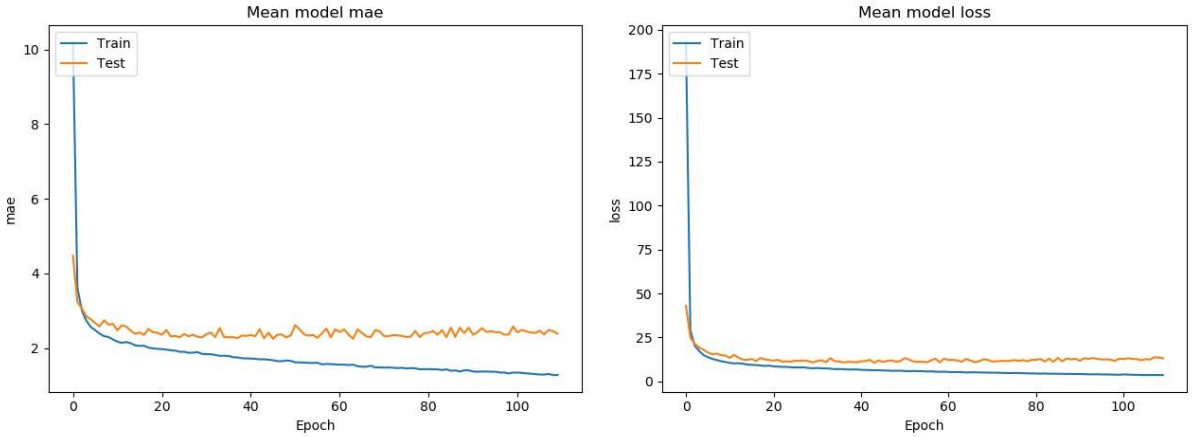

рисунок 9 – график точности и обучения усредненной модели

Вывод.

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

ПРИЛОЖЕНИЕ А ИСХОДНЫЙ

КОД ПРОГРАММЫ

```
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

from tensorflow.keras.datasets import
boston_housing

import numpy as np
import matplotlib.pyplot as plt


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()

mean = train_data.mean(axis=0)
std = train_data.std(axis=0)

train_data -= mean
train_data /= std

test_data -= mean
test_data /= std

k = 7
num_val_samples = len(train_data) // k
num_epochs = 150
all_mae_histories = []
mean_val_mae = []
for i in range(k):
    print(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),
verbose=0)


mean_val_mae.append(history.history['val_mean_
absolute_error'])

plt.plot(history.history['mean_absolute_error'
])

plt.plot(history.history['val_mean_absolute_er
ror'])
    plt.title('Block #' + str(i+1))
    plt.ylabel('mae')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'])
    plt.show()



def smooth_curve(points, factor=0.9):
    smoothed_points = []
    for point in points:
        if smoothed_points:
            prev = smoothed_points[-1]

smoothed_points.append(prev*factor+point*(1-
factor))
        else:
            smoothed_points.append(point)
    return smoothed_points
```

```
average_mae_history = [np.mean([x[i] for x in
mean_val_mae]) for i in range(num_epochs)]
smooth_mae_history =
smooth_curve(average_mae_history)
plt.plot(range(1, len(smooth_mae_history)+1),
smooth_mae_history)
plt.xlabel('EPOCHS')
plt.ylabel("Validation MAE")
plt.show()
```