**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”**

| | | |
|---|---|---|
| Студентка гр. 7381 | | Кревчик А.Б. |
| 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.

**Постановка 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

**Ход работы.**

Рассмотрим различия задач классификации и регрессии.

Классификация - это задача прогнозирования метки дискретного класса.

Регрессия - это задача прогнозирования непрерывного количества.

Для классификационных моделей характерно предсказывать непрерывное значение как вероятность данного примера, принадлежащего каждому выходному классу. Вероятности могут быть интерпретированы как вероятность или достоверность данного примера, принадлежащего каждому классу. Прогнозируемая вероятность может быть преобразована в значение класса путем выбора метки класса, которая имеет наибольшую вероятность.

Алгоритм регрессии может прогнозировать дискретное значение, но дискретное значение в виде целочисленной величины.

Изучим влияние количества эпох на результат обучения модели:

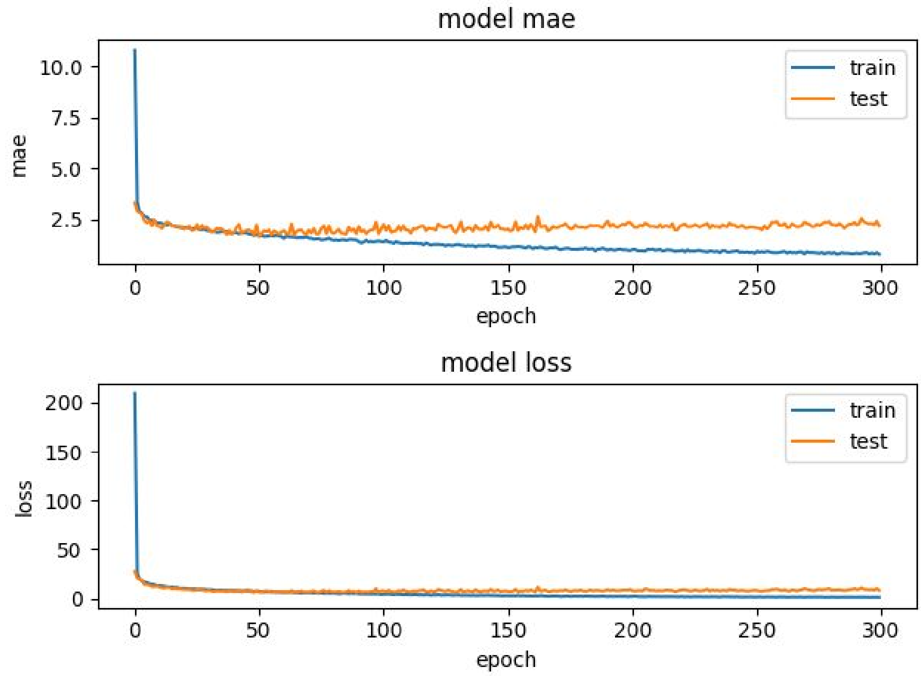


Рисунок 1 - Графики для 1-го блока

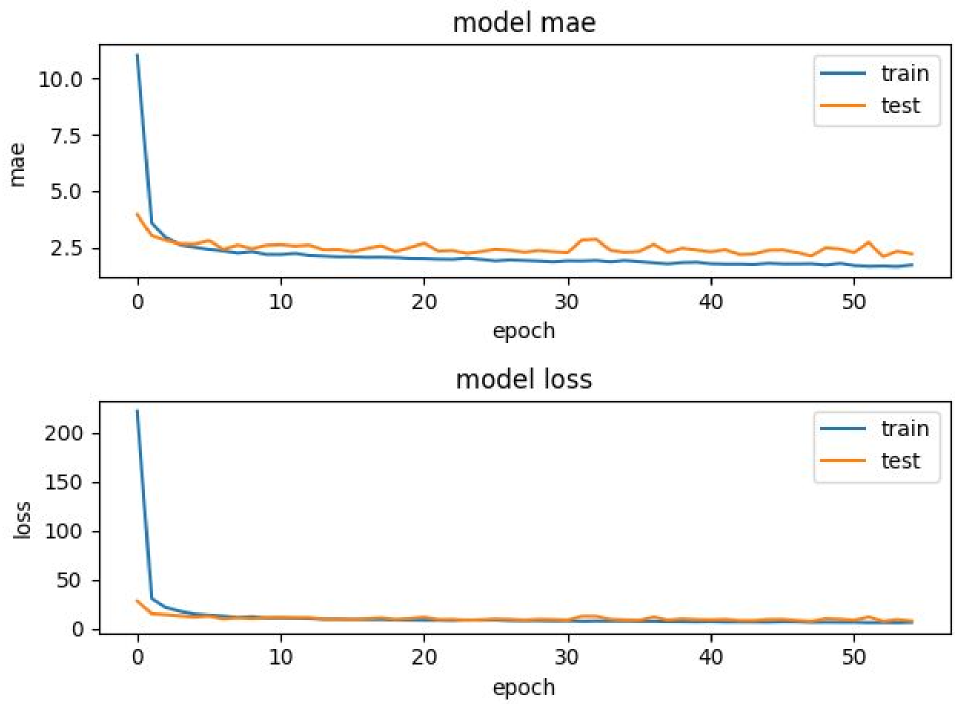


Рисунок 2 - Графики для 2-го блока

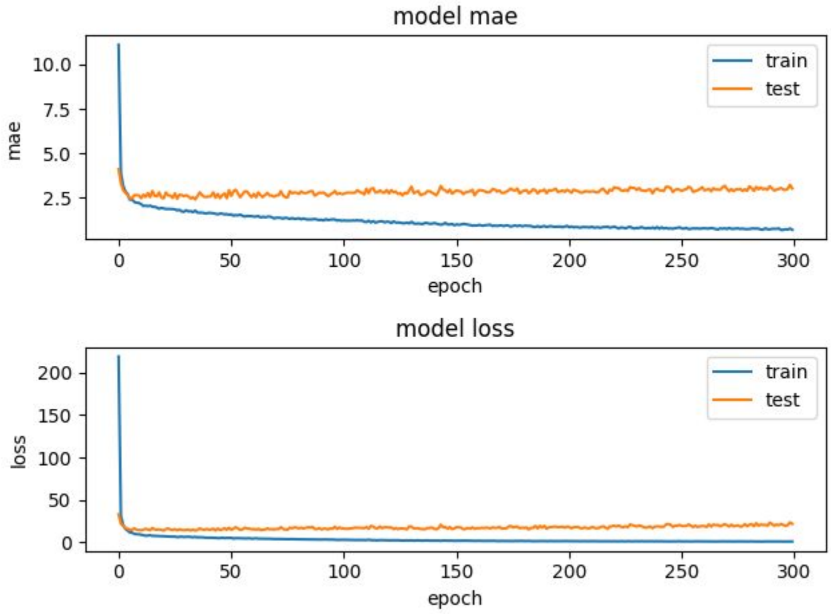


Рисунок 3 - Графики для 3-го блока

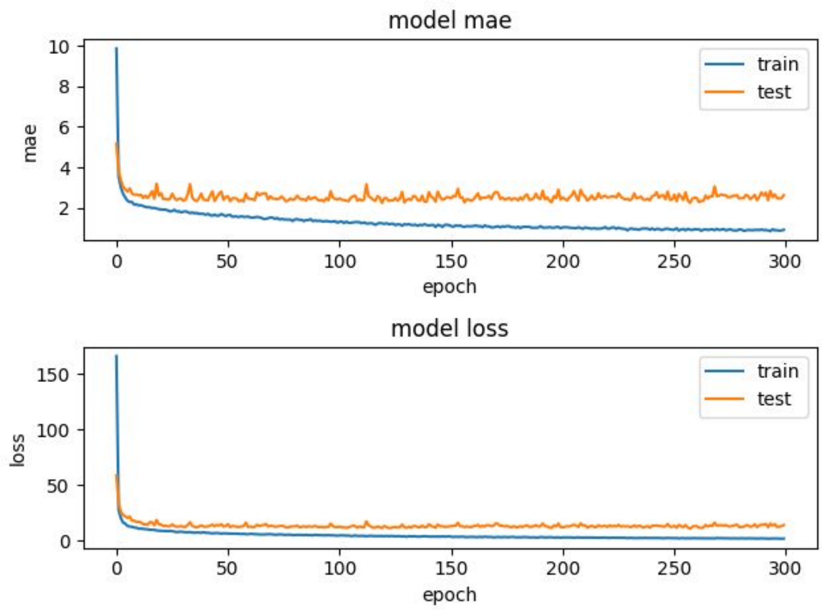


Рисунок 4 - Графики для 4-го блока

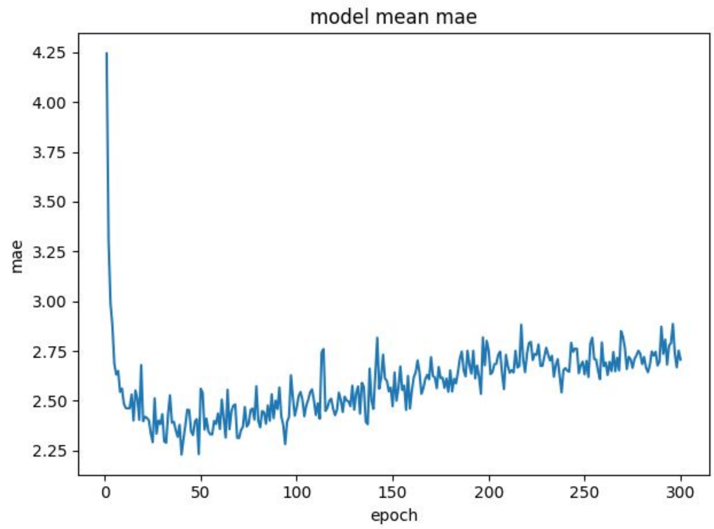


Рисунок 5 - Усредненный по всем моделям график среднеквадратичной ошибки

Проанализировав графики, можно сделать вывод, что точка переобучения находится около 50-ой эпохи, так как после нее среднеквадратичная ошибка растет.

Рассмотрим различное графики ошибок для разного количества К-блоков.

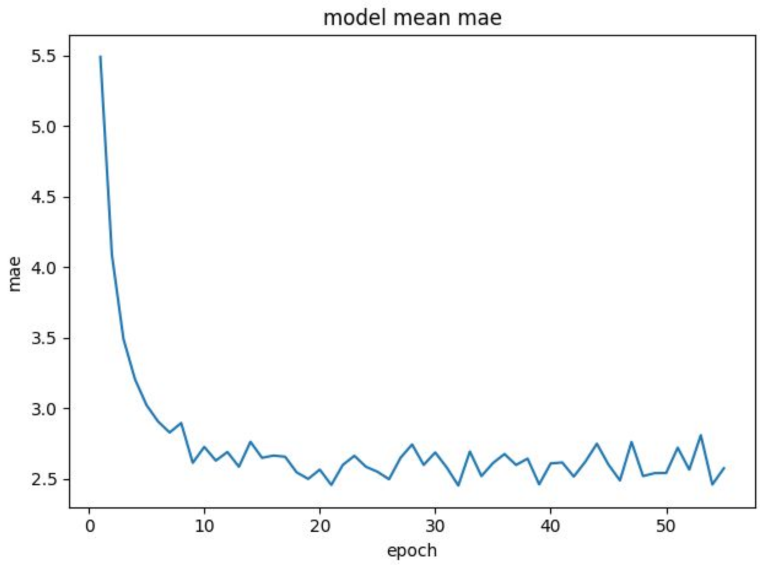


Рисунок 6 - Усредненный по всем моделям график среднеквадратичной ошибки при К = 2

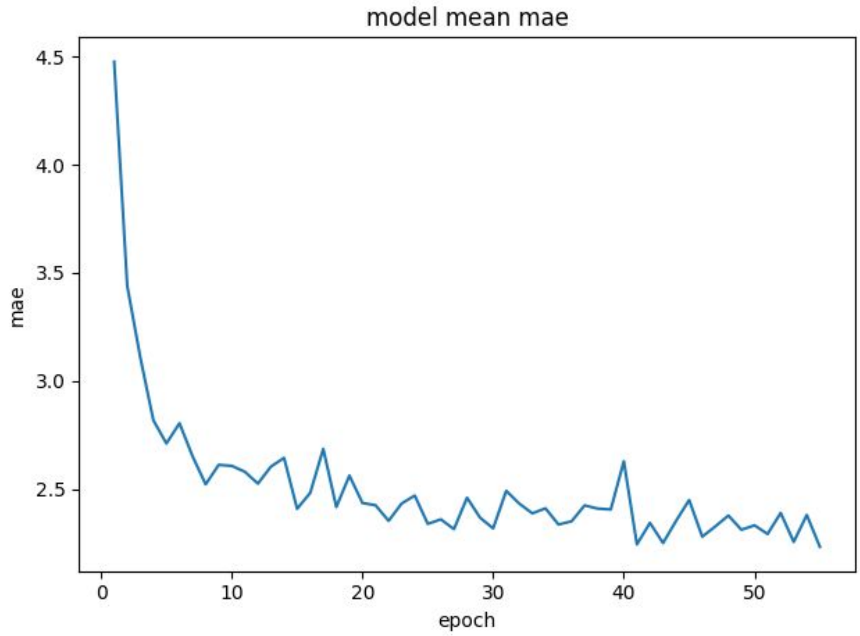


Рисунок 7 - Усредненный по всем моделям график  среднеквадратичной ошибки при K = 4

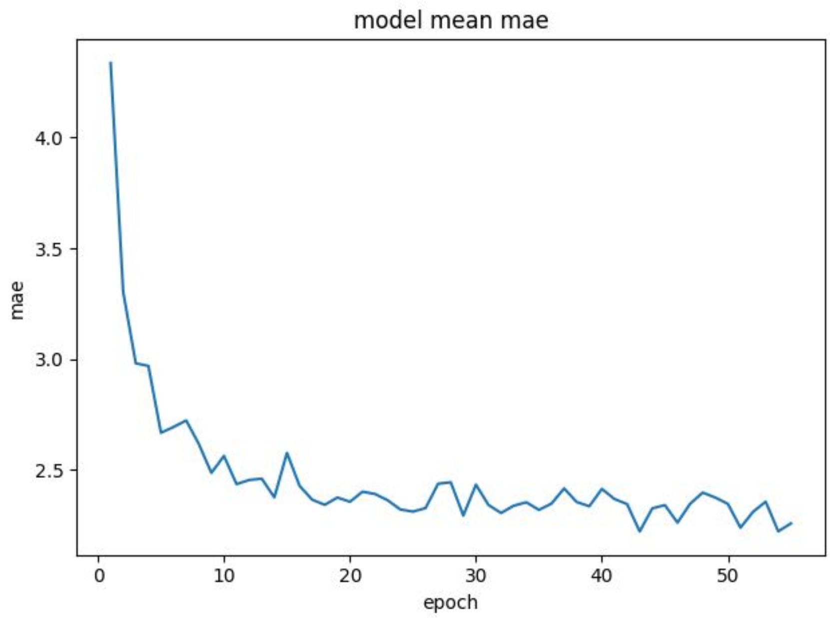


Рисунок 8 - Усредненный по всем моделям график  среднеквадратичной ошибки при K = 6

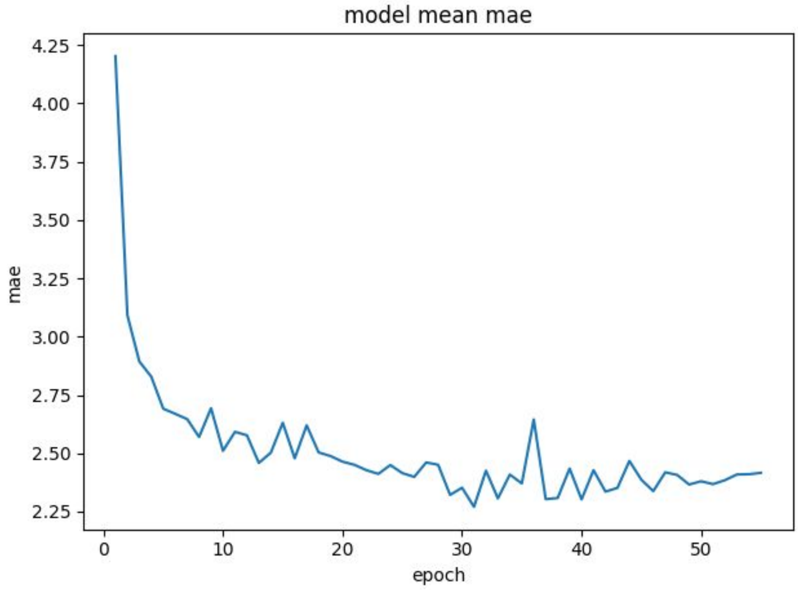


Рисунок 9 - Усредненный по всем моделям график среднеквадратичной ошибки при К = 8

Проанализировав графики, можно сделать вывод, что наименьшее среднеквадратичное отклонение получается при К = 6.

**Выводы.**

В ходе выполнения лабораторной работы была изучена задача прогнозирующего регрессионного моделирования. Также была изучена перекрёстная проверка по К блокам.

## Приложение А
## Исходный код программы

```
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
import matplotlib.pyplot as plt

(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


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 = 4
num_val_samples = len(train_data) // k
num_epochs = 15
all_scores = []
mae = []

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, validation_data=(val_data,
val_targets),verbose=0)
        val_mse, val_mae = model.evaluate(val_data, val_targets,
verbose=0)
        all_scores.append(val_mae)

        mae.append(history.history['val_mae'])
        plt.subplot(211)
        plt.plot(history.history['mae'])
        plt.plot(history.history['val_mae'])
        plt.title('model mae')
        plt.ylabel('mae')
        plt.xlabel('epoch')
        plt.legend(['train', 'test'], loc='upper right')

        plt.subplot(212)
        plt.plot(history.history['loss'])
        plt.plot(history.history['val_loss'])
        plt.title('model loss')
        plt.ylabel('loss')
        plt.xlabel('epoch')
        plt.legend(['train', 'test'], loc='upper right')
        plt.show()



    print(np.mean(all_scores))
    mean_mae_history = [np.mean([x[i] for x in mae]) for i in
range(num_epochs)]
    plt.plot(range(1, num_epochs + 1), mean_mae_history)
    plt.title('model mean mae')
    plt.ylabel('mae')
    plt.xlabel('epoch')
    plt.show()
```