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

Студент гр. 7383 _____ Власов Р.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.

Порядок выполнения работы.

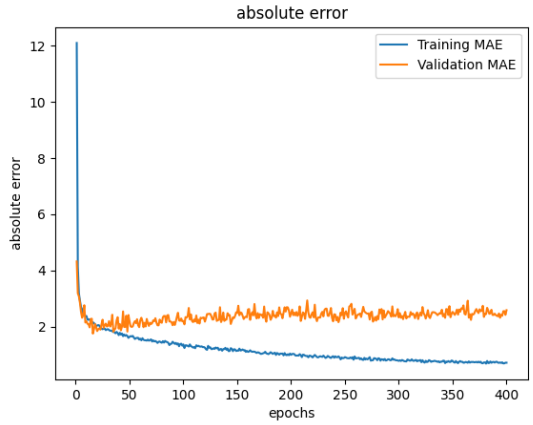
1. Ознакомиться с задачей регрессии
2. Изучить отличие задачи регрессии от задачи классификации
3. Загрузить данные
4. Создать модель
5. Настроить параметры обучения
6. Обучить и оценить модель
7. Ознакомиться с перекрестной проверкой

Ход работы.

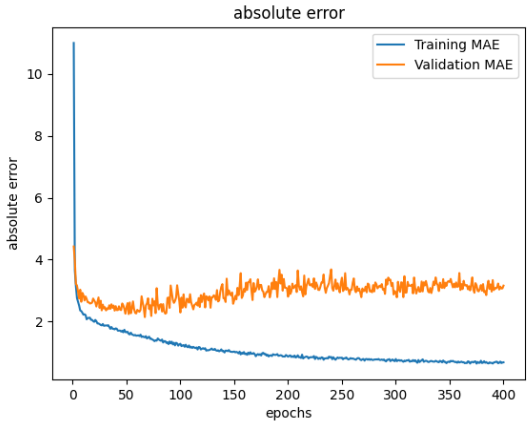
Для изучения регрессионной модели ИНС была разработана и использована программа. Код программы приведен в приложении А.

В задаче классфикации предсказывается принадлежность объекта к одному из заданных классов, причем набор значений ограничен. В задачи регрессии предсказывается некоторая характеристика объекта, значения которой не ограничены

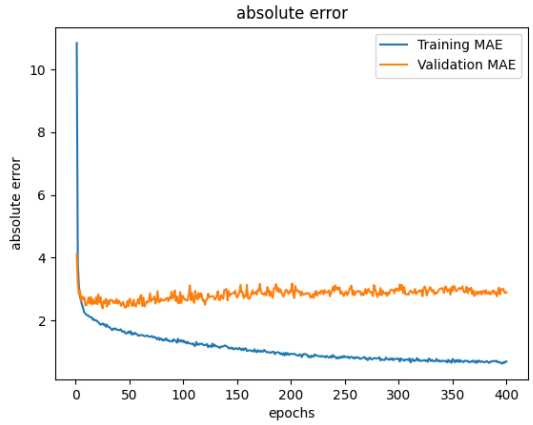
Изначально была рассмотрена модель с перекрестной проверкой на 4 блоках и 400 эпохами. Графики оценки MAE для каждого блока приведены на рис. 1. График средних значений MAE, приведен на рис. 2. Из графика видно, что значение оценки MAE на проверочных данных начинает возрастать примерно после 25 эпохи, тогда как на тестовых данных оно продолжает уменьшаться — это символизирует о том, что начинается переобучение нейронной сети.



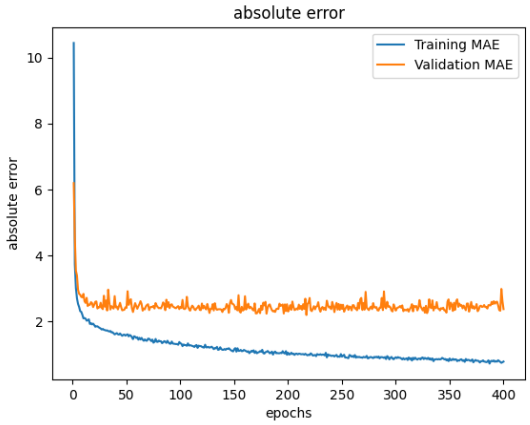
(а)



(б)



(в)



(г)

Рисунок 1 — График оценки MAE для блока (а) 1 (б) 2 (в) 3 (г) 4

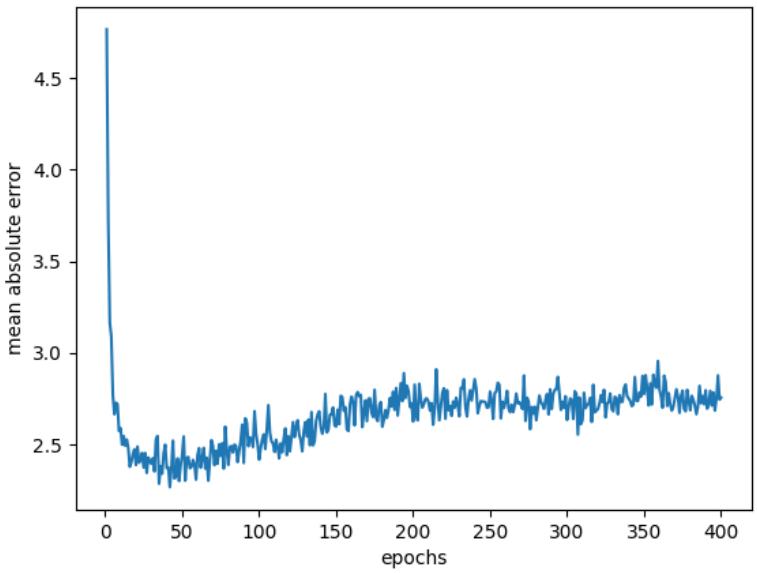
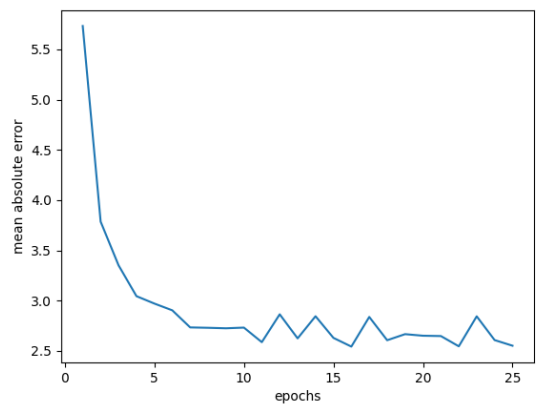
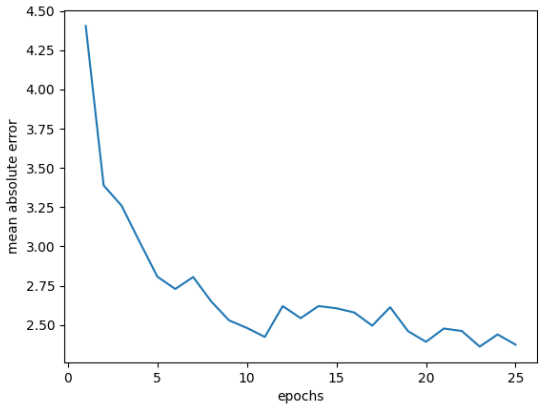


Рисунок 2 — График среднего значения MAE для модели с 400 эпохами и 4 блоками для перекрестной проверки

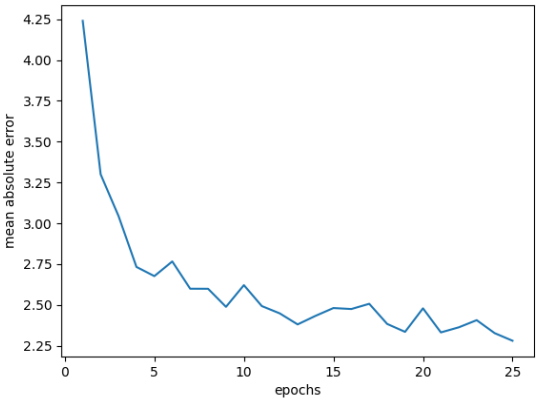
Далее рассмотрим модели с 25 эпохами и 2, 4, 6 и 8 блоками для перекрестной проверки. Графики среднего значения оценки MAE для этих моделей приведены на рис. 3. Из графиков видно, что наилучшие значения средней оценки MAE достигается в модели с 6 блоками перекрестной проверки, а наихудшие — в модели с 2 блоками.



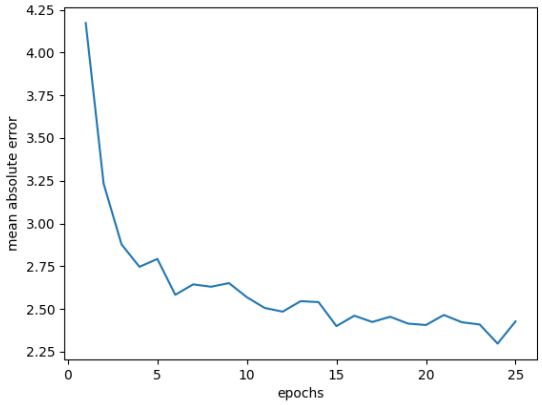
(а)



(б)



(в)



(г)

Рисунок 3 — Графики среднего значения MAE для моделей с 25 эпохами и (а) 2 блоками перекрестной проверки (б) 4 блоками перекрестной проверки (в) 6 блоками перекрестной проверки (г) 8 блоками перекрестной проверки

Выводы.

В ходе выполнения данной работы была изучена задача регрессии и ее отличие от задачи классификации. Была изучена и проведена перекрестная проверка модели.

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 = 8
num_val_samples = len(train_data) // k
num_epochs = 25
mae_histories = []
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))
    mae = history.history['mae']
```

```
    v_mae = history.history['val_mae']
    x = range(1, num_epochs + 1)
    mae_histories.append(v_mae)
    plt.figure(i + 1)
    plt.plot(x, mae, label='Training MAE')
    plt.plot(x, v_mae, label='Validation MAE')
    plt.title('absolute error')
    plt.ylabel('absolute error')
    plt.xlabel('epochs')
    plt.legend()
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)
plt.xlabel('epochs')
plt.ylabel("mean absolute error")
figs = [plt.figure(n) for n in plt.get_fignums()]
for i in range(len(figs)):
    figs[i].savefig("./Graphics/%d.png" %(i), format='png')
```