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. 8383 _____ Костарев К.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

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.

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.

Регрессия — это процесс поиска модели или функции для разделения данных на непрерывные реальные значения вместо использования классов. Математически, с проблемой регрессии, каждый пытается найти приближение функции с минимальным отклонением ошибки. В регрессии предсказывается, что числовая зависимость данных будет отличать ее.

Классификация — это процесс поиска или обнаружения модели (функции), которая помогает разделить данные на несколько категориальных классов. При классификации определяется членство группы в проблеме, что означает, что данные классифицируются под разными метками в соответствии с некоторыми параметрами, а затем метки прогнозируются для данных.

Данные подходы различаются тем, что процесс классификации моделирует функцию, с помощью которой данные прогнозируются в метках дискретных классов, а регрессия — это процесс создания модели, которая предсказывает непрерывное количество.

Построим модель, состоящую из трех слоев, где первый и второй имеют по 64 нейрона с функцией активации relu, и запустим ее со 100 эпохами, разделяя данные на 4 блока. На рис. 1-8 показаны графики средней абсолютной ошибки (mae) и среднего квадратичного отклонения (mse) для каждого блока.

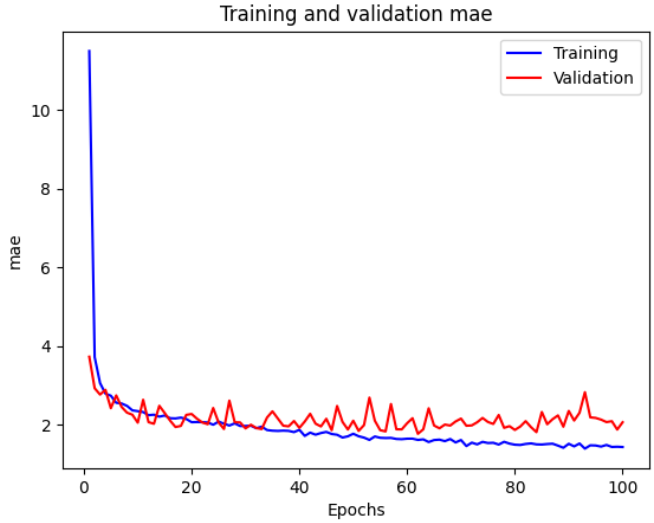

Рисунок 1 – mae первого блока

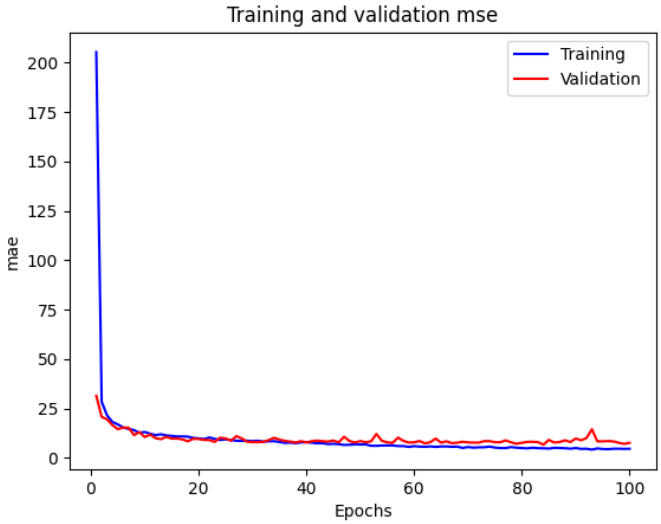

Рисунок 2 – mse первого блока

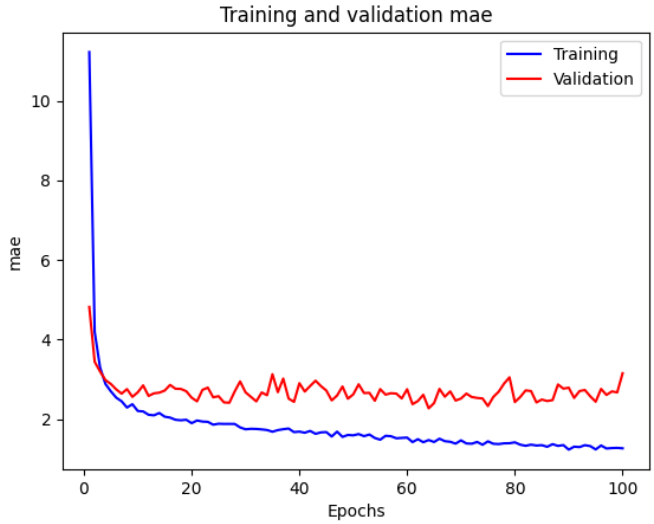

Рисунок 3 – mae второго блока

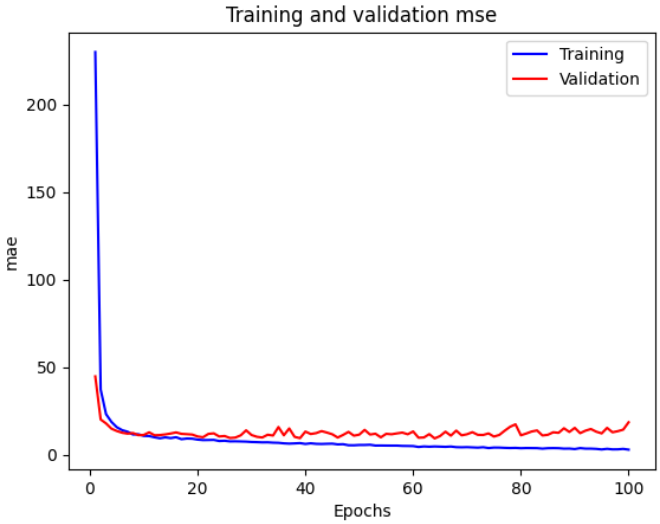

Рисунок 4 – mse второго блока

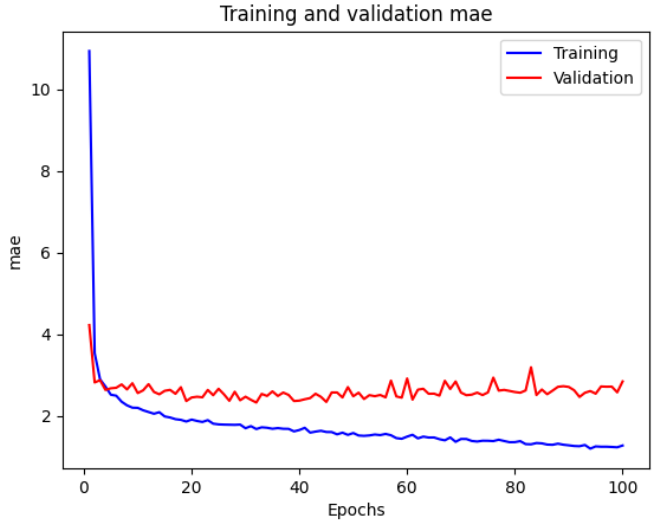

Рисунок 5 – mae третьего блока

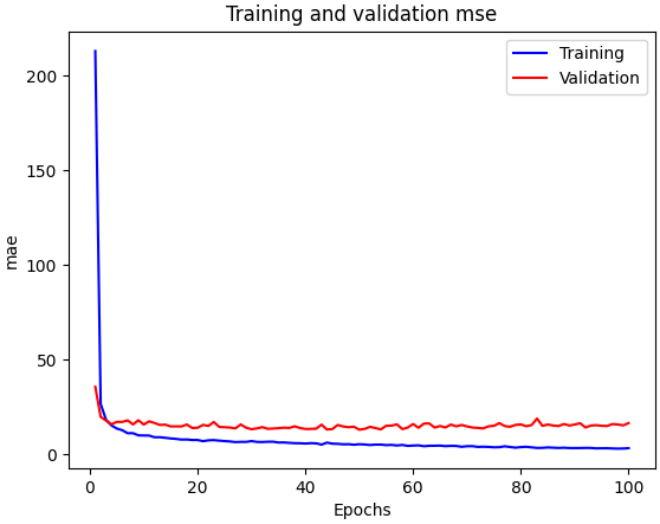

Рисунок 6 – mse третьего блока

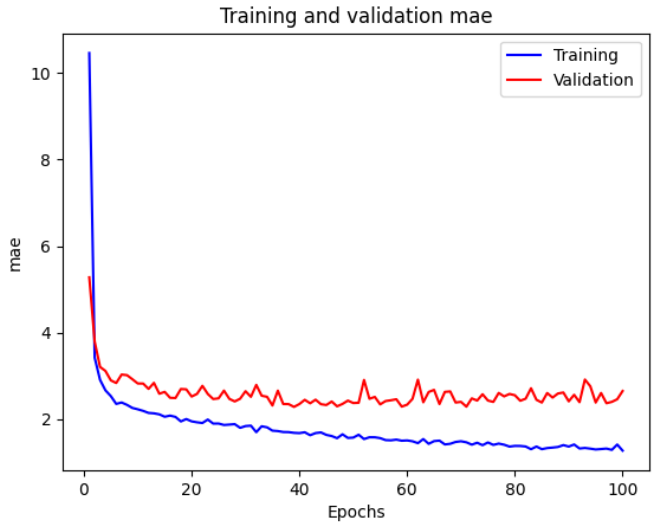

Рисунок 7 – mae четвертого блока

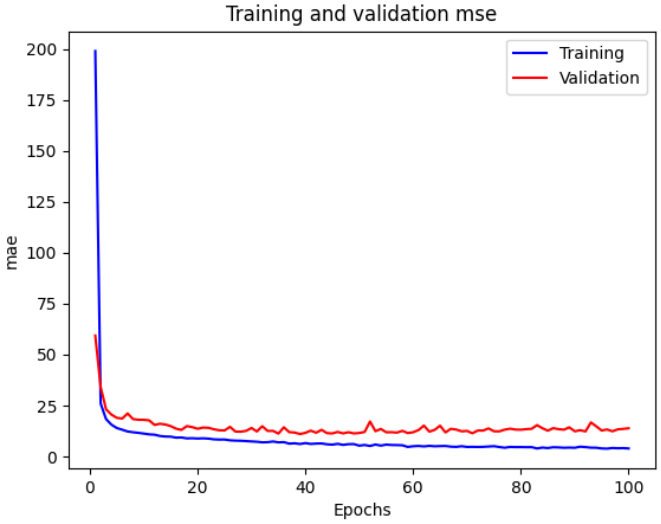

Рисунок 8 – mse четвертого блока

На рис. 9 представлен график средней абсолютной ошибки по всем блокам.

Оценка модели равна 2.678699314594269.

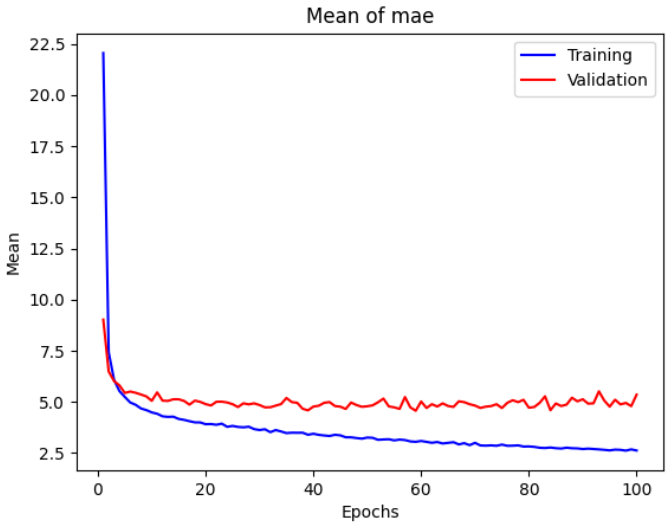

Рисунок 9 - mae по всем блокам

Можно заметить, что после 30 эпох ошибка не уменьшается. Оценим, что 35 эпоха это точка переобучения, поэтому запустим модель с 35 эпохами и 4 блоками.

На рис. 10 показан график средней абсолютной ошибки по всем блокам. Оценка модели равна 2.64430233001709.

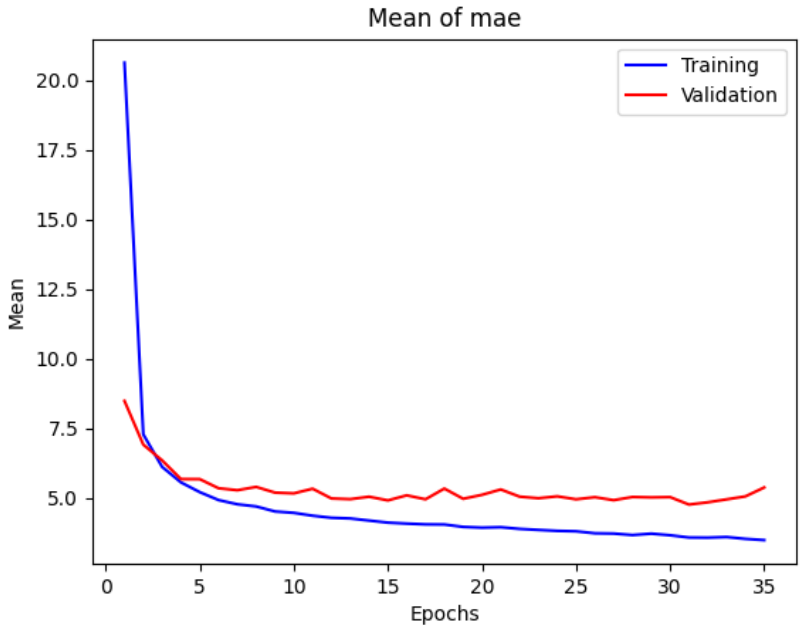

Рисунок 10 - Средняя абсолютная ошибка по всем блокам при 35 эпохах

Заметим, что значение ошибки уменьшилось, и более нет признаков переобучения, поэтому уменьшение количества эпох сказалось положительным образом.

Теперь изменим количество блоков до 3. Количество эпох оставим прежним. График ошибки по всем блокам представлен на рис. 11.

Оценка модели равна 2.451664845148722.

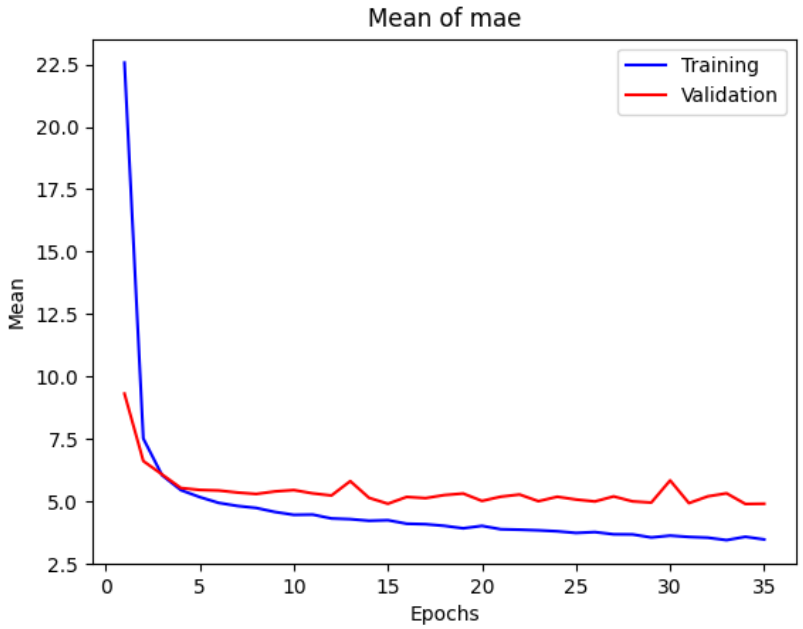

Рисунок 11 - mae по всем блокам при 3 блоках

Уменьшения ошибки добиться не удалось.

Увеличим число блоков до 5. График ошибки по всем блокам представлен на рис. 12.

Оценка модели равна 2.376707911491394.

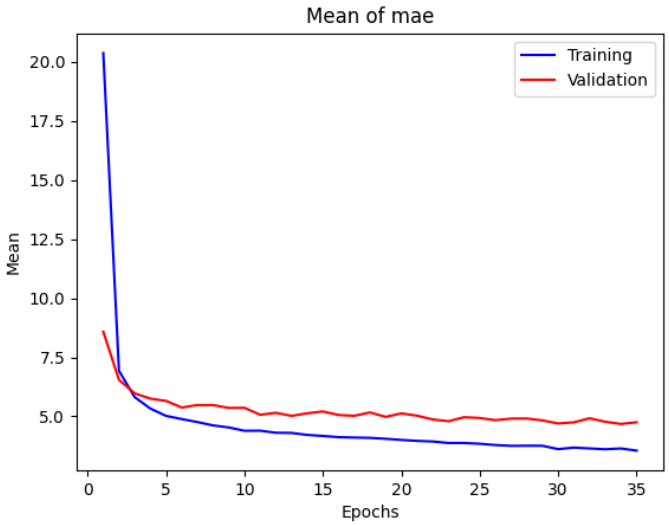

Рисунок 12 – mae по всем блокам при 5 блоках

Оценка стала лучше, хотя значение ошибки по всем блокам не уменьшилось.

Увеличим количество блоков до 7. График ошибки по всем блокам представлен на рис. 13.

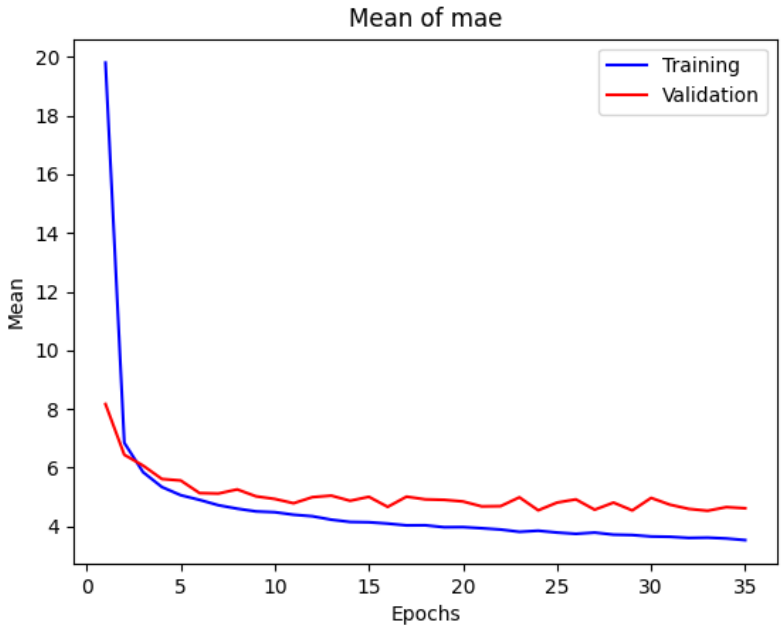

Рисунок 13 - mae по всем блокам при 7 блоках

Оценка модели равна 2.308245931352888.

Вывод.

Во время выполнения лабораторной работы была создана ИНС, которая строит регрессионную предсказание медианной цены на дома в пригороде Бостона, так же можно сделать вывод, что оптимальной является модель с 6 блоками перекрестной проверки и 40 эпохами обучения.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

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


(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 = 7
num_val_samples = len(train_data) // k
num_epochs = 35
all_scores = []
all_mae = np.zeros(num_epochs)
all_val_mae = np.zeros(num_epochs)
epochs = range(1, num_epochs + 1)

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

```
    all_mae += np.array(history['mae'])
    all_val_mae += np.array(history['val_mae'])
    all_scores.append(val_mae)

    all_mae += np.array(history['mae'])
    all_val_mae += np.array(history['val_mae'])

    plt.plot(epochs, history['mae'], 'b', label='Training')
    plt.plot(epochs, history['val_mae'], 'r', label='Validation')
    plt.title('Training and validation mae')
    plt.xlabel('Epochs')
    plt.ylabel('mae')
    plt.legend()
    plt.show()
    plt.clf()

    plt.plot(epochs, history['loss'], 'b', label='Training')
    plt.plot(epochs, history['val_loss'], 'r', label='Validation')
    plt.title('Training and validation mse')
    plt.xlabel('Epochs')
    plt.ylabel('mae')
    plt.legend()
    plt.show()


plt.plot(epochs, all_mae / k, 'b', label='Training')
plt.plot(epochs, all_val_mae / k, 'r', label='Validation')
plt.title('Mean of mae')
plt.xlabel('Epochs')
plt.ylabel('Mean')
plt.legend()
plt.show()

print(np.mean(all_scores))
```