МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Искусственные нейронные сети»

Тема: Предсказание медианной цены на дома в пригороде Бостона

Студентка гр. 8382 _____ Бердникова А. А.

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.

Не путайте регрессию с алгоритмом логистической регрессии. Как ни странно, логистическая регрессия не является регрессионным алгоритмом — это алгоритм классификации.

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

Требования:

- Объяснить различия задач классификации и регрессии
- 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

Ход работы.

1. Загрузка файла с данными

Набор данных присутствует в составе Keras. 404 обучающих и 102 контрольных образца, каждый с 13 числовыми признаками.

Цены в основном находятся в диапазоне от 10 000 до 50 000 долларов США.

2. Написание программы

С помощью методических указаний была построена нейросеть, имеющая следующую архитектуру слоёв:

1) Входной слой из 1 нейрона
2) Скрытый слой из 64 нейронов и функции активации relu
3) Скрытый слой из 64 нейронов и функции активации relu
4) Выходной слой из одного нейрона, не имеющий функции активации

Сеть заканчивается одномерным слоем, не имеющим функции активации (это линейный слой). Это типичная конфигурация для скалярной регрессии (целью которой является предсказание одного значения на непрерывной числовой прямой). Применение функции активации могло бы ограничить диапазон выходных значений: например, если в последнем слое применить функцию активации sigmoid, сеть обучилась бы предсказывать только значения из диапазона между 0 и 1.

В данном случае, с линейным последним слоем, сеть способна предсказывать значения из любого диапазона.

Сеть компилируется с функцией потерь mse — mean squared error (среднеквадратичная ошибка), вычисляющей квадрат разности между предсказанными и целевыми значениями. Эта функция широко используется в задачах регрессии. Также добавлен новый параметр на этапе обучения: mae — mean absolute error (средняя абсолютная ошибка). Это абсолютное значение разности между предсказанными и целевыми значениями. Например, значение MAE, равное 0,5, в этой задаче означает, что в среднем прогнозы отклоняются на 500 долларов США.

Так как предоставленный набор данных имеет небольшой объем, то при разбиении его на обучающий и проверочный наборы, проверочный может оказаться очень маленького объема (~ 100 образцов). Из-за этого оценки при проверке могут сильно меняться в зависимости от данных, попавших в проверочный и обучающий наборы (оценки будут иметь большой разброс).

Поэтому применим перекрёстную проверку к исходным данным, разбив их на 4 и 5 блоков, создав 4(5) идентичных моделей сетей и обучив каждую на 3(4) наборах с оценкой по оставшимся блокам.

Проанализируем результаты, показываемые нейросетью, при этом изменяя количество эпох обучения. Результаты приведены ниже.

1-ый запуск:

Количество блоков: 4

Количество эпох: 100

Средняя ошибка: 2.425348699092865

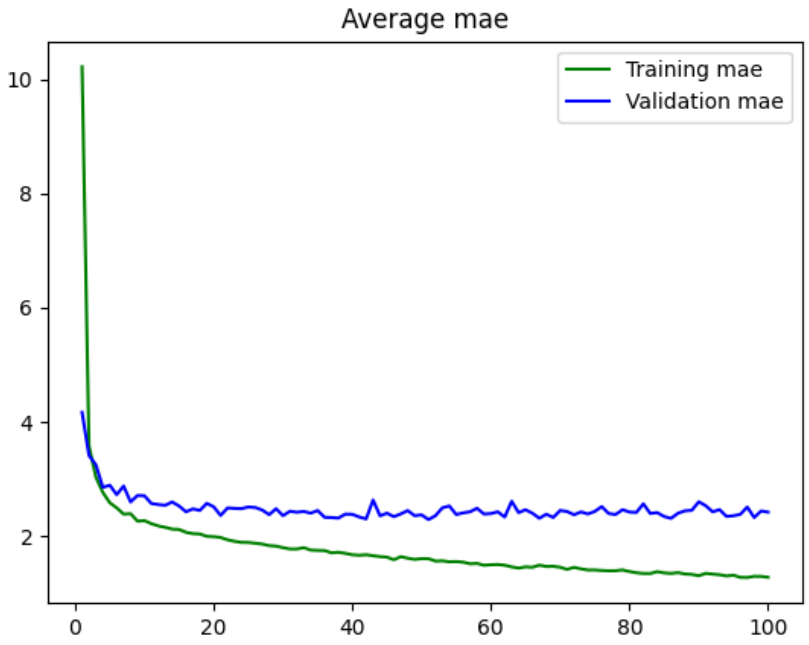

Рисунок 1 – средняя mae при 1-ом запуске

2-ой запуск:

Количество блоков: 4

Количество эпох: 200

Средняя ошибка: 2.6897128224372864

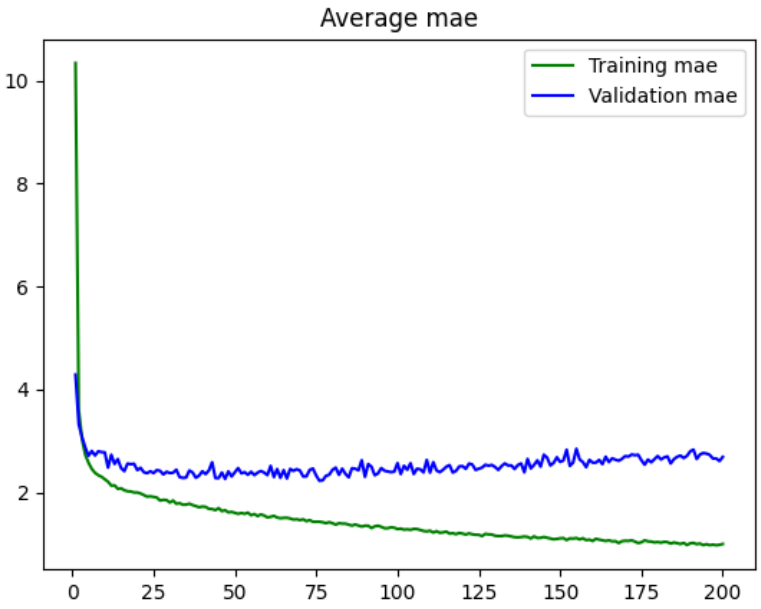

Рисунок 2 – средняя mae при 2-ом запуске

3-ий запуск:

Количество блоков: 4

Количество эпох: 50

Средняя ошибка: 2.479828655719757

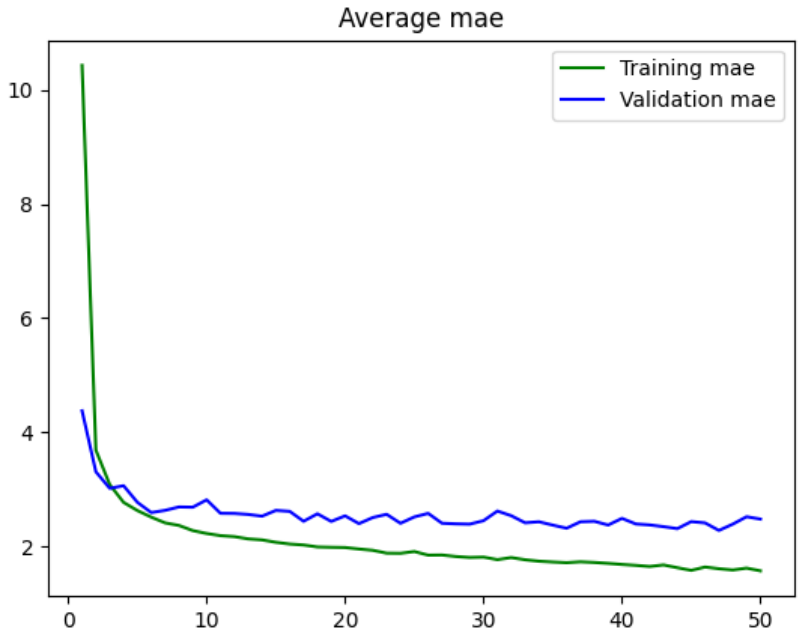

Рисунок 3 – средняя mae при 3-ем запуске

4-ый запуск:

Количество блоков: 5

Количество эпох: 100

Средняя ошибка: 2.4098995685577393

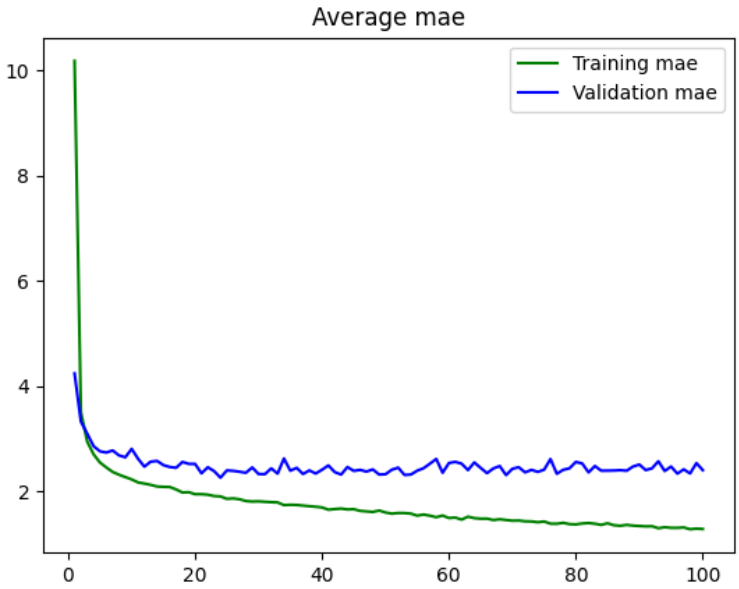

Рисунок 4 – средняя mae при 4-ом запуске

5-ый запуск:

Количество блоков: 5

Количество эпох: 200

Средняя ошибка:

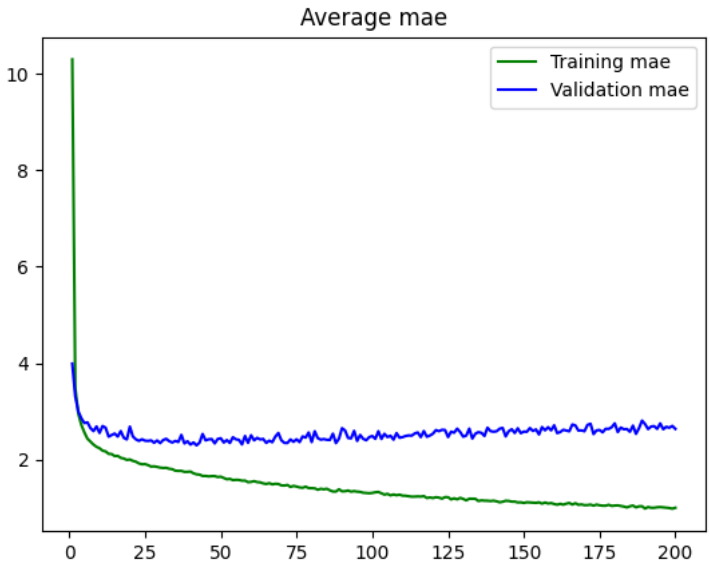

Рисунок 5 – средняя mae при 5-ом запуске

6-ой запуск:

Количество блоков: 5

Количество эпох: 50

Средняя ошибка:

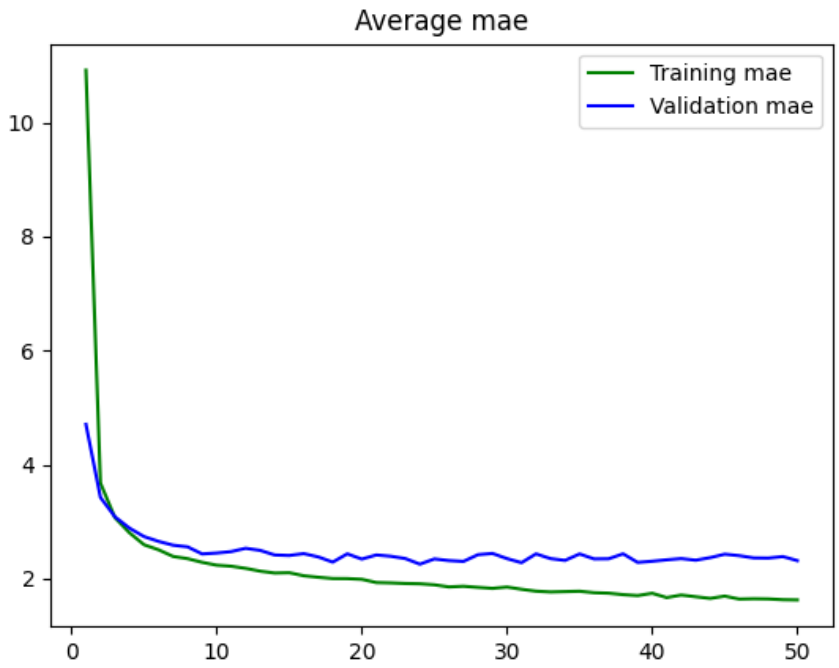

Рисунок 6 – средняя mae при 6-ом запуске

Вывод.

В ходе выполнения лабораторной работы было реализовано 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. Применена перекрестная проверку по K блокам при разном количестве эпох и блоков. Были построены графики ошибки и точности во время обучения для моделей, а также усредненные графики по всем моделям.

ПРИЛОЖЕНИЕ

В данном приложении приведён исходный код программы.

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

def print_average(mae_arr, val_mae_arr):
    avg_mae = np.average(mae_arr, axis=0)
    avg_val_mae = np.average(val_mae_arr, axis=0)
    epochs = range(1, len(avg_mae) + 1)
    plt.plot(epochs, avg_mae, color='green', label='Training mae')
    plt.plot(epochs, avg_val_mae, color='blue', label='Validation mae')
    plt.title('Average mae')
    plt.legend()
    plt.show()

def run(k=4, num_epochs=100):
    num_val_samples = len(train_data) // k
    all_scores = []
    mae_arr = []
    val_mae_arr = []

    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))

        val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=1)
        all_scores.append(val_mae)
        mae_arr.append(history.history['mae'])
        val_mae_arr.append(history.history['val_mae'])
    print("k=", k, "num_epochs=", num_epochs)
    print(np.mean(all_scores))
    print_average(mae_arr, val_mae_arr)


run(5, 50)
```