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. 8383 _____ Кормщикова А.О.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы.

Реализовать предсказание медианой 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.

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

Отличие задачи регрессии от задачи классификации:

Задача классификации состоит в получении категориального ответа на основе признаков. Такая задача имеет конечное число ответов.

Задача регрессии - прогноз на основе выборки объектов с различными признаками. На выходе должно получиться вещественное число, к примеру цена квартиры, стоимость ценной бумаги по прошествии полугода и др.

Были импортированы необходимые для работы классы и функции.

```
import matplotlib.pyplot as plt
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
```

Т.к. в нейронную сеть проблематично передать значения, имеющие самый разные диапазоны, к данным была применена нормализация: для каждого признака во входных данных (столбца в матрице входных данных) из каждого значения вычитается среднее по этому признаку, и разность делится на стандартное отклонение, в результате признак центрируется по нулевому значению и имеет стандартное отклонение, равное единице.

```
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /= std
test_data -= mean
test_data /= std
```

Была построена следующая модель ИНС:

```
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'])
```

Сеть заканчивается одномерным слоем, не имеющим функции активации, это типичная конфигурация для скалярной регрессии. Сеть способна предсказывать значения из любого диапазона.

Т.к. набор данных небольшой, разбитие исходных данных на обучающий и проверочный наборы было бы не эффективно. Была применена перекрестная проверка по K блокам. Данные разделяются на K блоков, создаются K идентичных моделей, которые обучаются на K-1 блоках с оценкой по оставшимся. По полученным K оценкам вычисляется среднее значение, которое принимается как оценка модели.

В представленном наборе данных присутствует некоторая избыточность, т.к. с разных углов описывается один и тот же сигнал. Вероятно, что некоторые углы отражения сигнала имеют большую значимость, чем другие.

Протестируем модель на 100 эпохах и K= 4. Полученные результаты средних абсолютных ошибок всех моделей представлены на рис. 1. Так же представлены графики среднего значения средней абсолютной ошибки и среднего значения среднеквадратичной ошибки на рис. 2

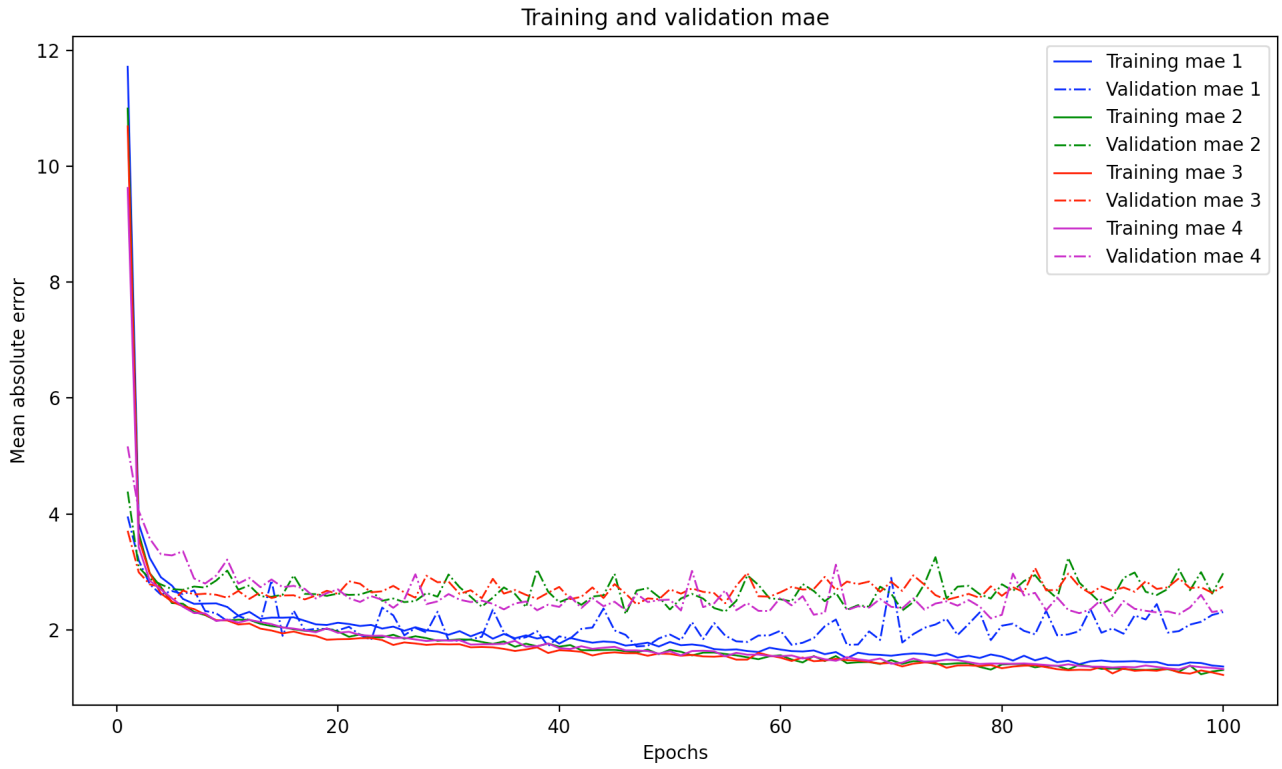

Рисунок 1 - График оценки средней абсолютной ошибки для 4х моделей

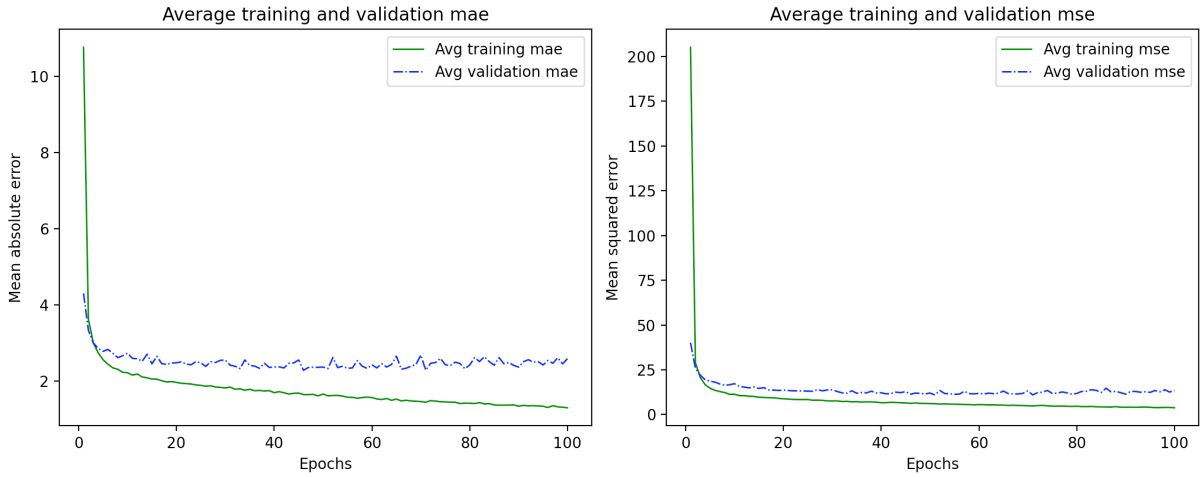

Рисунок 2 - Графики среднего значения mae и mse

На графиках среднего значения средней абсолютной ошибки видно, что на проверочных данных значение после 20 эпохи не сильно колебалось.

Рассматривая рисунок 1, можно заметить, что после 40 эпохи значение средней абсолютной ошибки на проверочных данных у первой модели начали расти, у остальных моделей значения ошибки вырастают после 60 эпохи. В основном модели ведут себя приблизительно одинаково, первая показывала лучшие результаты. Однако в период обучения ошибки для всех моделей падают, что наталкивает на вывод о том, что модель склонна к переобучению в районе 40-60 эпох.

Было уменьшено количество эпох до 40. На рис.3-4 представлены полученные результаты

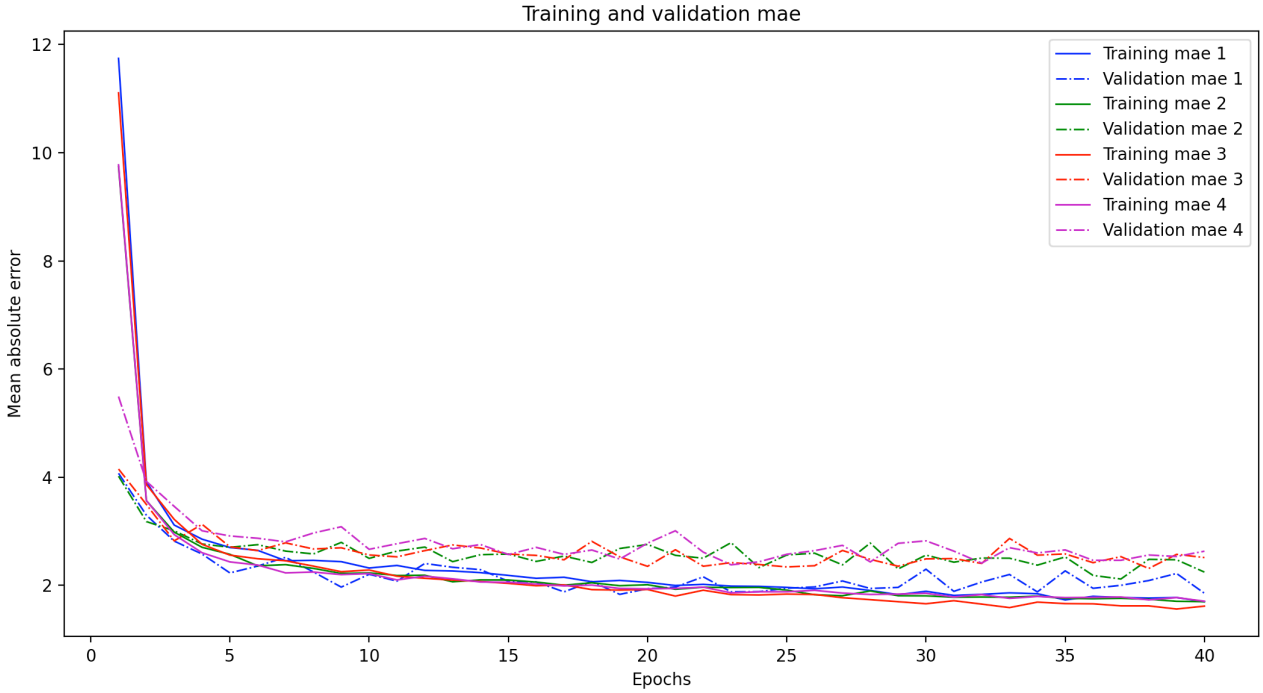

Рисунок 3 - График оценки средней абсолютной ошибки для 4х моделей

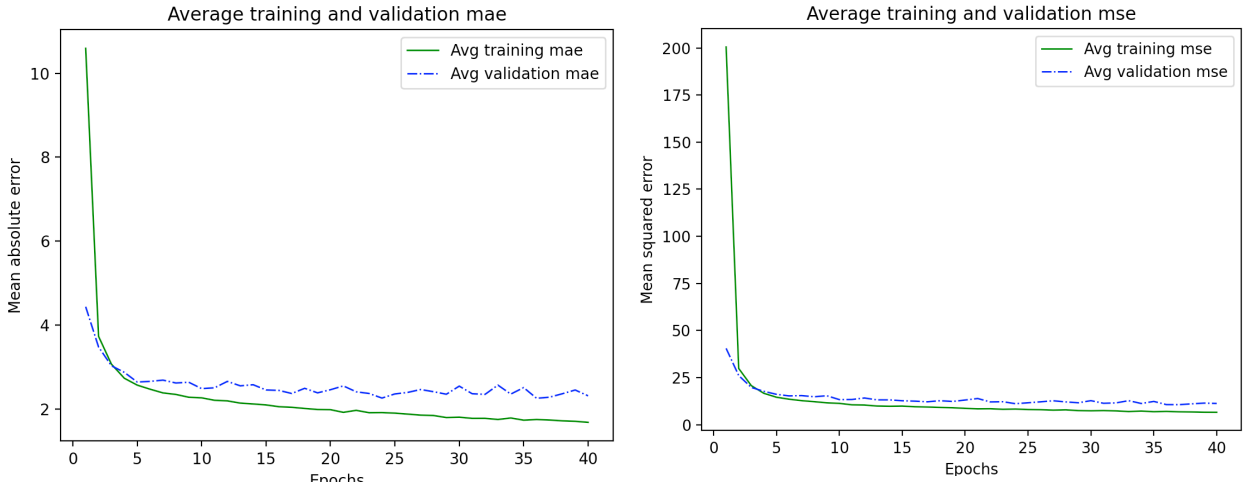

Рисунок 4 - Графики среднего значения mae и mse

Оценка модели составляет 2.5509389638900757

На графиках среднего значения средней абсолютной ошибки видно, что значения не возрастают, переобучение не происходит. На общем графике у первой модели после 30 эпохи ошибки имеют свойство возрастать, однако затем снова падают. Можно сделать вывод, что уменьшение количества эпох оптимизировало модель, избавив ее от лишних вычислений и переобучения.

Было изменено количество блоков разбиения. Графики среднего значения средней абсолютной ошибки показаны на рис. 5-7. Новые параметры K = 2,6,8

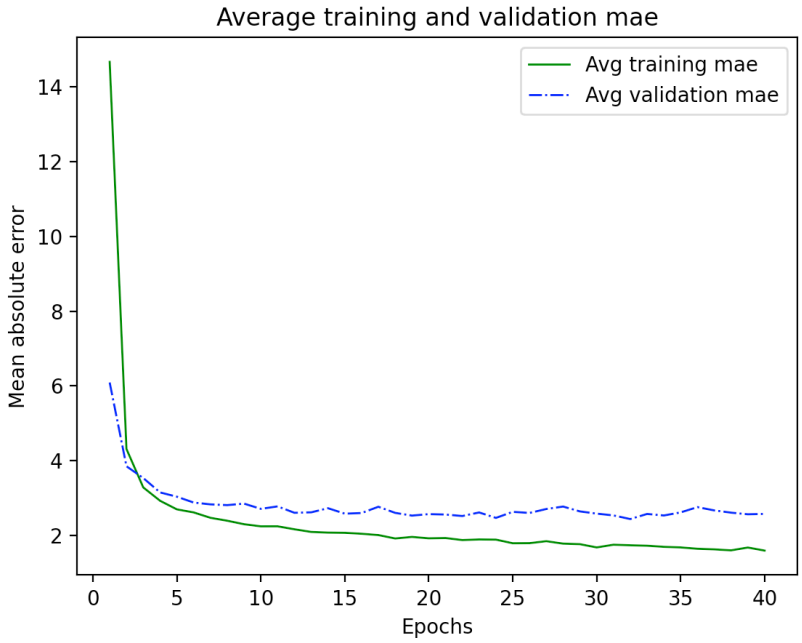

Рисунок 5 - Графики среднего значения mae при K = 2

Оценка модели 2.5777262449264526

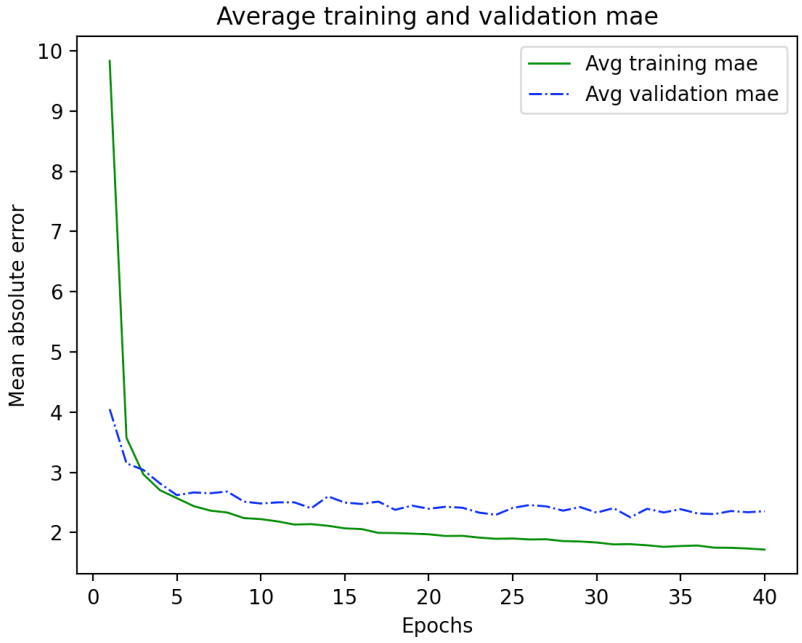

Рисунок 6 - Графики среднего значения mae при K = 6

Оценка модели 2.3549842039744058

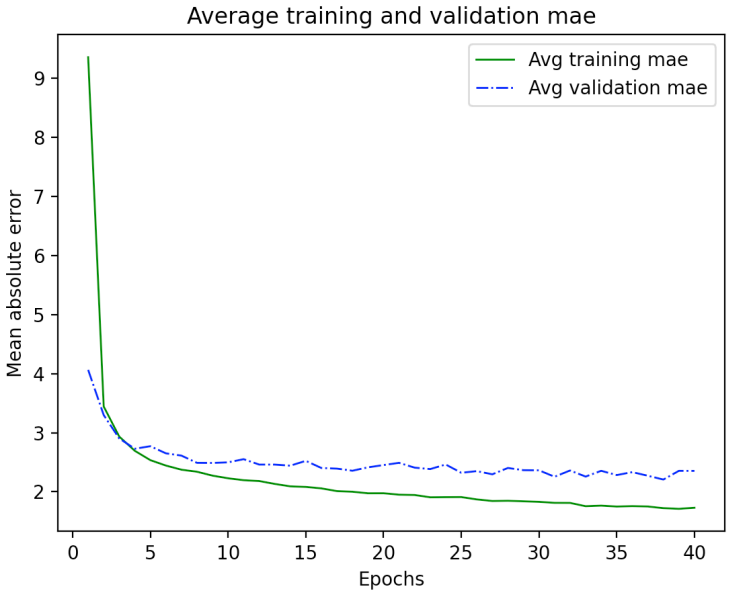

Рисунок 7 - Графики среднего значения mae при K = 8

Оценка модели 2.3535963594913483

При увеличении количества блоков с 6 до 8 значительных улучшений в работе модели не произошло. Наихудшее среднее значение mae показано при модели в 4 блока. Из этого можно сделать вывод, что оптимальным параметром для сети является K = 6.

Выводы.

Во время выполнения лабораторной работы была реализована модель, предсказывающая медиану цены дома в пригороде Бостона в середине 1970-х. Была изучена задача регрессии и выяснено ее отличие от задачи классификации, также изучена перекрестная проверка. Исследовано влияние количества эпох на работу модели и переобучение. Также исследовано влияние количества блоков в перекрестной проверке. Выяснено, что оптимальной конфигурацией модели - 40 эпох, 6 блоков для перекрестной модели.

ПРИЛОЖЕНИЕ А

```
import numpy as np
import matplotlib.pyplot as plt  #импорт модуля для графиков
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


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 plot_mae(history):
    color = "bgrmcyk"
    for i in range(len(history)):
        history_dict = history[i].history
        mae_values = history_dict['mae']
        val_mae_values = history_dict['val_mae']
        epochs = range(1, len(mae_values) + 1)
        tmpstr = 'Training mae '+str(i+1)
        tmpstr_2 = 'Validation mae '+str(i+1)
        plt.plot(epochs, mae_values, color=color[i], linestyle="-",
linewidth=1, label=tmpstr)
        plt.plot(epochs, val_mae_values, color=color[i],
linestyle="-.", linewidth=1, label=tmpstr_2)
    plt.title('Training and validation mae')
    plt.xlabel('Epochs')
    plt.ylabel('Mean absolute error')
    plt.legend()
    plt.show()


def plot_avg_mae(history, epochs):
    avg_mae = np.zeros(epochs)
    avg_val_mae = np.zeros(epochs)
    avg_mse = np.zeros(epochs)
    avg_val_mse = np.zeros(epochs)
    for i in range(len(history)):
        history_dict = history[i].history
```

```
        avg_mae += history_dict['mae']
        avg_mse += history_dict['loss']
        avg_val_mae += history_dict['val_mae']
        avg_val_mse += history_dict['val_loss']
    avg_mae /= len(history)
    avg_val_mae /= len(history)
    avg_mse /= len(history)
    avg_val_mse /= len(history)
    epochs_ = range(1, epochs + 1)
    #msa
    plt.plot(epochs_, avg_mae, color='g', linestyle="-",
linewidth=1, label='Avg training mae')
    plt.plot(epochs_, avg_val_mae, color='b', linestyle="-.",
linewidth=1, label='Avg validation mae')
    plt.title('Average training and validation mae')
    plt.xlabel('Epochs')
    plt.ylabel('Mean absolute error')
    plt.legend()
    plt.show()
    #mse
    plt.clf()
    plt.plot(epochs_, avg_mse, color='g', linestyle="-",
linewidth=1, label='Avg training mse')
    plt.plot(epochs_, avg_val_mse, color='b', linestyle="-.",
linewidth=1, label='Avg validation mse')
    plt.title('Average training and validation mse')
    plt.xlabel('Epochs')
    plt.ylabel('Mean squared error')
    plt.legend()
    plt.show()

(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

k = 4
num_val_samples = len(train_data) // k
print(num_val_samples)
num_epochs = 40
all_scores = []
```

```
all_history = []
for i in range(k):
    print('processing fold #', i)
    #проверочные данные
    val_data = train_data[i * num_val_samples: (i + 1) *
num_val_samples] #i часть данных
    val_targets = train_targets[i * num_val_samples: (i + 1) *
num_val_samples]
    #обучающие данные
    partial_train_data = np.concatenate([train_data[:i *
num_val_samples], train_data[(i + 1) * num_val_samples:]],
axis=0)#данные без i части
    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)
    all_history.append(history)

print(np.mean(all_scores))
# plot_mae(all_history)
plot_avg_mae(all_history, num_epochs)
```