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: «Регрессионная модель изменения цен на дома в Бостоне»

Студент гр. 7381 _____ Тарасенко Е.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.

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.

1. Объяснить различия задач классификации и регрессии;
2. Изучить влияние кол-ва эпох на результат обучения модели;
3. Выявить точку переобучения;
4. Применить перекрестную проверку по K блокам при различных K;
5. Построить графики ошибки и точности во время обучения для моделей, а также усредненные графики по всем моделям.

Ход работы.

Сначала необходимо исследовать результаты обучения модели на разном количестве эпох и в ходе исследования выявить точку переобучения модели. Изначально количество эпох было равно 100 (графики ошибки и оценки mae (mean absolute error) приведены на рис. 1). Сама же оценка в итоге равна примерно 2,5. Позже это количество было уменьшено вдвое, т. е. до 50-ти (результаты обучения приведены на рис. 2). Оценка – примерно 2,31. Далее снизим количество еще вдвое – до 25-ти (рис. 3). Оценка – 2,52.

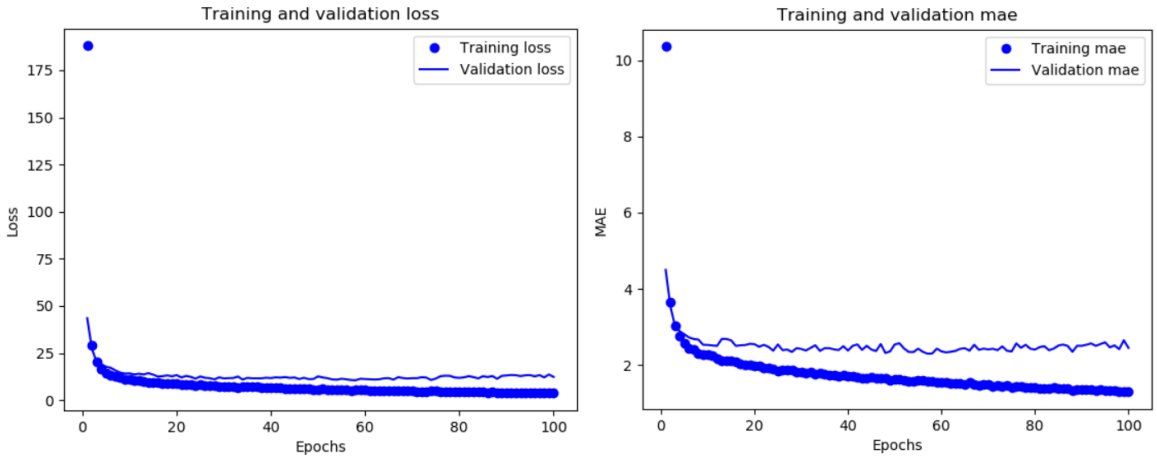

Рис. 1 – Графики ошибки и оценки mae модели, обучившейся на 100 эпохах

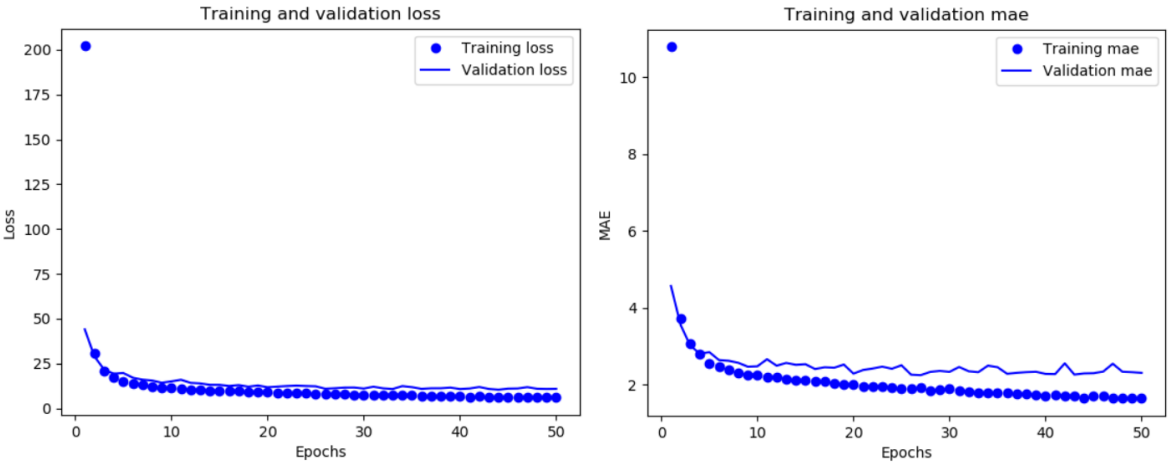

Рис. 2 – Графики ошибки и оценки mae модели, обучившейся на 50 эпохах

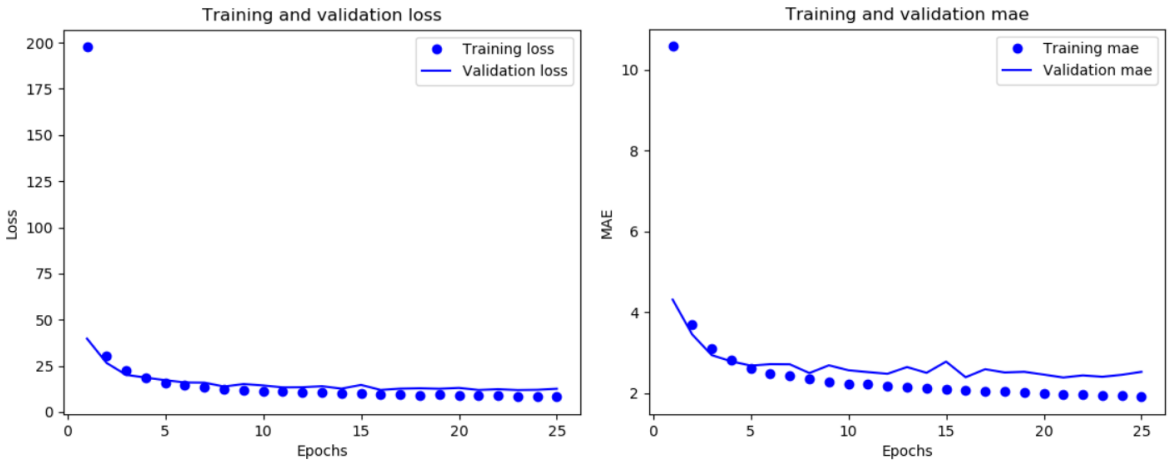

Рис. 3 – Графики ошибки и оценки mae модели, обучившейся на 25 эпохах

Т. о., самое оптимальное решение касательно количества эпох – это 50. После 50-ти эпох начинается переобучение модели, т. е. прекращение уменьшения потерь на тестовых данных при продолжении их уменьшения на

тренировочных. Модель «заучивает» ответы на тренировочные данные и плохо справляется с тестовыми.

Теперь выявим зависимость результатов обучения от количества блоков, на которые делится выборка с данными. Изначально количество блоков равно 4-м. Начнем исследование, пожалуй, с 2-х, постепенно увеличивая это число. Результаты обучения при разных значениях K (количества блоков данных) приведены на рис. 4 – 10.

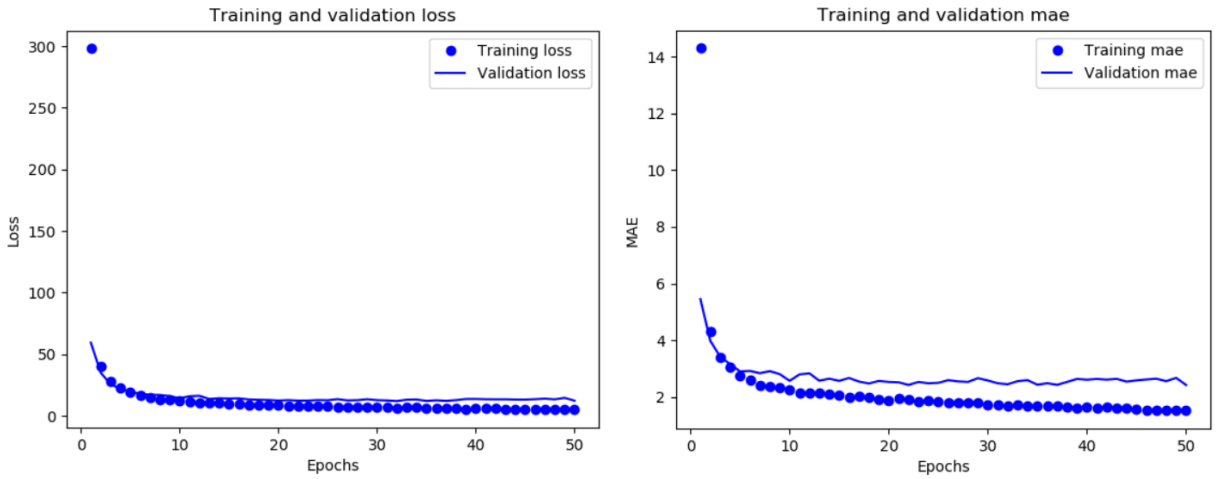

Рис. 4 – Графики ошибки и оценки mae модели при K = 2

(Оценка: 2.523992878287586)

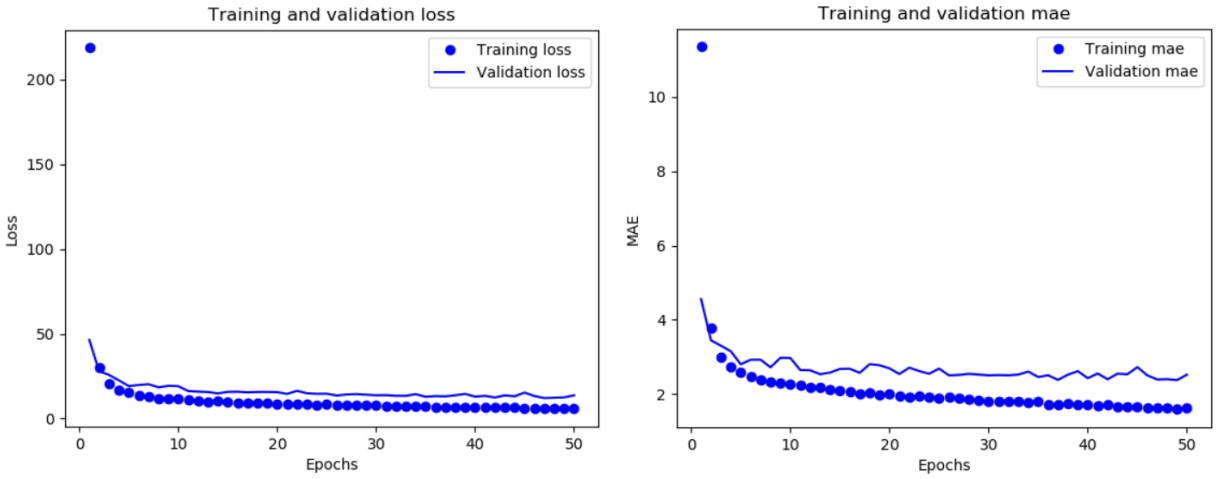

Рис. 5 – Графики ошибки и оценки mae модели при K = 3

(Оценка: 2.477401094271405)

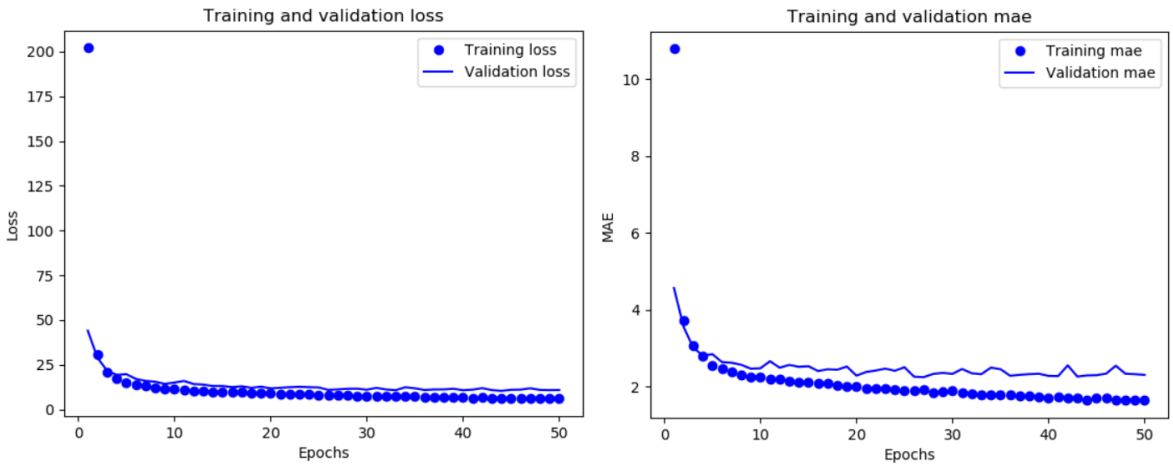

Рис. 6 – Графики ошибки и оценки mae модели при K = 4

(Оценка: 2.425012168317738)

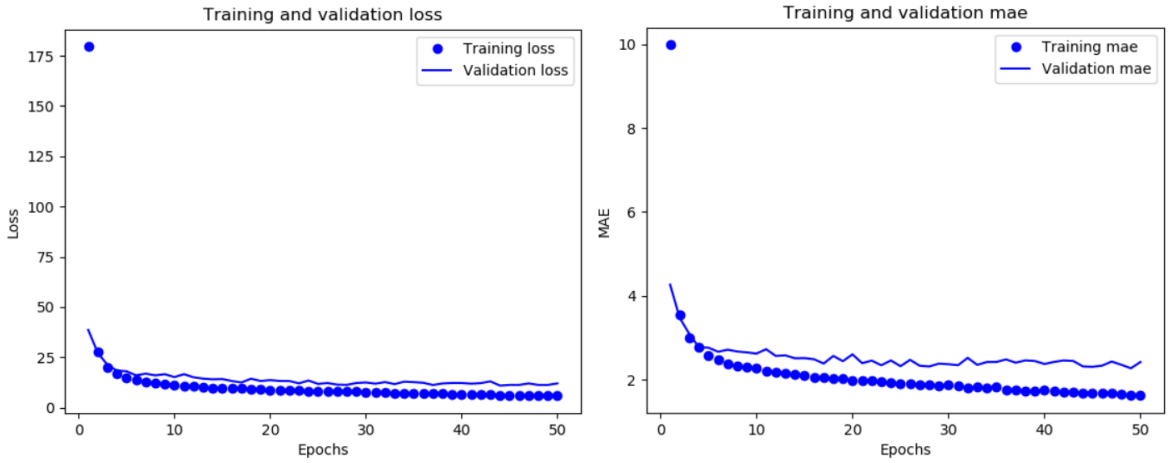

Рис. 7 – Графики ошибки и оценки mae модели при K = 5

(Оценка: 2.418513875007629)

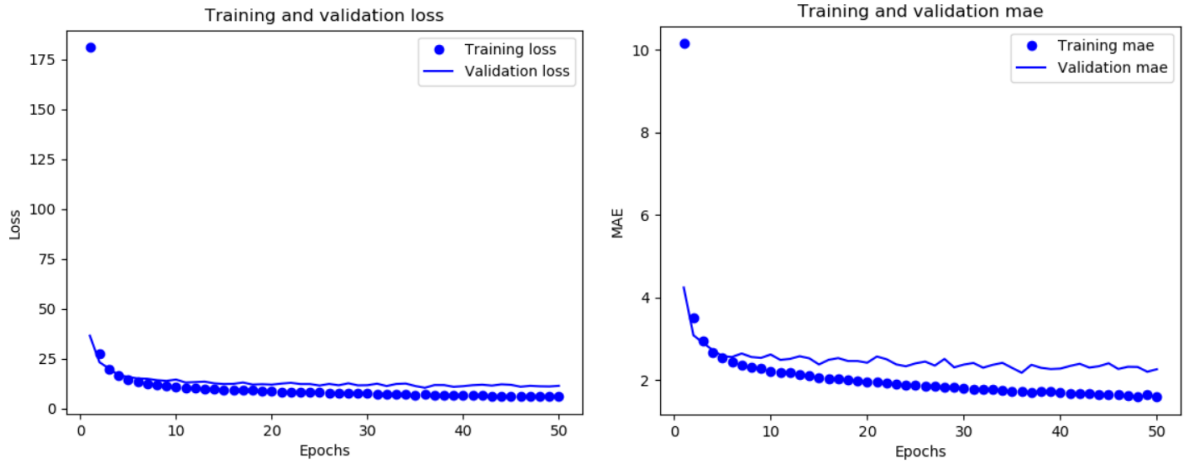

Рис. 8 – Графики ошибки и оценки mae модели при K = 6

(Оценка: 2.261990709165435)

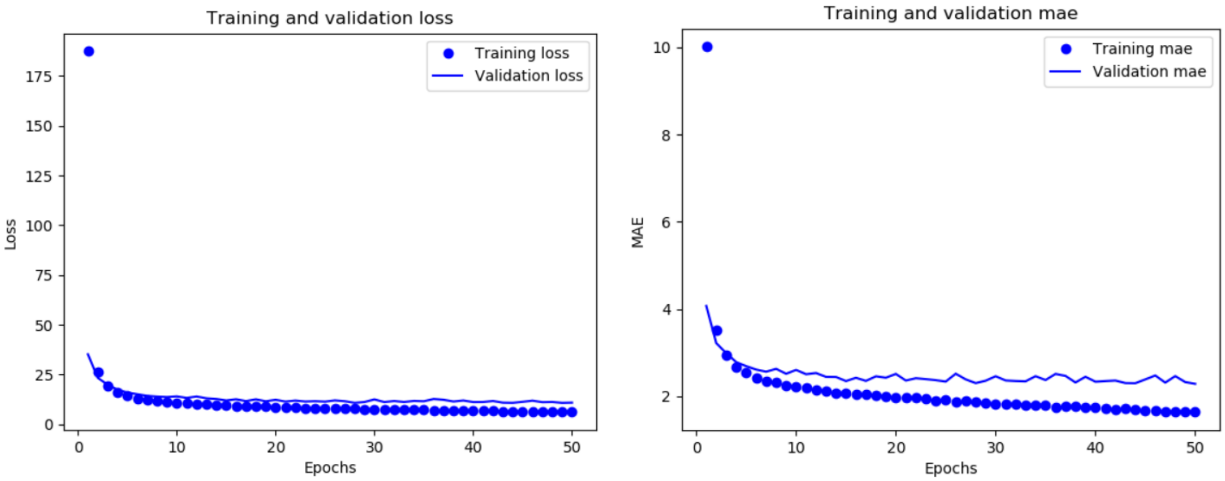

Рис. 9 – Графики ошибки и оценки mae модели при K = 7

(Оценка: 2.2843218826709832)

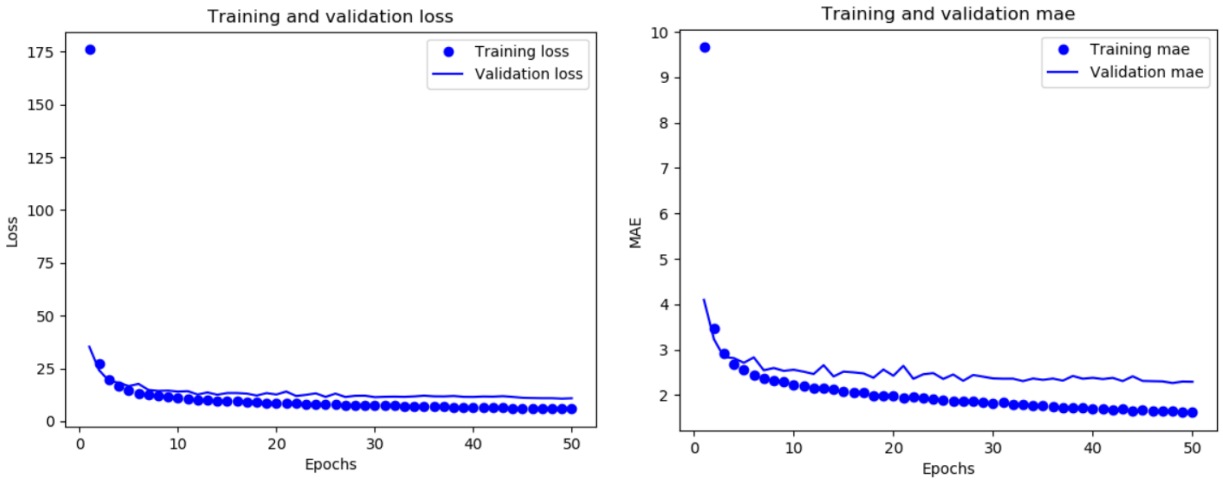

Рис. 10 – Графики ошибки и оценки mae модели при K = 8

(Оценка: 2.294295192956924)

В ходе наблюдений выявлено, что при 6-ти блоках модель дает наиболее низкую среднюю оценку mae (mean absolute error), что соответствует наиболее низкому отклонению прогнозов стоимости домов в Бостоне.

Выводы.

В ходе выполнения данной лабораторной работы был изучен принцип перекрестной проверки при создании и обучении нейронных сетей. В рамках выполнения задания была создана и обучена модель, предсказывающая медианную цену на дома в пригороде Бостона в середине 1970-х по таким данным, как уровень преступности, ставка местного имущественного налога и т. д. Также в ходе выполнения исследования были подобраны наилучшие параметры для обучения данной модели.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow


def build_model():
    model = tensorflow.keras.Sequential()
    model.add(tensorflow.keras.layers.Dense(64, activation='relu',
input_shape=(train_data.shape[1],)))
    model.add(tensorflow.keras.layers.Dense(64, activation='relu'))
    model.add(tensorflow.keras.layers.Dense(1))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model


def plots():
    epochs = range(1, len(loss[len(loss) - 1]) + 1)

    # построение графика ошибки
    plt.plot(epochs, loss[len(loss) - 1], 'bo', label='Training loss')
    plt.plot(epochs, val_loss[len(loss) - 1], 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    # построение графика точности
    plt.clf()
    plt.plot(epochs, mae[len(loss) - 1], 'bo', label='Training mae')
    plt.plot(epochs, val_mae[len(loss) - 1], 'b', label='Validation mae')
    plt.title('Training and validation mae')
    plt.xlabel('Epochs')
    plt.ylabel('MAE')
    plt.legend()
    plt.show()


def add(x, y):
    for i in range(len(x)):
        x[i] += y[i]


# подготовка рабочих данных
(train_data, train_targets), (test_data, test_targets) =
tensorflow.keras.datasets.boston_housing.load_data()

print(train_data.shape)
print(test_data.shape)

print(test_targets)

# нормализация данных
mean = train_data.mean(axis=0)
train_data -= mean
```

```
std = train_data.std(axis=0)
train_data /= std

test_data -= mean
test_data /= std

# статистические данные для построения графиков
loss = []
val_loss = []
mae = []
val_mae = []
avg_loss = []
avg_val_loss = []
avg_mae = []
avg_val_mae = []

# перекрестная проверка по K блокам
k = 4
num_val_samples = len(train_data) // k
num_epochs = 100
all_scores = []

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()
    H = model.fit(partial_train_data, partial_train_targets, epochs=num_epochs,
batch_size=1, verbose=0,
                  validation_data=(val_data, val_targets))

    # получение ошибки и точности в процессе обучения, построение графиков
    loss.append(H.history['loss'])
    val_loss.append(H.history['val_loss'])
    mae.append(H.history['mean_absolute_error'])
    val_mae.append(H.history['val_mean_absolute_error'])
    if len(avg_loss) == 0:
        avg_loss += H.history['loss']
        avg_val_loss += H.history['val_loss']
        avg_mae += H.history['mean_absolute_error']
        avg_val_mae += H.history['val_mean_absolute_error']
    else:
        add(avg_loss, H.history['loss'])
        add(avg_val_loss, H.history['val_loss'])
        add(avg_mae, H.history['mean_absolute_error'])
        add(avg_val_mae, H.history['val_mean_absolute_error'])
    plots()

    # результат работы текущей модели
    val_mse, val_MAE = model.evaluate(val_data, val_targets, verbose=0)
    all_scores.append(val_MAE)

# вывод усредненных графиков и ответа
for i in range(len(avg_loss)):
    avg_loss[i] /= k
```

```
    avg_val_loss[i] /= k
    avg_mae[i] /= k
    avg_val_mae[i] /= k
loss.append(avg_loss)
val_loss.append(avg_val_loss)
mae.append(avg_mae)
val_mae.append(avg_val_mae)
plots()
print(np.mean(all_scores))
```