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 модеь изменения цен на дома в Бостоне

Студент гр. 8383 _____ Федоров И.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Реализовать предсказание медианной цена на дома в пригороде Бостона в середине 1970-х по таким данным, как уровень преступности, ставка местного имущественного налога и т.д. (всего 13 параметров).

Задачи

- Ознакомиться с задачей регрессии
- Изучить отличие задачи регрессии от задачи классификации
- Создать модель
- Настроить параметры обучения
- Обучить и оценить модель
- Ознакомиться с перекрестной проверкой

Выполнение работы.

Цель задач классификации состоит в предсказании одной дискретной метки для образца входных данных. В отличие от них, цель регресии заключается в предсказании не дискретной метки, а значения на непрерывной числовой прямой. В данной работе значениями являются цены на дома (другим примером могут служить предсказания температуры, времени и т.д.).

Были импортированы необходимые для работы класы, модули, функции, а также данные, которые являются частью библиотеки *keras*.

```
import pandas
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import plot_model
from tensorflow.keras.datasets import boston_housing
```

Исходный 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. Т.к. нежелательно передавать в есть значения, которые имеют разные диапазоны, то к ним была применена нормализация: для каждого признака из каждого значения вычитается среднее по этому признаку, и разность делится на стандартное отклонение:

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

Сама сеть заканчивается одномерным слоев, не имеющим функции активации, т.к. она могла бы ограничить диапазон выходных значений. В данном случае, с линейным последним слоем, сеть может предсказывать значения из любого диапазона.

Т.к. объем данных достаточно небольшой, то для более качественного оценивания модели был применен ***метод перекрестной проверки по k блокам***. Он заключается в разбиении данных на *k* блоков равного размера. Для каждого блока выполняется обучение исследуемой модели на остальных *k-1* блоках, а оценка на текущем блоке. Конечная оценка рассчитывается как среднее всех промежуточных оценок. Схематично метод выглядит как на рис. 1. Полный текст программы показан в приложении.

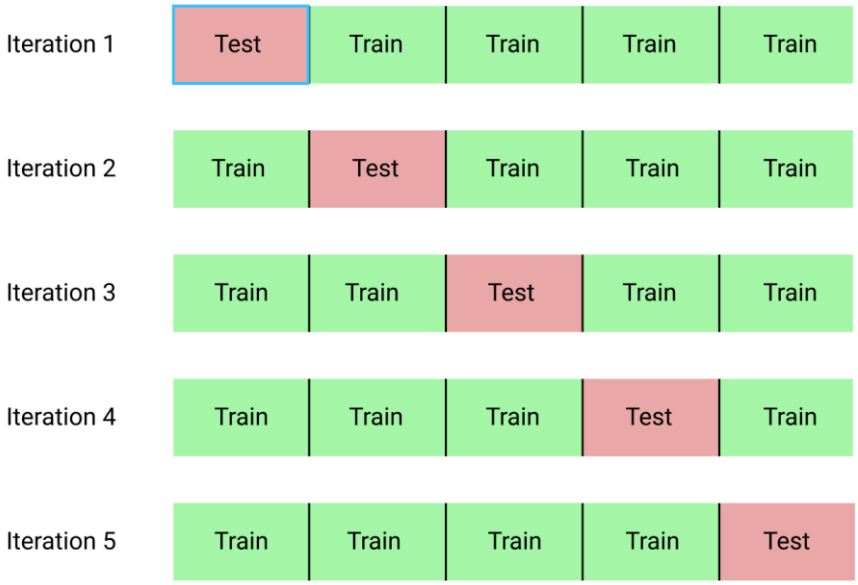

Рисунок 1 - Перекрестная проверка

Схема модели показана на рис. 2.

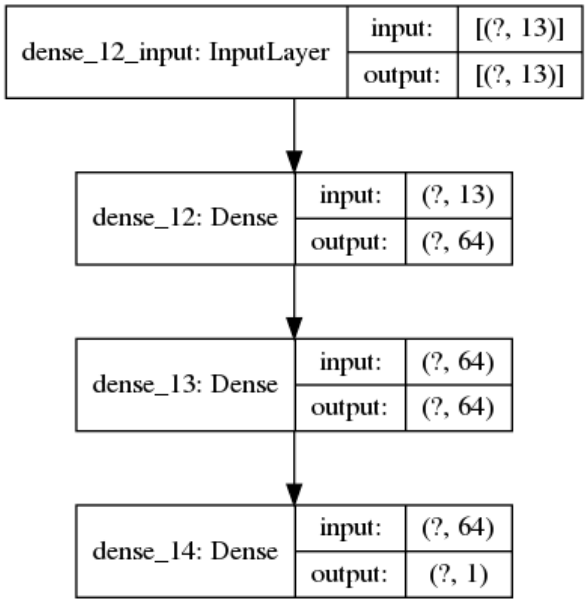

Рисунок 2 - Схема модели

Была выполнена прогонка модели при 500 эпохах при значении числа блоков k=4. Графики каждой модели показаны на рис. 3.1-3.4. График среднего значения метрики `mae` для всех прогонов показан на рис. 3.5.

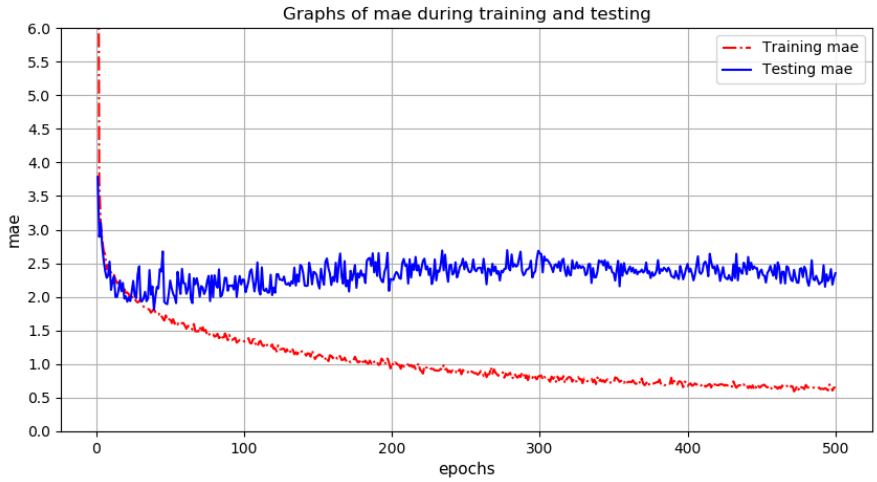

Рисунок 3.1 - Первая модель

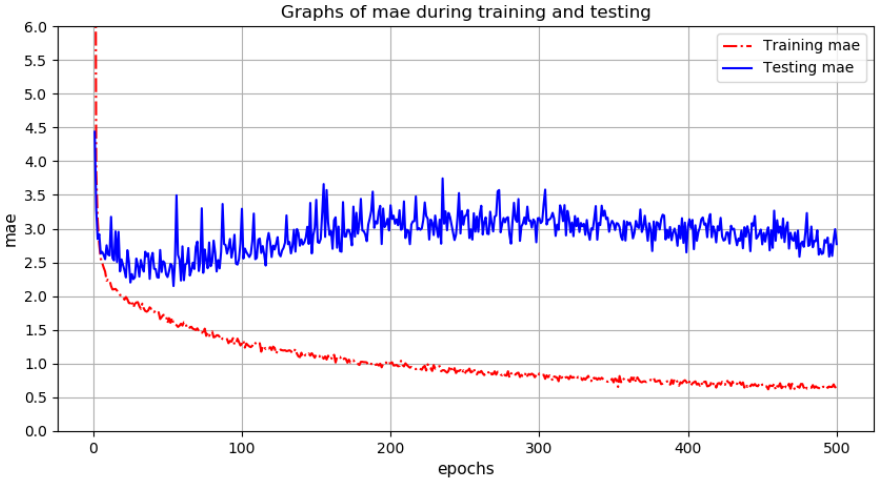

Рисунок 3.2 - Вторая модель

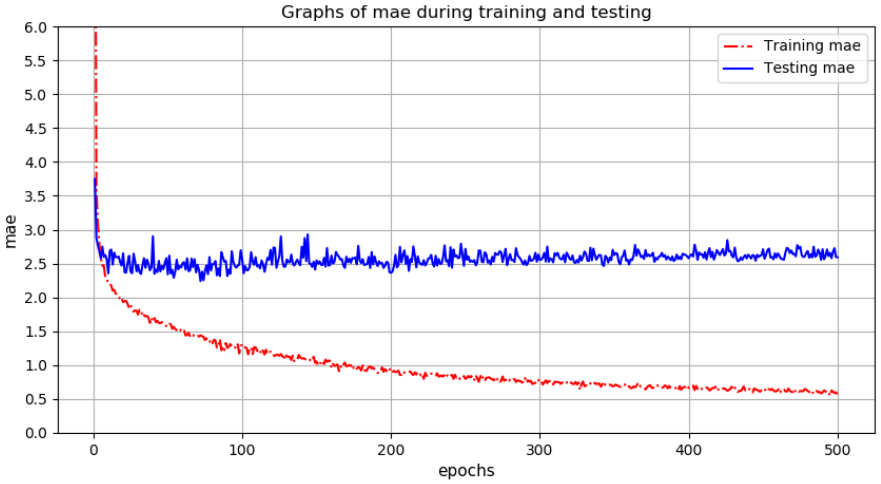

Рисунок 3.3 - Третья модель

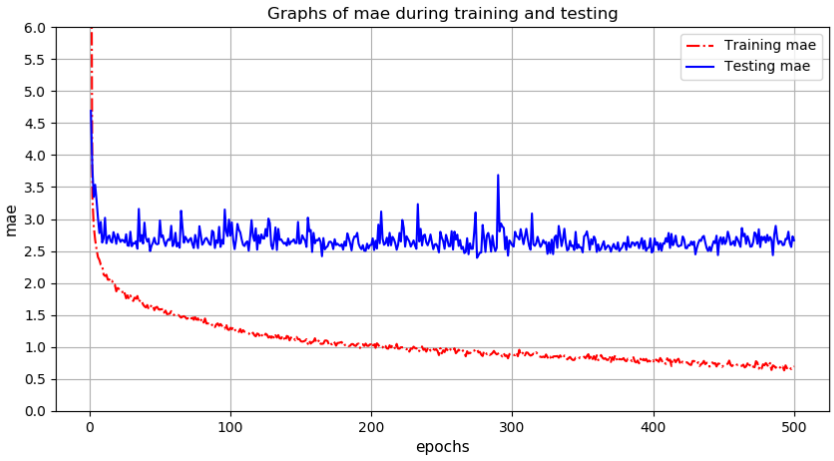

Рисунок 3.4 - Четвертая модель

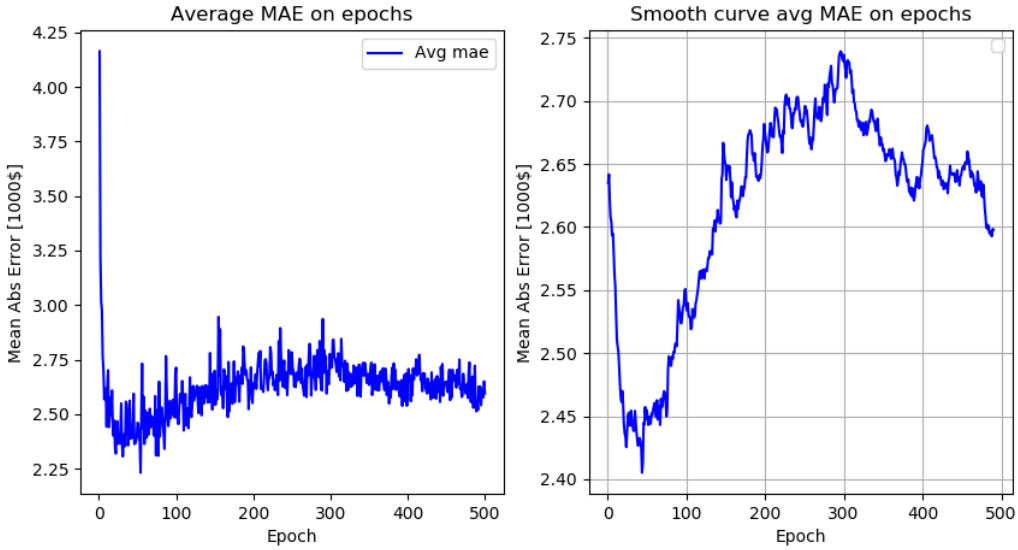

Рисунок 3.5 - Графики среднего значения mae для всех прогонов

Правый график для большей наглядности общей тенденции был получен с помощью функции `smooth_curve_mae()` (также представлена в приложении), которая заменяет каждую оценку экпоненциальным скользящим средним по предыдущим оценкам, чтобы получить более гладкую кривую (сглаживание краткосрочных колебаний). Вычисляется по формуле:

$$EMA_t = \alpha \times P_t + (1 - \alpha) \times EMA_{t-1}$$

где α - коэффициент в интервале от 0 до 1.

Прогонка дала следующие оценки и результаты на контрольных данных:

Mean MAE	2.7612715
Test MSE	18.1549365473
Test MAE	2.7583997

По рисунку 3.5. видно, что переобучение наступает примерно в области 50-60 эпох.

Была выполнена прогонка модели при 100 эпохах при значении числа блоков k=4. Графики каждой модели показаны на рис. 4.1-4.4. График среднего значения метрики `mae` для всех прогонов показан на рис. 4.5.

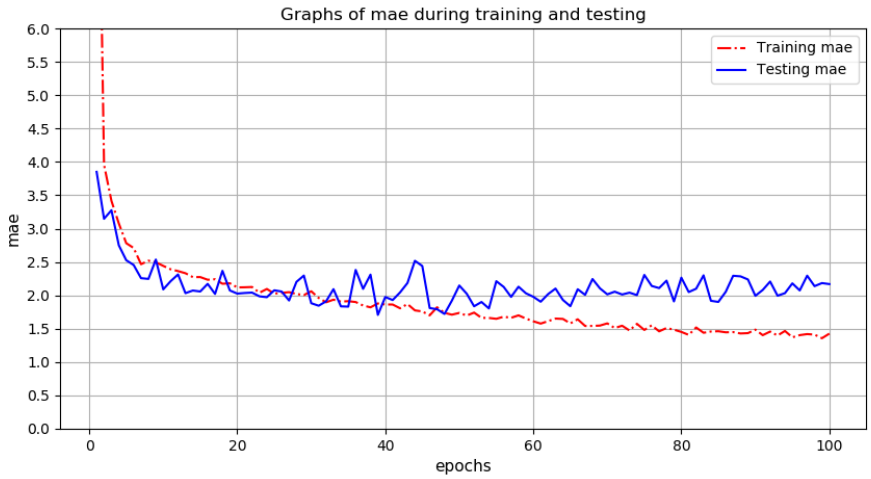

Рисунок 4.1 - Первая модель

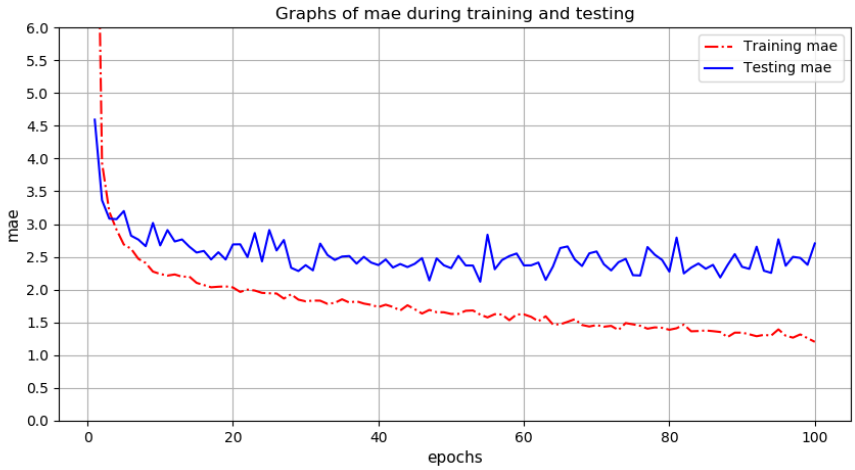

Рисунок 4.2 - Вторая модель

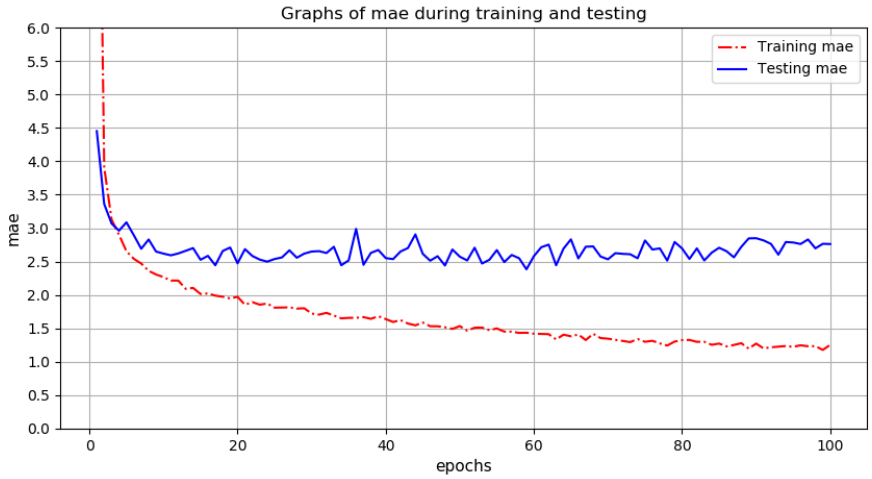

Рисунок 4.3 - Третья модель

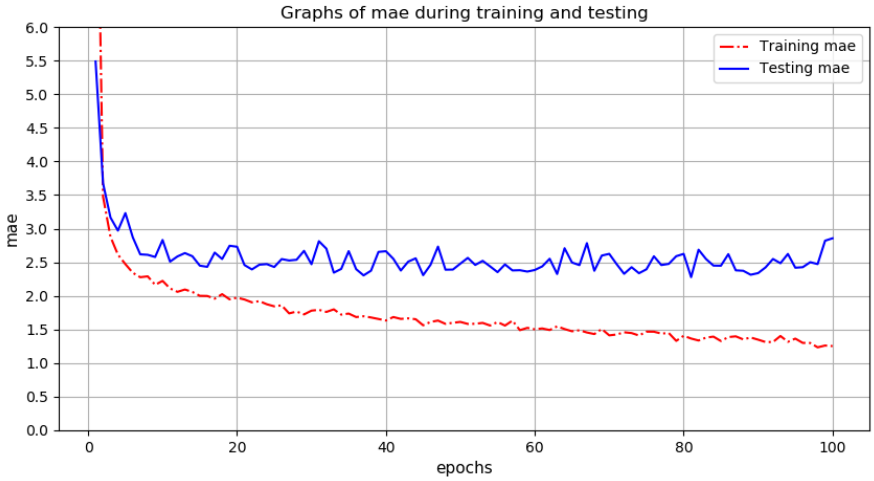

Рисунок 4.4 - Четвертая модель

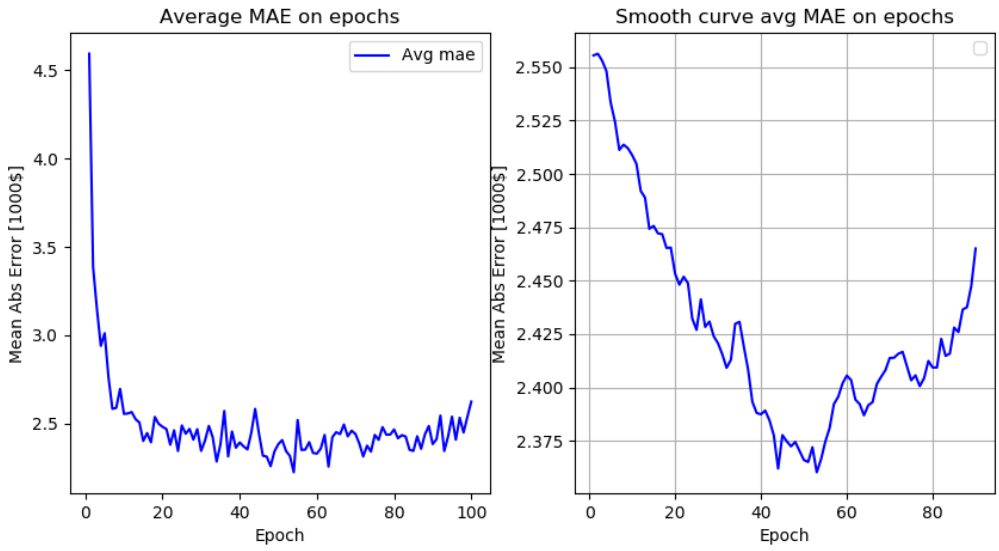

Рисунок 4.5 - Графики среднего значения mae для всех прогонов

Прогонка дала следующие оценки и результаты на контрольных данных:

Mean MAE	2.62268213
Test MSE	17.67669617
Test MAE	2.55164511

Видно, что значение ошибок уменьшилось, также подтвердилось предыдущее наблюдение о переобучении модели на 50-60 эпохах.

Была выполнена прогонка модели при 60 эпохах при значении числа блоков k=4. Графики каждой модели показаны на рис. 5.1-5.4. График среднего значения метрики `mae` для всех прогонов показан на рис. 5.5.

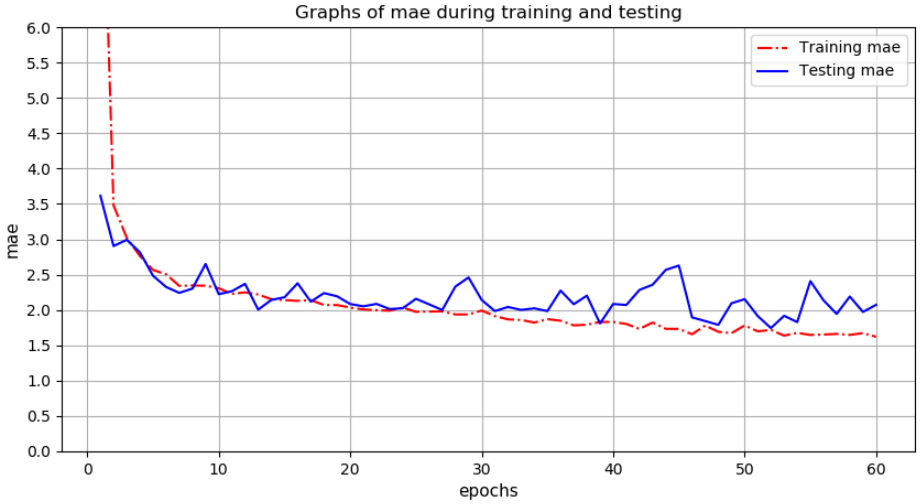

Рисунок 5.1. - Первая модель

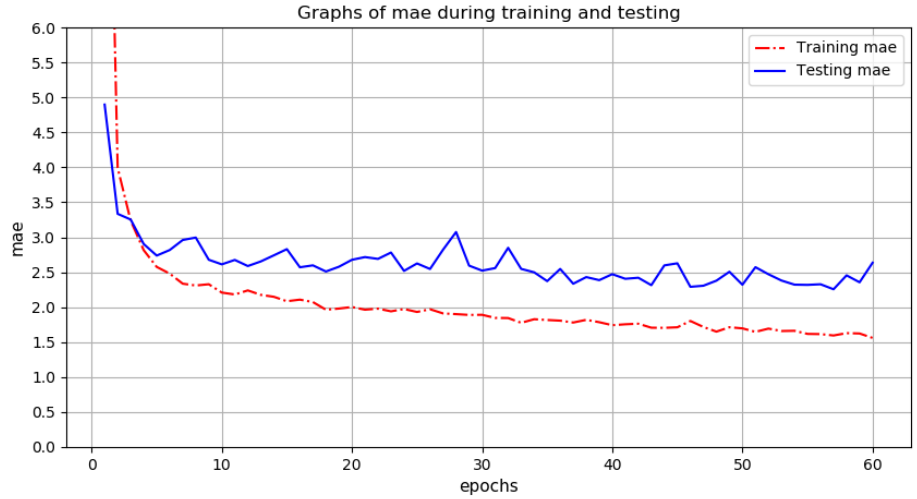

Рисунок 5.2. - Вторая модель

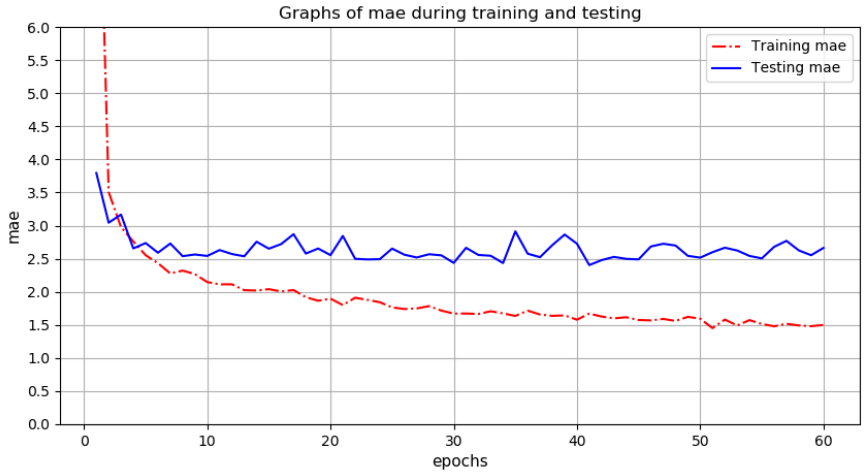

Рисунок 5.3 - Третья модель

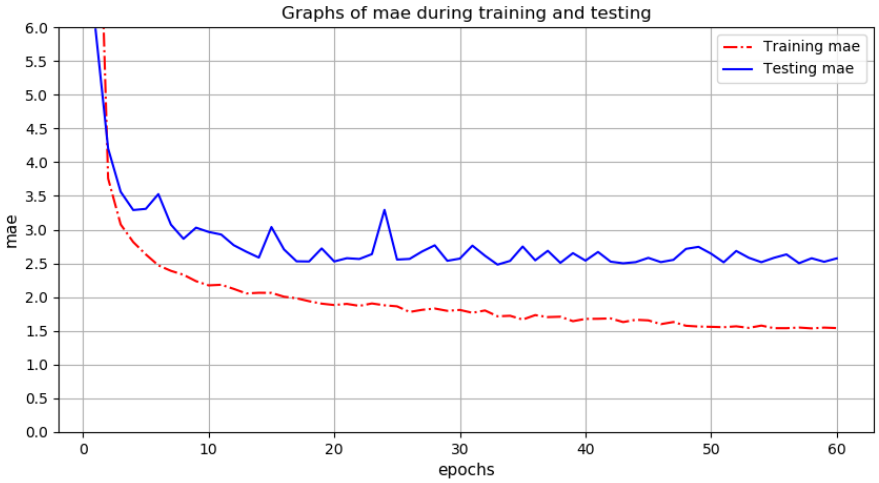

Рисунок 5.4 - Четвертая модель

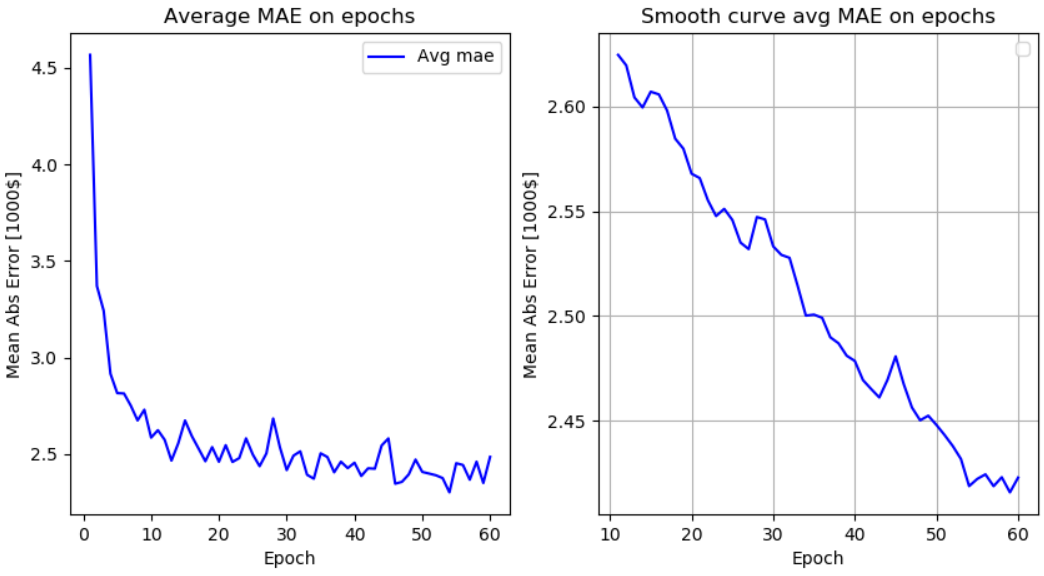

Рисунок 5.5 - Графики среднего значения mae для всех прогонов

Прогонка дала следующие оценки и рез-ты на контрольных данных:

Mean MAE	2.38068824
Test MSE	13.119851902
Test MAE	2.31521645

Видно, что значение ошибок еще раз уменьшилось, также можно заметить по графику, что выбор примерно 60 эпох является оптимальным.

Были проведены эксперименты по использованию **метода перекрестной проверки по *k* блокам** при разных значениях ***k***, число эпох равно 100.

Было установлено значения k=15. Результаты:

Mean MAE	2.266119738
Test MSE	13.588868207
Test MAE	2.512855215

В данном случае число mae колебалось от 1.5794984 до 3.188624, что является достаточно большим разбросом. Также оценка по сравнению с k=4 и с результатом полученном на контрольном наборе является чрезмерно оптимистичной.

Было установлено значение k=2. Результаты:

Mean MAE (1-й запуск)	2.4591484
Mean MAE (2-й запуск)	2.7134762
Test MAE	2.5430353

Видно, что значения среднего сильно различаются в двух экспериментах.

Можно сделать вывод, что слишком маленькое число разбиений делает применение метода бесполезным, т.к. в ходе работы будет получено слишком мало значения для вычисления среднего, которые могут значительно отличаться. Чрезмерное увеличение может привести к тому, что будет получена слишком оптимистичная оценка, т.к. уменьшается размер проверочного набора на каждом прогоне.

Выводы.

В ходе выполнения данной работы была реализована ИНС для предсказания медианной цены на дома в пригороде Бостона. Было изучен метод перекрестной проверки по К блокам, проведены эксперементы с определением эпохи наступления переобучения, влияния количества блоков. Были построены необходимые графики и приведены необходимые результаты.

Приложение

```
import pandas
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import plot_model

# X - входные данные, У-выходные, данные перемешиваются
dataframe = pandas.read_csv("sonar.csv", header=None)
dataset = dataframe.values
np.random.shuffle(dataset)

X = dataset[:,0:60].astype(float)
Y = dataset[:,60]


#Переход от R,M меток к категориальному вектору
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)   #[1, 1, 1, 1, ... , 0 , 0, 0]

###### отбор тестовых данных
test_index = int(len(X)-len(X)*0.2)
test_data_x = X[test_index:]

X = X[:test_index]
test_index = int(len(encoded_Y)-len(encoded_Y)*0.2)
test_data_y = encoded_Y[test_index:]
encoded_Y = encoded_Y[:test_index]


# графики потерь и точности при обучении и тестирования
def plot_model_loss_and_accuracy(history, figsize_=(10,5)):
    plt.figure(figsize=figsize_)
    train_loss = history.history['loss']
    test_loss = history.history['val_loss']
    train_acc = history.history['acc']
    test_acc = history.history['val_acc']
    epochs = range(1, len(train_loss) + 1)

    plt.subplot(121)
    plt.plot(epochs, train_loss, 'r--', label='Training loss')
    plt.plot(epochs, test_loss, 'b-', label='Testing loss')
    plt.title('Graphs of losses during training and testing')
    plt.xlabel('epochs')
    plt.ylabel('loss')
    plt.legend()
    plt.subplot(122)
    plt.plot(epochs, train_acc, 'r-.', label='Training acc')
    plt.plot(epochs, test_acc, 'b-', label='Testing acc')
    plt.title('Graphs of accuracy during training and testing')
    plt.xlabel('epochs', fontsize=11, color='black')
    plt.ylabel('accuracy', fontsize=11, color='black')
    plt.legend()
    plt.grid(True)

    plt.show()
```

```
def get_model():
    model = Sequential()
    model.add(Dense(60, input_dim=60, activation='relu'))
    model.add(Dense(15, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam',loss='binary_crossentropy',
metrics=['accuracy'])
    return model

k = 4
num_valid_data = len(X) // k
valid_scores_loss = []
valid_scores_acc = []

#X = X[:, 0:50]
#test_data_x = test_data_x[:, 0:50]
#перекрестная проверка по К блокам
for fold in range(k):
    validation_data_x = X[num_valid_data * fold: num_valid_data * (fold + 1)]
    validation_data_y = encoded_Y[num_valid_data * fold: num_valid_data *
(fold + 1)]

    train_data_x = np.concatenate((X[:num_valid_data * fold] ,
X[num_valid_data * (fold + 1):]))
    train_data_y = np.concatenate((encoded_Y[:num_valid_data * fold] ,
encoded_Y[num_valid_data * (fold + 1):]))

    model = get_model()
    history = model.fit(train_data_x, train_data_y, epochs=100,
batch_size=10, verbose=0)
    results = model.evaluate(validation_data_x, validation_data_y,
batch_size=10,verbose=0)
    valid_scores_loss.append(results[0])
    valid_scores_acc.append(results[1])

valid_score_acc = np.average(valid_scores_acc)
valid_score_loss = np.average(valid_scores_loss)
print("Loss: ", valid_score_loss)
print("Accuracy: ", valid_score_acc)

#обучение на всех данных, кроме контрольного набора
model = get_model()
history = model.fit(X, encoded_Y, epochs=100, batch_size=10,
validation_data=(test_data_x, test_data_y),verbose=2)
test_score = model.evaluate(test_data_x,test_data_y, batch_size=10,verbose=0)
print(test_score)
plot_model_loss_and_accuracy(history)
plot_model(model, to_file='model.png', show_shapes=True)
```