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. 7382 _____ Бахеров Д.В.

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.

Порядок выполнения работы.

1. Ознакомиться с задачей регрессии
2. Изучить отличие задачи регрессии от задачи классификации
3. Создать модель
4. Настроить параметры обучения
5. Обучить и оценить модели
6. Ознакомиться с перекрестной проверкой

Требования к выполнению задания.

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

Основные теоретические положения.

1. *Классификационное моделирование* - это задача приближения функции отображения f от входных переменных (X)) к дискретным выходным переменным (Y).

Задача классификации требует, разделения объектов в один или два класса.

Классификация может иметь действительные или дискретные входные переменные.

Проблема с двумя классами часто называется проблемой двухклассной или двоичной классификации.

Проблема с более чем двумя классами часто называется проблемой классификации нескольких классов.

Проблема, когда для примера назначается несколько классов, называется проблемой классификации по нескольким меткам.

2. *Регрессионное моделирование* - это задача приближения функции отображения *f* от входных переменных (*X)*) к непрерывной выходной переменной (*Y*).

Задача регрессии требует предсказания количества.

Регрессия может иметь действительные или дискретные входные переменные.

Проблема с несколькими входными переменными часто называется проблемой многомерной регрессии.

Проблема регрессии, когда входные переменные упорядочены по времени, называется проблемой прогнозирования временных рядов.

Ход работы.

Была создана и обучена модель искусственной нейронной сети. Код предоставлен в приложении А.

Для выполнения поставленной задачи были опробованы разнообразные архитектуры сети, обучение проводилось при различных параметрах, было изменено количество блоков 4,5 и количество эпох.

Рассмотрим модель с 5-ю блоками. Точность будем оценивать с помощью средней абсолютной ошибки. Графики ошибок и точности предоставлены на рис. 1-12.

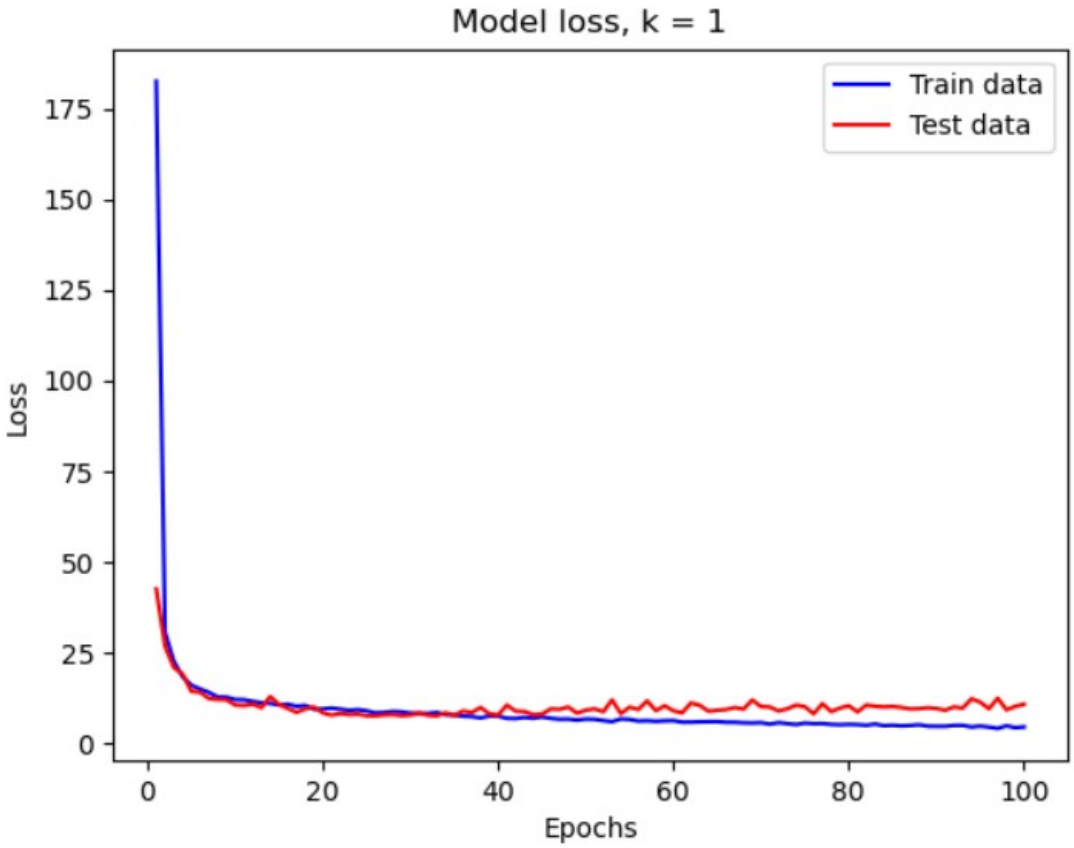

Рисунок 1 – График ошибки k=1

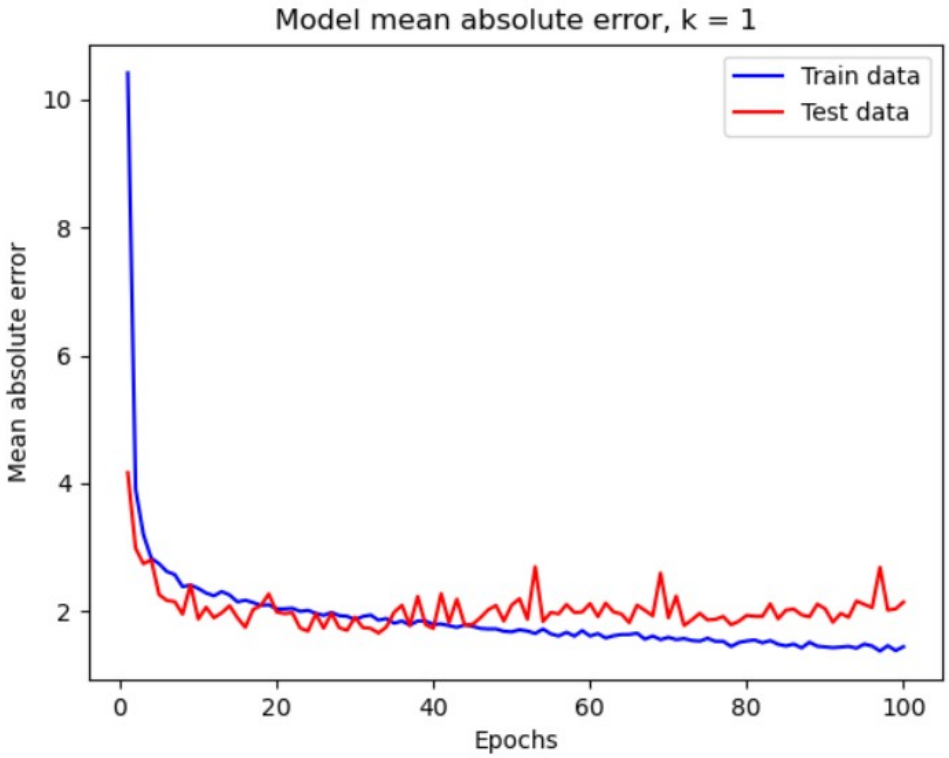

Рисунок 2 – График оценки mae k=1

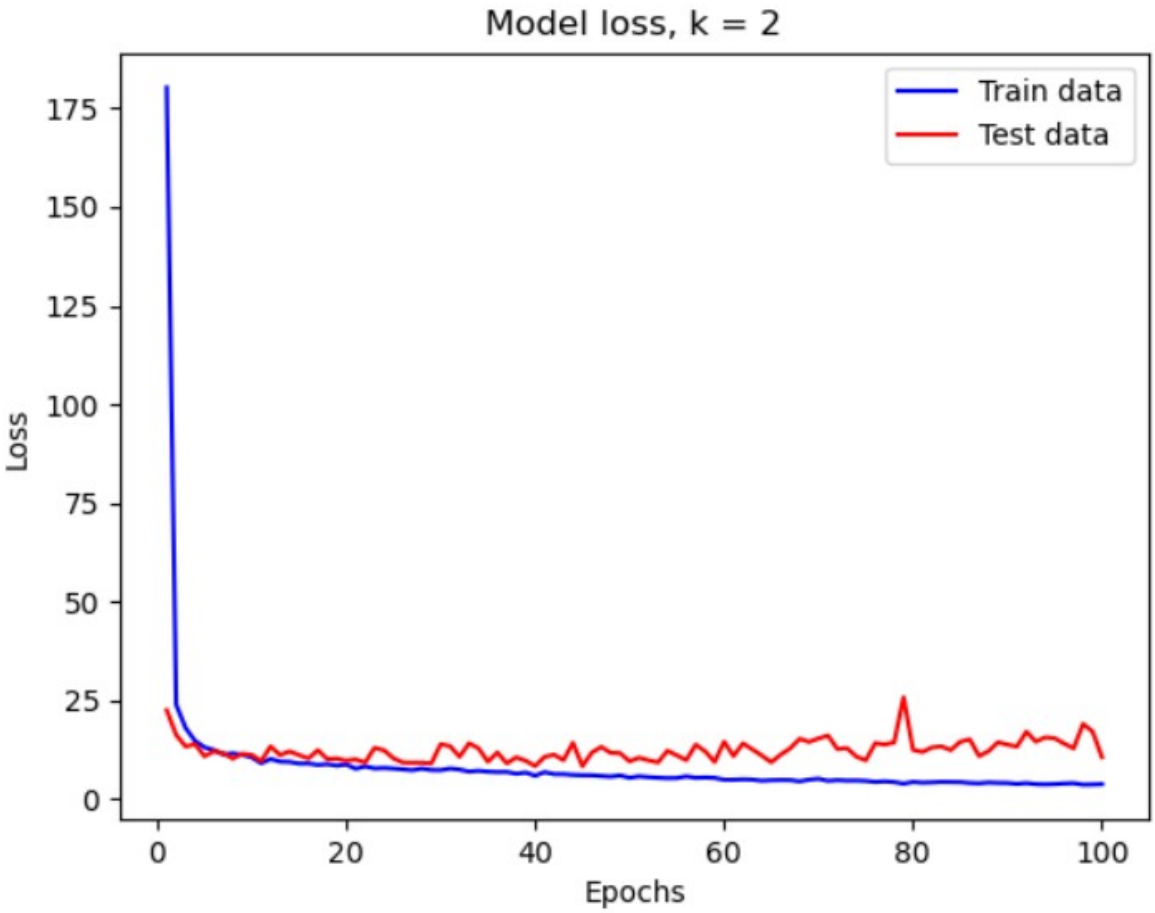

Рисунок 3 – График ошибки k=2

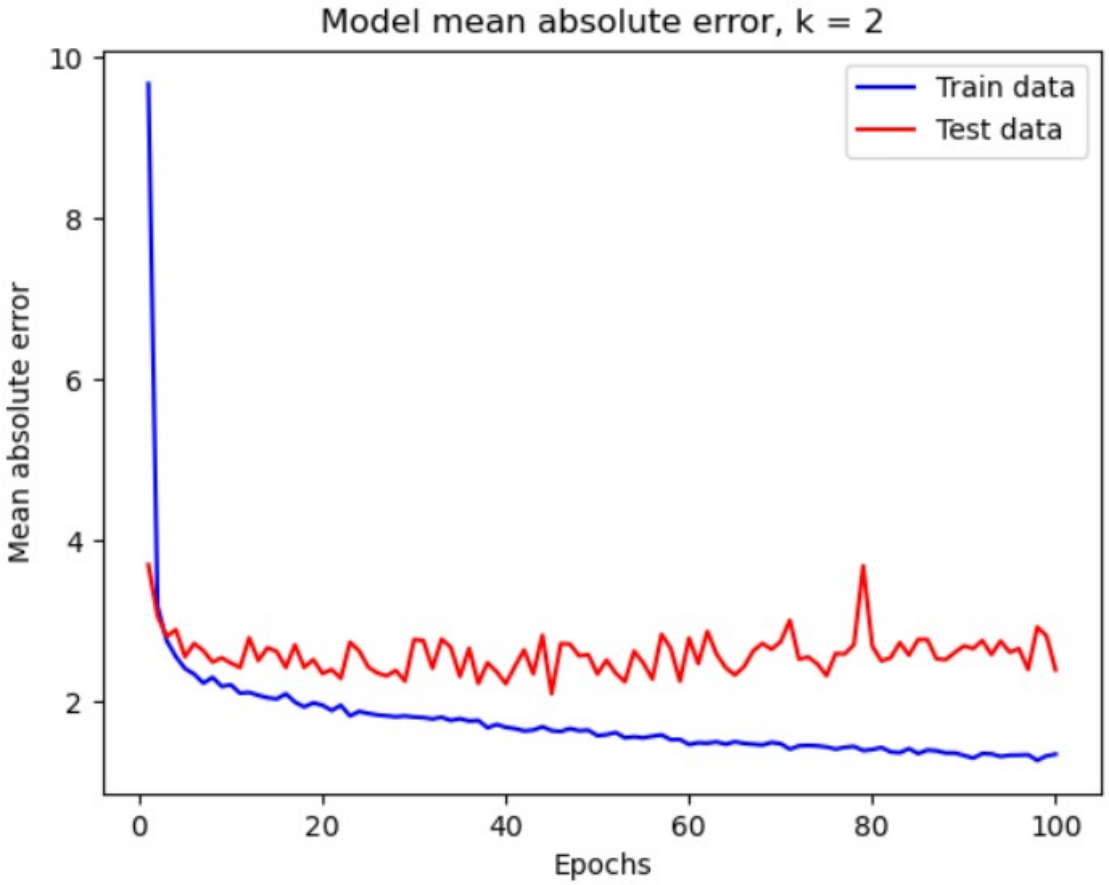

Рисунок 4 – График оценки mae k=2

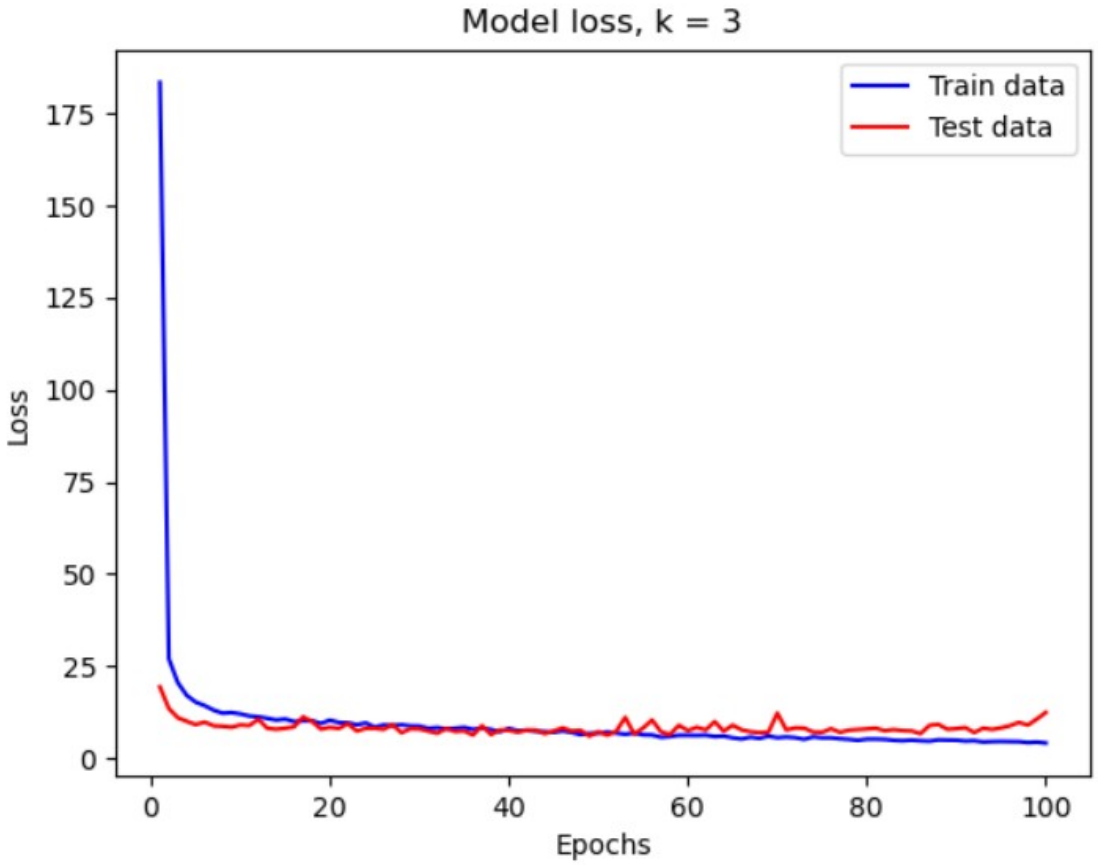

Рисунок 5 – График ошибки k=3

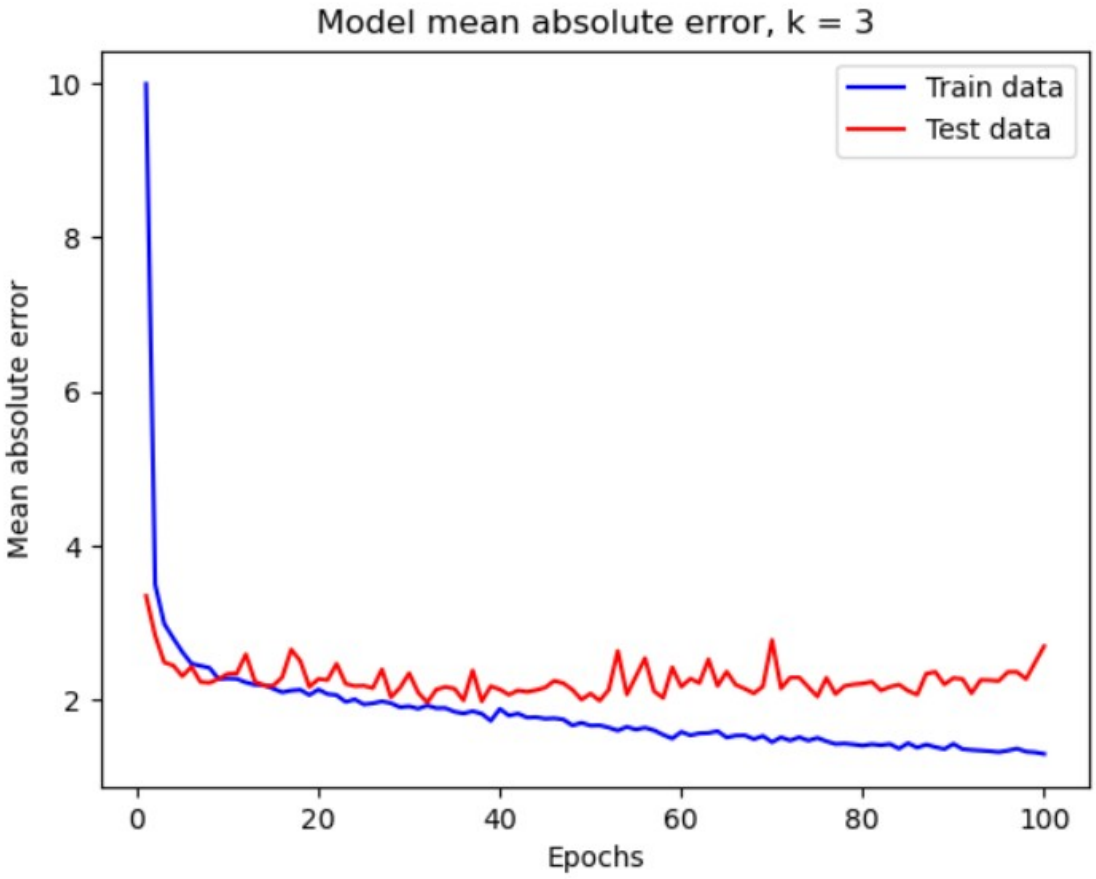

Рисунок 6 – График оценки mae k=3

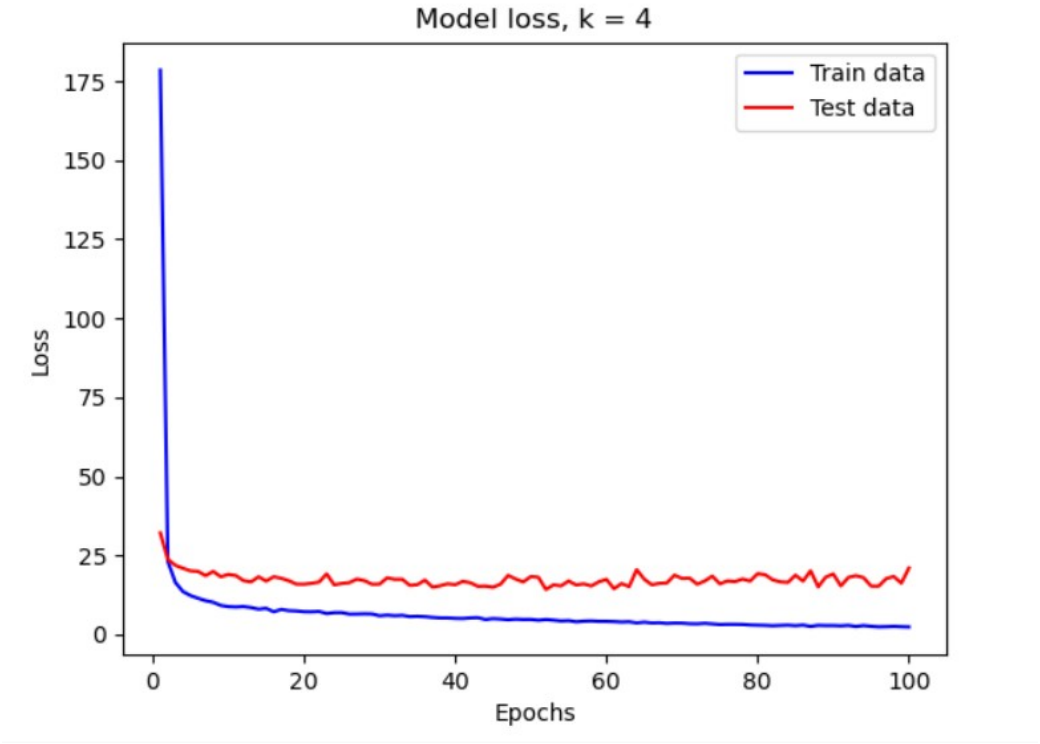

Рисунок 7 – График ошибки k=4

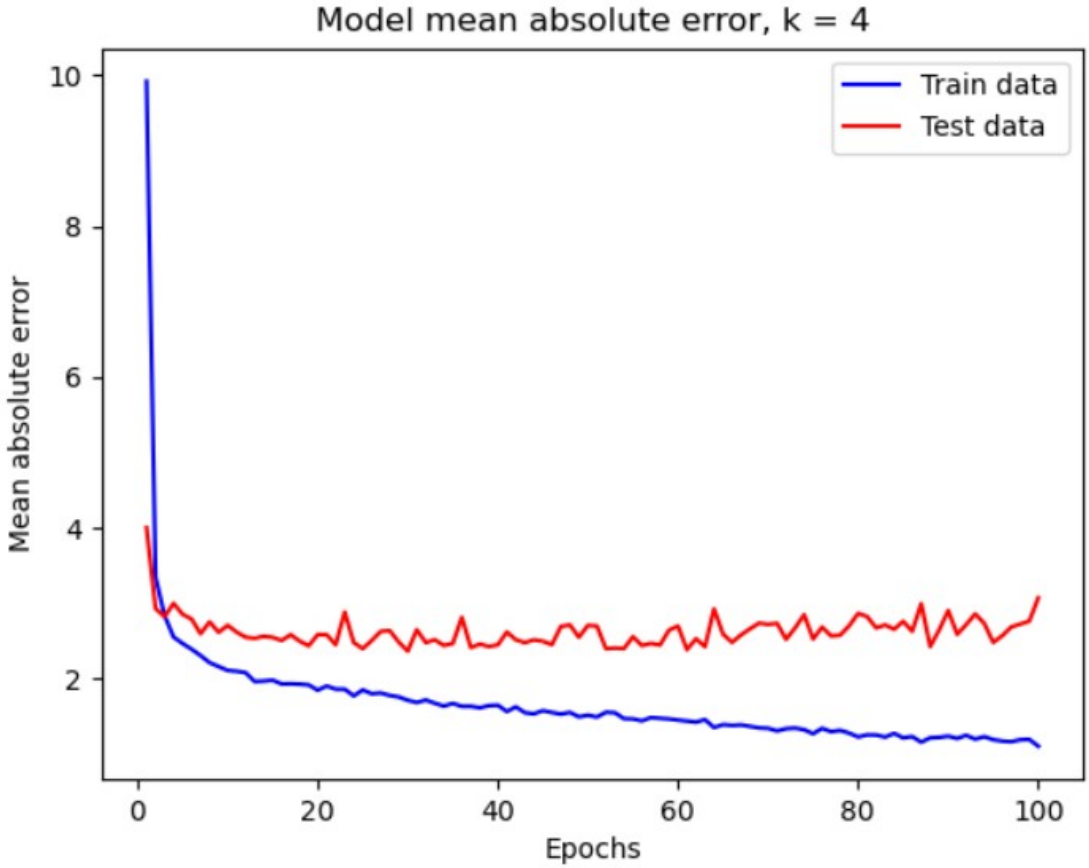

Рисунок 8 – График оценки mae k=4

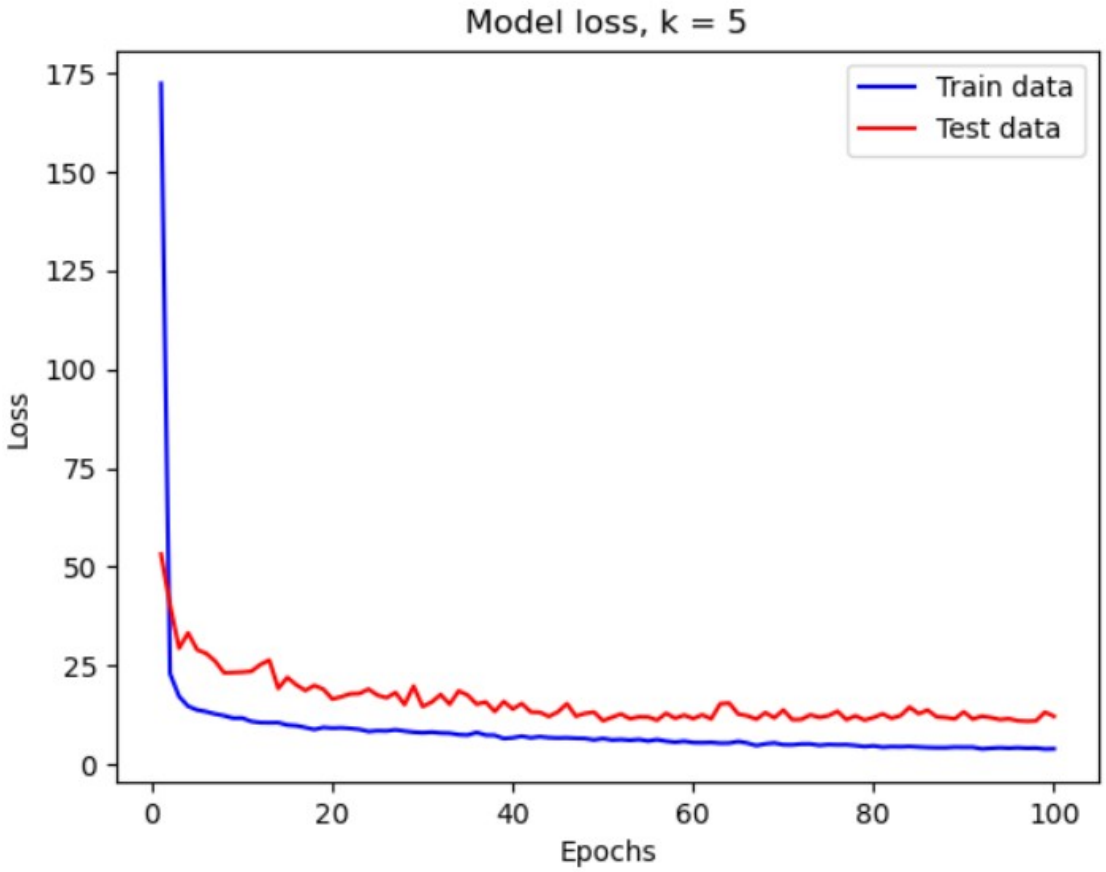

Рисунок 9 – График ошибки k=5

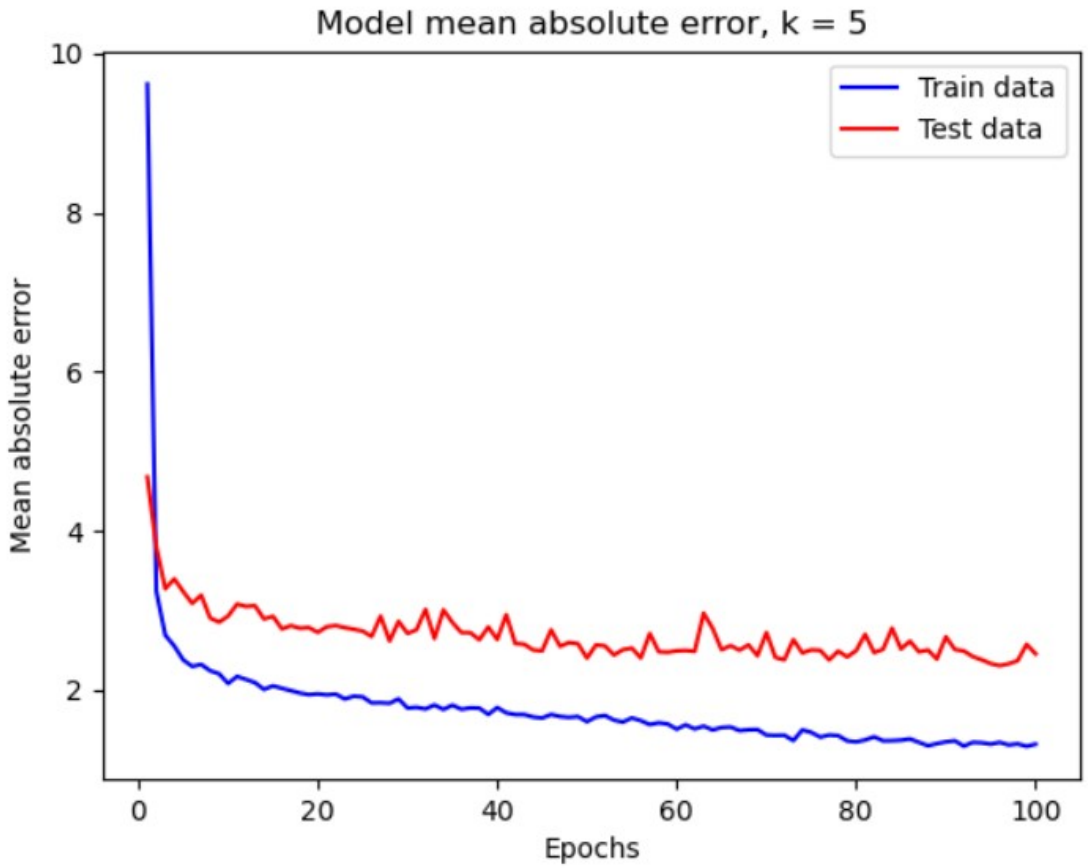

Рисунок 10 – График оценки mae k=5

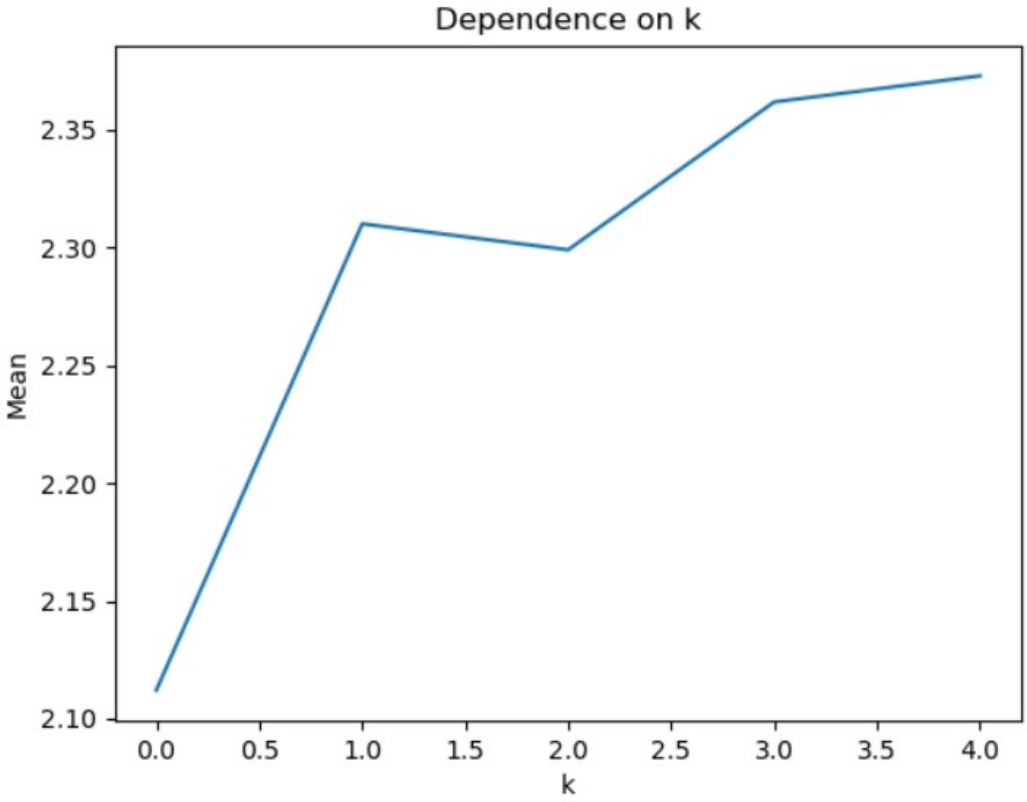

Рисунок 13 – Зависимость средней точности от числа блоков K при перекрестной проверке

Из графиков видно, что на 40 эпохах модель начинает переобучаться, так как потери на тренировочных данных продолжали уменьшаться, а на тестовых оставались прежними. Поэтому оптимальным вариантом является модель с 2-мя блоками, так как на этом k отклонение минимально, а значит достигается максимальная точность.

Выводы.

В ходе работы было изучено влияние числа эпох на результат обучения в задаче регрессии, найдена точка переобучения, которое происходит на 40 эпохах. Оптимальным вариантом будет модель с 2-мя блоками и 40 эпохами.

ПРИЛОЖЕНИЕ А

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

k = 5
num_val_samples = len(train_data) // k
num_epochs = 100
all_scores = []
res = []

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=0)
    all_scores.append(val_mae)
    res.append(np.mean(all_scores))
    history_dict = history.history
    loss = history_dict['loss']
    mae = history_dict['mean_absolute_error']
    vl_loss = history_dict['val_loss']
    vl_mae = history_dict['val_mean_absolute_error']
    epochs = range(1, num_epochs + 1)

    plt.plot(epochs, loss, 'b')
    plt.plot(epochs, vl_loss, 'r')
    plt.title('Model loss, k = ' + str(i+1))
    plt.ylabel('Loss')
    plt.xlabel('Epochs')
    plt.legend(['Train data', 'Test data'], loc='upper right')
    plt.show()
```

```
    plt.plot(epochs, mae, 'b')
    plt.plot(epochs, vl_mae, 'r')
    plt.title('Model mean absolute error, k = ' + str(i+1))
    plt.ylabel('Mean absolute error')
    plt.xlabel('Epochs')
    plt.legend(['Train data', 'Test data'], loc='upper right')
    plt.show()

plt.plot(range(k), res)
plt.title('Dependence on k')
plt.ylabel('Mean')
plt.xlabel('k')
plt.show()

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