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. 7382 _____ Дерябина П.С.

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

- 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 к выполнению задания

- 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

Задачей классификации является предсказание дискретного значения, которое является обозначением какого-либо класса, для набора входных данных. Задачи регрессии же является предсказание самого значения на непрерывной числовой прямой.

Код программы, реализующий структуру ИНС, приведен в приложении А.

Так как предоставленный датасет имеет маленький размер, было решено применить перекрестную проверку по какому-то количеству блоков. Было протестировано количество блоков с 4 по 6, рассмотрим метрики точности модели и функции потерь для проверки с 6 блоками. Точность оценивается с помощью средней абсолютной ошибки (рис. 1-3).

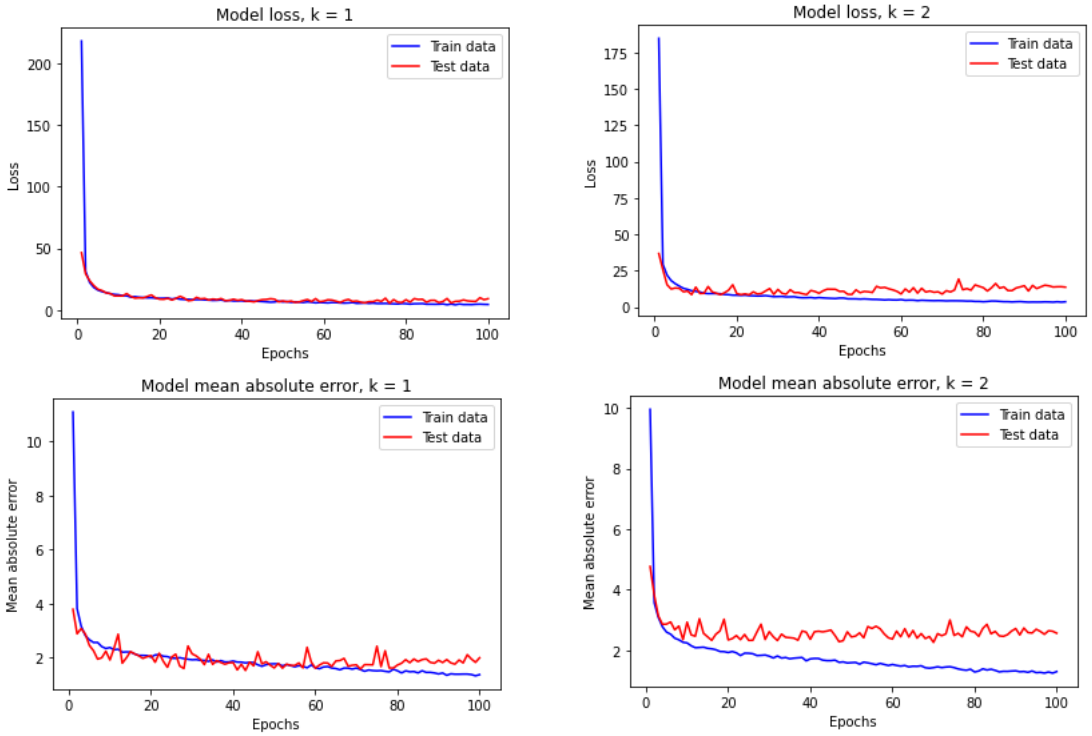

Рисунок 1 – Функция потерь и точность модели для k=1,2

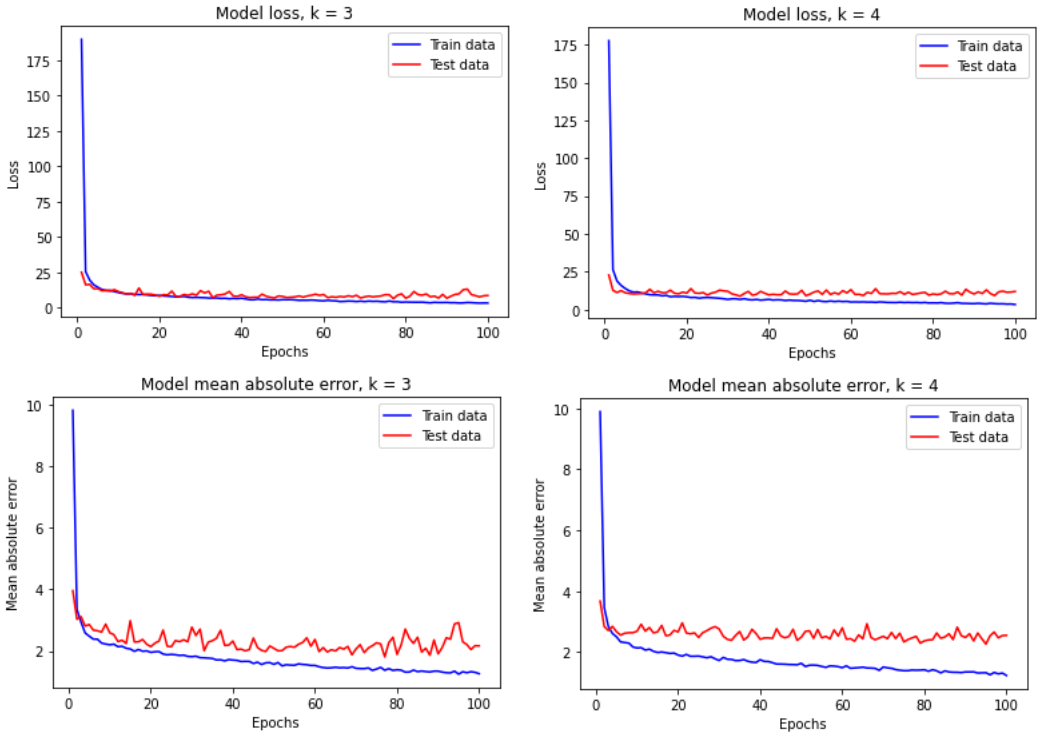

Рисунок 2 – Функция потерь и точность модели для k=3,4

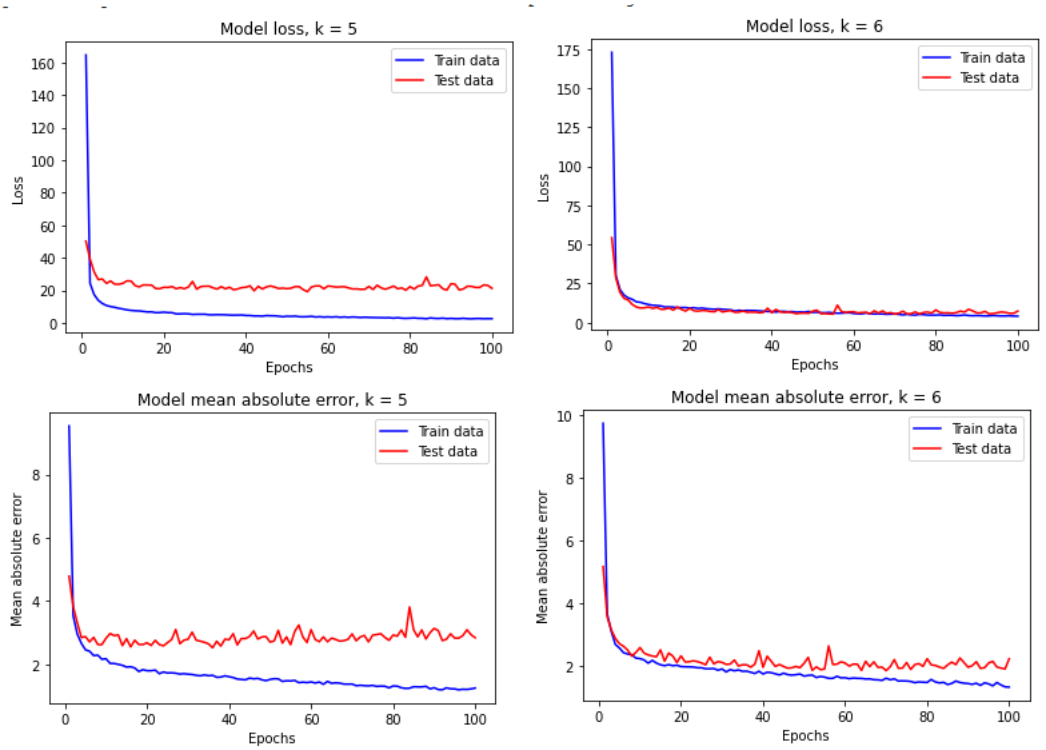

Рисунок 3 – Функция потерь и точность модели для k=5,6

Посмотрим на зависимость средней точности от количества блоков на рис. 4.

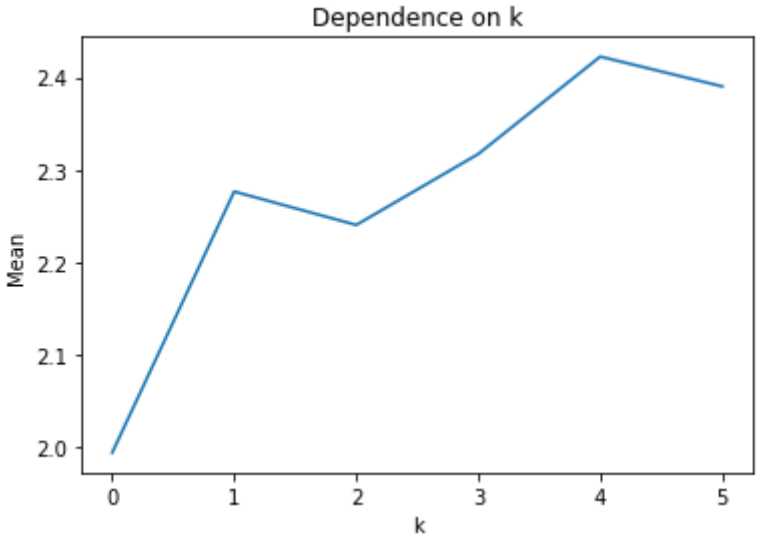

Рисунок 4 – Средняя точность

Из графиков 1-3 видно, что область, где кривые для тренировочных и тестовых данных начинаются расходиться (причем тренировочные расположены ниже), находится примерно между 40 и 60 эпохой, поэтому точкой переобучения будем считать 50. Оптимальным вариантом количества блоков является 2, так как на этом значении отклонение минимально, а значит достигается максимальная точность.

Вывод

В ходе работы было изучено влияние числа эпох на результат обучения в задаче регрессии, найдена точка переобучения и оптимальное число блоков для перекрестной проверки.

ПРИЛОЖЕНИ А
ИСХОДНЫЙ КОД

```
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

#1
(train_data, train_targets), (test_data, test_targets) = boston_housing.load
_data()
print(train_data.shape)
print(test_data.shape)
print(test_targets)

#2
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /= std
test_data -= mean
test_data /= std

#3
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

#4
k = 6
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_sampl
es]
    partial_train_data = np.concatenate([train_data[:i * num_val_samples],
                                         train_data[(i + 1) * num_val_sample
s:]],
                                         axis=0)
```

```
    partial_train_targets = np.concatenate([train_targets[:i * num_val_sampl
es],
                                            train_targets[(i + 1) * num_val_
samples:]],
                                           axis=0)
    model = build_model()
    h = model.fit(partial_train_data, partial_train_targets, epochs=num_epoc
hs, batch_size=1, verbose=0, validation_data=(val_data, val_targets))
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    history = h.history
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
    all_scores.append(val_mae)
    res.append(np.mean(all_scores))
    loss = history['loss']
    mae = history['mae']
    val_loss = history['val_loss']
    val_mae = history['val_mae']
    epochs = range(1, num_epochs + 1)

    plt.plot(epochs, loss, 'b')
    plt.plot(epochs, val_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, val_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()

print(np.mean(all_scores))

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

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