**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 |  | Дрозд А. 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.**

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.**

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

1. Задача классификации требует, разделения объектов в один или два класса.
2. Классификация может иметь действительные или дискретные входные переменные.
3. Проблема с двумя классами часто называется проблемой двухклассной или двоичной классификации.
4. Проблема с более чем двумя классами часто называется проблемой классификации нескольких классов.
5. Проблема, когда для примера назначается несколько классов, называется проблемой классификации по нескольким меткам.

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

1. Задача регрессии требует предсказания количества.
2. Регрессия может иметь действительные или дискретные входные переменные.
3. Проблема с несколькими входными переменными часто называется проблемой многомерной регрессии.
4. 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 модели, обучение проводилось при различных значениях количества эпох и k – количество блоков, на которые делились тренировочные данные.

Рассмотрим модель с 6-ю блоками. Точность будем оценивать с помощью средней абсолютной ошибки. Графики ошибок и точности предоставлены на рис. 1-12.

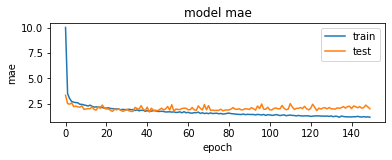


Рисунок 1 – График оценки mae k=1

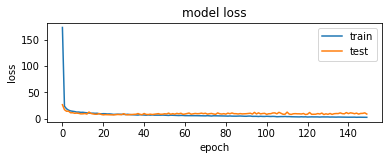


Рисунок 2 – График ошибки k=1

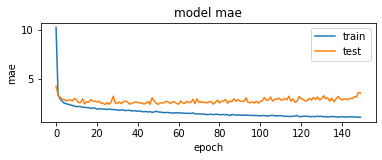


Рисунок 3 – График оценки mae k=2

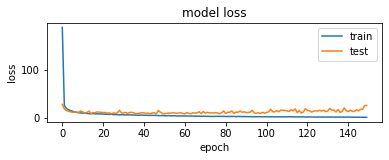


Рисунок 4 – График ошибки k=2

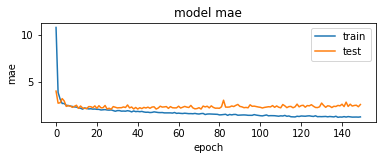


Рисунок 5 – График оценки mae k=3

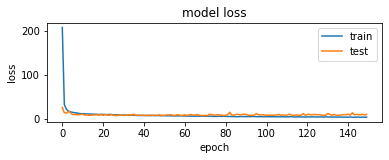


Рисунок 6 – График ошибки k=3

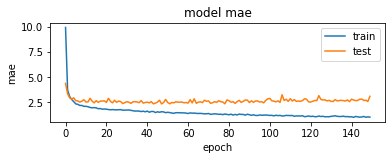


Рисунок 7 – График оценки mae k=4

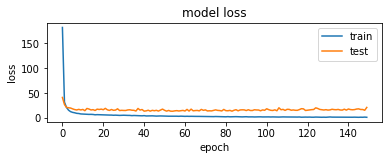


Рисунок 8 – График ошибки k=4

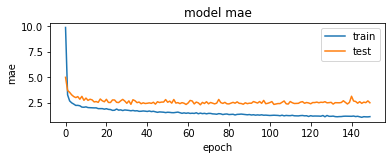


Рисунок 9 – График оценки mae k=5

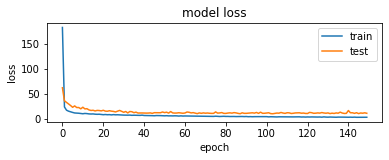


Рисунок 10 – График ошибки k=5

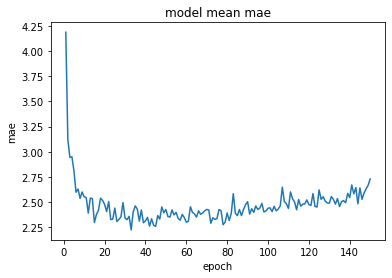


Рисунок 11 – Усредненный по всем моделям график среднеквадратичной ошибки

По графикам определили, что примерно на 40 эпохе модель начинает переобучаться, так как потери на тренировочных данных продолжались уменьшаться, а на тестовых не изменялись, это означает, что модель начинает излишне обучаться на этих данных и не дает результатов на незнакомых данных.

Рассмотрим графики ошибок при разном количестве k-блоков.

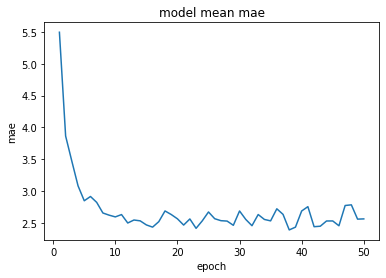


Рисунок 12 – Усредненный по всем моделям график среднеквадратической ошибки при k=2

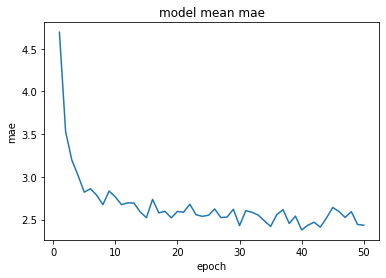


Рисунок 13 – Усредненный по всем моделям график среднеквадратической ошибки при k=3

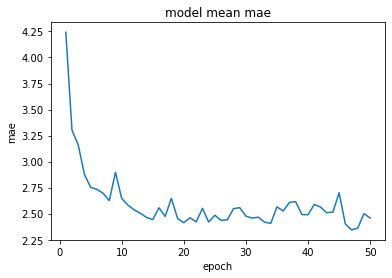


Рисунок 14 – Усредненный по всем моделям график среднеквадратической ошибки при k=4

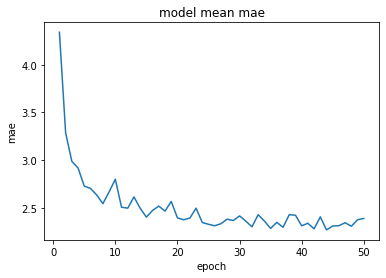


Рисунок 15 – Усредненный по всем моделям график среднеквадратической ошибки при k=5

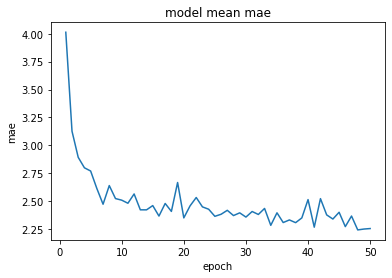


Рисунок 16 – Усредненный по всем моделям график среднеквадратической ошибки при k=6

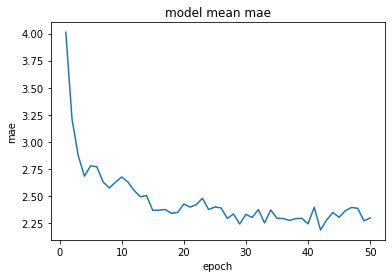


Рисунок 17 – Усредненный по всем моделям график среднеквадратической ошибки при k=7

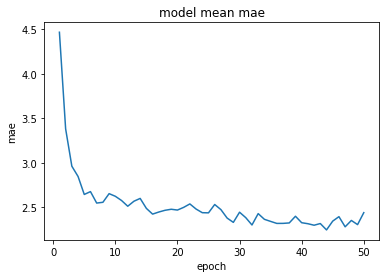


Рисунок 18 – Усредненный по всем моделям график среднеквадратической ошибки при k=8

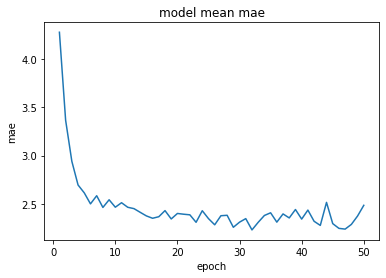


Рисунок 19 – Усредненный по всем моделям график среднеквадратической ошибки при k=9

Исходя из рис. 12-19 можно сказать что оптимальное k для нашей задачи 7, так как именно при этом значении среднеквадратическая ошибка минимальна.

**Выводы.**

В ходе работы было изучено влияние числа эпох на результат обучения в задаче регрессии, найдена точка переобучения, которое происходит на 40 эпохе. Оптимальным вариантом будет модель с 7-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.utils import to\_categorical

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 = 9

num\_val\_samples = len(train\_data) // k

num\_epochs = 50

all\_scores = []

mae = []

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, 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)

mae.append(history.history['val\_mean\_absolute\_error'])

plt.subplot(211)

plt.plot(history.history['mean\_absolute\_error'])

plt.plot(history.history['val\_mean\_absolute\_error'])

plt.title('model mae')

plt.ylabel('mae')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper right')

plt.show()

plt.subplot(212)

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper right')

plt.show()

print(np.mean(all\_scores))

mean\_mae\_history = [np.mean([x[i] for x in mae]) for i in range(num\_epochs)]

plt.plot(range(1, num\_epochs + 1), mean\_mae\_history)

plt.title('model mean mae')

plt.ylabel('mae')

plt.xlabel('epoch')

plt.show()