Федеральное агентство связи

Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет

телекоммуникаций и информатики» (СибГУТИ)

# Лабораторная работа №4

Выполнил: студент IV курса ИВТ,

гр. ИП-813

Бурдуковский И.А. Проверила: ассистент кафедры ПМиК

Морозова К.И.

Новосибирск, 2021 г.

Оглавление

[Задание 3](#_Toc58850857)

[Выполнение 4](#_Toc58850858)

[Код программы 5](#_Toc58850859)

# Задание

Целью данной лабораторной работы является разработка нейронной сети для решения задачи классификации или регрессии в зависимости от набора данных в рамках варианта. Лабораторная работа предполагает разработку на языке программирования Python с использованием библиотеки Keras.

Определение цены недвижимости (Boston Housing price regression dataset) Все наборы данных доступны по ссылке: https://keras.io/api/datasets/

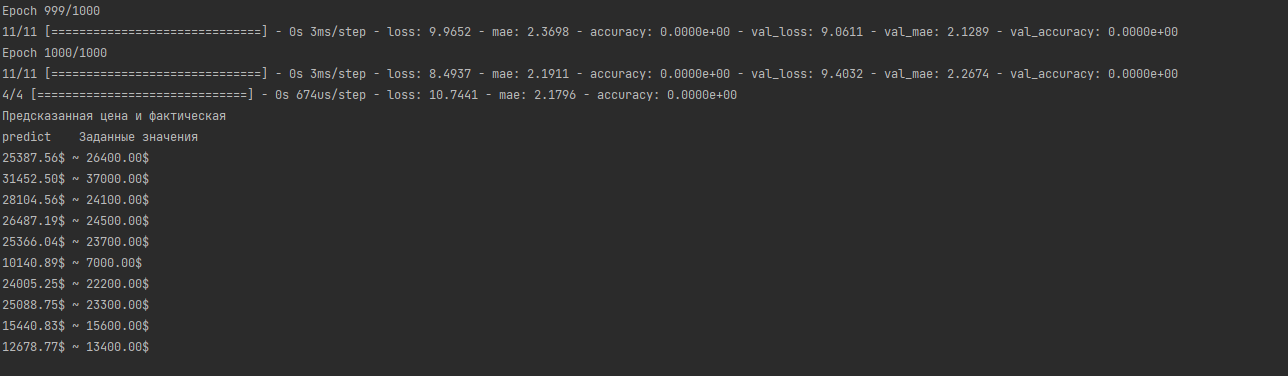
При разработке нейронной сети следует соблюсти наличие необходимых составляющих исходя из следующих вариантов:

Нейросеть должна состоять из четырех полносвязных слоёв, обязательное использование GaussianDropout, в качестве оптимизатора использовать SGD.

Выбор количества нейронов на всех внутренних слоях, функций активации и других параметров должен быть обусловлен оптимальностью работы модели.

# Выполнение

Результат работы нейросети:



# Код программы

import random  
from tensorflow import metrics  
from tensorflow.keras.datasets import cifar10  
from sklearn.preprocessing import StandardScaler  
from tensorflow.keras import models, layers, constraints  
from tensorflow.keras import utils  
from keras.constraints import maxnorm  
from tensorflow.keras.datasets import boston\_housing  
import scipy  
from sklearn.preprocessing import StandardScaler  
from tensorflow.keras import models, layers  
  
(X\_train, y\_train), (X\_test, y\_test) = boston\_housing.load\_data(seed=1)  
# print(X\_train[0], y\_train[0])  
scaler = StandardScaler()  
scaler.fit(X\_train)  
X\_train\_scaled = scaler.transform(X\_train)  
X\_test\_scaled = scaler.transform(X\_test)  
# print(X\_train\_scaled[0])  
  
model = models.Sequential()  
model.add(layers.Dense(64, activation='relu', input\_shape=[X\_train.shape[1]]))  
model.add(layers.GaussianDropout(0.2))  
model.add(layers.Dense(64, activation='relu'))  
model.add(layers.GaussianDropout(0.2))  
model.add(layers.Dense(1))  
  
model.compile(optimizer='adam', loss='mse', metrics=['mae', 'accuracy'])  
# model.compile(optimizer='sgd', loss='mse', metrics=['mae'])  
model.fit(X\_train\_scaled, y\_train, validation\_split=0.2, epochs=500, batch\_size=32)  
model.evaluate(X\_test\_scaled, y\_test)  
  
num = 10;  
to\_predict = X\_test\_scaled[:num]  
predictions = model.predict(to\_predict)  
  
  
print("Предсказанная цена и фактическая")  
print("predict Заданные значения");  
for i in range(num):  
 print('%.2f$ ~ %.2f$' %(predictions[i]\*1000, y\_test[i]\*1000))  
  
# print(X\_test\_scaled.shape)  
# to\_predict\_sum = X\_test\_scaled  
# predictions\_sum = model.predict(to\_predict\_sum)  
# print("Проц = ", metrics.mean\_absolute\_error(y\_test, predictions\_sum))

))