Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

**09**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 01 | Write a python program which can read any dataset and implement Neural Network |
| 02 |  |
|  |  |

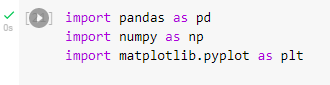
Submitted On:

**Date: 17 JUNE 2022**

**EXAMPLE:**

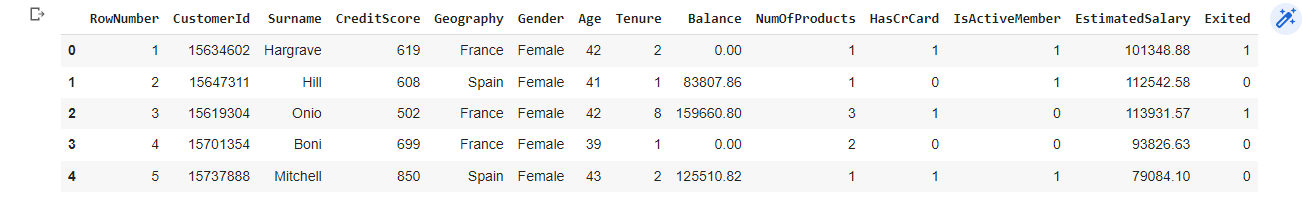
*“IMPLEMENTATION OF ANN”*

*Solution:*



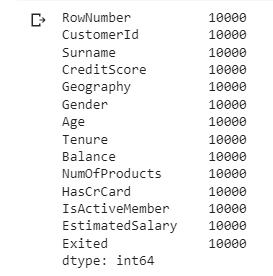
*Output:*



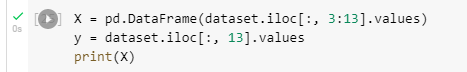
*Solution:*



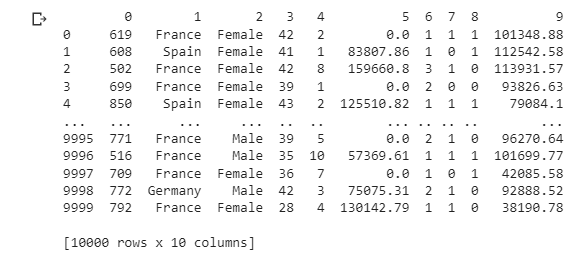
*Output:*



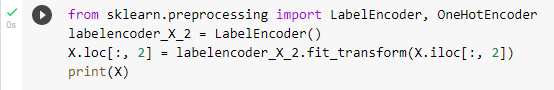
*Solution:*



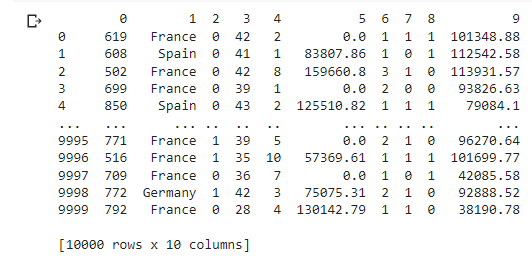
*Output:*



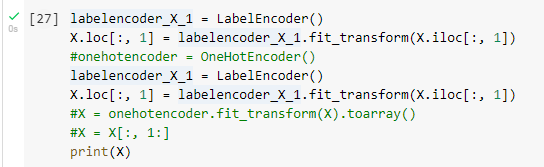
*Solution:*



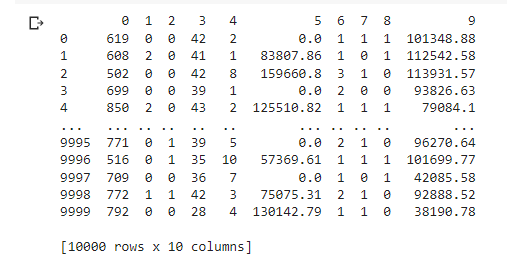
*Output:*



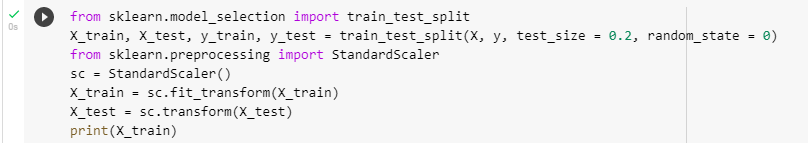
*Solution:*



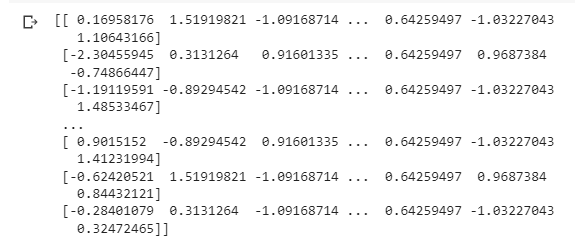
*Output:*



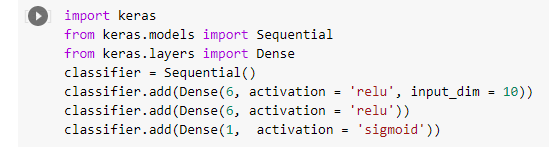
*Solution:*



*Output:*



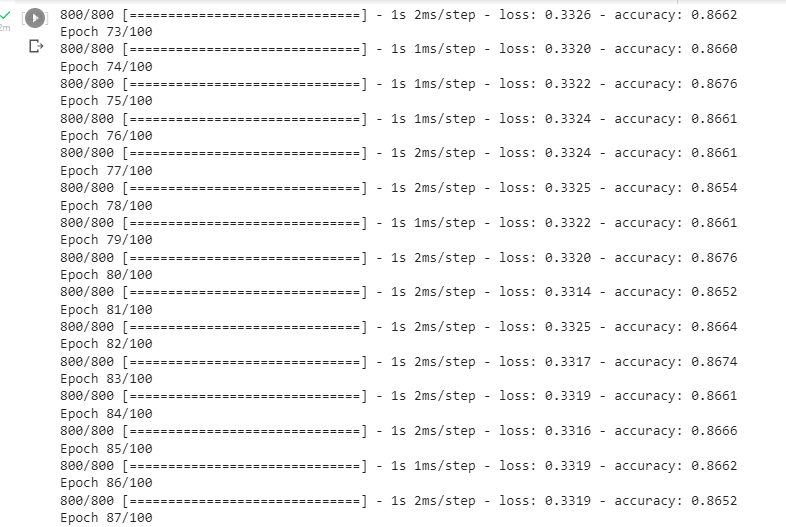
*Solution:*

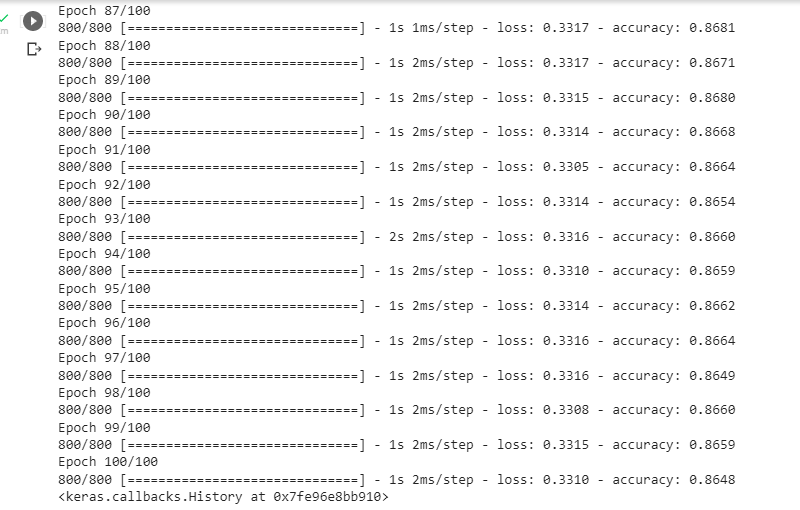




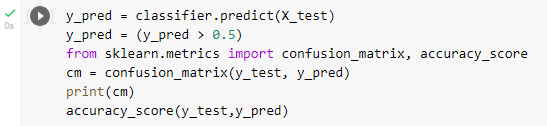


*Output:*

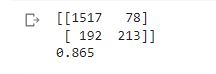




*Solution:*



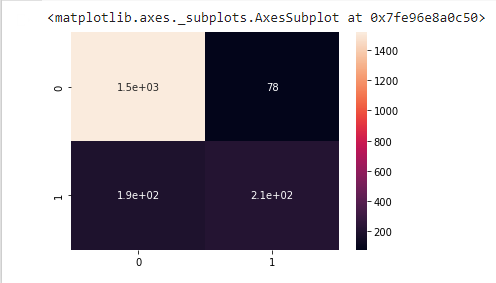
*Output:*



*Solution:*



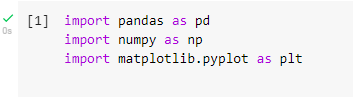
*Output:*



**Task No. 1:**

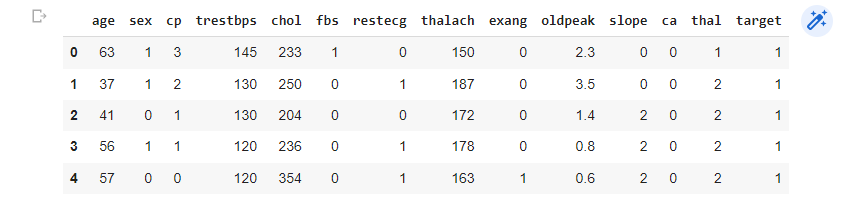
Write a python program which can read any dataset and implement Neural Network

*Solution:*



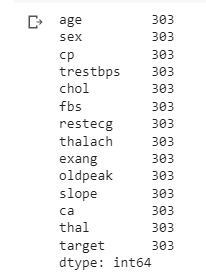
*Output:*



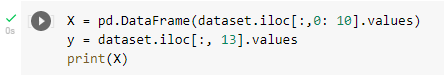
*Solution:*



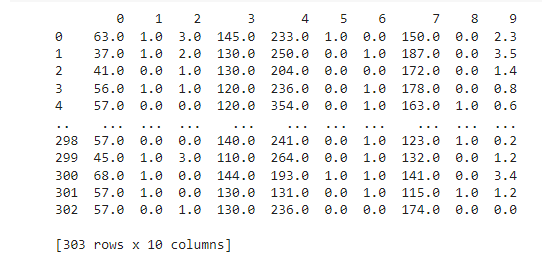
*Output:*



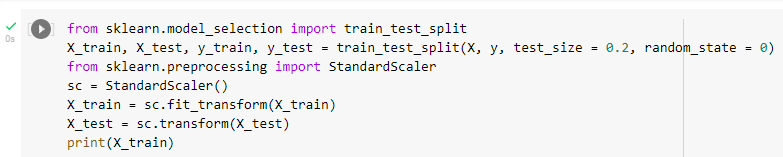
*Solution:*



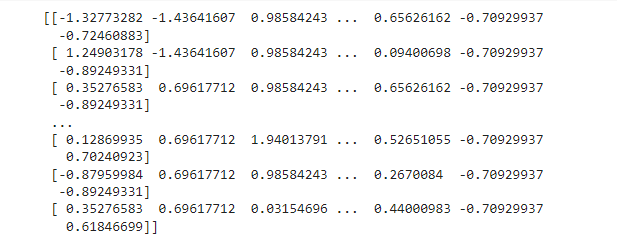
*Output:*



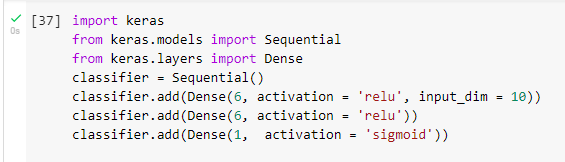
*Solution:*



*Output:*



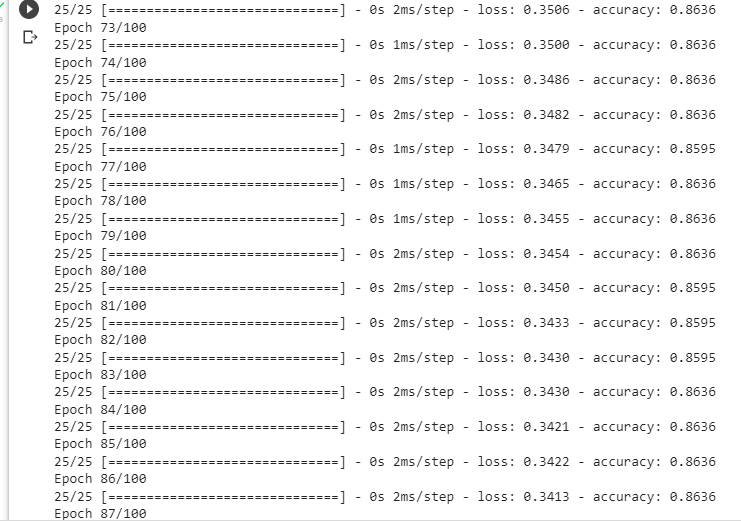
*Solution:*

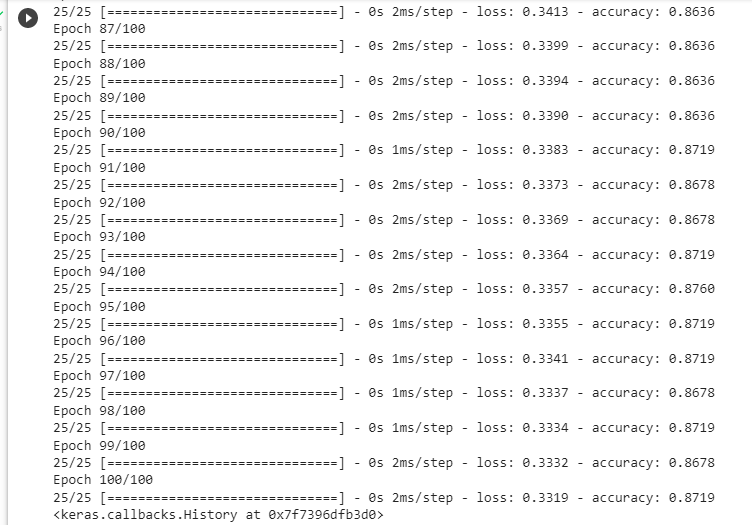




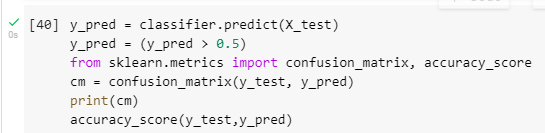


*Output:*

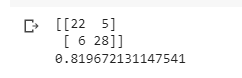




*Solution:*



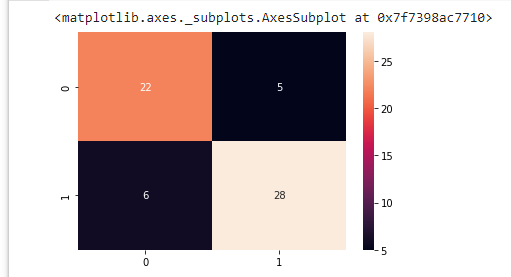
*Output:*



*Solution:*



*Output:*



**Task No. 2:**

Write a python program which can read any dataset and implement Neural Network

*Solution:*

import pandas as pd

import numpy as np

# To remove the scientific notation from numpy arrays

np.set\_printoptions(suppress=True)

CarPricesDataNumeric=pd.read\_pickle('/content/sample\_data/CarPricesData.pkl')

CarPricesDataNumeric.head()

Table

Description automatically generated

TargetVariable=['Price']

Predictors=['Age', 'KM', 'Weight', 'HP', 'MetColor', 'CC', 'Doors']

X=CarPricesDataNumeric[Predictors].values

y=CarPricesDataNumeric[TargetVariable].values

### Sandardization of data ###

from sklearn.preprocessing import StandardScaler

PredictorScaler=StandardScaler()

TargetVarScaler=StandardScaler()

# Storing the fit object for later reference

PredictorScalerFit=PredictorScaler.fit(X)

TargetVarScalerFit=TargetVarScaler.fit(y)

# Generating the standardized values of X and y

X=PredictorScalerFit.transform(X)

y=TargetVarScalerFit.transform(y)

# Split the data into training and testing set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Quick sanity check with the shapes of Training and testing datasets

print(X\_train.shape)

print(y\_train.shape)

print(X\_test.shape)

print(y\_test.shape)

Text

Description automatically generated

from keras.models import Sequential

from keras.layers import Dense

# create ANN model

model = Sequential()

# Defining the Input layer and FIRST hidden layer, both are same!

model.add(Dense(units=5, input\_dim=7, kernel\_initializer='normal', activation='relu'))

# Defining the Second layer of the model

# after the first layer we don't have to specify input\_dim as keras configure it automatically

model.add(Dense(units=5, kernel\_initializer='normal', activation='tanh'))

# The output neuron is a single fully connected node

# Since we will be predicting a single number

model.add(Dense(1, kernel\_initializer='normal'))

# Compiling the model

model.compile(loss='mean\_squared\_error', optimizer='adam')

# Fitting the ANN to the Training set

model.fit(X\_train, y\_train ,batch\_size = 20, epochs = 50, verbose=1)

A screenshot of a computer

Description automatically generated with medium confidence

model.fit(X\_train, y\_train ,batch\_size = 15, epochs = 5, verbose=0)

# Generating Predictions on testing data

Predictions=model.predict(X\_test)

# Scaling the predicted Price data back to original price scale

Predictions=TargetVarScalerFit.inverse\_transform(Predictions)

# Scaling the y\_test Price data back to original price scale

y\_test\_orig=TargetVarScalerFit.inverse\_transform(y\_test)

# Scaling the test data back to original scale

Test\_Data=PredictorScalerFit.inverse\_transform(X\_test)

TestingData=pd.DataFrame(data=Test\_Data, columns=Predictors)

TestingData['Price']=y\_test\_orig

TestingData['PredictedPrice']=Predictions

TestingData.head()

Table

Description automatically generated

APE=100\*(abs(TestingData['Price']-TestingData['PredictedPrice'])/TestingData['Price'])

TestingData['APE']=APE

print('The Accuracy of ANN model is:', 100-np.mean(APE))

TestingData.head()

Table

Description automatically generated

Table

Description automatically generated