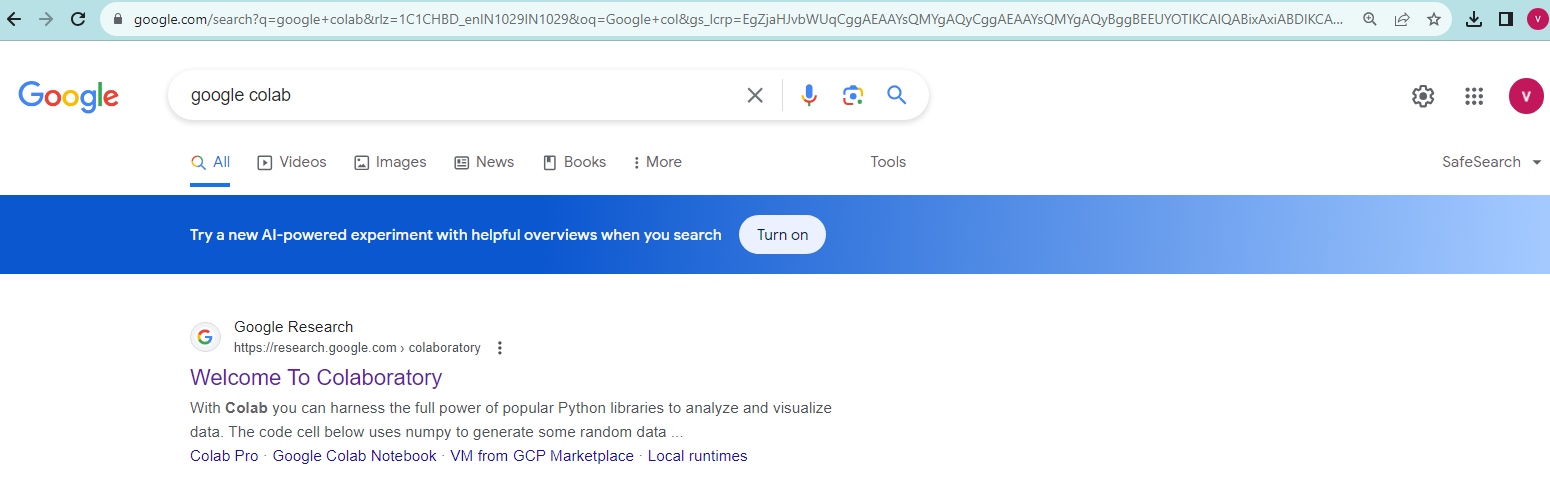
**Experiment No: 1(A)**

**Name:** Prem S. Kharat

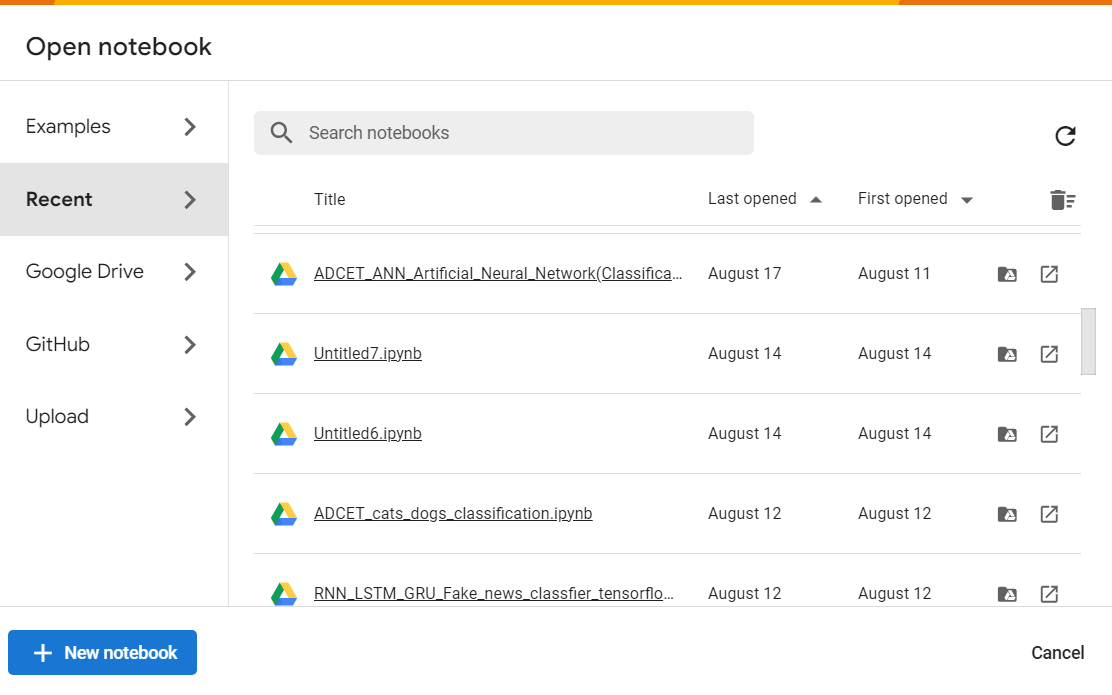
**Roll No:** 4102

**Title:** Google Colab Implementation Steps

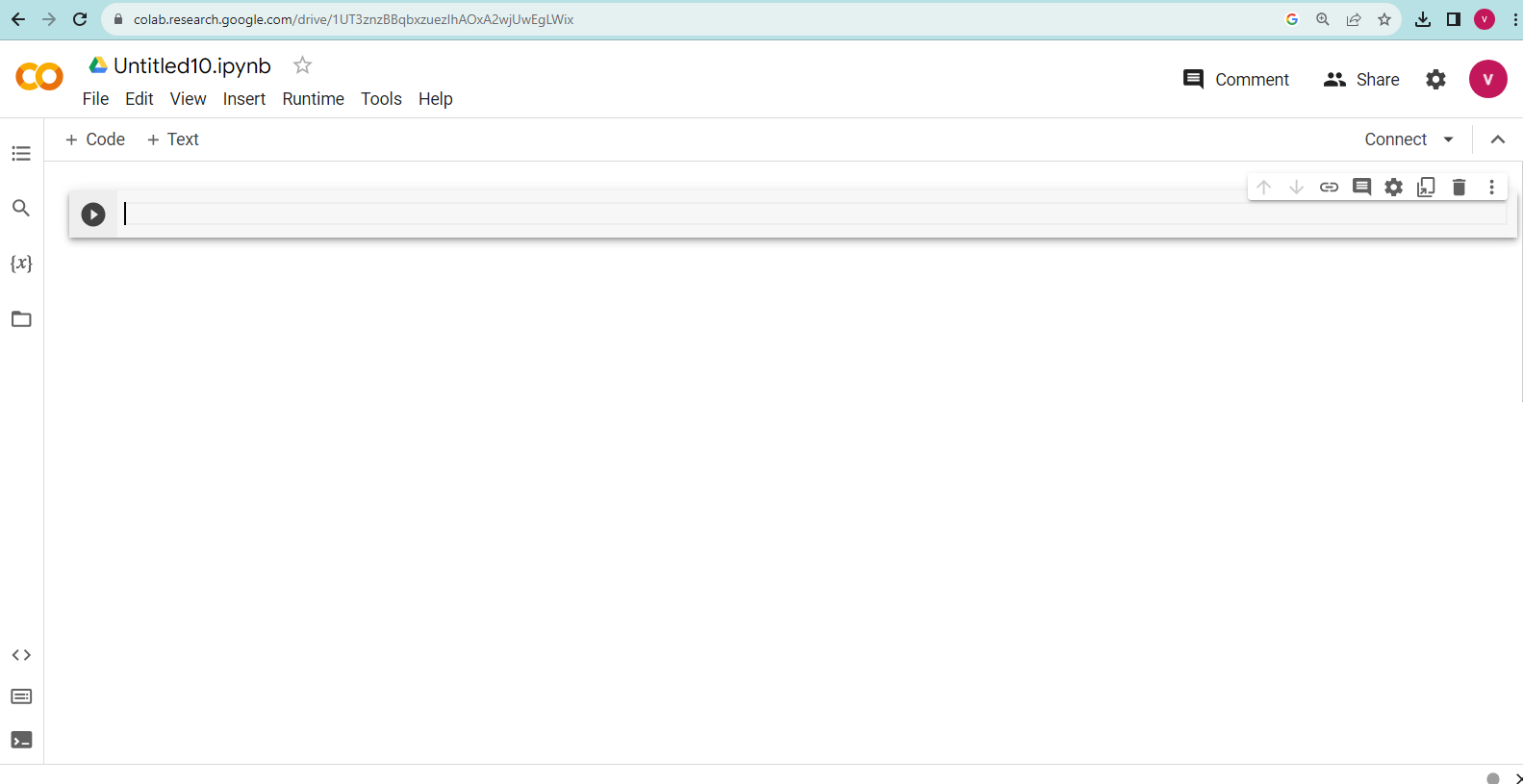
Step 1: Open Browser and type Google colab and open first link.



Step 2: In this link there is an option new notebook click on open notebook.



Step 3: It will open the interface to write a python code.



**Experiment No: 1(B)**

**Title:** Take Dataset Analyze and Classify it.

**Code:**

#Importing necessary Libraries

import numpy as np

import pandas as pd

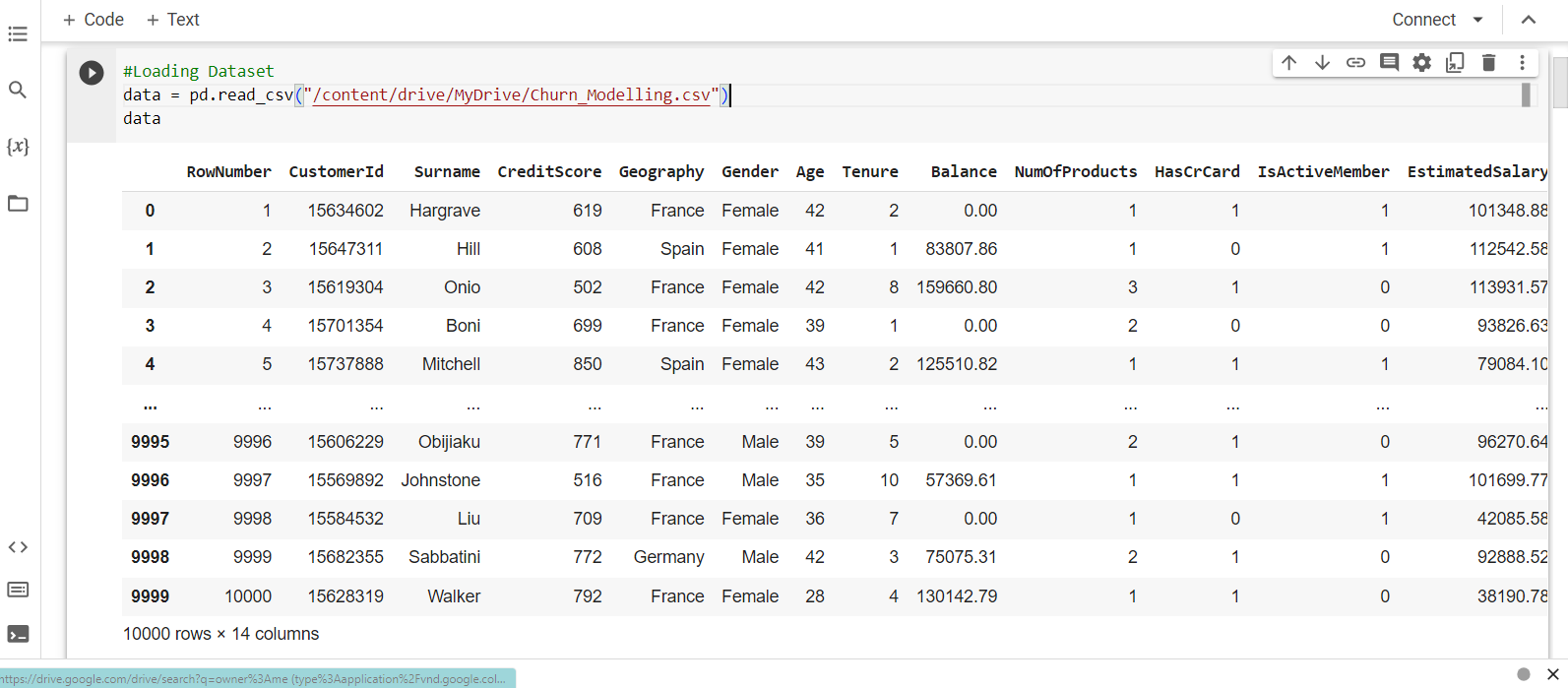
import tensorflow as tf

import matplotlib.pyplot as plt

#Loading Dataset

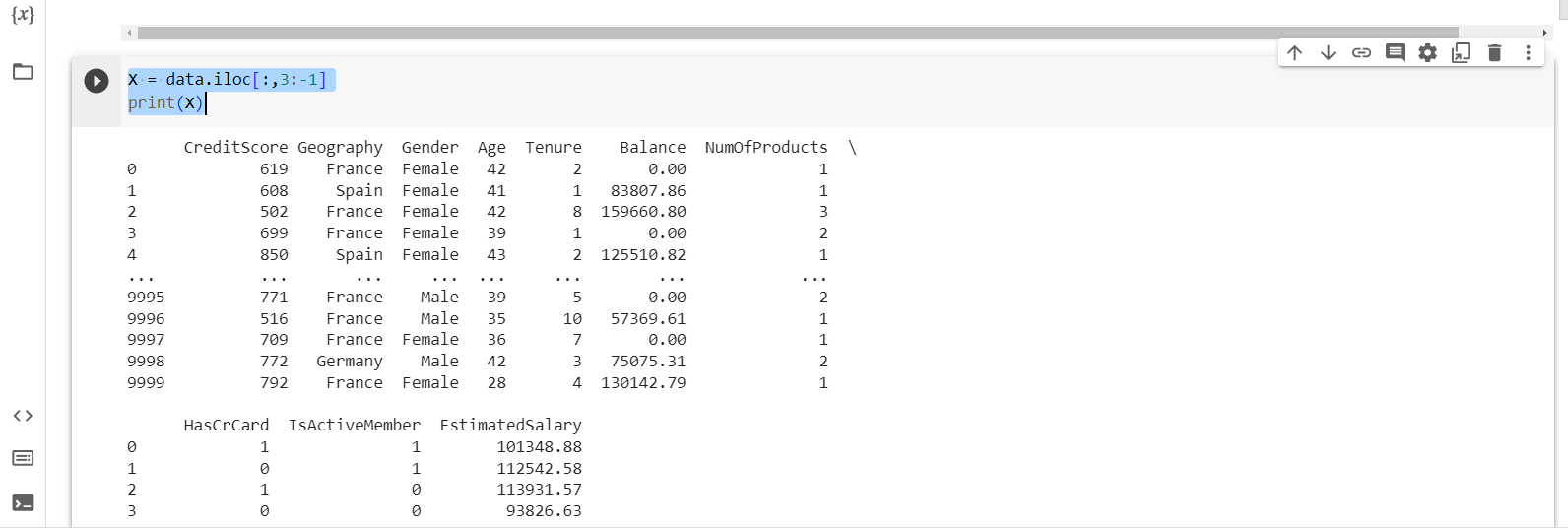
data = pd.read\_csv("/content/drive/MyDrive/Churn\_Modelling.csv")

data



X = data.iloc[:,3:-1]

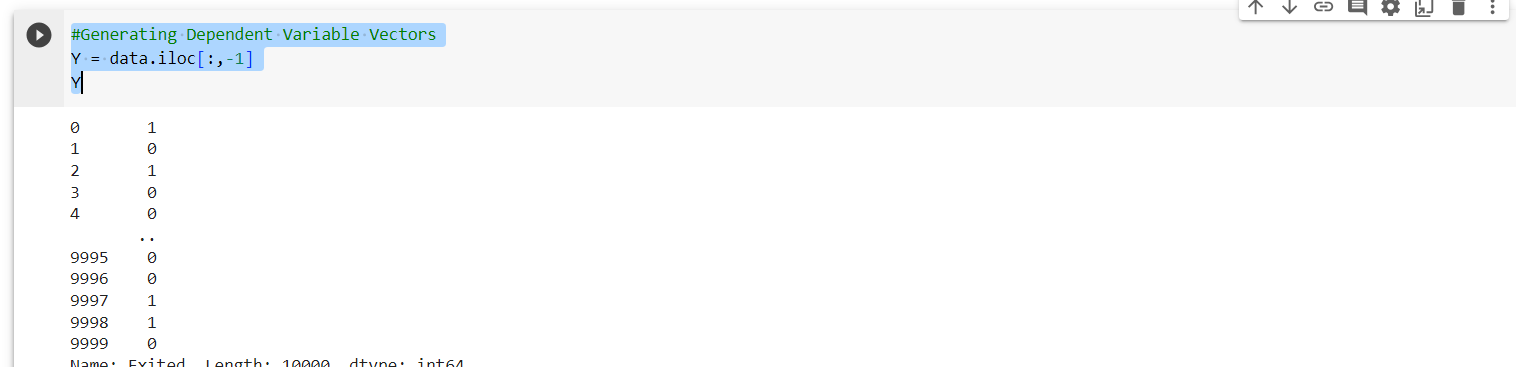
print(X)



#Generating Dependent Variable Vectors

Y = data.iloc[:,-1]

Y



#Create dummy variables

geography=pd.get\_dummies(X["Geography"],drop\_first=True)

gender=pd.get\_dummies(X['Gender'],drop\_first=True)

## Concatenate the Data Frames

X=pd.concat([X,geography,gender],axis=1)

## Drop Unnecessary columns

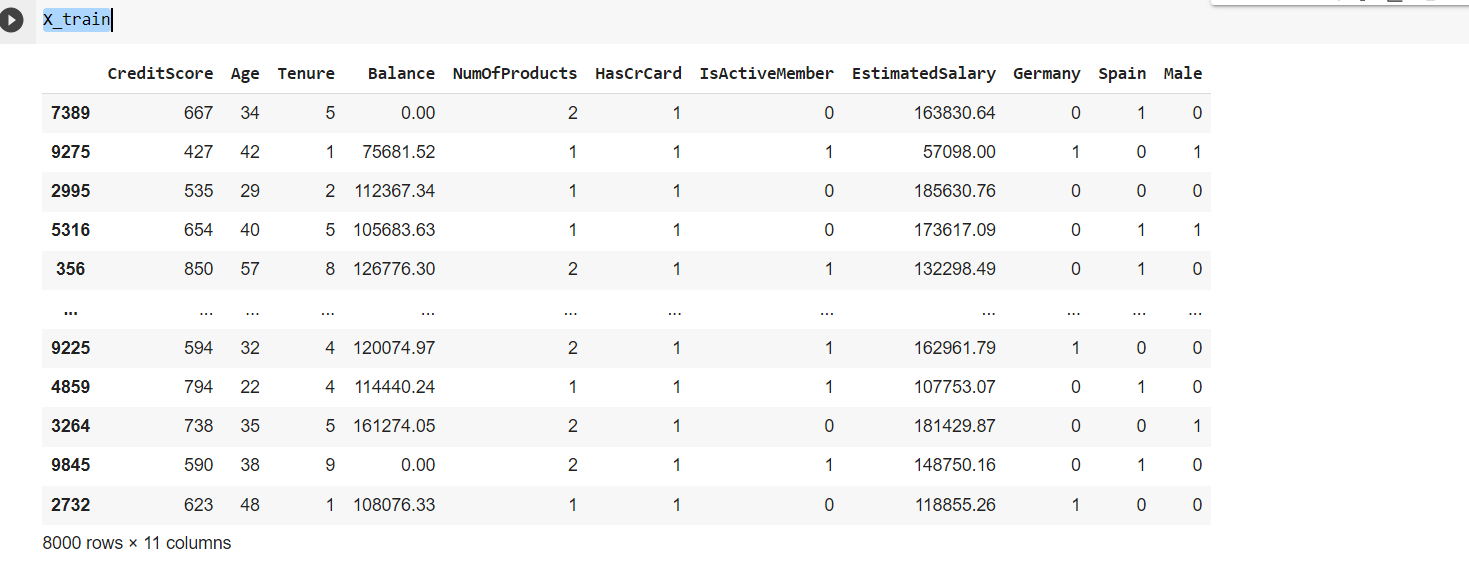
X=X.drop(['Geography','Gender'],axis=1)

# Splitting the dataset into the Training set and Test set

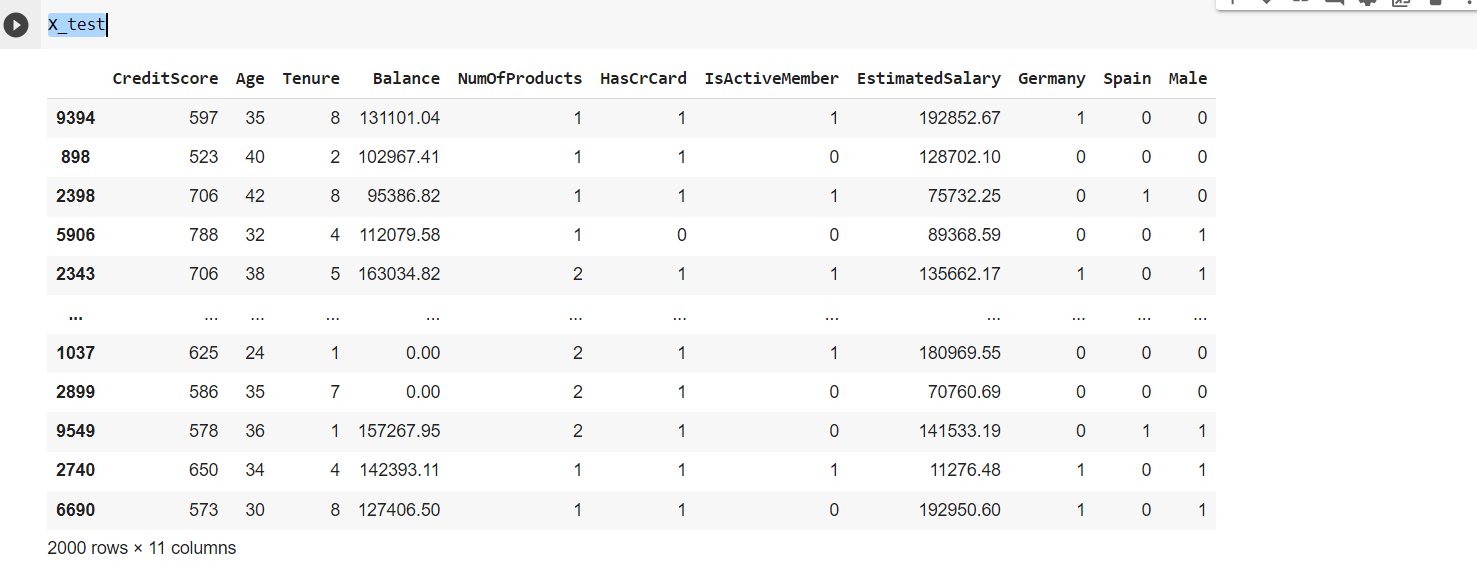
from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size = 0.2, random\_state = 0)

X\_train



X\_test



# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

from sklearn.linear\_model import LogisticRegression

LR=LogisticRegression()

LR

LR.fit(X\_train,y\_train)

y\_pred=LR.predict(X\_test)

y\_pred

LR.score(X\_train,y\_train)\*100

LR.score(X\_test,y\_test)\*100

# Importing the Keras libraries and packages

import keras

from keras.models import Sequential

from keras.layers import Dense



# Initialising the ANN

classifier = Sequential()

# Adding the input layer and the first hidden layer

#Initializers define the way to set the initial random weights of Keras layers.

classifier.add(Dense(units = 6, kernel\_initializer = 'he\_uniform',activation='relu',input\_dim = 11))

# Adding the second hidden layer

classifier.add(Dense(units = 6, kernel\_initializer = 'he\_uniform',activation='relu'))

# Adding the output layer

classifier.add(Dense(units= 1, kernel\_initializer = 'glorot\_uniform', activation = 'sigmoid'))

# Compiling the ANN

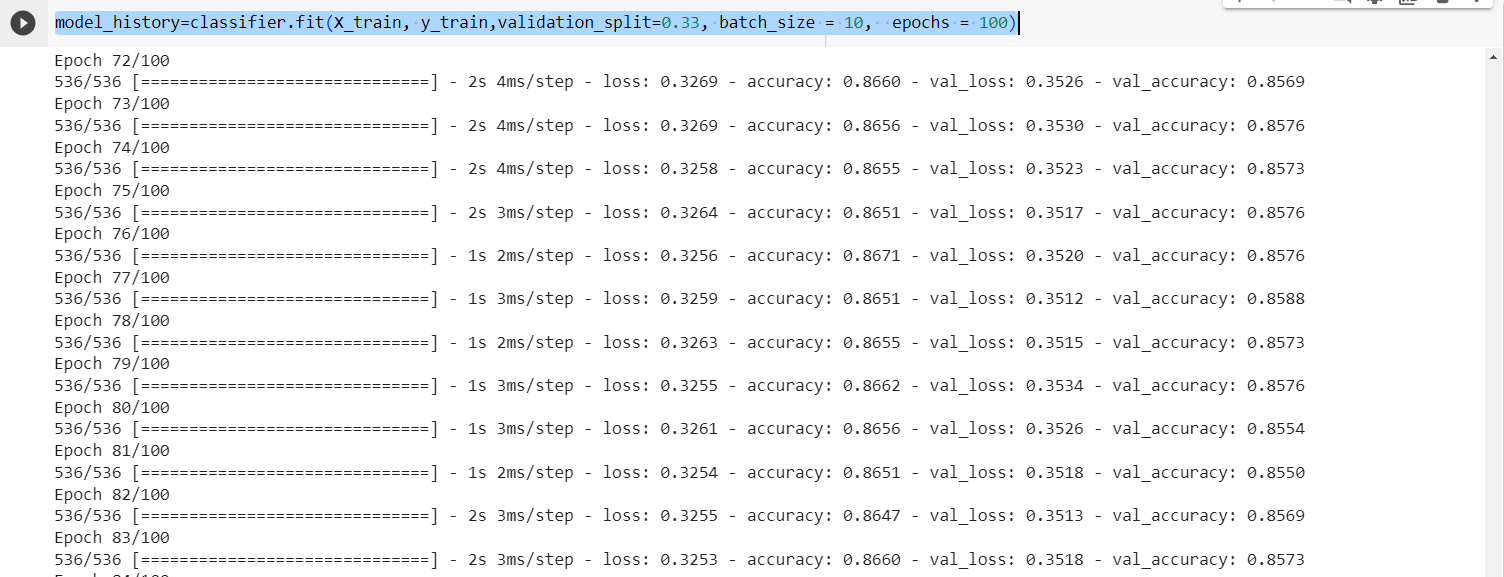
#Computes the cross-entropy loss between true labels and predicted labels.

#Use this cross-entropy loss for binary (0 or 1) classification applications.

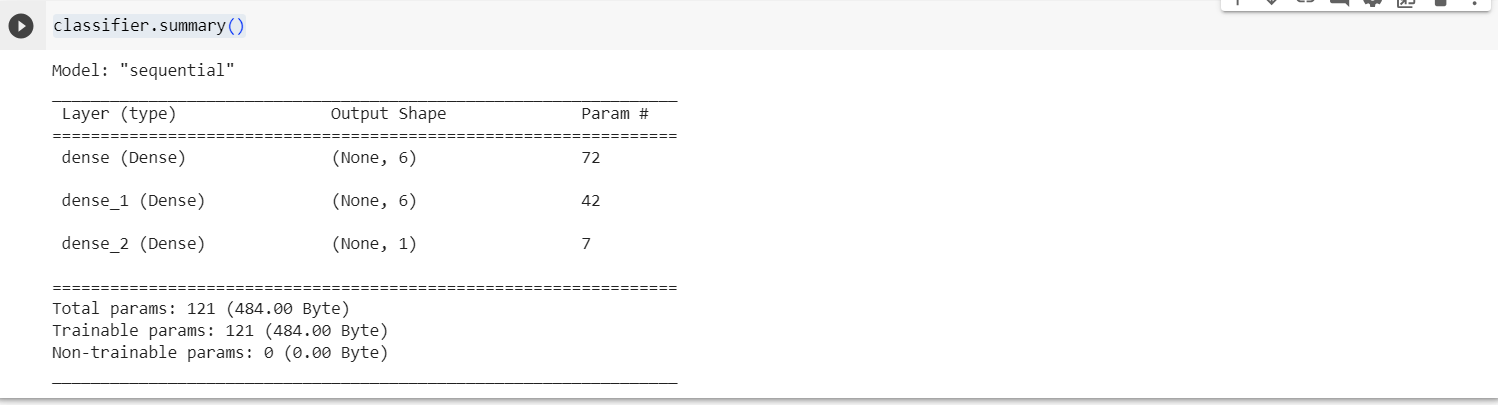
#Calculates how often predictions equal labels.

classifier.compile(optimizer = 'Adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

model\_history=classifier.fit(X\_train, y\_train,validation\_split=0.33, batch\_size = 10, epochs = 100)



classifier.summary()



# Predicting the Test set results

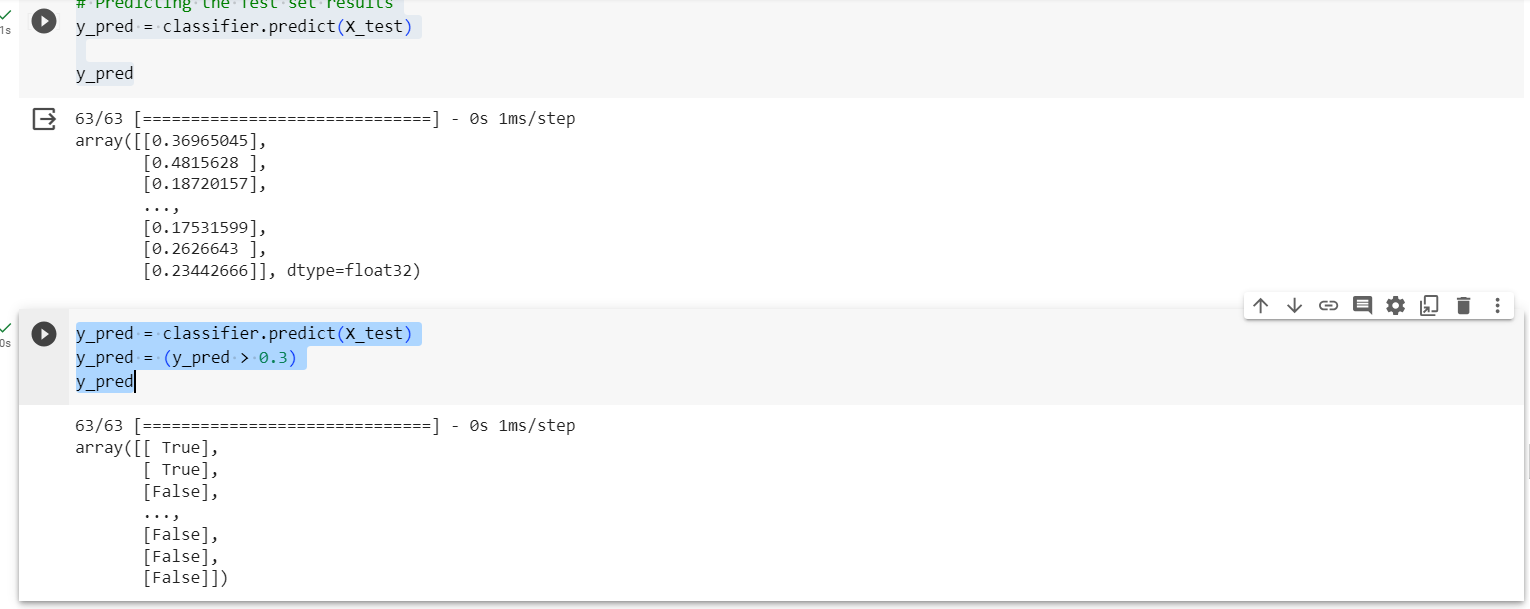
y\_pred = classifier.predict(X\_test)

y\_pred

y\_pred = classifier.predict(X\_test)

y\_pred = (y\_pred > 0.3)

y\_pred



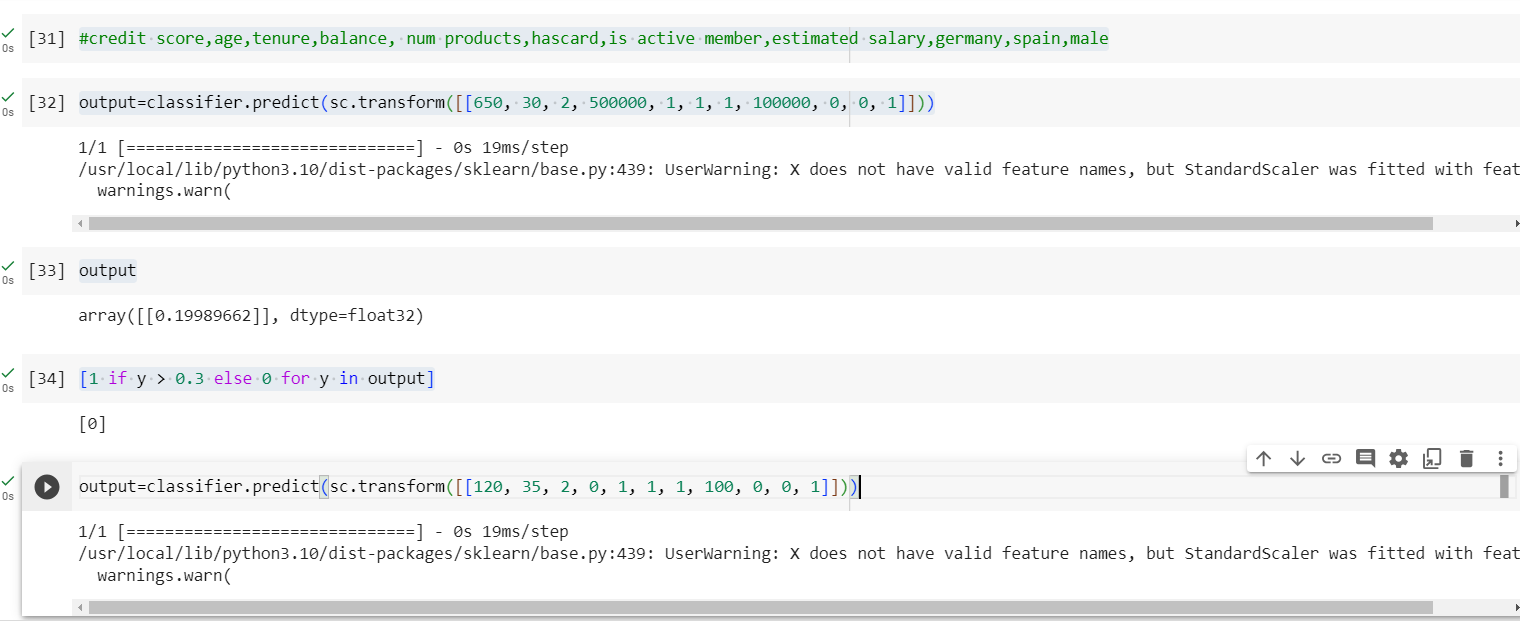
#credit score,age,tenure,balance, num products,hascard,is active member,estimated salary,germany,spain,male

output=classifier.predict(sc.transform([[650, 30, 2, 500000, 1, 1, 1, 100000, 0, 0, 1]]))

output

[1 if y > 0.3 else 0 for y in output]

output=classifier.predict(sc.transform([[120, 35, 2, 0, 1, 1, 1, 100, 0, 0, 1]]))



output

[1 if y > 0.3 else 0 for y in output]

# Calculate the Accuracy

from sklearn.metrics import accuracy\_score

score=accuracy\_score(y\_pred,y\_test)

score

print(model\_history.history.keys())

# summarize history for accuracy

plt.plot(model\_history.history['accuracy'])

plt.plot(model\_history.history['val\_accuracy'])

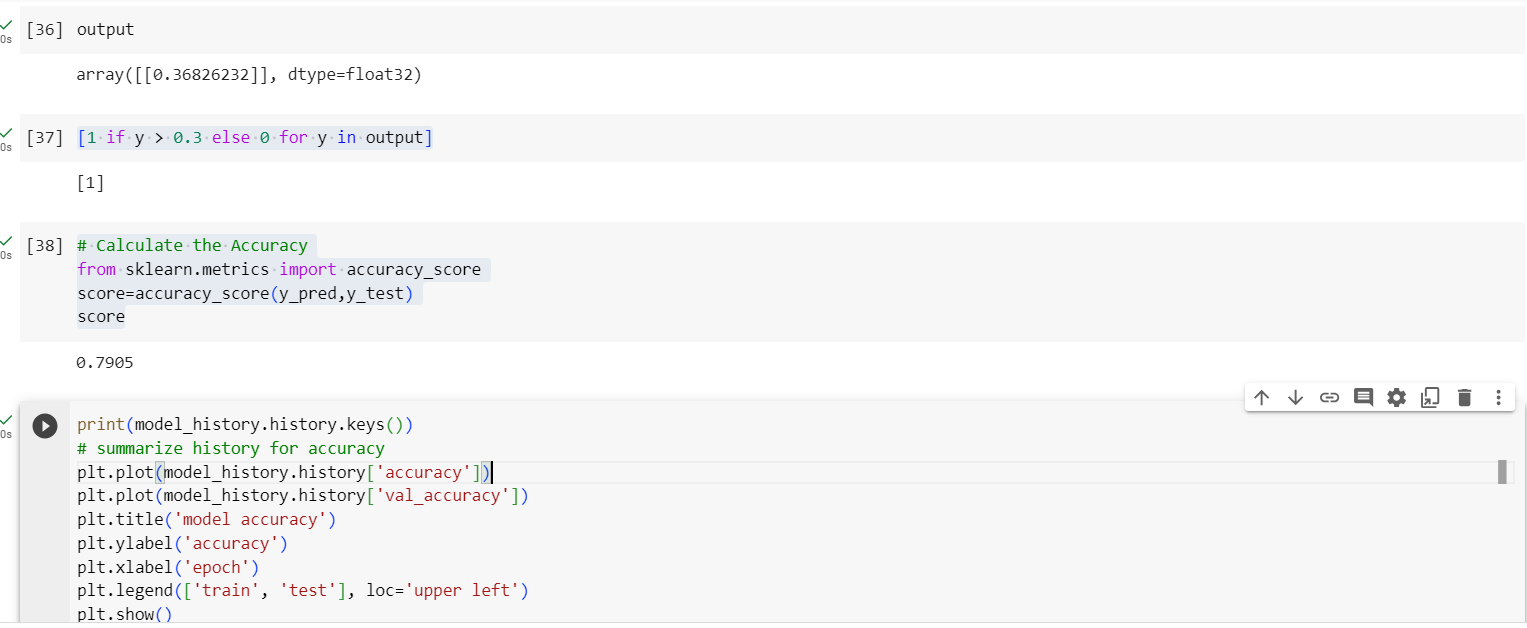
plt.title('model accuracy')

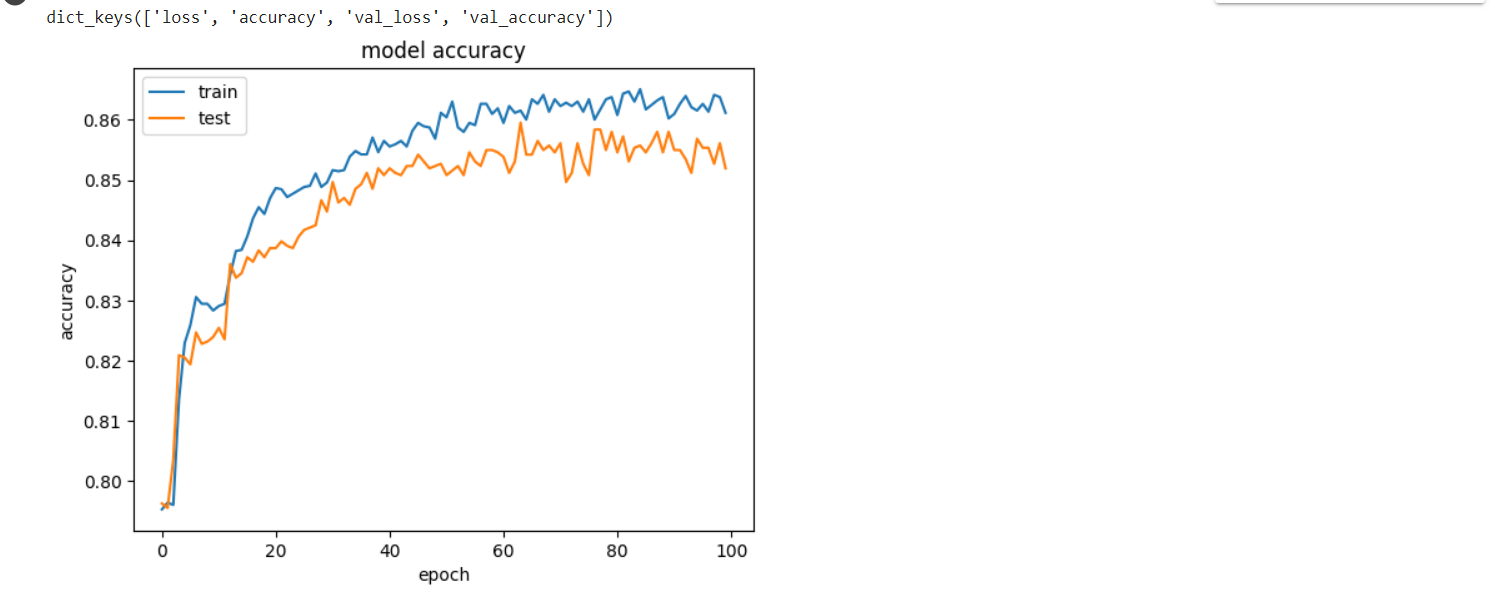
plt.ylabel('accuracy')

plt.xlabel('epoch')

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

plt.show()





# summarize history for loss

plt.plot(model\_history.history['loss'])

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

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

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

plt.show()

