محمد کاظم رجبی  
96222039

main idea:

the data set is realted to bank customers who they want to exit.

first divide the dataset to two category one of them is independent and the other is dependent:

dataset = pd.read\_csv('Proj1\_Data.csv')

X = dataset.iloc[:, 3:-1].values

y = dataset.iloc[:, -1].values

convert the gender column to number :

from sklearn.preprocessing import LabelEncoder

label = LabelEncoder()

X[:, 2] = label.fit\_transform(X[:, 2])

The last column is dependent and also we do not need the firs three columns and omit them.(rownumber,customerid,surname)

use labalencoder to to convert the categorical columns to numbers that can use in learning our neural networks:

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')

X = np.array(ct.fit\_transform(X))

then i divide the dataset to two category the first one is train dataset and the second category is test dataset and doing this dividion with 0.2 rate .

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)

Then with standardScaler convert the dataset like age to scale between 0 and 1 because the column related to cities has the same scale ,thus the our model learn better.

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

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

firs model has 1 hidden layer and 6 nodes and use sigmoid function .becuse we need binary out put i use 1 node in last layer and relu function.

ann1 = tf.keras.models.Sequential()

ann1.add(tf.keras.layers.Dense(units=6, activation='sigmoid'))

ann1.add(tf.keras.layers.Dense(units=6, activation='sigmoid'))

ann1.add(tf.keras.layers.Dense(units=1, activation='relu'))

from sklearn.metrics import confusion\_matrix, accuracy\_score

confusion\_matrix = confusion\_matrix(y\_test, y\_pred1)

print(confusion\_matrix)

accuracy\_score(y\_test, y\_pred1)

In addintion i use adam optimizer .also use validation set to prevent overfiting in 80 epoch

and i get 0.79% accuracy.



The second model has more nodes and out put a number between 0 and 1 ,if the number is greater than 0,5 it converts to 1 and if it fewer than 0.5 it converts to 0.

ann2 = tf.keras.models.Sequential()

ann2.add(tf.keras.layers.Dense(units=14, activation='relu'))

ann2.add(tf.keras.layers.Dense(units=14, activation='relu'))

ann2.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

ann2.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

history2 =ann2.fit(X\_train, y\_train,validation\_split=0.1, batch\_size = 32, epochs = 100)





The third model is like this

ann3 = tf.keras.models.Sequential()

ann3.add(tf.keras.layers.Dense(units=6, activation='relu'))

ann3.add(tf.keras.layers.Dropout(0.1))

ann3.add(tf.keras.layers.Dense(units=6, activation='relu'))

ann3.add(tf.keras.layers.Dense(units=6, activation='relu'))

ann3.add(tf.keras.layers.Dropout(0.1))

ann3.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

ann3.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

history3 =ann3.fit(X\_train, y\_train, batch\_size = 32, epochs = 120)



