**Given a bank customer build a neural network based classifer that can determine whether they will leave or not**

**pip install numpy**

**pip install pandas**

**pip install matplotlib**

**pip install seaborn**

**pip install sklearn**

**pip install keras**

**pip install tenserflow**

**Churn\_Modelling.csv is required**

**import pandas as pd**

**import numpy as np**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**data = pd.read\_csv('Churn\_Modelling.csv\Churn\_Modelling.csv')**

**data.head**

**data.shape**

**data.isnull().sum()**

**for col in data.columns:**

**if data[col].dtype != 'object':**

**bp = sns.boxplot(data = data, x = col)**

**plt.show()**

**sp = sns.scatterplot(data = data, x = col, y=data["Exited"])**

**plt.show()**

**bp = sns.boxplot(data = data, x = 'Age')**

**plt.show()**

**sp = sns.scatterplot(data = data, x = 'Age', y=data["Exited"])**

**plt.show()**

**Q1\_age = data['Age'].quantile(0.25)**

**Q3\_age = data['Age'].quantile(0.75)**

**IQR\_age = Q3\_age - Q1\_age**

**data.drop(data[data['Age']>83].index, axis = 0, inplace =True)**

**bp = sns.boxplot(data = data, x = 'Age')**

**plt.show()**

**sp = sns.scatterplot(data = data, x = 'Age', y=data["Exited"])**

**plt.show()**

**data.shape**

**bp = sns.boxplot(data = data, x = 'NumOfProducts')**

**plt.show()**

**sp = sns.scatterplot(data = data, x = 'NumOfProducts', y=data["Exited"])**

**plt.show()**

**data.drop(data[data['NumOfProducts']>=4].index, axis = 0, inplace =True)**

**bp = sns.boxplot(data = data, x = 'NumOfProducts')**

**plt.show()**

**sp = sns.scatterplot(data = data, x = 'NumOfProducts', y=data["Exited"])**

**plt.show()**

**data.shape**

**sns.countplot(x="Exited", data=data)**

**geography = pd.get\_dummies(data['Geography'], drop\_first=True)**

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

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

**data.info()**

**X = data.drop(['Exited', 'CustomerId', 'Surname', 'RowNumber', 'Geography', 'Gender'], axis=1)**

**Y = data['Exited']**

**X.shape**

**from imblearn.over\_sampling  import RandomOverSampler**

**smoteOver = RandomOverSampler(sampling\_strategy=1)**

**X, Y = smoteOver.fit\_resample(X,Y)**

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

**X\_Train, X\_Test, Y\_Train, Y\_Test = train\_test\_split(X, Y, test\_size= 0.20, random\_state=42)**

**from sklearn.preprocessing import StandardScaler**

**sc = StandardScaler()**

**X\_Train = sc.fit\_transform(X\_Train)**

**X\_Test = sc.transform(X\_Test)**

**import keras**

**from keras.models import Sequential**

**from keras.layers import Dense**

**X\_Train.shape**

**model = Sequential()**

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

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

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

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

**# Fitting the ANN to the Training set**

**model\_history = model.fit(X\_Train, Y\_Train, batch\_size=10, validation\_split=0.33, epochs=100)**

**# Summarize history for loss**

**plt.figure(figsize=(8,8))**

**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 right')**

**plt.show()**

**# Summarize history for accuracy**

**plt.figure(figsize=(8,8))**

**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='lower right')**

**plt.show()**

**Y\_Pred = model.predict(X\_Test)**

**Y\_Pred = (Y\_Pred > 0.5)**

**Y\_Pred**

**from sklearn.metrics import confusion\_matrix, accuracy\_score, classification\_report**

**cm = confusion\_matrix(Y\_Test, Y\_Pred)**

**def cmatrix\_fun(model\_name, actual, predicted):**

**# check the confusion matrix**

**cm = confusion\_matrix(actual, predicted)**

**print(cm)**

**# Plot the CM**

**ax = sns.heatmap(cm/np.sum(cm), annot=True, fmt='.2%', cmap='Blues')**

**ax.set\_title(f'The confusion matrix using  Classifier \n\n');**

**ax.set\_xlabel('\nPredicted Values')**

**ax.set\_ylabel('Actual Values ');**

**# Ticket labels - List must be in alphabetical order**

**ax.xaxis.set\_ticklabels(['False','True'])**

**ax.yaxis.set\_ticklabels(['False','True'])**

**cmatrix\_fun('ANN model', Y\_Test, Y\_Pred)**

**print(classification\_report(Y\_Test, Y\_Pred))**