## Subject: Data Mining and Business Intelligence (2170715)

**Open Ended Problem**

**Title : Churn Modeling Using Random Forest (classiﬁcation problem) Group No: 1**

**Enrollment No. Name**

**170420107015 Kartik Gondaliya**

**170420107051 Sahil Shingala**

**180423107001 Bhumi Chavada**

**180423107002 Dhwani Desai**

**180423107005 Dhrumi Kansara**

**Importing the libraries**

In [13]:

**import numpy as np**

**import pandas as pd**

# Data Preprocessing

## Importing the dataset

In [4]:

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

X = dataset.iloc[:, 3:-1].values y = dataset.iloc[:, -1].values print(X)

print(y)

[[619 'France' 'Female' ... 1 1 101348.88]

[608 'Spain' 'Female' ... 0 1 112542.58]

[502 'France' 'Female' ... 1 0 113931.57]

...

[709 'France' 'Female' ... 0 1 42085.58]

[772 'Germany' 'Male' ... 1 0 92888.52]

[792 'France' 'Female' ... 1 0 38190.78]]

[1 0 1 ... 1 1 0]

# Encoding categorical data

## Label Encoding the "Gender" column

In [5]:

**from sklearn.preprocessing import** LabelEncoder le = LabelEncoder()

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

[[619 'France' 0 ... 1 1 101348.88]

[608 'Spain' 0 ... 0 1 112542.58]

[502 'France' 0 ... 1 0 113931.57]

...

[709 'France' 0 ... 0 1 42085.58]

[772 'Germany' 1 ... 1 0 92888.52]

[792 'France' 0 ... 1 0 38190.78]]

## One Hot Encoding the "Geography" column

In [6]:

**from sklearn.compose import** ColumnTransformer

**from sklearn.preprocessing import** OneHotEncoder

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

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [[1.0 | 0.0 | 0.0 | ... | 1 | 1 | 101348.88] |
| [0.0 | 0.0 | 1.0 | ... | 0 | 1 | 112542.58] |
| [1.0 | 0.0 | 0.0 | ... | 1 | 0 | 113931.57] |
| ... |  |  |  |  |  |  |
| [1.0 | 0.0 | 0.0 | ... | 0 | 1 | 42085.58] |
| [0.0 | 1.0 | 0.0 | ... | 1 | 0 | 92888.52] |
| [1.0 | 0.0 | 0.0 | ... | 1 | 0 | 38190.78]] |

# Feature Scaling

In [7]:

**from sklearn.preprocessing import** StandardScaler sc = StandardScaler()

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [[ 0.99720391 -0.57873591 | -0.57380915 | ... | 0.64609167 | 0.97024255 |
| 0.02188649] |  |  |  |  |
| [-1.00280393 -0.57873591 | 1.74273971 | ... | -1.54776799 | 0.97024255 |
| 0.21653375] |  |  |  |  |
| [ 0.99720391 -0.57873591 | -0.57380915 | ... | 0.64609167 | -1.03067011 |
| 0.2406869 ] |  |  |  |  |
| ... |  |  |  |  |
| [ 0.99720391 -0.57873591 | -0.57380915 | ... | -1.54776799 | 0.97024255 |
| -1.00864308] |  |  |  |  |
| [-1.00280393 1.72790383 | -0.57380915 | ... | 0.64609167 | -1.03067011 |
| -0.12523071] |  |  |  |  |
| [ 0.99720391 -0.57873591 | -0.57380915 | ... | 0.64609167 | -1.03067011 |
| -1.07636976]] |  |  |  |  |

# Splitting the dataset into the Training set and Test set

In [8]:

**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, rando m\_state = 0)

# Training the Random Forest Classiﬁcation model on the Training set

In [9]:

**from sklearn.ensemble import** RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators = 1000, criterion = 'entropy', random\_state = 0)

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

Out[9]:

RandomForestClassifier(criterion='entropy', n\_estimators=1000, rando m\_state=0)

# Prediction

In [10]:

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

# Making the Confusion Matrix

In [11]:

**from sklearn.metrics import** confusion\_matrix cm = confusion\_matrix(y\_test, y\_pred) print(cm)

[[1525 70]

[ 199 206]]

So we got accuracy = (TP+TN)/TOTAL

=(1525+206)/2000

=**86.55%**