```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
sns.set()

dataset = pd.read_csv('/content/Churn_Modelling.csv', index_col =
'RowNumber')
dataset.head()
```

```
           CustomerId   Surname  CreditScore Geography  Gender  Age
Tenure  \
RowNumber

1            15634602  Hargrave          619    France  Female   42
2
2            15647311      Hill          608     Spain  Female   41
1
3            15619304      Onio          502    France  Female   42
8
4            15701354      Boni          699    France  Female   39
1
5            15737888  Mitchell          850     Spain  Female   43
2

             Balance  NumOfProducts  HasCrCard  IsActiveMember  \
RowNumber
1               0.00              1          1               1
2           83807.86              1          0               1
3          159660.80              3          1               0
4               0.00              2          0               0
5          125510.82              1          1               1

           EstimatedSalary  Exited
RowNumber
1                101348.88       1
2                112542.58       0
3                113931.57       1
4                 93826.63       0
5                 79084.10       0
```

```
#Customer ID and Surname would not be relevant as features
X_columns = dataset.columns.tolist()[2:12]
Y_columns = dataset.columns.tolist()[-1:]
print(X_columns)
print(Y_columns)
```

```
['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance',
'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary']
['Exited']
```

```
X = dataset[X_columns].values
Y = dataset[Y_columns].values

#We need to encode categorical variables such as geography and gender
from sklearn.preprocessing import LabelEncoder
X_column_transformer = LabelEncoder()
X[:, 1] = X_column_transformer.fit_transform(X[:, 1])

#Lets Encode gender now
X[:, 2] = X_column_transformer.fit_transform(X[:, 2])
```

We are treating countries with ordinal values(0 < 1 < 2) but they are incomparable. To solve this we can use one hot encoding. We will perform some standardization

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

pipeline = Pipeline(
    [
        ('Categorizer', ColumnTransformer(
            [
                ("Gender Label Encoder", OneHotEncoder(categories =
'auto', drop = 'first'), [2]),
                ("Geography Label Encoder", OneHotEncoder(categories =
'auto', drop = 'first'), [1])
            ],
            remainder = 'passthrough', n_jobs = 1)),
        ('Normalizer', StandardScaler())
    ]
)

#Standardize the features
X = pipeline.fit_transform(X)

#Spilt the data
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)

#Let us create the Neural Network
from keras.models import Sequential
from keras.layers import Dense, Dropout

#Initialize ANN
classifier = Sequential()

#Add input layer and hidden layer
classifier.add(Dense(6, activation = 'relu', input_shape =
(X_train.shape[1], )))
classifier.add(Dropout(rate = 0.1))
```

```
#Add second layer
classifier.add(Dense(6, activation = 'relu'))
classifier.add(Dropout(rate = 0.1))

#Add output layer
classifier.add(Dense(1, activation = 'sigmoid'))

#Let us take a look at our network
classifier.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 6)                 72

 dropout (Dropout)           (None, 6)                 0

 dense_1 (Dense)             (None, 6)                 42

 dropout_1 (Dropout)         (None, 6)                 0

 dense_2 (Dense)             (None, 1)                 7

=================================================================
Total params: 121
Trainable params: 121
Non-trainable params: 0
_________________________________________________________________

#Optimize the weights
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy',
metrics = ['accuracy'])

#Fitting the Neural Network
history = classifier.fit(X_train, y_train, batch_size = 32, epochs =
200, validation_split = 0.1, verbose = 2)

y_pred = classifier.predict(X_test)
print(y_pred[:5])

63/63 [==============================] - 0s 1ms/step
[[0.21353428]
 [0.3550975 ]
 [0.1884149 ]
 [0.04963601]
 [0.2057534 ]]

#Let us use confusion matrix with cutoff value as 0.5
y_pred = (y_pred > 0.5).astype(int)
print(y_pred[:5])
```

```
[[0]
 [0]
 [0]
 [0]
 [0]]
```

```python
#Making the Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[1569   26]
 [ 293  112]]
```

```python
#Accuracy of our NN
print(((cm[0][0] + cm[1][1])* 100) / len(y_test), '% of data was 
classified correctly')
```

```
84.05 % of data was classified correctly
```