Build Model But we Handeled Data To Be Balanced Using OverSampling

In [46]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics         import classification_report
from sklearn.linear_model    import LogisticRegression
from sklearn.neighbors       import KNeighborsClassifier

# to ignore Errors
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

In [33]:
```
# Read Data into Dataframe

df = pd.read_csv(r'C:\Users\dell\Desktop\powerbi projects\datasets\telecom\telecom_data.csv')
```

In [5]:
```
# split the data into Features and Target

X = df.drop(['customerID','Churn'],axis=1)
y = df['Churn']
```

In [6]:
```
# Show The Target Distribution Before OverSampling

y.value_counts()
```

Out[6]:
```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

- Apply over_sampling:-

In [17]:
```
# over_sampling
from imblearn.over_sampling import RandomOverSampler

ros = RandomOverSampler()
X,y = ros.fit_resample(X,y)
```

In [18]:
```
# Show The Target Distribution After OverSampling

y.value_counts()
```

Out[18]:
```
No     5174
Yes    5174
Name: Churn, dtype: int64
```

`Now Data Is Balanced`

In [25]:
```
# ordinal encoding

from sklearn.preprocessing import OrdinalEncoder

oe = OrdinalEncoder()
X = oe.fit_transform(X)
```

In [26]:
```
# train test split

X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=10)
```

- Build Model:-

In [15]:
```
# RandomForestClassifier Model

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)

y_pred = rfc.predict(X_test)
```

In [45]:
```
# print The Evalution Metrics

print("RandomForestClassifier Results")
print("-------------------------------------------------------")
print(classification_report(y_test,y_pred))
print("-------------------------------------------------------")
```

```
RandomForestClassifier Results
-------------------------------------------------------
              precision    recall  f1-score   support

          No       0.95      0.84      0.89      1318
         Yes       0.85      0.95      0.90      1269

    accuracy                           0.90      2587
   macro avg       0.90      0.90      0.90      2587
weighted avg       0.90      0.90      0.90      2587

-------------------------------------------------------
```

`The Model precision = 95% with NO And 85% With Yes`

`The Model Recall = 85% with NO And 95% With Yes`

> The Accuracy is 90%

`Done`