```
# import necessary  libraries
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_sc
```

```
#import dataset
data = pd.read_csv(r"/content/Churn_Modelling[1].csv")
data
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

```
data.isnull().any()
```

```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

```
data.isnull().sum()
```

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
```

```
EstimatedSalary    0
Exited             0
dtype: int64
```

```
le = LabelEncoder()
data["Gender"] = le.fit_transform(data["Gender"])
data["CustomerId"] = le.fit_transform(data["CustomerId"])
data["Surname"] = le.fit_transform(data["Surname"])
data["Age"] = le.fit_transform(data["Age"])
data["Balance"] = le.fit_transform(data["Balance"])
data["CreditScore"] = le.fit_transform(data["CreditScore"])
data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balar
0	1	2736	1115	228	France	0	24	2	
1	2	3258	1177	217	Spain	0	23	1	7
2	3	2104	2040	111	France	0	24	8	57
3	4	5435	289	308	France	0	21	1	
4	5	6899	1822	459	Spain	0	25	2	36

```
x= data.iloc[:,0:19].values
y= data.iloc[:,19:20].values
```

```
x
```

```
array([[1, 2736, 1115, ..., 1, 101348.88, 1],
       [2, 3258, 1177, ..., 1, 112542.58, 0],
       [3, 2104, 2040, ..., 0, 113931.57, 1],
       ...,
       [9998, 717, 1570, ..., 1, 42085.58, 1],
       [9999, 4656, 2345, ..., 0, 92888.52, 1],
       [10000, 2497, 2751, ..., 0, 38190.78, 0]], dtype=object)
```

```
y
```

```
array([], shape=(10000, 0), dtype=float64)
```

```
one = OneHotEncoder()
a= one.fit_transform(x[:,6:7]).toarray()
b= one.fit_transform(x[:,7:8]).toarray()
c= one.fit_transform(x[:,8:9]).toarray()
d= one.fit_transform(x[:,9:10]).toarray()
e= one.fit_transform(x[:,10:11]).toarray()
x=np.delete(x,[6,7,8,9,10],axis=1)
```

```
x=np.concatenate((a,b,c,d,e,x),axis=1)
```

```
from imblearn.over_sampling import SMOTE
```

```
smt = SMOTE()
x
```

```
array([[1, 2736, 1115, ..., 1, 101348.88, 1],
       [2, 3258, 1177, ..., 1, 112542.58, 0],
       [3, 2104, 2040, ..., 0, 113931.57, 1],
       ...,
       [9998, 717, 1570, ..., 1, 42085.58, 1],
       [9999, 4656, 2345, ..., 0, 92888.52, 1],
       [10000, 2497, 2751, ..., 0, 38190.78, 0]], dtype=object)
```

```
y
```

```
array([], shape=(10000, 0), dtype=float64)
```

```
x.shape
```

```
(10000, 14)
```

```
y.shape
```

```
(10000, 0)
```

Colab paid products - Cancel contracts here

✓ 0s completed at 6:32 AM