

```
#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_score

#import dataset
data = pd.read_csv(r"Churn_Modelling.csv")
data
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Ba
1	2	15647311	Hargrave	619	France	Female	42	2
2	3	15619304	Hill	608	Spain	Female	41	1
3	4	15701354	Onio	502	France	Female	42	8
4	5	15737888	Boni	699	France	Female	39	1
...	Mitchell	850	Spain	Female	43	2
9995	9996	15606229	Obijaku	771	France	Male	39	5
9996	9997	15569892	Johnstone	516	France	Male	35	10
9997	9998	15584532	Liu	709	France	Female	36	7
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3
9999	10000	15628319	Walker	792	France	Female	28	4

10000 rows × 14 columns

```
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
```

```
#checking for null values
data.CustomerId=pd.to_numeric(data.CustomerId, errors='coerce')
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["CustomerId"].fillna(data["CustomerId"].median(), inplace=True)
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
```

Creating a copy...

```
Exited        0
dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data["RowNumber"] = le.fit_transform(data["RowNumber"])
data["CustomerId"] = le.fit_transform(data["CustomerId"])
data["Surname"] = le.fit_transform(data["Surname"])
data["CreditScore"] = le.fit_transform(data["CreditScore"])
data["Geography"] = le.fit_transform(data["Geography"])
data["Gender"] = le.fit_transform(data["Gender"])
data["Age"] = le.fit_transform(data["Age"])
data["Tenure"] = le.fit_transform(data["Tenure"])
data["Balance"] = le.fit_transform(data["Balance"])
data["NumOfProducts"] = le.fit_transform(data["NumOfProducts"])
data["HasCrCard"] = le.fit_transform(data["HasCrCard"])
data["IsActiveMember"] = le.fit_transform(data["IsActiveMember"])
data["EstimatedSalary"] = le.fit_transform(data["EstimatedSalary"])
data["Exited"] = le.fit_transform(data["Exited"])
```

```
data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	0	2736	1115	228	0	0	24	2	0
1	1	3258	1177	217	2	0	23	1	743
2	2	2104	2040	111	0	0	24	8	5793
3	3	5435	289	308	0	0	21	1	0
4	4	6899	1822	459	2	0	25	2	3696

```
x=data.iloc[:,0:14].values
y=data.iloc[:,13:14].values
```

```
x
```

```
array([[ 0, 2736, 1115, ..., 1, 5068, 1],
   [ 1, 3258, 1177, ..., 1, 5639, 0],
   [ 2, 2104, 2040, ..., 0, 5707, 1],
   ...,
   [9997, 717, 1570, ..., 1, 2062, 1],
```

```
[9998, 4656, 2345, ...], 0, 4639, 1],
[9999, 2497, 2751, ...], 0, 1878, 0]])
```

y

```
array([[1],
       [0],
       [1],
       ...,
       [1],
       [1],
       [0]])
```

```
from sklearn.preprocessing import OneHotEncoder
one = OneHotEncoder()
a= one.fit_transform(x[:,0:1]).toarray()
b= one.fit_transform(x[:,1:2]).toarray()
c= one.fit_transform(x[:,2:3]).toarray()
d= one.fit_transform(x[:,3:4]).toarray()
e= one.fit_transform(x[:,4:5]).toarray()
f= one.fit_transform(x[:,5:6]).toarray()
g= one.fit_transform(x[:, 6:7]).toarray()
h= one.fit_transform(x[:,7:8]).toarray()
i= one.fit_transform(x[:,8:9]).toarray()
j= one.fit_transform(x[:,9:10]).toarray()
x=np.delete(x, [0,1,2,3,4,5,6,7,8,9], axis=1)
x=np.concatenate((a,b,c,d,e,f,g,h,i,j,x), axis=1)
```

Creating a copy...

MOTE

```
smt = SMOTE ()
x_resample,y_resample=smt.fit_resample (x,y)
```

x_resample

```
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
       1.00000000e+00, 5.06800000e+03, 1.00000000e+00],
       [0.00000000e+00, 1.00000000e+00, 0.00000000e+00, ...,
       1.00000000e+00, 5.63900000e+03, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00, ...,
       0.00000000e+00, 5.70700000e+03, 1.00000000e+00],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
       1.00000000e+00, 5.46979923e+03, 1.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
       8.95483113e-01, 7.88685031e+03, 1.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
       0.00000000e+00, 4.57916667e+03, 1.00000000e+00]])
```

y_resample

```
array([1, 0, 1, ..., 1, 1, 1])
```

data.describe()

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	
count	10000.00000	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	1
mean	4999.50000	4999.50000	1507.774200	259.584600	0.746300	0.545700	
std	2886.89568	2886.89568	846.204311	96.496107	0.827529	0.497932	
min	0.00000	0.00000	0.000000	0.000000	0.000000	0.000000	
25%	2499.75000	2499.75000	773.750000	193.000000	0.000000	0.000000	
50%	4999.50000	4999.50000	1542.000000	261.000000	0.000000	1.000000	
75%	7499.25000	7499.25000	2238.250000	327.000000	1.000000	1.000000	
max	9999.00000	9999.00000	2931.000000	459.000000	2.000000	1.000000	

```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.distplot(data["Balance"])
plt.subplot(1,2,2)
sns.distplot(data["Tenure"])
```

<ipython-input-41-40ad5a6b3819>:3: UserWarning:
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

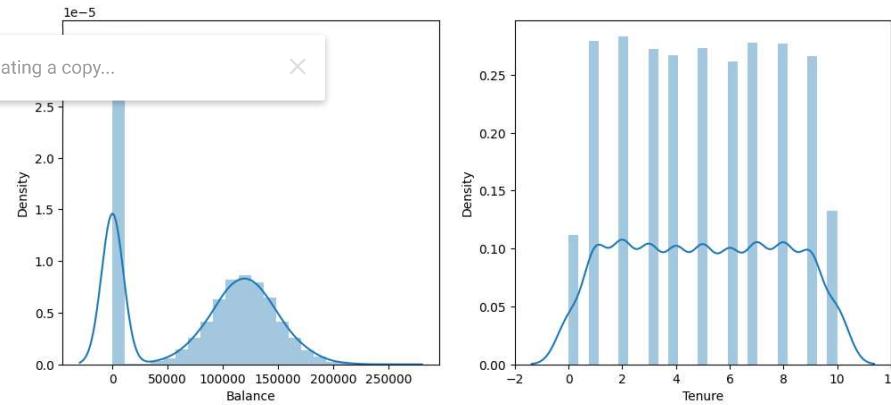
```
sns.distplot(data["Balance"])
<ipython-input-41-40ad5a6b3819>:5: UserWarning:  

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

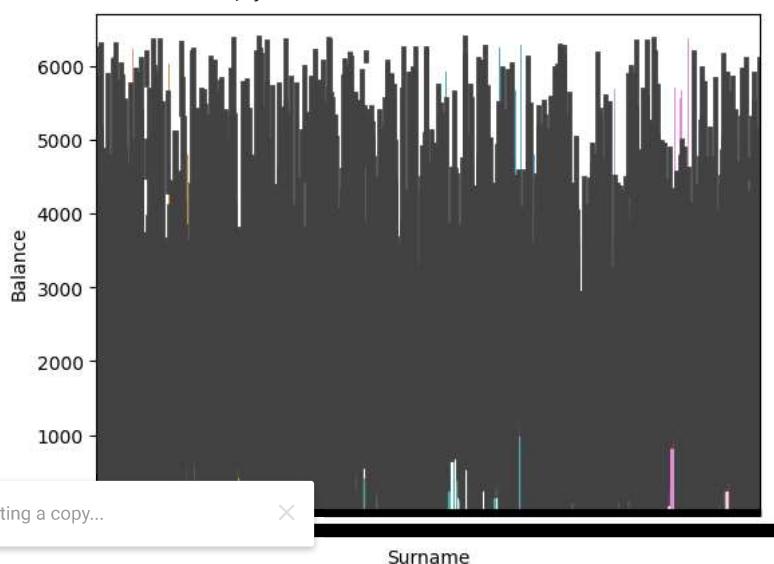
For a guide to updating your code to use the new functions, please see
<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(data["Tenure"])
<Axes: xlabel='Tenure', ylabel='Density'>
```

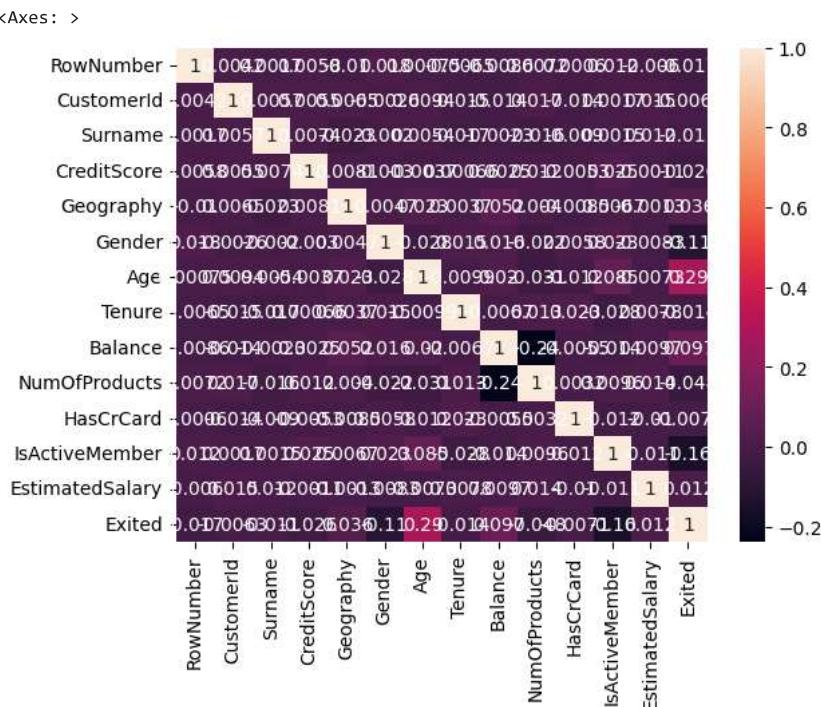


```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.countplot(data["Tenure"])
plt.subplot(1,2,2)
sns.countplot(data["Balance"])
```

A bar chart titled "Surname" on the x-axis and "Balance" on the y-axis. The y-axis has a label "ylabel='count'" and a value of 10000. The bars are blue, representing the count of each surname. The chart shows a single bar reaching the 10000 mark.



```
sns.heatmap(data.corr(), annot = True)
```



```
sns.pairplot(data=data, markers= ["^", "v"], palette= "inferno")
```

Creating a copy...

