

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

🔗

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	
	0	1	15634602	Hargrave	619	France	Female	42	2
	1	2	15647311	Hill	608	Spain	Female	41	1
	2	3	15619304	Onio	502	France	Female	42	8 1
	3	4	15701354	Boni	699	France	Female	39	1
	4	5	15737888	Mitchell	850	Spain	Female	43	2 1

	9995	9996	15606229	Obijiaku	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 1

10000 rows × 14 columns

```
data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	1	15634602	Hargrave	619	France	Female	42	2	0.00
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86
2	3	15619304	Onio	502	France	Female	42	8	159660.80
3	4	15701354	Boni	699	France	Female	39	1	0.00
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82

```
import warnings
warnings.filterwarnings("ignore")
from sklearn import metrics
from sklearn.metrics import accuracy_score
```

```
# importing .csv files using Pandas
train = pd.read_csv('/content/Churn_Modelling[1].csv')
test = pd.read_csv('/content/Churn_Modelling[1].csv')
```

```
train['Balance'] = train['Balance'].apply(lambda x: 1 if x == 'male' else 2)
```

```
train.drop(columns=['RowNumber', 'CustomerId', 'Surname', 'Gender', 'Age'], inplace=True)
```

```
X = train.drop(["Balance"], axis=1)
y = train.Balance
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=21)
```

```
#importing and building the Decision tree model
def logreg(x_train,x_test,y_train,y_test):
    Ir = LogisticRegression(random_state=0)
    Ir.fit(x_train,y_train)
    y_Ir_tr = Ir.predict(x_train)
    print(accuracy_score(y_Ir_tr,y_train))
    ypred_Ir = Ir.predict(x_test)
    print(accuracy_score(ypred_tr,y_train))
    print("****LogisticRegression****")
    print("Confusion_Matrix")
    print(Confusion_Matrix(y_test,ypred_Ir))
    print("Classification Report")
    print(classification_report(y_test,ypred_Ir))
```

```
!pip install matplotlib-venn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: matplotlib-venn in /usr/local/lib/python3.9/dist-packages (0.11.9)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from matplotlib-venn) (1.22.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages (from matplotlib-venn) (3.7.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from matplotlib-venn) (1.10.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->matplotlib-venn) (1.0.7)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/dist-packages (from matplotlib->matplotlib-venn) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->matplotlib-venn) (5.12.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->matplotlib-venn) (4.38.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->matplotlib-venn) (1.4.5)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->matplotlib-venn) (3.0.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->matplotlib-venn) (23.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (from matplotlib->matplotlib-venn) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->matplotlib-venn) (8.4.0)
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (from importlib-resources->=3.2.0->matplotlib-venn) (3.15.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil->=2.7->matplotlib-venn) (1.16.0)
```

```
!apt-get -qq install -y libfluidsynth1
```

```
E: Package 'libfluidsynth1' has no installation candidate
```

```
# https://pypi.python.org/pypi/libarchive
```

```
!apt-get -qq install -y libarchive-dev && pip install -U libarchive
import libarchive
```

```
Selecting previously unselected package libarchive-dev:amd64.
(Reading database ... 122349 files and directories currently installed.)
Preparing to unpack .../libarchive-dev_3.4.0-2ubuntu1.2_amd64.deb ...
Unpacking libarchive-dev:amd64 (3.4.0-2ubuntu1.2) ...
Setting up libarchive-dev:amd64 (3.4.0-2ubuntu1.2) ...
Processing triggers for man-db (2.9.1-1) ...
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting libarchive
  Downloading libarchive-0.4.7.tar.gz (23 kB)
  Preparing metadata (setup.py) ... done
Collecting nose
  Downloading nose-1.3.7-py3-none-any.whl (154 kB)
    |#####| 154.7/154.7 kB 5.3 MB/s eta 0:00:00
Building wheels for collected packages: libarchive
  Building wheel for libarchive (setup.py) ... done
  Created wheel for libarchive: filename=libarchive-0.4.7-py3-none-any.whl size=31644 sha256=c4b4a7dbf9b52924689d8407a2aa36e903d1f
  Stored in directory: /root/.cache/pip/wheels/c9/a5/cc/cb20f1314d4cdce001fd72baa1efe93e1542a81bdea2fc639
Successfully built libarchive
Installing collected packages: nose, libarchive
Successfully installed libarchive-0.4.7 nose-1.3.7
```

```
pip install lazypredict
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: lazypredict in /usr/local/lib/python3.9/dist-packages (0.2.12)
```

Requirement already satisfied: lightgbm in /usr/local/lib/python3.9/dist-packages (from lazypredict) (3.3.5)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from lazypredict) (4.65.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-packages (from lazypredict) (1.2.2)
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from lazypredict) (8.1.3)
Requirement already satisfied: xgboost in /usr/local/lib/python3.9/dist-packages (from lazypredict) (1.7.5)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from lazypredict) (1.5.3)
Requirement already satisfied: joblib in /usr/local/lib/python3.9/dist-packages (from lazypredict) (1.2.0)
Requirement already satisfied: wheel in /usr/local/lib/python3.9/dist-packages (from lightgbm->lazypredict) (0.40.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from lightgbm->lazypredict) (1.10.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from lightgbm->lazypredict) (1.22.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn->lazypredict) (3.1.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas->lazypredict) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas->lazypredict) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas->lazypredict) (1.16.0)

```
### importing lazypredict library
import lazypredict
### importing LazyClassifier for classification problem
from lazypredict.Supervised import LazyClassifier
### importing LazyClassifier for classification problem because here we are solving Classification use case.
from lazypredict.Supervised import LazyClassifier
### importing breast Cancer Dataset from sklearn
from sklearn.datasets import load_breast_cancer
### splitting dataset into training and testing part
from sklearn.model_selection import train_test_split
```

```
import lazypredict
from lazypredict.Supervised import LazyClassifier
```

```
clf = LazyClassifier(verbose=0,ignore_warnings=True)
models, predictions = clf.fit(X_train, X_test, y_train, y_test)
models
```

100%|██████████| 29/29 [00:11<00:00, 2.45it/s]

	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
Model					
AdaBoostClassifier	1.00	1.00	None	1.00	0.10
BaggingClassifier	1.00	1.00	None	1.00	0.13
BernoulliNB	1.00	1.00	None	1.00	0.08
DecisionTreeClassifier	1.00	1.00	None	1.00	0.05
DummyClassifier	1.00	1.00	None	1.00	0.05
ExtraTreeClassifier	1.00	1.00	None	1.00	0.05
ExtraTreesClassifier	1.00	1.00	None	1.00	0.43
GaussianNB	1.00	1.00	None	1.00	0.07
KNeighborsClassifier	1.00	1.00	None	1.00	0.39
LabelPropagation	1.00	1.00	None	1.00	3.17
LabelSpreading	1.00	1.00	None	1.00	4.76
LinearDiscriminantAnalysis	1.00	1.00	None	1.00	0.31
RandomForestClassifier	1.00	1.00	None	1.00	0.82
RidgeClassifier	1.00	1.00	None	1.00	0.10
RidgeClassifierCV	1.00	1.00	None	1.00	0.11
LGBMClassifier	1.00	1.00	None	1.00	0.23

```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.models import load_model
```

```
#empty model
classifier = Sequential()
```

```
from tensorflow.keras.models import Model
import numpy as np
import pandas as pd
import tensorflow
```

```
import keras
import tensorflow.keras
```

```
veri=pd.read_csv("/content/Churn_Modelling[1].csv")
```

```
data=veri.copy()
```

```
data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bal
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	8380
2	3	15619304	Onio	502	France	Female	42	8	15960
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	12557

```
len(data.columns)
```

14

```
data.columns
```

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
      'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
      'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

```
data.isnull()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...
9995	False	False	False	False	False	False	False	False	False
9996	False	False	False	False	False	False	False	False	False
9997	False	False	False	False	False	False	False	False	False
9998	False	False	False	False	False	False	False	False	False
9999	False	False	False	False	False	False	False	False	False

10000 rows × 10 columns

```
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	

```
x=data.iloc[:,3:-1].values
y=data.Exited.values
```

x

```
array([[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]], dtype=object)
```

y

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

```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
x[:,2]=le.fit_transform(x[:,2])
```

x

```
array([[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]], dtype=object)
```

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer(transformers=[("encoder",OneHotEncoder(),[1])],remainder="passthrough")
x=np.array(ct.fit_transform(x))
```

x

```
array([[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]], dtype=object)
```

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=43)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
xtrain1=sc.fit_transform(xtrain)
xtest1=sc.transform(xtest)
```

```
ann=tensorflow.keras.models.Sequential()
```

```
ann.add(tensorflow.keras.layers.Dense(units=6,activation="relu"))
ann.add(tensorflow.keras.layers.Dense(units=6,activation="relu"))
ann.add(tensorflow.keras.layers.Dense(units=1,activation="sigmoid"))
```

```
ann.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"])
```

```
ann.fit(xtrain1,ytrain,epochs=100)
```

```
Epoch 1/100
250/250 [=====] - 1s 2ms/step - loss: 0.3278 - accuracy: 0.8640
Epoch 2/100
250/250 [=====] - 0s 2ms/step - loss: 0.3278 - accuracy: 0.8656
Epoch 3/100
250/250 [=====] - 1s 3ms/step - loss: 0.3277 - accuracy: 0.8654
Epoch 4/100
250/250 [=====] - 1s 3ms/step - loss: 0.3278 - accuracy: 0.8645
Epoch 5/100
250/250 [=====] - 1s 3ms/step - loss: 0.3278 - accuracy: 0.8641
Epoch 6/100
250/250 [=====] - 1s 3ms/step - loss: 0.3281 - accuracy: 0.8649
Epoch 7/100
250/250 [=====] - 1s 3ms/step - loss: 0.3275 - accuracy: 0.8649
```

```

Epoch 8/100
250/250 [=====] - 0s 2ms/step - loss: 0.3277 - accuracy: 0.8648
Epoch 9/100
250/250 [=====] - 0s 2ms/step - loss: 0.3274 - accuracy: 0.8649
Epoch 10/100
250/250 [=====] - 0s 2ms/step - loss: 0.3277 - accuracy: 0.8646
Epoch 11/100
250/250 [=====] - 0s 2ms/step - loss: 0.3279 - accuracy: 0.8646
Epoch 12/100
250/250 [=====] - 0s 2ms/step - loss: 0.3276 - accuracy: 0.8636
Epoch 13/100
250/250 [=====] - 0s 2ms/step - loss: 0.3276 - accuracy: 0.8649
Epoch 14/100
250/250 [=====] - 0s 2ms/step - loss: 0.3277 - accuracy: 0.8659
Epoch 15/100
250/250 [=====] - 0s 2ms/step - loss: 0.3273 - accuracy: 0.8658
Epoch 16/100
250/250 [=====] - 0s 2ms/step - loss: 0.3276 - accuracy: 0.8659
Epoch 17/100
250/250 [=====] - 0s 2ms/step - loss: 0.3273 - accuracy: 0.8652
Epoch 18/100
250/250 [=====] - 0s 2ms/step - loss: 0.3273 - accuracy: 0.8639
Epoch 19/100
250/250 [=====] - 0s 2ms/step - loss: 0.3275 - accuracy: 0.8650
Epoch 20/100
250/250 [=====] - 0s 2ms/step - loss: 0.3275 - accuracy: 0.8648
Epoch 21/100
250/250 [=====] - 0s 2ms/step - loss: 0.3278 - accuracy: 0.8654
Epoch 22/100
250/250 [=====] - 0s 2ms/step - loss: 0.3275 - accuracy: 0.8655
Epoch 23/100
250/250 [=====] - 0s 2ms/step - loss: 0.3278 - accuracy: 0.8643
Epoch 24/100
250/250 [=====] - 0s 2ms/step - loss: 0.3272 - accuracy: 0.8633
Epoch 25/100
250/250 [=====] - 0s 2ms/step - loss: 0.3271 - accuracy: 0.8661
Epoch 26/100
250/250 [=====] - 0s 2ms/step - loss: 0.3272 - accuracy: 0.8655
Epoch 27/100
250/250 [=====] - 0s 2ms/step - loss: 0.3272 - accuracy: 0.8651
Epoch 28/100
250/250 [=====] - 1s 2ms/step - loss: 0.3268 - accuracy: 0.8652
Epoch 29/100
250/250 [=====] - 1s 3ms/step - loss: 0.3271 - accuracy: 0.8669

```

```

ypred=ann.predict(xtest1)
ypred=(ypred>0.5)

```

```

from sklearn.metrics import accuracy_score
accuracy_score(ytest,ypred)

```

```
#Let's Import the Packages...
```

```

%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

```

```

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression

```

```

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import warnings
warnings.filterwarnings('ignore')

```

```

from sklearn.linear_model import LogisticRegression # for Logistic Regression Algorithm
from sklearn.model_selection import train_test_split # to split the dataset for training and testing
from sklearn import metrics # for checking the model accuracy

```

```

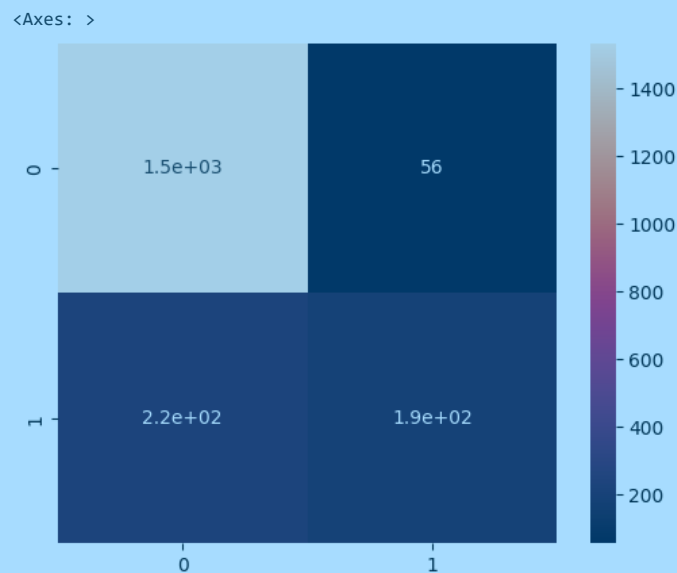
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

```

```
print(classification_report(ytest, ypred))
```

	precision	recall	f1-score	support
0	0.87	0.96	0.92	1588
1	0.77	0.46	0.57	412
accuracy			0.86	2000
macro avg	0.82	0.71	0.74	2000
weighted avg	0.85	0.86	0.85	2000

```
cm =confusion_matrix(ytest, ypred)
sns.heatmap(cm, square=True , annot=True)
```



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=5)
print(xtrain.shape)
print(ytrain.shape)
print(xtest.shape)
print(ytest.shape)
```

```
(8000, 12)
(8000,)
(2000, 12)
(2000,)
```

```
from sklearn.tree import DecisionTreeClassifier
classifier_Decicsion = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier_Decicsion.fit(xtrain, ytrain)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

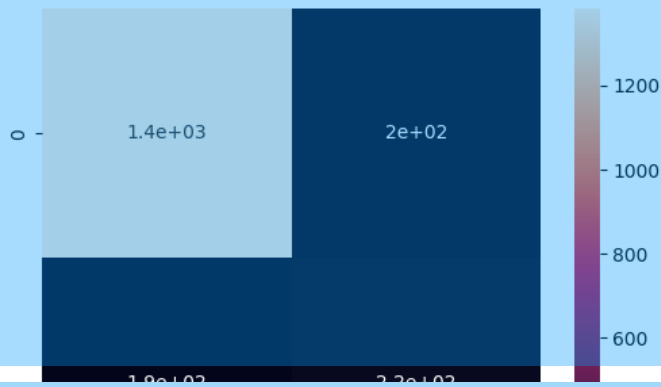
```
ypred_Decision = classifier_Decicsion.predict(xtest)
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(ytest, ypred_Decision)
print(cm)
accuracy_score(ytest,ypred_Decision)
```

```
[[1383 205]
 [ 194 218]]
0.8005
```

```
sns.heatmap(cm, square=True , annot=True)
```

<Axes: >



```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size =0.25, random_state = 4)
print(xtrain.shape)
print(ytrain.shape)
print(xtest.shape)
print(ytest.shape)
```

```
(8000, 12)
(8000,)
(2000, 12)
(2000,)
```

```
# Building Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(criterion = 'entropy', random_state = 42)
rfc.fit(xtrain, ytrain)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', random_state=42)
```

```
from sklearn.metrics import f1_score
rfc_pred_test = rfc.predict(xtest)
print('Testing Set Evaluation F1-Score=>',f1_score(ytest,rfc_pred_test))
```

```
Testing Set Evaluation F1-Score=> 0.5805471124620062
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=5)
print(xtrain.shape)
print(ytrain.shape)
print(xtest.shape)
print(ytest.shape)
```

```
(8000, 12)
(8000,)
(2000, 12)
(2000,)
```

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(xtrain, ytrain)
```

```
SVC
SVC(kernel='linear')
```

```
ypred_svm = svclassifier.predict(xtest)
```

```
ypred_svm
```

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

```
accuracy = accuracy_score(ytest, ypred_svm) * 100
print("Accuracy of the Logistic Regression Model: ",accuracy)
```

```
Accuracy of the Logistic Regression Model: 77.8
```

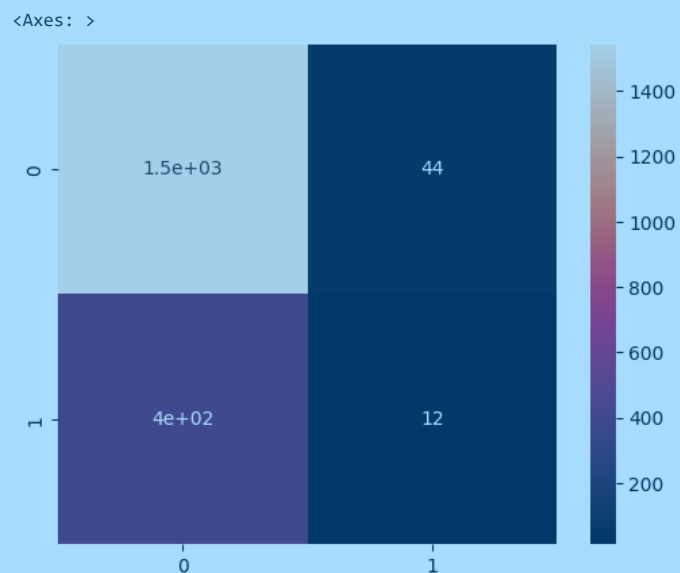
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(ytest,ypred_svm))
print(classification_report(ytest,ypred_svm))
```



```
[[1544  44]
 [ 400  12]]
```

	precision	recall	f1-score	support
0	0.79	0.97	0.87	1588
1	0.21	0.03	0.05	412
accuracy			0.78	2000
macro avg	0.50	0.50	0.46	2000
weighted avg	0.67	0.78	0.70	2000

```
cm =confusion_matrix(ytest,ypred_svm)
sns.heatmap(cm, square=True , annot=True)
```



```
model.save("telcom_churn")
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolutio
```