MACHINE LEARNING LAB

EXERCISE 6

Aim :

1. Use the attached file and run SVM, Decision tree, Random Forest and any one boosting algorithm.
2. Find out the different tunable parameters for each algorithms mentioned above.
3. Apply gridsearchCV and randomizedsearchCV for all the above classification algorithms and get the best parameter

Algorithm :

1. **Data Loading, preprocessing, splitting**
2. **Algorithm Selection and Parameter Exploration:**
    - Run the algorithms, Support Vector Machine (SVM), Decision Tree, Random Forest, AdaBoost and find the accuracies.
    - Identify the tunable parameters for each selected algorithm.
3. **GridSearchCV:**
    - For each algorithm:
        - Define a parameter grid that includes various values for the tunable parameters identified in step 2.
        - Initialize a GridSearchCV object for the algorithm with the defined parameter grid and cross-validation (CV) setting (e.g., 5-fold CV).
        - Fit the GridSearchCV object to the training data to search for the best combination of parameters.
        - Retrieve and record the best parameters found for each algorithm.
4. **RandomizedSearchCV:**
    - For each algorithm:
        - Define a parameter distribution that includes ranges or distributions for the tunable parameters identified in step 2.
        - Initialize a RandomizedSearchCV object for the algorithm with the defined parameter distribution, cross-validation (CV) setting (e.g., 5-fold CV), and the number of iterations (e.g., 10).
        - Fit the RandomizedSearchCV object to the training data to randomly search for the best combination of parameters.
        - Retrieve and record the best parameters found for each algorithm.

Code and Output :

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
C:\Users\TEJU\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

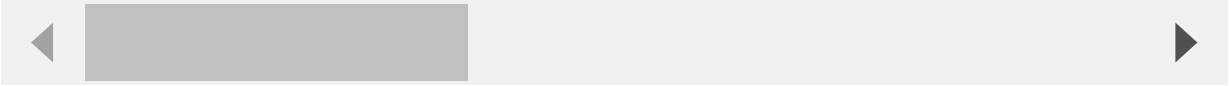
In [2]:
```
df=pd.read_csv(r"C:\Users\TEJU\Downloads\Telco-Customer-Churn (1).csv")
```

In [3]:
```
df.head()
```

Out[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Int
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	
4	9237-HQITU	Female	0	No	No	2	Yes	No	

5 rows × 21 columns



In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64 
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64 
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
```

```
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

In [5]:
```
df['TotalCharges'].replace(" ",0,inplace=True)
df['TotalCharges']=df['TotalCharges'].astype('float64')
```

In [6]:
```
df=df.drop('customerID',axis=1)
```

In [7]:
```
df['SeniorCitizen']=df['SeniorCitizen'].map({0:'No',1:'Yes'})
```

In [8]:
```
num_features=df.select_dtypes(include='number')
cat_features=df.select_dtypes(exclude='number')
```

In [9]:
```
cat_features_encoded = pd.get_dummies(data=cat_features, dtype=int)
churn_corr = cat_features_encoded.corr()['Churn_Yes'].drop(['Churn_Yes', 'Churn_No']
```

In [10]:
```
df_final=pd.get_dummies(data=df,drop_first=True,dtype=int)
```

In [11]:
```
df_final = df_final.drop(['gender_Male', 'PhoneService_Yes',
                'MultipleLines_No phone service',
                'MultipleLines_Yes'], axis=1)
```

In [12]:
```
df_final.head()
```

Out[12]:

	tenure	MonthlyCharges	TotalCharges	SeniorCitizen_Yes	Partner_Yes	Dependents_Yes	InternetSe
0	1	29.85	29.85	0	1	0	
1	34	56.95	1889.50	0	0	0	
2	2	53.85	108.15	0	0	0	
3	45	42.30	1840.75	0	0	0	
4	2	70.70	151.65	0	0	0	

5 rows × 27 columns

In [13]:
```
df_final['Churn_Yes'].value_counts()
```

Out[13]:
```
0    5174
1    1869
```

```
Name: Churn_Yes, dtype: int64
```

In [14]:
```
X=df_final.drop('Churn_Yes',axis=1)
y=df_final['Churn_Yes']
```

In [19]:
```
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_stat
```

In [20]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

SVM :

In [21]:
```
from sklearn.svm import SVC
svcModel = SVC()
svcModel.fit(X_train,y_train)
accuracy = svcModel.score(X_test, y_test)
print("Accuracy of SVM:", accuracy)
```

```
Accuracy: 0.7823000473260767
```

Decision Tree :

In [22]:
```
from sklearn.tree import DecisionTreeClassifier
treeModel=DecisionTreeClassifier()
treeModel.fit(X_train,y_train)
accuracy=treeModel.score(X_test,y_test)
print("Accuracy of decision tree:",accuracy)
```

```
Accuracy of decision tree: 0.7221959299574066
```

Random Forest :

In [23]:
```
from sklearn.ensemble import RandomForestClassifier
rfcModel=RandomForestClassifier()
rfcModel.fit(X_train,y_train)
accuracy=rfcModel.score(X_test,y_test)
print("Accuracy of random forest:",accuracy)
```

```
Accuracy of random forest: 0.7799337434926644
```

Boosting algorithm - Adaboost Classifier :

In [26]:
```
from sklearn.ensemble import AdaBoostClassifier
adbModel=AdaBoostClassifier()
adbModel.fit(X_train,y_train)
accuracy=adbModel.score(X_test,y_test)
print("Accuracy of adaboost:",accuracy)
```

```
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
```

```
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
Accuracy of adaboost: 0.7974443918599148
```

Tunable parameters of each algorithm :

1. Support Vector Machine (SVM):
   - `C` : Penalty parameter of the error term.
   - `kernel` : Specifies the kernel type to be used in the algorithm (linear, polynomial, radial basis function (RBF), sigmoid).
   - `gamma` : Kernel coefficient for 'rbf', 'poly', and 'sigmoid'.
   - `degree` : Degree of the polynomial kernel function ('poly').
   - `coef0` : Independent term in the polynomial kernel function ('poly' and 'sigmoid').
2. Decision Trees:
   - `max_depth` : Maximum depth of the tree.
   - `min_samples_split` : Minimum number of samples required to split an internal node.
   - `min_samples_leaf` : Minimum number of samples required to be at a leaf node.
   - `max_features` : Number of features to consider when looking for the best split.
   - `criterion` : Function to measure the quality of a split (e.g., 'gini' for Gini impurity or 'entropy' for information gain).
3. Random Forest:
   - All parameters of Decision Trees.
   - `n_estimators` : The number of trees in the forest.
   - `bootstrap` : Whether bootstrap samples are used when building trees.
   - `max_samples` : The number of samples to draw from X to train each base estimator.
4. AdaBoost (Adaptive Boosting):
   - `base_estimator` : The base estimator from which the boosted ensemble is built.
   - `n_estimators` : The maximum number of estimators at which boosting is terminated.
   - `learning_rate` : Weight applied to each classifier at each boosting iteration.
   - `algorithm` : The algorithm used to update weights ('SAMME' or 'SAMME.R').

Hyper parameter tuning :

In [28]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

SVM :

In [30]:
```
# GridSearchCV :

param_grid_svc = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf'], 'gamma': ['scale',
grid_search_svc = GridSearchCV(SVC(), param_grid_svc, cv=5)
grid_search_svc.fit(X_train, y_train)
best_params_svc = grid_search_svc.best_params_
print("Best Parameters according to GridSearchCV for SVC:", best_params_svc)
```

```
Best Parameters according to GridSearchCVfor SVC: {'C': 1, 'gamma': 'scale', 'kerne
l': 'rbf'}
```

In [31]:
```
# RandomisedSearchCV :

from scipy.stats import uniform
param_dist_svc = {'C': uniform(loc=0, scale=10), 'kernel': ['linear', 'rbf'], 'gamma
random_search_svc = RandomizedSearchCV(SVC(), param_dist_svc, cv=5, n_iter=10)
random_search_svc.fit(X_train, y_train)
best_params_rand_svc = random_search_svc.best_params_
print("Best Parameters to RandomisedSearchCV for SVC :", best_params_rand_svc)
```

```
Best Parameters to RandomisedSearchCV for SVC : {'C': 0.9555803888742154, 'gamma': 's
cale', 'kernel': 'rbf'}
```

Decision Trees :

In [32]:
```
#GridSearchCV :

param_grid_dt = {'criterion': ['gini', 'entropy'], 'splitter': ['best', 'random'],
                 'max_depth': [None, 10, 20], 'min_samples_split': [2, 5, 10],
                 'min_samples_leaf': [1, 2, 4]}
grid_search_dt = GridSearchCV(DecisionTreeClassifier(), param_grid_dt, cv=5)
grid_search_dt.fit(X_train, y_train)
best_params_dt = grid_search_dt.best_params_
print("Best Parameters for Decision Tree:(grid)", best_params_dt)
```

```
Best Parameters for Decision Tree:(grid) {'criterion': 'entropy', 'max_depth': 10, 'm
in_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

In [33]:
```
#RandomizedSearchCV :

param_dist_dt = {'criterion': ['gini', 'entropy'], 'splitter': ['best', 'random'],
                 'max_depth': [None, 10, 20], 'min_samples_split': [2, 5, 10],
                  'min_samples_leaf': [1, 2, 4]}
random_search_dt = RandomizedSearchCV(DecisionTreeClassifier(), param_dist_dt, cv=5,
random_search_dt.fit(X_train, y_train)
best_params_rand_dt = random_search_dt.best_params_
print("Best Parameters for Decision Tree (RandomizedSearchCV):", best_params_rand_dt
```

```
Best Parameters for Decision Tree (RandomizedSearchCV): {'splitter': 'best', 'min_sam
ples_split': 2, 'min_samples_leaf': 4, 'max_depth': 10, 'criterion': 'entropy'}
```

Random Forest :

In [40]:
```
#GridSearchCV :

param_grid_rf = {'n_estimators': [100], 'criterion': ['gini', 'entropy'],
                 'max_depth': [None, 10, 20], 'min_samples_split': [2, 10],
                 'min_samples_leaf': [1,2]}
grid_search_rf = GridSearchCV(RandomForestClassifier(), param_grid_rf, cv=2)
grid_search_rf.fit(X_train, y_train)
best_params_rf = grid_search_rf.best_params_
print("Best Parameters for Random Forest:(grid)", best_params_rf)
```

```
Best Parameters for Random Forest:(grid) {'criterion': 'entropy', 'max_depth': 10, 'm
in_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}
```

In [42]:
```
#RandomizedSearchCV :

param_dist_ada = {'n_estimators': [50, 100], 'learning_rate': [0.01, 0.1]}
random_search_ada = RandomizedSearchCV(AdaBoostClassifier(), param_dist_ada, cv=3, n
random_search_ada.fit(X_train, y_train)
best_params_rand_ada = random_search_ada.best_params_
print("Best Parameters for AdaBoost (RandomizedSearchCV):", best_params_rand_ada)
```

```
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:318: Use
rWarning: The total space of parameters 4 is smaller than n_iter=5. Running 4 iterati
ons. For exhaustive searches, use GridSearchCV.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
```

```
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
Best Parameters for AdaBoost (RandomizedSearchCV): {'n_estimators': 100, 'learning_ra
te': 0.1}
```

Adaboost :

In [43]:

```
param_grid_ada = {
    'n_estimators': [50, 100],
    'learning_rate': [0.01, 0.1]
}
grid_search_ada = GridSearchCV(AdaBoostClassifier(), param_grid_ada, cv=2)
grid_search_ada.fit(X_train, y_train)
best_params_grid_ada = grid_search_ada.best_params_
print("Best Parameters for AdaBoost (GridSearchCV):", best_params_grid_ada)
```

```
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: F
utureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed i
n 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
Best Parameters for AdaBoost (GridSearchCV): {'learning_rate': 0.1, 'n_estimators': 1
00}
```

In [44]:

```
param_dist_ada = {
    'n_estimators': [100, 150],
    'learning_rate': [0.1, 1.0]
}
random_search_ada = RandomizedSearchCV(AdaBoostClassifier(), param_dist_ada, cv=2, n
random_search_ada.fit(X_train, y_train)
```

```
best_params_rand_ada = random_search_ada.best_params_
print("Best Parameters for AdaBoost (RandomizedSearchCV):", best_params_rand_ada)
```

```
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:318: UserWarning: The total space of parameters 4 is smaller than n_iter=10. Running 4 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\TEJU\anaconda3\lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
Best Parameters for AdaBoost (RandomizedSearchCV): {'n_estimators': 100, 'learning_rate': 0.1}
```

Results :

Therefore, we were successfully able to run SVM, Decision tree, Random Forest and adaboost, check their accuracies, learn about tunable parameters and then apply gridSearchCV and randomizedSearchCV again on each algorithm to check what are the best hyper parameters to apply to increase accuracy.