In [5]:
```
#Importing Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [48]:
```
# import warnings filter
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)
```

In [11]:
```
#Importing the dataset
dataset = pd.read_csv('pulsar_stars.csv')
X = dataset.iloc[:, :-1].values #Independent vector
Y = dataset.iloc[:, 8].values   #Dependent vector
```

In [24]:
```
#Handling Missing data
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer = imputer.fit(X[:, :])
X[:, :] = imputer.transform(X[:, :])
```

In [26]:
```
#Splitting the dataset into training and test set
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, ra
ndom_state=0)
```

In [28]:
```
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

In [29]:
```
#Fitting Naive Bayes  to Training set
from sklearn.naive_bayes import GaussianNB
classifier_NB = GaussianNB()
classifier_NB.fit(X_train, Y_train)
```

Out[29]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [43]:
```
#Fitting SVM to Training set
from sklearn.svm import SVC
classifier_SVM = SVC(kernel = 'linear', random_state = 0, gamma = 'auto')
classifier_SVM.fit(X_train, Y_train)
```

Out[43]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
    max_iter=-1, probability=False, random_state=0, shrinking=True, tol=0.00
1,
    verbose=False)
```

In [33]:
```
#Fitting Kernel SVM  to Training set
from sklearn.svm import SVC
classifier_KSVM = SVC(kernel='rbf', random_state = 0)
classifier_KSVM.fit(X_train, Y_train)
```

Out[33]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=0,
    shrinking=True, tol=0.001, verbose=False)
```

In [34]:
```
#Fitting Decision Tree Classification  to Training set
from sklearn.tree import DecisionTreeClassifier
classifier_DT = DecisionTreeClassifier(criterion = 'entropy', random_state = 0
)
classifier_DT.fit(X_train, Y_train)
```

Out[34]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=Non
e,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

In [37]:
```
#Fitting Random Forest Classification  to Training set
from sklearn.ensemble import RandomForestClassifier
classifier_RFC = RandomForestClassifier(n_estimators = 10, criterion = 'entrop
y', random_state = 0)
classifier_RFC.fit(X_train, Y_train)
```

Out[37]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entrop
y',
                       max_depth=None, max_features='auto', max_leaf_nodes=No
ne,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=0, verbose=
0,
                       warm_start=False)
```

In [44]:
```
#Predicting the results
Y_pred_NB = classifier_NB.predict(X_test)
Y_pred_SVM = classifier_SVM.predict(X_test)
Y_pred_KSVM = classifier_KSVM.predict(X_test)
Y_pred_DT = classifier_DT.predict(X_test)
Y_pred_RFC = classifier_RFC.predict(X_test)
```

In [53]:
```
#Applying K-Fold Cross Validation
from sklearn.model_selection import cross_val_score
accuracies_NB = cross_val_score(estimator = classifier_NB, X = X_train, y = Y_
train, cv = 10)
accuracies_SVM = cross_val_score(estimator = classifier_SVM, X = X_train, y =
Y_train, cv = 10)
accuracies_KSVM = cross_val_score(estimator = classifier_KSVM, X = X_train, y
= Y_train, cv = 10)
accuracies_DT = cross_val_score(estimator = classifier_DT, X = X_train, y = Y_
train, cv = 10)
accuracies_RFC = cross_val_score(estimator = classifier_RFC, X = X_train, y =
Y_train, cv = 10)

print('Accuracy for Classification Models:')
print('Naive Bayes: ',round(accuracies_NB.mean()*100,3), '%')
print('Support Vector Machine: ',round(accuracies_SVM.mean()*100,3), '%')
print('Kernel SVM: ',round(accuracies_KSVM.mean()*100,3), '%')
print('Decision Trees: ',round(accuracies_DT.mean()*100,3), '%')
print('Random Forest: ',round(accuracies_RFC.mean()*100,3), '%')
```

```
Accuracy for Classification Models:
Naive Bayes:  94.287 %
Support Vector Machine:  97.807 %
Kernel SVM:  97.758 %
Decision Trees:  96.836 %
Random Forest:  97.779 %
```

In [64]:
```
#Calculating Precision, Recall and F1 score
from sklearn.metrics import average_precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

average_precision_NB = average_precision_score(Y_test, Y_pred_NB)
average_precision_SVM = average_precision_score(Y_test, Y_pred_SVM)
average_precision_KSVM = average_precision_score(Y_test, Y_pred_KSVM)
average_precision_DT = average_precision_score(Y_test, Y_pred_DT)
average_precision_RFC = average_precision_score(Y_test, Y_pred_RFC)
print('Precision: Naive Bayes',round(average_precision_NB,3))
print('Precision: Support Vector Machine',round(average_precision_SVM,3))
print('Precision: Kernel SVM',round(average_precision_KSVM,3))
print('Precision: Decision Trees',round(average_precision_DT,3))
print('Precision: Random Forest',round(average_precision_RFC,3))
```

```
Precision: Naive Bayes 0.575
Precision: Support Vector Machine 0.794
Precision: Kernel SVM 0.791
Precision: Decision Trees 0.686
Precision: Random Forest 0.801
```

In [65]:
```
recall_score_NB = recall_score(Y_test, Y_pred_NB, average='macro')
recall_score_SVM = recall_score(Y_test, Y_pred_SVM, average='macro')
recall_score_KSVM = recall_score(Y_test, Y_pred_KSVM, average='macro')
recall_score_DT = recall_score(Y_test, Y_pred_DT, average='macro')
recall_score_RFC = recall_score(Y_test, Y_pred_RFC, average='macro')
print('Recall: Naive Bayes',round(recall_score_NB,3))
print('Recall: Support Vector Machine',round(recall_score_SVM,3))
print('Recall: Kernel SVM',round(recall_score_KSVM,3))
print('Recall: Decision Trees',round(recall_score_DT,3))
print('Recall: Random Forest',round(recall_score_RFC,3))
```

```
Recall: Naive Bayes 0.915
Recall: Support Vector Machine 0.917
Recall: Kernel SVM 0.919
Recall: Decision Trees 0.908
Recall: Random Forest 0.921
```

In [66]:
```
f1_score_NB = f1_score(Y_test, Y_pred_NB, average='macro')
f1_score_SVM = f1_score(Y_test, Y_pred_SVM, average='macro')
f1_score_KSVM = f1_score(Y_test, Y_pred_KSVM, average='macro')
f1_score_DT = f1_score(Y_test, Y_pred_DT, average='macro')
f1_score_RFC = f1_score(Y_test, Y_pred_RFC, average='macro')
print('F1 Score: Naive Bayes',round(f1_score_NB,3))
print('F1 Score: Support Vector Machine',round(f1_score_SVM,3))
print('F1 Score: Kernel SVM',round(f1_score_KSVM,3))
print('F1 Score: Decision Trees',round(f1_score_DT,3))
print('F1 Score: Random Forest',round(f1_score_RFC,3))
```

```
F1 Score: Naive Bayes 0.859
F1 Score: Support Vector Machine 0.937
F1 Score: Kernel SVM 0.936
F1 Score: Decision Trees 0.903
F1 Score: Random Forest 0.939
```