HW2_Hardik_2678294168

September 20, 2021

1 HW2 - EE660 : Hardik Prajapati (2678294168)

[66]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn import preprocessing
import random
from sklearn.model_selection import train_test_split
```

[2]:
```
def readfile(filename,head):
    filepath="../data/"
    file=filepath+filename
    df=pd.read_csv(file,header=head)
    return df
```

[3]:
```
Xtrain=readfile('x_train.csv',0)
Ytrain=readfile('y_train.csv',None)
Xtest=readfile('x_test.csv',0)
Ytest=readfile('y_test.csv',None)
```

[4]:
```
le = preprocessing.LabelEncoder()
le.fit(Ytrain)
le_dict = dict(zip(le.classes_, le.transform(le.classes_)))
Ytrain_enc=le.transform(Ytrain)
Ytest_enc=le.transform(Ytest)
le_dict
```

```
c:\users\hp631\appdata\local\programs\python\python39\lib\site-
packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

[4]:
```
{'CYT': 0,
 'ERL': 1,
 'EXC': 2,
 'ME1': 3,
```

```
 'ME2': 4,
 'ME3': 5,
 'MIT': 6,
 'NUC': 7,
 'POX': 8,
 'VAC': 9}
```

[5]:
```
lab_0=0
lab_1=0
lab_2=0
lab_3=0
lab_4=0
lab_5=0
lab_6=0
lab_7=0
lab_8=0
lab_9=0

for lab in Ytrain_enc:
    if lab==0:
        lab_0=lab_0+1
    elif lab==1:
        lab_1=lab_1+1
    elif lab==2:
        lab_2=lab_2+1
    elif lab==3:
        lab_3=lab_3+1
    elif lab==4:
        lab_4=lab_4+1
    elif lab==5:
        lab_5=lab_5+1
    elif lab==6:
        lab_6=lab_6+1
    elif lab==7:
        lab_7=lab_7+1
    elif lab==8:
        lab_8=lab_8+1
    elif lab==9:
        lab_9=lab_9+1
```

[6]:
```
def frequency(lab_count):
    fre=lab_count/1000
    return fre
```

[8]:
```
lab_0_fre=frequency(lab_0)
lab_1_fre=frequency(lab_1)
lab_2_fre=frequency(lab_2)
```

```
lab_3_fre=frequency(lab_3)
lab_4_fre=frequency(lab_4)
lab_5_fre=frequency(lab_5)
lab_6_fre=frequency(lab_6)
lab_7_fre=frequency(lab_7)
lab_8_fre=frequency(lab_8)
lab_9_fre=frequency(lab_9)
prob_arr=[lab_0_fre,lab_1_fre,lab_2_fre,lab_3_fre,lab_4_fre,lab_5_fre,lab_6_fre,lab_7_fre,lab_
prob_arr
```

[8]:
```
[0.313, 0.005, 0.023, 0.03, 0.038, 0.106, 0.159, 0.292, 0.015, 0.019]
```

[13]:
```
def classification_error(Ytrue,Ypred):
    misclasf=0
    for i in range(len(Ytrue)):
        if Ytrue[i]!=Ypred[i]:
            misclasf=misclasf+1
    misclasf_err=misclasf/len(Ytrue)
    return misclasf_err
```

[14]:
```
def classifier1_random(seed_val,trainY,testY,target_label,pro):
    random.seed(seed_val)
    train_pred=random.choices(target_label, weights=pro, k=len(trainY))
    test_pred=random.choices(target_label, weights=pro, k=len(testY))
    train_error=classification_error(trainY,train_pred)
    test_error=classification_error(testY,test_pred)
    return train_error,test_error
```

[30]:
```
seed_value=[1,10,100,500,1000,2500,5000,7500,10000,50000]
train_err_clf1=[]
test_err_clf1=[]
for value in seed_value:
    ␣
 ↪train_err,test_err=classifier1_random(value,Ytrain_enc,Ytest_enc,[0,1,2,3,4,5,6,7,8,9],prob
    train_err_clf1.append(train_err)
    test_err_clf1.append(test_err)
mean_train_error_clf1=np.mean(train_err_clf1)
std_train_error_clf1=np.std(train_err_clf1)
mean_test_error_clf1=np.mean(test_err_clf1)
std_test_error_clf1=np.std(test_err_clf1)

print("Mean training error and STD for trivial CLF 1 = {:.4f} +/- {:.2f} ".
 ↪format(mean_train_error_clf1,std_train_error_clf1))
print("...")
print("Mean test error and STD for trivial CLF 1 = {:.4f} +/- {:.2f} ".
 ↪format(mean_test_error_clf1,std_test_error_clf1))
```

```
Mean training error and STD for trivial CLF 1 = 0.7837 +/- 0.01
```

…
Mean test error and STD for trivial CLF 1 = 0.7771 +/- 0.03

[37]:
```
def classifier2_populated(trainY,testY,pro):
    label_high_freq=np.argmax(pro)
    train_pred=[label_high_freq]*len(trainY)
    test_pred=[label_high_freq]*len(testY)
    train_error=classification_error(trainY,train_pred)
    test_error=classification_error(testY,test_pred)
    return train_error,test_error
```

[38]:
```
train_err_clf2,test_err_clf2=classifier2_populated(Ytrain_enc,Ytest_enc,prob_arr)
print("Training error for trivial CLF 2 = {:.4f} ".format(train_err_clf2))
print("...")
print("Test error for trivial CLF 2 = {:.4f} ".format(test_err_clf2))
```

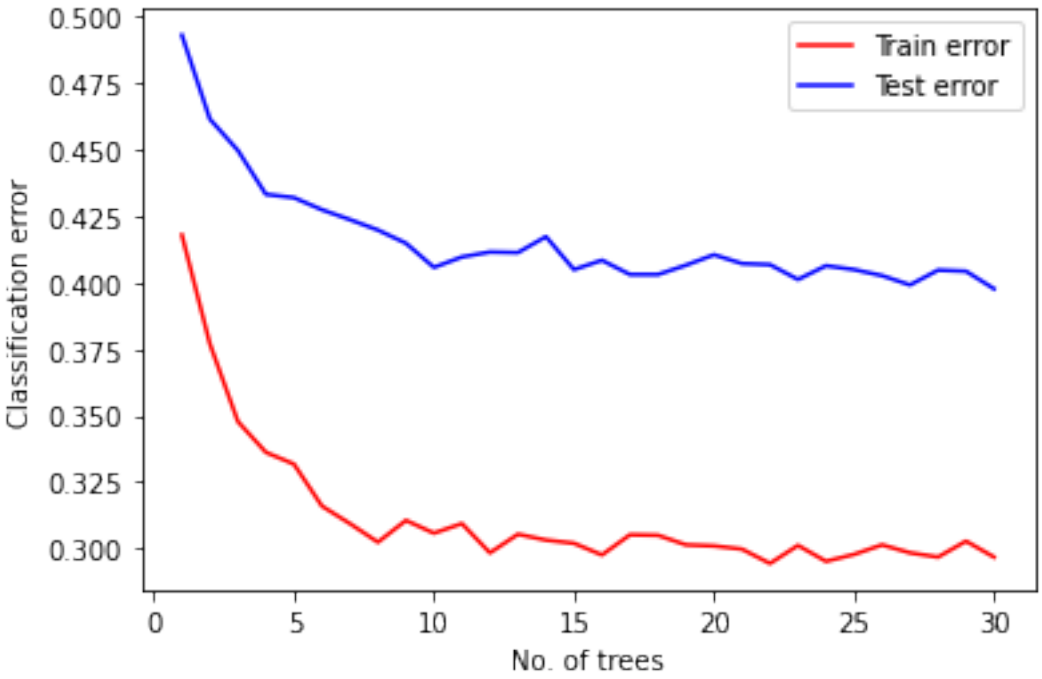
Training error for trivial CLF 2 = 0.6870
…
Test error for trivial CLF 2 = 0.6901

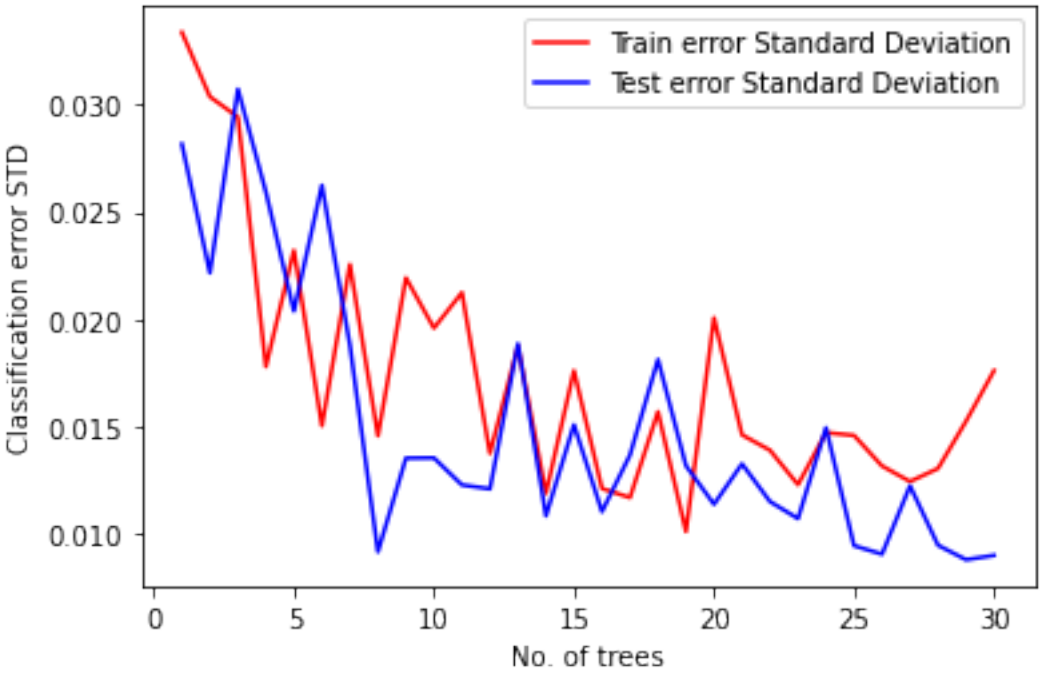
[50]:
```
def␣
↪classifier3_randomforest(trainX,trainY,testX,testY,bag_size,n_trees,depth,features,random_v
↪
    testsize_bag=1-bag_size
    Xtrain_bag, Xtest_bag, Ytrain_bag, Ytest_bag =␣
↪train_test_split(trainX,trainY, test_size=testsize_bag,␣
↪random_state=random_val)
    clf=RandomForestClassifier(n_estimators=n_trees,criterion='entropy',␣
↪max_depth=depth, max_features=features, bootstrap=True)
    clf.fit(Xtrain_bag,Ytrain_bag)
    train_pred=clf.predict(Xtrain_bag)
    test_pred=clf.predict(testX)
    train_error=classification_error(Ytrain_bag,train_pred)
    test_error=classification_error(testY,test_pred)
    return train_error,test_error
```

[79]:
```
estimators=np.arange(1,31,1)
bag_random_state=np.linspace(10,1000,10,dtype=int)
mean_train_err_randomforest=[]
mean_test_err_randomforest=[]
std_train_err_randomforest=[]
std_test_err_randomforest=[]

for b in estimators:
    train_error_array=[]
    test_error_array=[]
    for s in bag_random_state:
```

```
      ␣
↪train_err,test_err=classifier3_randomforest(Xtrain,Ytrain_enc,Xtest,Ytest_enc,0.
↪5,b,5,3,s)
       train_error_array.append(train_err)
       test_error_array.append(test_err)
   mean_train_error=np.mean(train_error_array)
   std_train_error=np.std(train_error_array)
   mean_test_error=np.mean(test_error_array)
   std_test_error=np.std(test_error_array)
   mean_train_err_randomforest.append(mean_train_error)
   std_train_err_randomforest.append(std_train_error)
   mean_test_err_randomforest.append(mean_test_error)
   std_test_err_randomforest.append(std_test_error)
```

[81]:
```
plt.figure(1)
plt.plot(estimators,mean_train_err_randomforest,color='red',label='Train error')
plt.plot(estimators,mean_test_err_randomforest,color='blue',label='Test error')
plt.xlabel('No. of trees')
plt.ylabel('Classification error')
plt.legend()

plt.figure(2)
plt.plot(estimators,std_train_err_randomforest,color='red',label='Train error␣
↪Standard Deviation')
plt.plot(estimators,std_test_err_randomforest,color='blue',label='Test error␣
↪Standard Deviation')
plt.xlabel('No. of trees')
plt.ylabel('Classification error STD')
plt.legend()

plt.show()

best_err_idx=np.argmin(mean_test_err_randomforest)
print("Minimun test error rate = {} +/- {} ".
↪format(mean_test_err_randomforest[best_err_idx],std_test_err_randomforest[best_err_idx]))
```






```
Minimun test error rate = 0.39772727272727265 +/- 0.008970369151277854
```

2 Q3: b) i) :

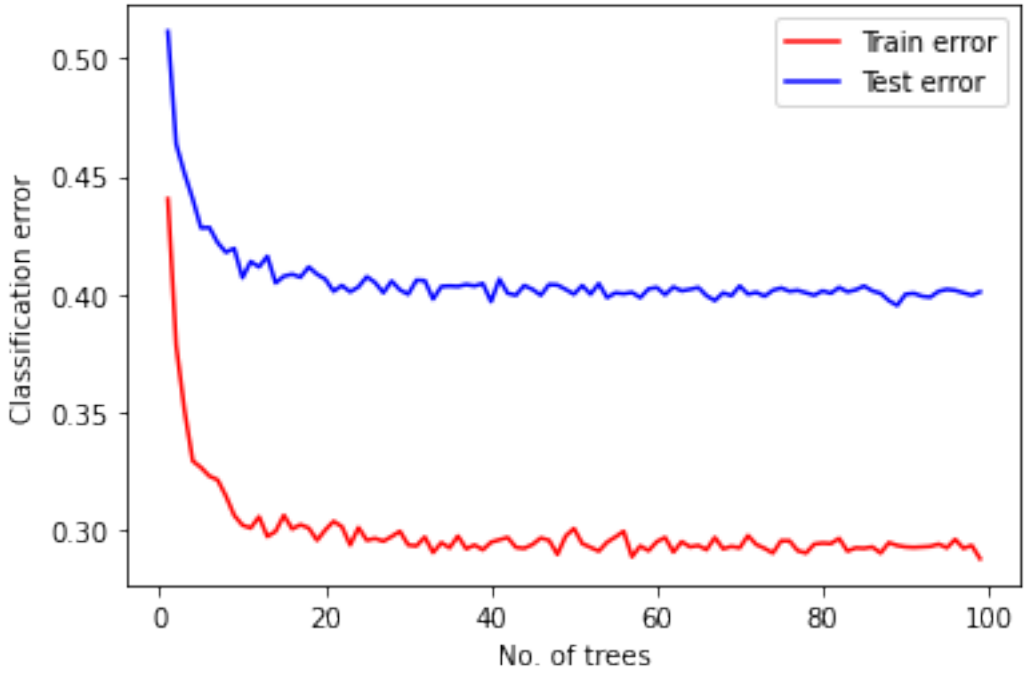
3 1)The best testing error rate occurs at No.of trees = 19, with error=0.3977 and std=0.00897

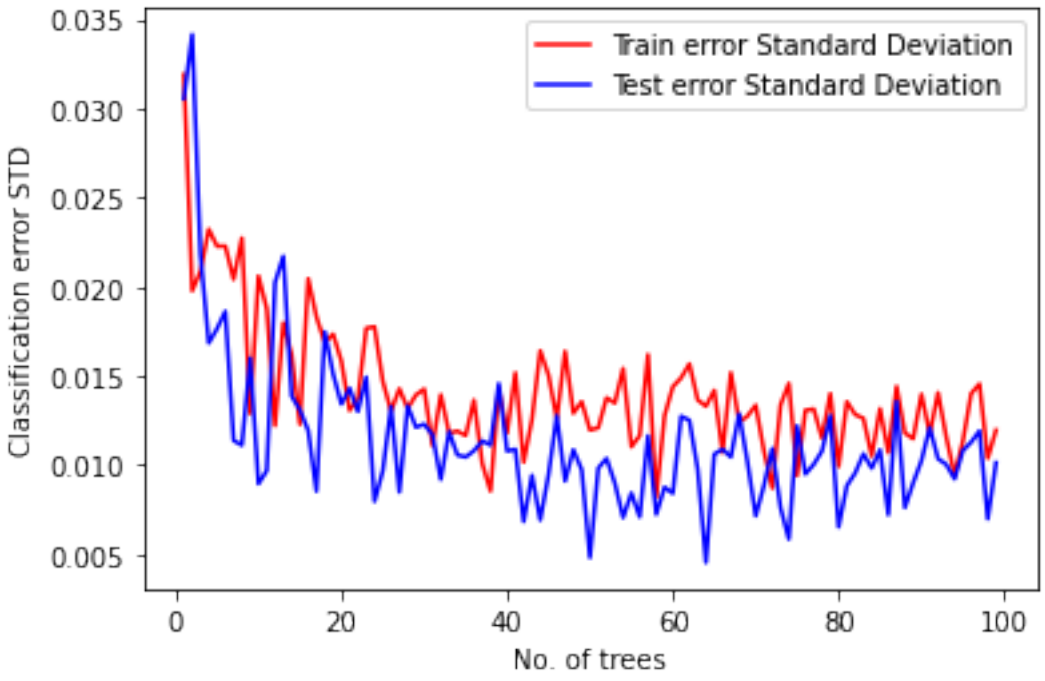
4 2)With increase in no. of trees, Training error and test error decreases very rapidly at the start and then steadies down.

[82]:
```
#setting B=100
estimators=np.arange(1,100,1)
bag_random_state=np.linspace(10,1000,10,dtype=int)
mean_train_err_randomforest=[]
mean_test_err_randomforest=[]
std_train_err_randomforest=[]
std_test_err_randomforest=[]

for b in estimators:
    train_error_array=[]
    test_error_array=[]
    for s in bag_random_state:
        ␣
 ↪train_err,test_err=classifier3_randomforest(Xtrain,Ytrain_enc,Xtest,Ytest_enc,0.
 ↪5,b,5,3,s)
        train_error_array.append(train_err)
        test_error_array.append(test_err)
    mean_train_error=np.mean(train_error_array)
    std_train_error=np.std(train_error_array)
    mean_test_error=np.mean(test_error_array)
    std_test_error=np.std(test_error_array)
    mean_train_err_randomforest.append(mean_train_error)
    std_train_err_randomforest.append(std_train_error)
    mean_test_err_randomforest.append(mean_test_error)
    std_test_err_randomforest.append(std_test_error)
```

[83]:
```
plt.figure(3)
plt.plot(estimators,mean_train_err_randomforest,color='red',label='Train error')
plt.plot(estimators,mean_test_err_randomforest,color='blue',label='Test error')
plt.xlabel('No. of trees')
plt.ylabel('Classification error')
plt.legend()

plt.figure(4)
plt.plot(estimators,std_train_err_randomforest,color='red',label='Train error␣
 ↪Standard Deviation')
plt.plot(estimators,std_test_err_randomforest,color='blue',label='Test error␣
 ↪Standard Deviation')
```

```
plt.xlabel('No. of trees')
plt.ylabel('Classification error STD')
plt.legend()

plt.show()

best_err_idx=np.argmin(mean_test_err_randomforest)
print("Minimun test error rate = {} +/- {} ".
 ↪format(mean_test_err_randomforest[best_err_idx],std_test_err_randomforest[best_err_idx]))
```





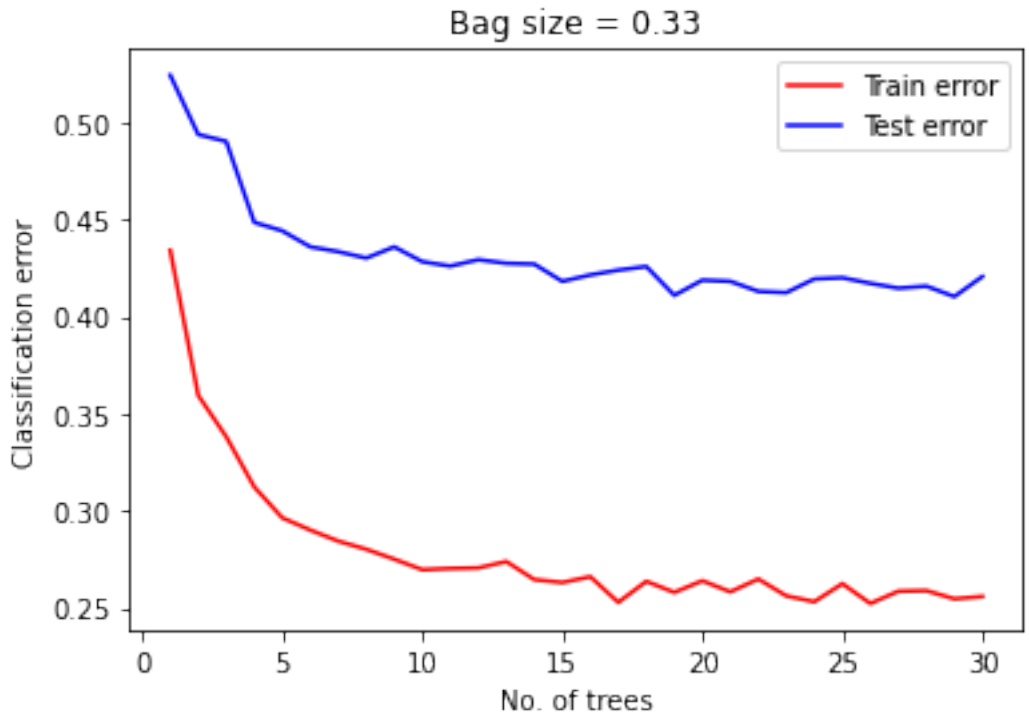

```
Minimun test error rate = 0.3952479338842975 +/- 0.00900835923826101
```

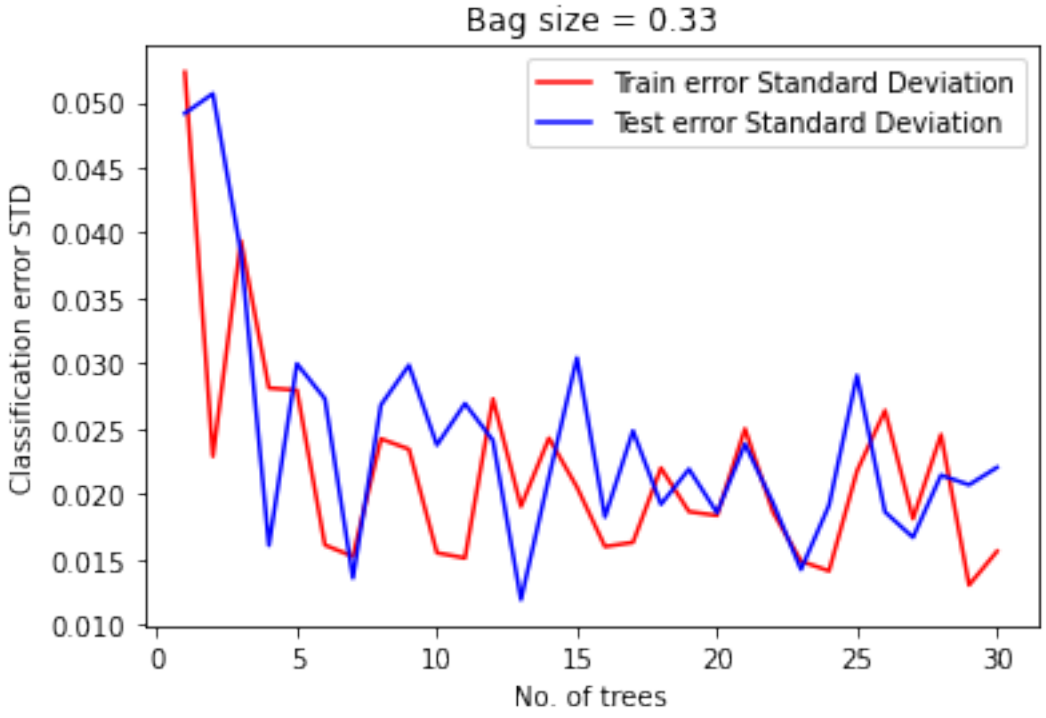
5 Q3:b:ii)

6 On changing B=100 , the Mean error rate converges but the standard deviation curve does not converge as seen from the defined scale on Y-axis. The test error Standard Deviation curve varries from 0.005 to 0.015.

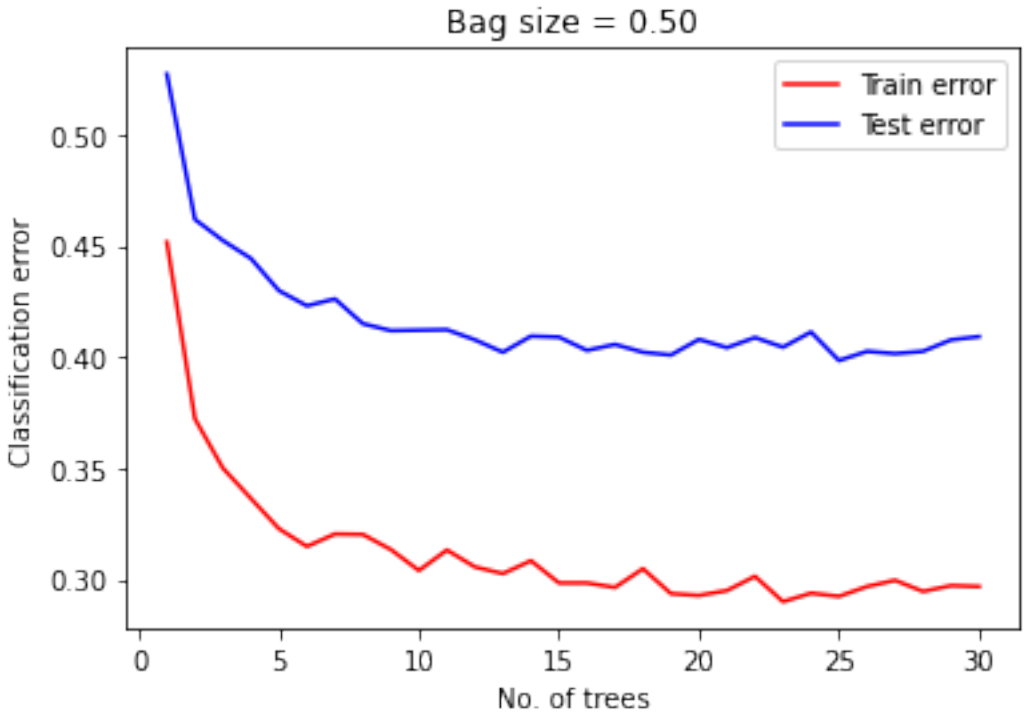
[93]:
```
bag_size_=[1/3,1/2,2/3]
estimators=np.arange(1,31,1)
bag_random_state=np.linspace(10,1000,10,dtype=int)

for bag in bag_size_:
    mean_train_err_randomforest=[]
    mean_test_err_randomforest=[]
    std_train_err_randomforest=[]
    std_test_err_randomforest=[]
    for b in estimators:
        train_error_array=[]
        test_error_array=[]
        for s in bag_random_state:
```

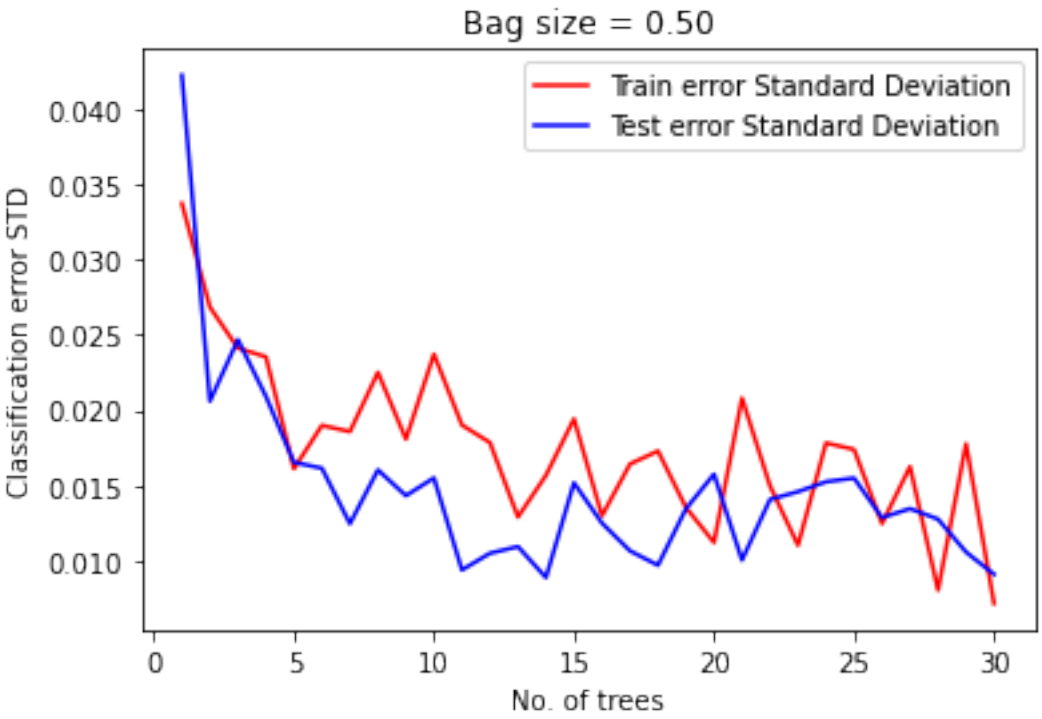
```
            ␣
↪train_err,test_err=classifier3_randomforest(Xtrain,Ytrain_enc,Xtest,Ytest_enc,bag,b,5,3,s)
            train_error_array.append(train_err)
            test_error_array.append(test_err)
        mean_train_error=np.mean(train_error_array)
        std_train_error=np.std(train_error_array)
        mean_test_error=np.mean(test_error_array)
        std_test_error=np.std(test_error_array)
        mean_train_err_randomforest.append(mean_train_error)
        std_train_err_randomforest.append(std_train_error)
        mean_test_err_randomforest.append(mean_test_error)
        std_test_err_randomforest.append(std_test_error)
    plt.figure()
    plt.plot(estimators,mean_train_err_randomforest,color='red',label='Train␣
↪error')
    plt.plot(estimators,mean_test_err_randomforest,color='blue',label='Test␣
↪error')
    plt.xlabel('No. of trees')
    plt.ylabel('Classification error')
    plt.legend()
    plt.title('Bag size = {:.2f}'.format(bag))

    plt.figure()
    plt.plot(estimators,std_train_err_randomforest,color='red',label='Train␣
↪error Standard Deviation')
    plt.plot(estimators,std_test_err_randomforest,color='blue',label='Test␣
↪error Standard Deviation')
    plt.xlabel('No. of trees')
    plt.ylabel('Classification error STD')
    plt.legend()
    plt.title('Bag size = {:.2f}'.format(bag))
    plt.show()
    best_err_idx=np.argmin(mean_test_err_randomforest)
    print("--> Minimun test error rate for bag size:{:.2f}  == {} +/- {} ".
↪format(bag,mean_test_err_randomforest[best_err_idx],std_test_err_randomforest[best_err_idx]
    print("            ")
    print("            ")
    print("            ")
```


Bag size = 0.33
Train error
Test error
Classification error
0.50
0.45
0.40
0.35
0.30
0.25
0
5
10
15
20
25
30
No. of trees

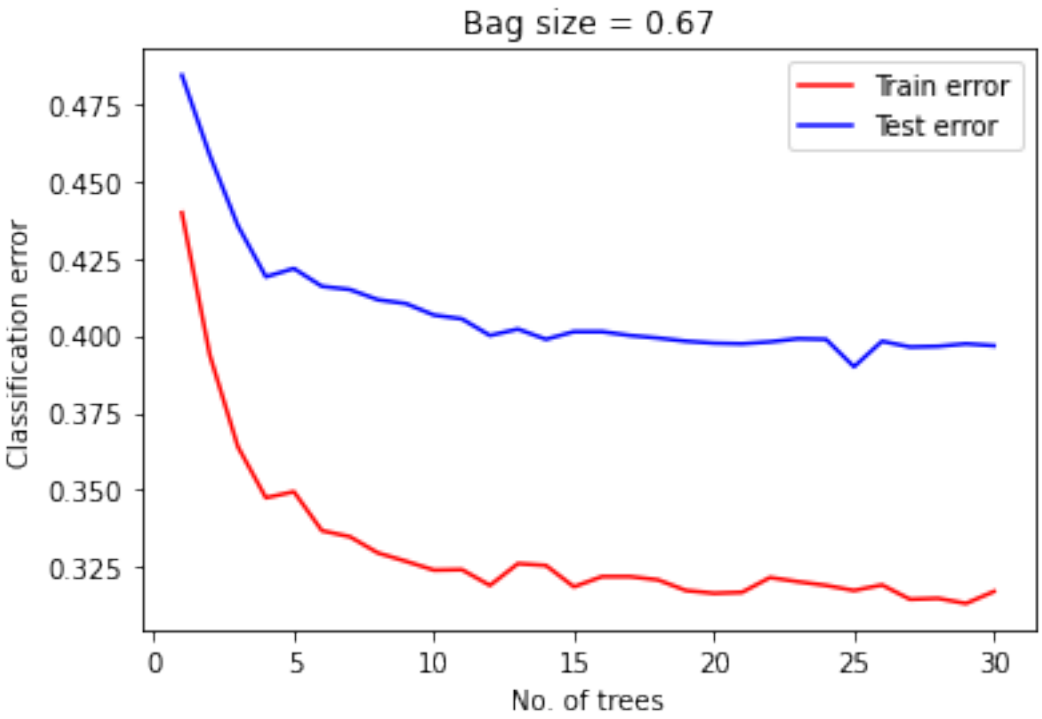

Bag size = 0.33
Train error Standard Deviation
Test error Standard Deviation
Classification error STD
0.050
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0
5
10
15
20
25
30
No. of trees

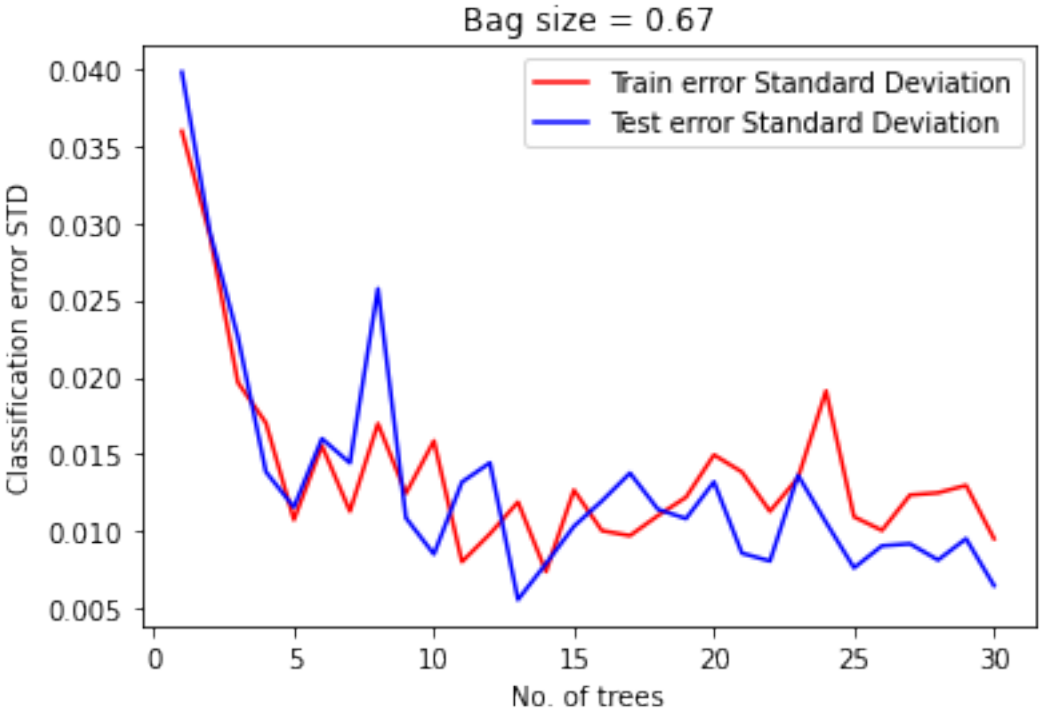
```
--> Minimun test error rate for bag size:0.33  == 0.4103305785123966 +/-
0.020685935555014585
```






```
--> Minimun test error rate for bag size:0.50  == 0.3983471074380166 +/-
0.015483466111731573
```


Bag size = 0.67
Train error
Test error
Classification error
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0
5
10
15
20
25
30
No. of trees


Bag size = 0.67
Train error Standard Deviation
Test error Standard Deviation
Classification error STD
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0
5
10
15
20
25
30
No. of trees

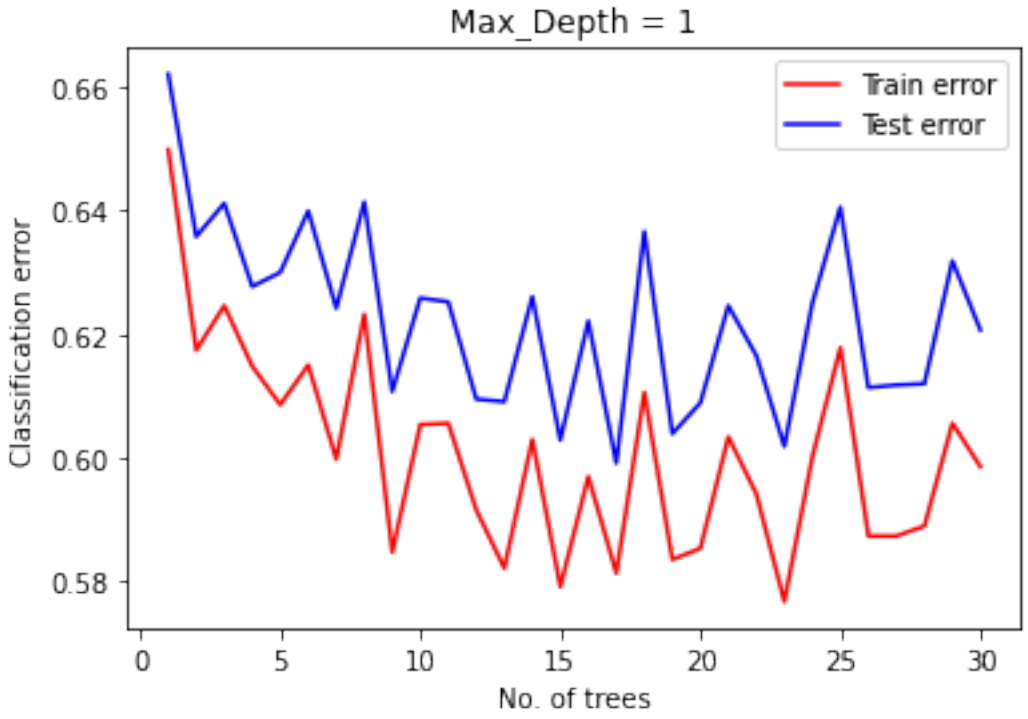
```
--> Minimun test error rate for bag size:0.67  == 0.38987603305785123 +/-
0.007622258985825088
```

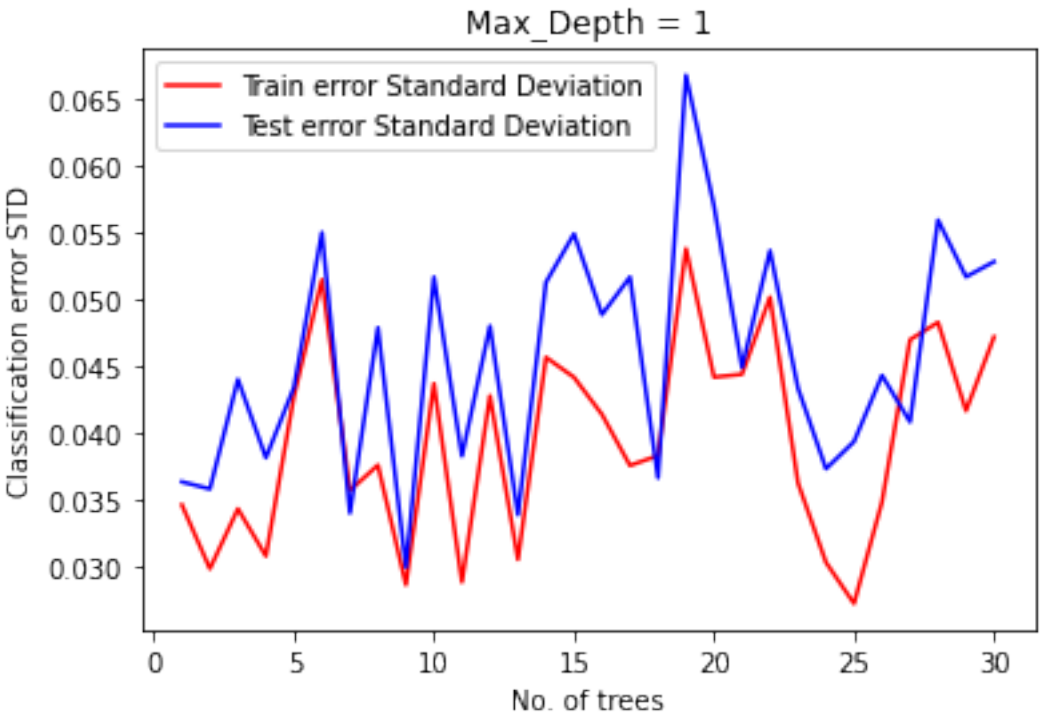
7 Q3: b) iii)

8 1) changing Bag size: We get the best error rate when Bag_Size =2/3. With increase in bag size, performance improves gradually. From bag_size= 1/3 to 2/3, one can observe significant difference in error rate, after accounting the standard deviation.

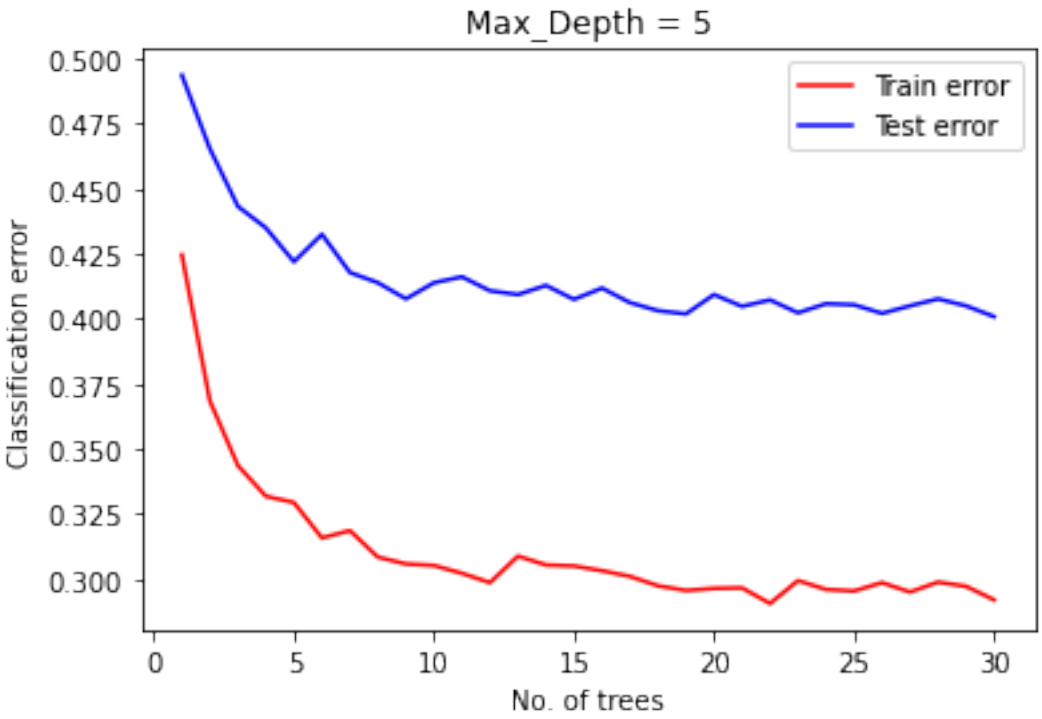
[94]:
```
max_depth_=[1,5,None]
estimators=np.arange(1,31,1)
bag_random_state=np.linspace(10,1000,10,dtype=int)

for deep in max_depth_:
    mean_train_err_randomforest=[]
    mean_test_err_randomforest=[]
    std_train_err_randomforest=[]
    std_test_err_randomforest=[]
    for b in estimators:
        train_error_array=[]
        test_error_array=[]
        for s in bag_random_state:
            ␣
 ↪train_err,test_err=classifier3_randomforest(Xtrain,Ytrain_enc,Xtest,Ytest_enc,0.
 ↪5,b,deep,3,s)
            train_error_array.append(train_err)
            test_error_array.append(test_err)
        mean_train_error=np.mean(train_error_array)
        std_train_error=np.std(train_error_array)
        mean_test_error=np.mean(test_error_array)
        std_test_error=np.std(test_error_array)
        mean_train_err_randomforest.append(mean_train_error)
        std_train_err_randomforest.append(std_train_error)
        mean_test_err_randomforest.append(mean_test_error)
        std_test_err_randomforest.append(std_test_error)
    plt.figure()
    plt.plot(estimators,mean_train_err_randomforest,color='red',label='Train␣
 ↪error')
    plt.plot(estimators,mean_test_err_randomforest,color='blue',label='Test␣
 ↪error')
    plt.xlabel('No. of trees')
```

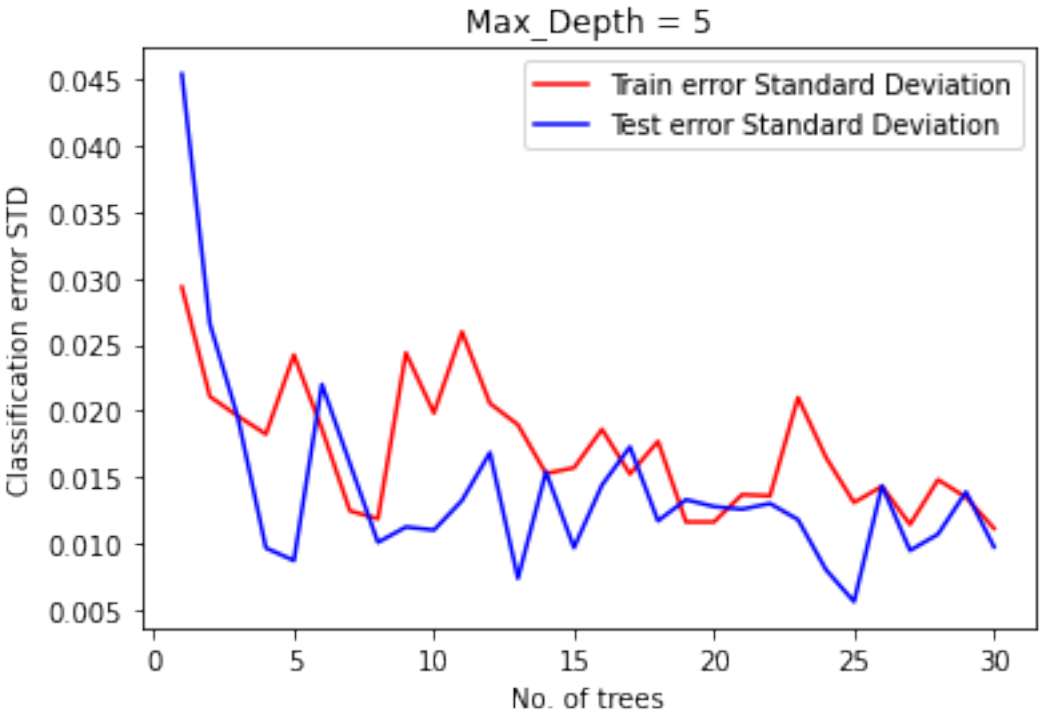
```
    plt.ylabel('Classification error')
    plt.legend()
    plt.title('Max_Depth = {}'.format(deep))

    plt.figure()
    plt.plot(estimators,std_train_err_randomforest,color='red',label='Train␣
↪error Standard Deviation')
    plt.plot(estimators,std_test_err_randomforest,color='blue',label='Test␣
↪error Standard Deviation')
    plt.xlabel('No. of trees')
    plt.ylabel('Classification error STD')
    plt.legend()
    plt.title('Max_Depth = {}'.format(deep))
    plt.show()
    best_err_idx=np.argmin(mean_test_err_randomforest)
    print("--> Minimun test error rate for Max_Depth:{}  == {} +/- {} ".
↪format(deep,mean_test_err_randomforest[best_err_idx],std_test_err_randomforest[best_err_idx
    print("             ")
    print("             ")
    print("             ")
```



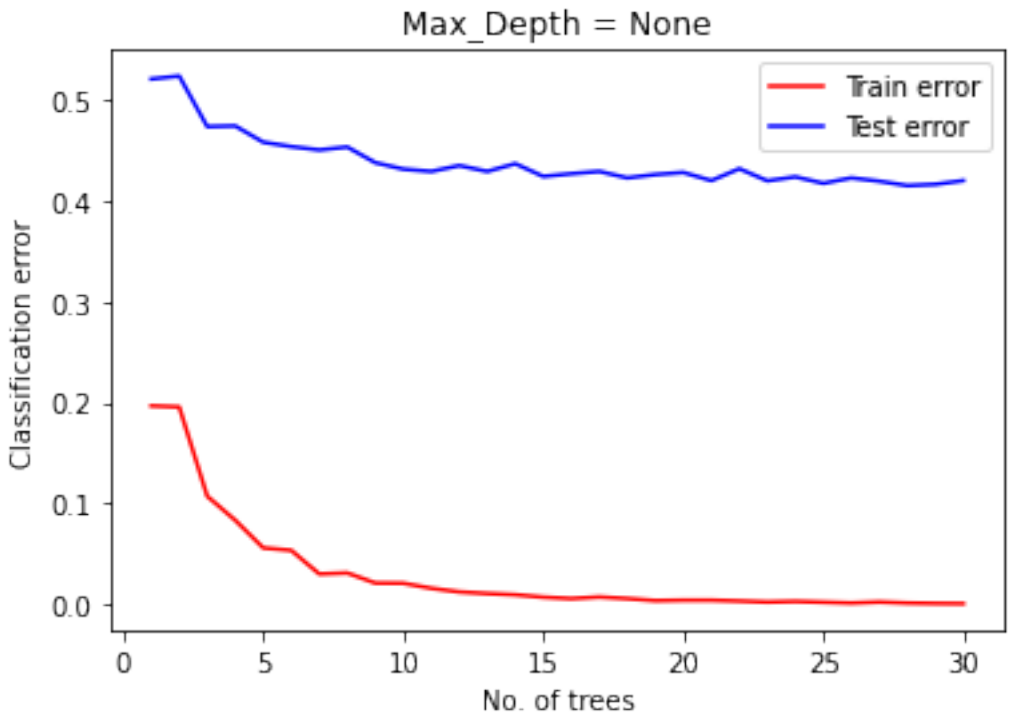


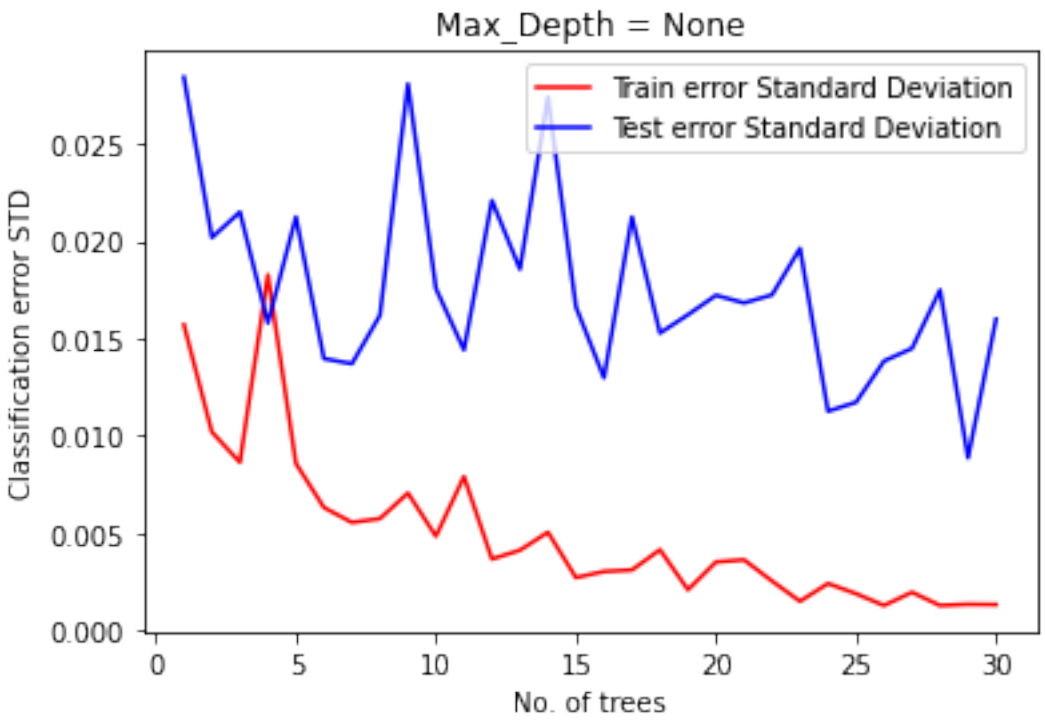
--> Minimun test error rate for Max_Depth:1 == 0.5991735537190083 +/- 0.051669418843821174


Max_Depth = 5
Train error
Test error
Classification error
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0
5
10
15
20
25
30
No. of trees


Max_Depth = 5
Train error Standard Deviation
Test error Standard Deviation
Classification error STD
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0
5
10
15
20
25
30
No. of trees

```
--> Minimun test error rate for Max_Depth:5  == 0.40082644628099173 +/-
0.009734891726768391
```





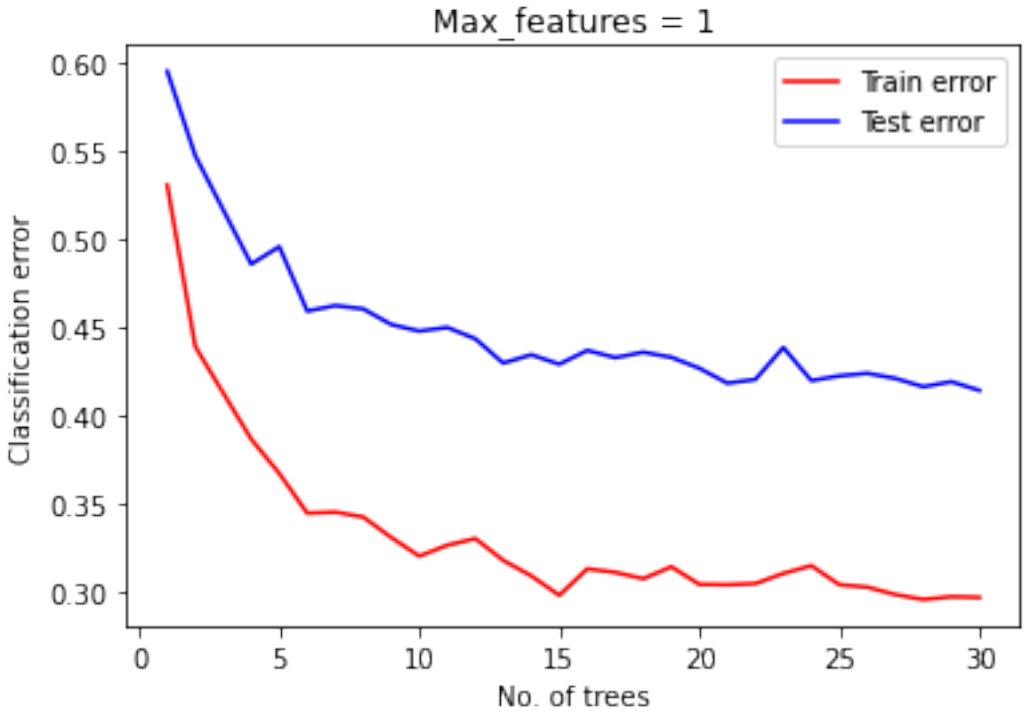

```
--> Minimun test error rate for Max_Depth:None  == 0.41466942148760333 +/-
0.017484027271107466
```

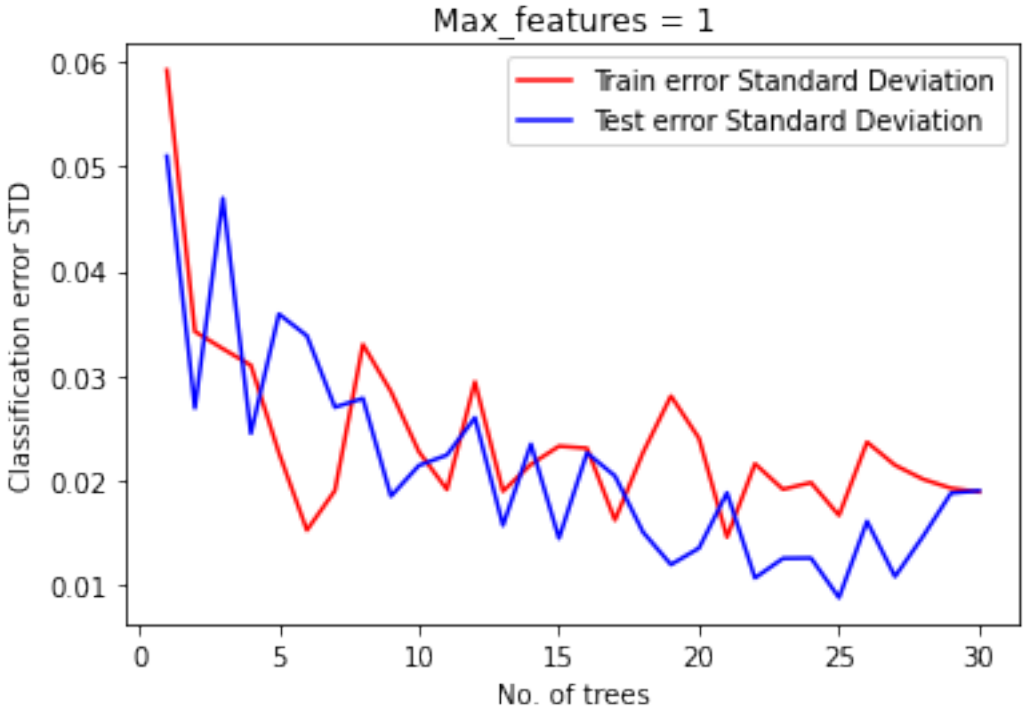
9 Q3: b) iii)

10 2) changing Max Depth: We get the best error rate when Max_Depth =5. We get a very high error rate when Max_Depth=1. Also When Max_depth=None, error rate doesn't improve rather worsens when compared to Max_Depth=5 setting.

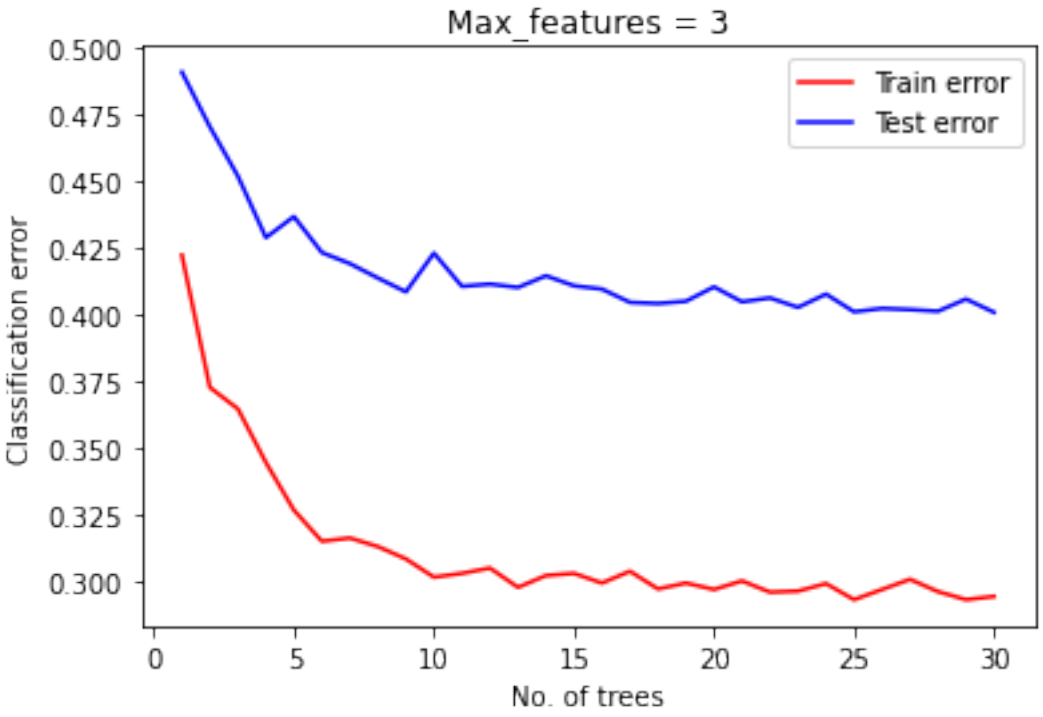
[95]:
```
max_features_=[1,3,8]
estimators=np.arange(1,31,1)
bag_random_state=np.linspace(10,1000,10,dtype=int)

for feat in max_features_:
    mean_train_err_randomforest=[]
    mean_test_err_randomforest=[]
```

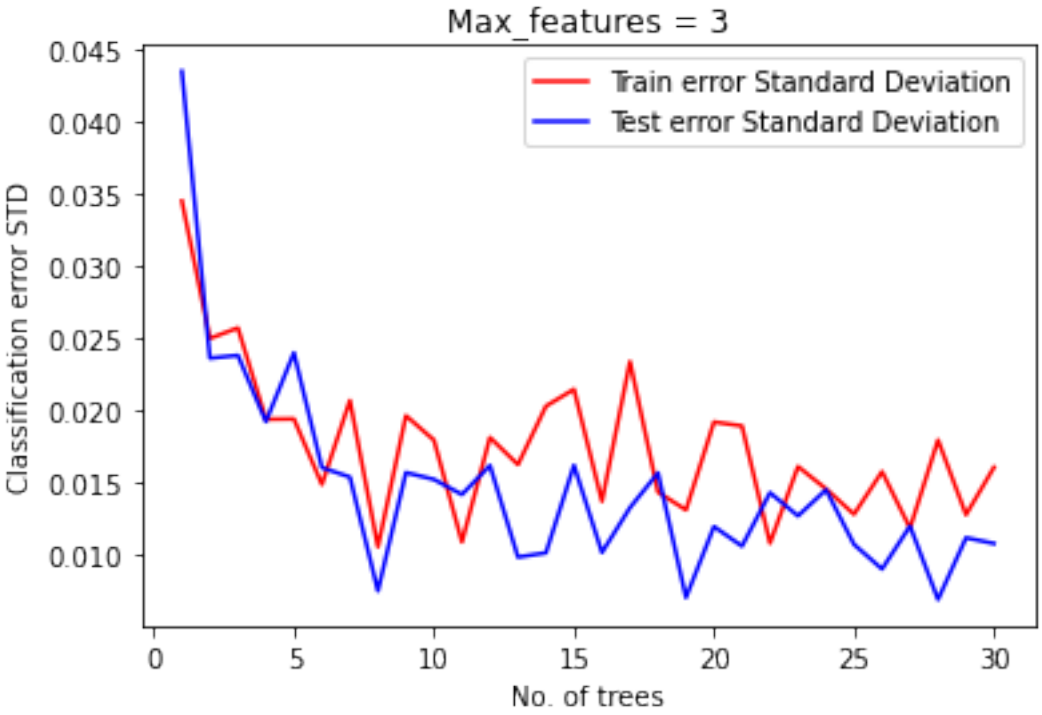
```
    std_train_err_randomforest=[]
    std_test_err_randomforest=[]
    for b in estimators:
        train_error_array=[]
        test_error_array=[]
        for s in bag_random_state:
            ␣
↪train_err,test_err=classifier3_randomforest(Xtrain,Ytrain_enc,Xtest,Ytest_enc,0.
↪5,b,5,feat,s)
            train_error_array.append(train_err)
            test_error_array.append(test_err)
        mean_train_error=np.mean(train_error_array)
        std_train_error=np.std(train_error_array)
        mean_test_error=np.mean(test_error_array)
        std_test_error=np.std(test_error_array)
        mean_train_err_randomforest.append(mean_train_error)
        std_train_err_randomforest.append(std_train_error)
        mean_test_err_randomforest.append(mean_test_error)
        std_test_err_randomforest.append(std_test_error)
    plt.figure()
    plt.plot(estimators,mean_train_err_randomforest,color='red',label='Train␣
↪error')
    plt.plot(estimators,mean_test_err_randomforest,color='blue',label='Test␣
↪error')
    plt.xlabel('No. of trees')
    plt.ylabel('Classification error')
    plt.legend()
    plt.title('Max_features = {}'.format(feat))

    plt.figure()
    plt.plot(estimators,std_train_err_randomforest,color='red',label='Train␣
↪error Standard Deviation')
    plt.plot(estimators,std_test_err_randomforest,color='blue',label='Test␣
↪error Standard Deviation')
    plt.xlabel('No. of trees')
    plt.ylabel('Classification error STD')
    plt.legend()
    plt.title('Max_features = {}'.format(feat))
    plt.show()
    best_err_idx=np.argmin(mean_test_err_randomforest)
    print("--> Minimun test error rate for Max_features:{}  == {} +/- {} ".
↪format(feat,mean_test_err_randomforest[best_err_idx],std_test_err_randomforest[best_err_idx
    print("            ")
    print("            ")
    print("            ")
```


Max_features = 1
Train error
Test error
Classification error
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
30
No. of trees

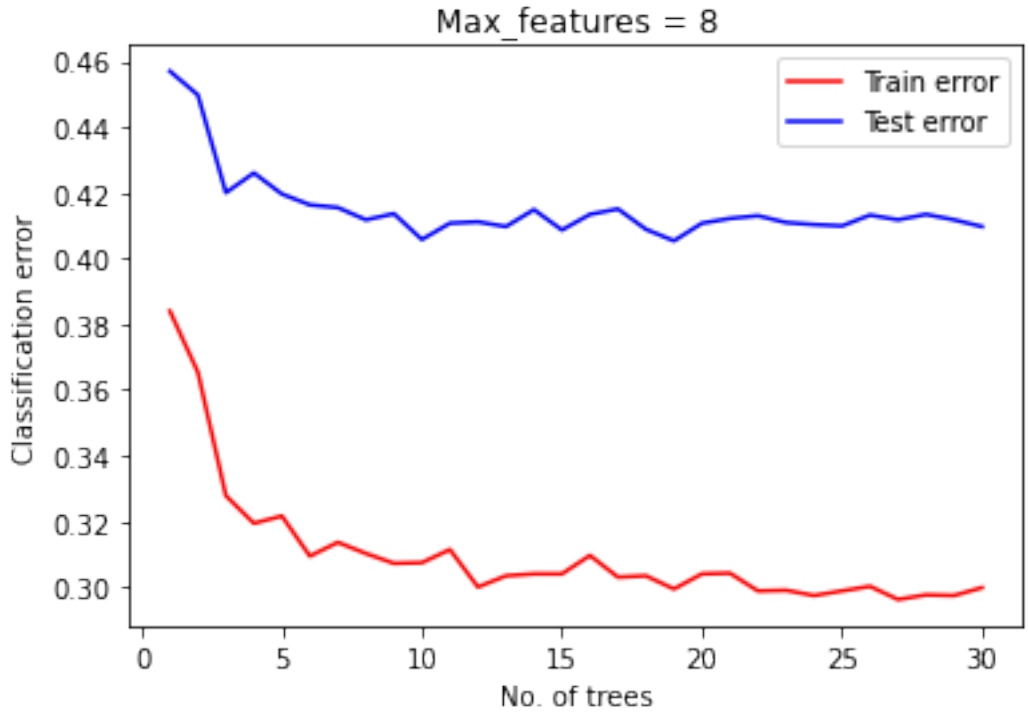

Max_features = 1
Train error Standard Deviation
Test error Standard Deviation
Classification error STD
0.06
0.05
0.04
0.03
0.02
0.01
0
5
10
15
20
25
30
No. of trees

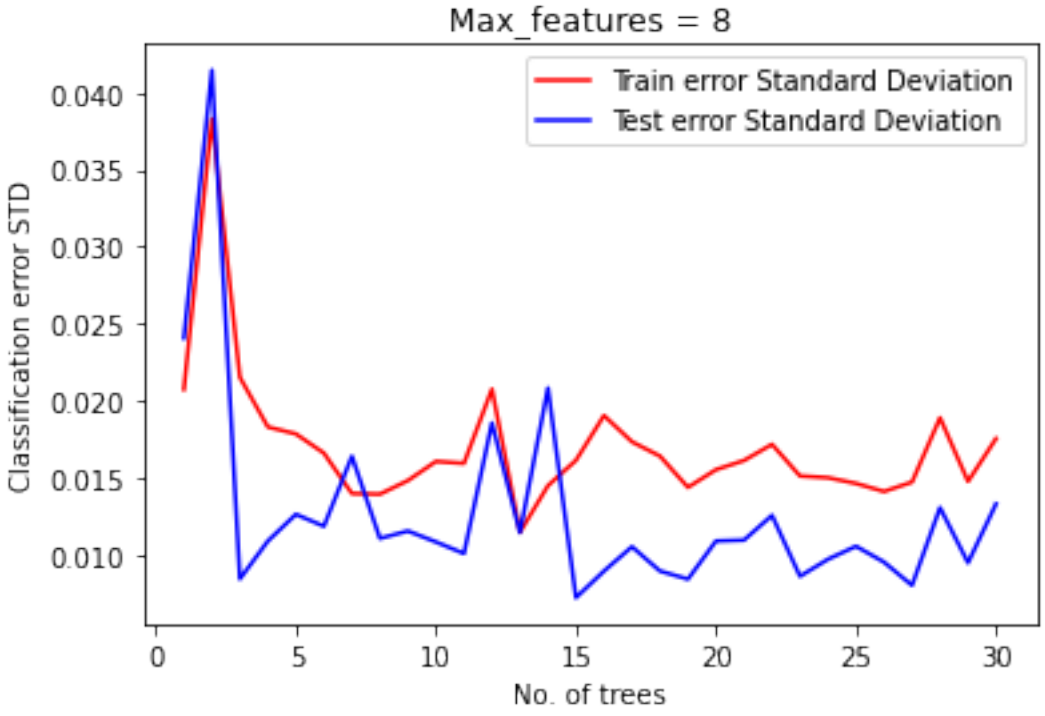
```
--> Minimun test error rate for Max_features:1  == 0.4144628099173554 +/-
0.019030708991189176
```






```
--> Minimun test error rate for Max_features:3  == 0.4008264462809918 +/-
0.010775541165624215
```


Max_features = 8
Train error
Test error
Classification error
0.46
0.44
0.42
0.40
0.38
0.36
0.34
0.32
0.30
0
5
10
15
20
25
30
No. of trees


Max_features = 8
Train error Standard Deviation
Test error Standard Deviation
Classification error STD
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0
5
10
15
20
25
30
No. of trees

```
--> Minimun test error rate for Max_features:8  == 0.4053719008264463 +/-
0.008407847086518932
```

Q3: b) iii)

3) changing Max_features: We get the best error rate when Max_feature = 3. There is not much of a difference seen with different settings of Max_features as seen in above plots.The only visible aspect that can be seen is, with increase in Max_features, the Test error curve steadies more quickly.

Q3: b) iv)

Based on comparision in (i) & (iii), we see that the best settings for setting up the random forest are as follows:

–> Bag_size = 2/3

–> Max_depth = 5

–> Max_features = 3

–> Minimun test error rate for the above settings for Random Forest == 0.38987603305785123 +/- 0.007622258985825088

–> Mean test error and STD for trivial CLF 1 = 0.7771 +/- 0.03

–> Test error for trivial CLF 2 = 0.6901

–> Yes, we see the Random Forest Classifer has learned from the data as it's quite evident that the test error has reduced when appyly Random forest classifier with the above settings on the given dataset.

[ ]: