In [110…
```
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler , LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings('ignore')
```

In [111…
```
df=pd.read_csv("NVDA_weekly_return_volatility.csv")
year=df['Year'].unique()
Q1_label=[]
yearly_mean=df.groupby('Year')['mean_return'].mean().values
for i in range(len(year)):
    for j in range(len(df)):
        if df['Year'][j]==year[i] and df["mean_return"][j]>yearly_mean[i]:
            Q1_label.append('green')
        elif df['Year'][j]==year[i]:
            Q1_label.append('red')
df['label']=Q1_label
Q1_X=df[df["Year"]==2017][["mean_return","volatility"]]
Q1_y=df[df["Year"]==2017]["label"]
Q2_X=df.loc[df["Year"]==2018][["mean_return","volatility"]]
Q2_y=df.loc[df["Year"]==2018]["label"]
```

1. take N = 1,...,10 and d = 1,2,...,5. For each value of N and d construct a random tree classifier (use "entropy" as splitting criteria - this is the default) use your year 1 labels as training set and compute the error rate for year 2. Plot your error rates and find the best combination of N and d.

In [112…
```
from sklearn.ensemble import RandomForestClassifier
errorrate=[]
for i in range(1,11):
    for j in range(1,6):
        rf=RandomForestClassifier(n_estimators=i,max_depth=j)
        rf.fit(Q1_X,Q1_y)
        errorrate.append(1-accuracy_score(Q2_y,rf.predict(Q2_X)))
        print(" when n_estimators=",i,"max_depth=",j,",the accuracy=",1-accurac
#plot the error rate
plt.plot(range(1,11),errorrate[:10],label="max_depth=1")
plt.plot(range(1,11),errorrate[10:20],label="max_depth=2")
plt.plot(range(1,11),errorrate[20:30],label="max_depth=3")
plt.plot(range(1,11),errorrate[30:40],label="max_depth=4")
plt.plot(range(1,11),errorrate[40:50],label="max_depth=5")
plt.legend()
plt.xlabel("n_estimators")
plt.ylabel("error rate")
plt.show()
```

```
#the best n_estimators and max_depth
print("the best n_estimators and max_depth are",errorrate.index(min(errorrate))
```

```
  File "/var/folders/by/3jf2bh0x2ks63rycd0ctbzsh0000gn/T/ipykernel_81199/4277732179.py", line 21
    print("the best n_estimators and max_depth are",errorrate.index(min(errorrate))10+1,"and",errorrate.index(min(errorrate))%5+1)

^
SyntaxError: invalid syntax
```

1. using the optimal values from year 1, compute the confusion matrix for year 2

In [ ]:

```
rf=RandomForestClassifier(n_estimators=errorrate.index(min(errorrate))%10+1,max
rf.fit(Q1_X,Q1_y)
print("the confusion matrix for year 2 is")
print(confusion_matrix(Q2_y,rf.predict(Q2_X)))
```

```
the confusion matrix for year 2 is
[[28  1]
 [ 0 24]]
```

1. what is true positive rate and true negative rate for year 2?

In [ ]:

```
# what is true positive rate and true negative rate for year 2?
print("the true positive rate for year 2 is",confusion_matrix(Q2_y,rf.predict(Q
print("the true negative rate for year 2 is",confusion_matrix(Q2_y,rf.predict(Q
```

```
the true positive rate for year 2 is 0.9655172413793104
the true negative rate for year 2 is 1.0
```