# **a)** Training 12 decisions trees on the madelon dataset. Obtaining misclassification errors, then plotting the results on a graph and reporting the minimum in a table.

## Import our needed modules and locate our datasets.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

import os

os.chdir("C:/Users/rique/Downloads")
```

## Create our dataframes using the madelon dataset files.

Attempted with pandas but it led to alot of noise in the data. And plotted graph was a nearly perfect horizontal line.

## Use numpy to load the text

```
#train_data = pd.read_csv("madelon_train.data")
#train_labels = pd.read_csv("madelon_train.labels")
X = np.loadtxt("madelon_train.data")
Y = np.loadtxt("madelon_train.labels")

#valid_data = pd.read_csv("madelon_valid.data")
#valid_labels = pd.read_csv("madelon_valid.labels")
Xtest = np.loadtxt("madelon_valid.data")
Ytest = np.loadtxt("madelon_valid.labels")
```

## It's good to have some functions just to visualize our data and see whats in it, or what its about.

```
print(X)
#train_data.tail()
#train_data.shape
```

```
[[485. 477. 537. ... 479. 475. 496.]
 [483. 458. 460. ... 492. 510. 517.]
 [487. 542. 499. ... 489. 499. 498.]
 ...
 [480. 517. 631. ... 500. 523. 481.]
 [484. 481. 505. ... 473. 527. 485.]
 [474. 493. 469. ... 489. 516. 516.]]
```

## Train our Decision Trees using sklearn's DecisionTreeClassifier, with max depth of 12.

train using the fit(X,y) function

make our predictions via the predict(X) function

find the misclassification error by subtracting the accuracy_score(y, prediction) from 1

## Complete the predictions and error calculations for both the test and training sets.

```
train_misclass_errors = []
test_misclass_errors = []

maxDepth = np.arange(12)+1

for i in range(maxDepth.shape[0]):
    dt = DecisionTreeClassifier(criterion='entropy',
max_depth=maxDepth[i])

    dt.fit(X, Y)

    y_predict = dt.predict(X)
    yTest_predict = dt.predict(Xtest)

    train_error = 1 - accuracy_score(Y, y_predict)
    train_misclass_errors.append(train_error)
    print(f"train error: {train_error}")

    test_error = 1 - accuracy_score(Ytest, yTest_predict)
    test_misclass_errors.append(test_error)
    print(f"test error: {test_error}\n")

    #train_predictions = tree.predict(train_data)
    #train_error = 1 - accuracy_score(train_labels, train_predictions)
    #train_misclass_errors.append(train_error)
```

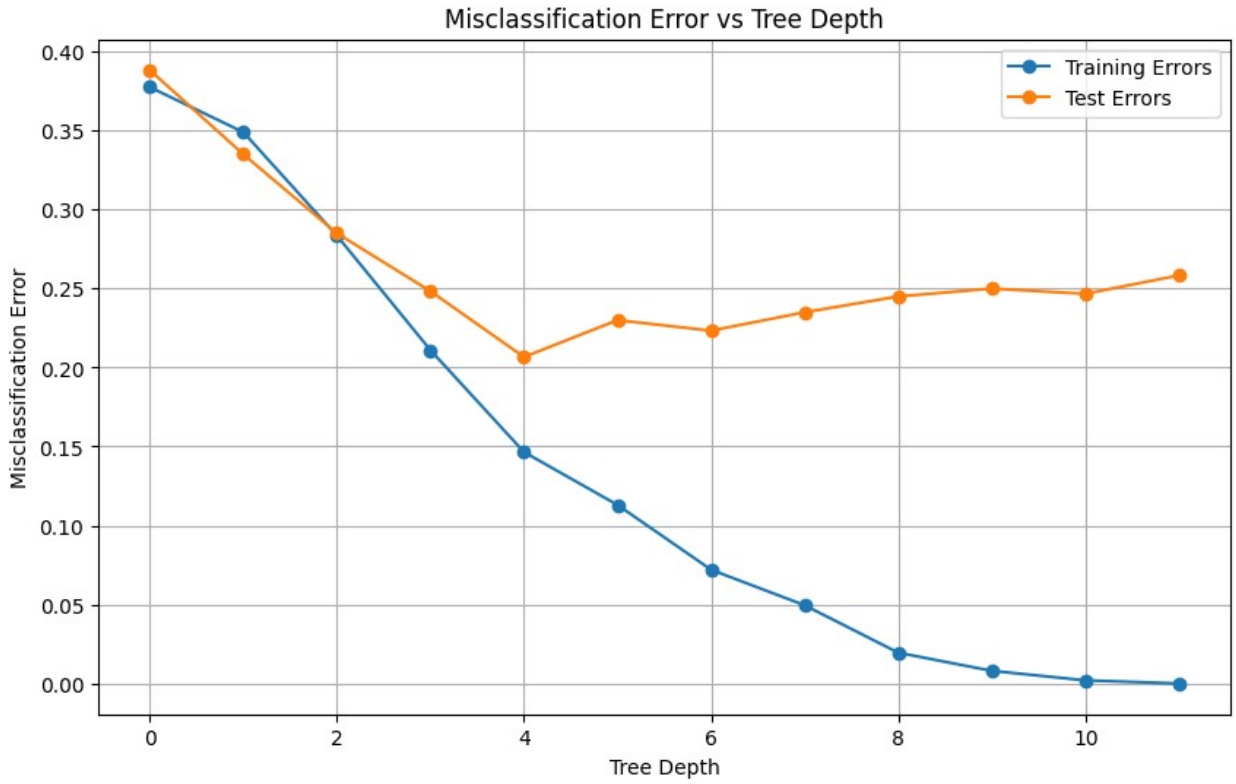
```
    #test_predictions = tree.predict(valid_data)
    #test_error = 1 - accuracy_score(valid_labels, test_predictions)
    #test_misclass_errors.append(test_error)
```

```
train error: 0.37749999999999995
test error: 0.3883333333333333

train error: 0.349
test error: 0.33499999999999996

train error: 0.2835
test error: 0.28500000000000003

train error: 0.21099999999999997
test error: 0.2483333333333333

train error: 0.14649999999999996
test error: 0.20666666666666667

train error: 0.11299999999999999
test error: 0.22999999999999998

train error: 0.07199999999999995
test error: 0.22333333333333338

train error: 0.04949999999999999
test error: 0.235

train error: 0.019499999999999962
test error: 0.245

train error: 0.008000000000000007
test error: 0.25

train error: 0.0020000000000000018
test error: 0.2466666666666667

train error: 0.0
test error: 0.2583333333333333
```

## Plot our results in a fancy graph. Error v Depth

```
tree_depths = range(0,12)
plt.figure(figsize=(10,6))
plt.plot(tree_depths, train_misclass_errors, marker='o',
label="Training Errors")
plt.plot(tree_depths, test_misclass_errors, marker='o', label="Test
Errors")
```

```
plt.xlabel('Tree Depth')
plt.ylabel('Misclassification Error')
plt.title('Misclassification Error vs Tree Depth')
plt.legend()
plt.grid(True)
plt.show()
```




## Showcase the minimum error by making a table

```
table_results = pd.DataFrame({
    'Tree_Depth': tree_depths,
    'Training Error': train_misclass_errors,
    'Testing Error': test_misclass_errors
})

train_min_error_row = table_results.loc[table_results['Training
Error'].idxmin()]
test_min_error_row = table_results.loc[table_results['Testing
Error'].idxmin()]
print(table_results)
print("\nTraining Minimum Error: ")
print(train_min_error_row)
print("\nTesting Minimum Error: ")
print(test_min_error_row)
```

```
    Tree_Depth  Training Error  Testing Error
0            0          0.3775       0.388333
1            1          0.3490       0.335000
2            2          0.2835       0.285000
3            3          0.2110       0.248333
4            4          0.1465       0.206667
5            5          0.1130       0.230000
6            6          0.0720       0.223333
7            7          0.0495       0.235000
8            8          0.0195       0.245000
9            9          0.0080       0.250000
10          10          0.0020       0.246667
11          11          0.0000       0.258333

Training Minimum Error:
Tree_Depth        11.000000
Training Error     0.000000
Testing Error      0.258333
Name: 11, dtype: float64

Testing Minimum Error:
Tree_Depth        4.000000
Training Error    0.146500
Testing Error     0.206667
Name: 4, dtype: float64
```

# b) Training 12 decisions trees on the satimage dataset. Obtaining misclassification errors, then plotting the results on a graph and reporting the minimum in a table. (exact same thing as a, just different dataset)

```
X = np.loadtxt("X.dat")
Y = np.loadtxt("Y.dat")

Xtest = np.loadtxt("Xtest.dat")
Ytest = np.loadtxt("Ytest.dat")

train_misclass_errors = []
test_misclass_errors = []

maxDepth = np.arange(12)+1

for i in range(maxDepth.shape[0]):
```

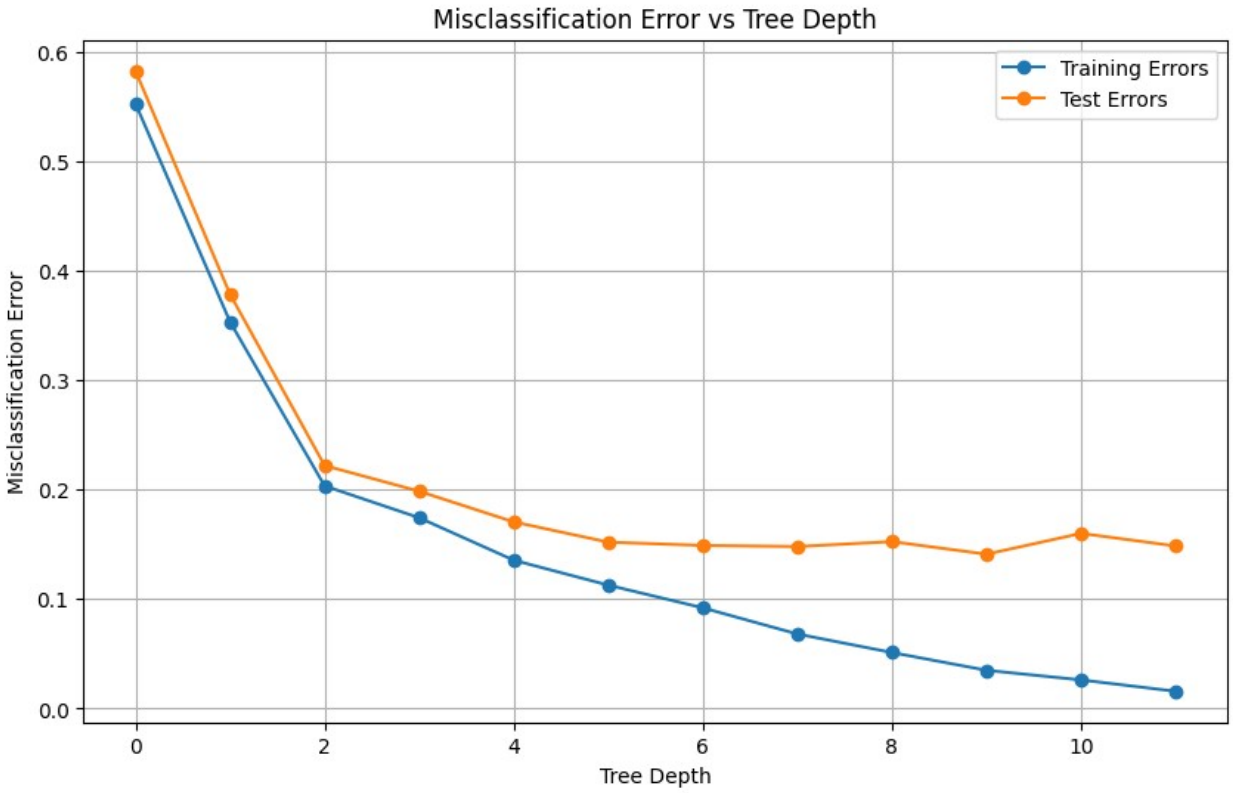
```
    dt = DecisionTreeClassifier(criterion='entropy',
max_depth=maxDepth[i])

    dt.fit(X, Y)

    y_predict = dt.predict(X)
    yTest_predict = dt.predict(Xtest)

    train_error = 1 - accuracy_score(Y, y_predict)
    train_misclass_errors.append(train_error)
    print(f"train error: {train_error}")

    test_error = 1 - accuracy_score(Ytest, yTest_predict)
    test_misclass_errors.append(test_error)
    print(f"test error: {test_error}\n")
```

```
train error: 0.5526493799323562
test error: 0.5825

train error: 0.35332581736189406
test error: 0.37849999999999995

train error: 0.20338218714768885
test error: 0.22199999999999998

train error: 0.17429537767756487
test error: 0.1985

train error: 0.13551296505073285
test error: 0.17049999999999998

train error: 0.112739571589628
test error: 0.15200000000000002

train error: 0.0919954904171364
test error: 0.14900000000000002

train error: 0.06809470124013528
test error: 0.14800000000000002

train error: 0.051183765501691125
test error: 0.15249999999999997

train error: 0.03494926719278468
test error: 0.14100000000000001

train error: 0.026155580608793638
test error: 0.16000000000000003

train error: 0.015783540022547893
```

```
test error: 0.14849999999999997
```

## Plot the Graph

```
tree_depths = range(0,12)
plt.figure(figsize=(10,6))
plt.plot(tree_depths, train_misclass_errors, marker='o',
label="Training Errors")
plt.plot(tree_depths, test_misclass_errors, marker='o', label="Test
Errors")
plt.xlabel('Tree Depth')
plt.ylabel('Misclassification Error')
plt.title('Misclassification Error vs Tree Depth')
plt.legend()
plt.grid(True)
plt.show()
```




## Showcase the errors in a table

```
table_results = pd.DataFrame({
    'Tree_Depth': tree_depths,
    'Training Error': train_misclass_errors,
    'Testing Error': test_misclass_errors
```

```
})

train_min_error_row = table_results.loc[table_results['Training
Error'].idxmin()]
test_min_error_row = table_results.loc[table_results['Testing
Error'].idxmin()]
print(table_results)
print("\nTraining Minimum Error: ")
print(train_min_error_row)
print("\nTesting Minimum Error: ")
print(test_min_error_row)
```

```
    Tree_Depth  Training Error  Testing Error
0            0        0.552649         0.5825
1            1        0.353326         0.3785
2            2        0.203382         0.2220
3            3        0.174295         0.1985
4            4        0.135513         0.1705
5            5        0.112740         0.1520
6            6        0.091995         0.1490
7            7        0.068095         0.1480
8            8        0.051184         0.1525
9            9        0.034949         0.1410
10          10        0.026156         0.1600
11          11        0.015784         0.1485

Training Minimum Error:
Tree_Depth        11.000000
Training Error     0.015784
Testing Error      0.148500
Name: 11, dtype: float64

Testing Minimum Error:
Tree_Depth        9.000000
Training Error    0.034949
Testing Error     0.141000
Name: 9, dtype: float64
```

# c) Using the madelon dataset, for each k {3,10,30,100,300} train Random Forest with these k values. Obtain the misclass errors, plot results and report the errors in a table.

We need to import a new module for this.

```
from sklearn.ensemble import RandomForestClassifier
```

Use our given k_values, train our RandomForestClassifier with sqrt(500) which just requires max_features to be a certain value

and then find the rest of our needed info, predictions, misclass error, etc.....

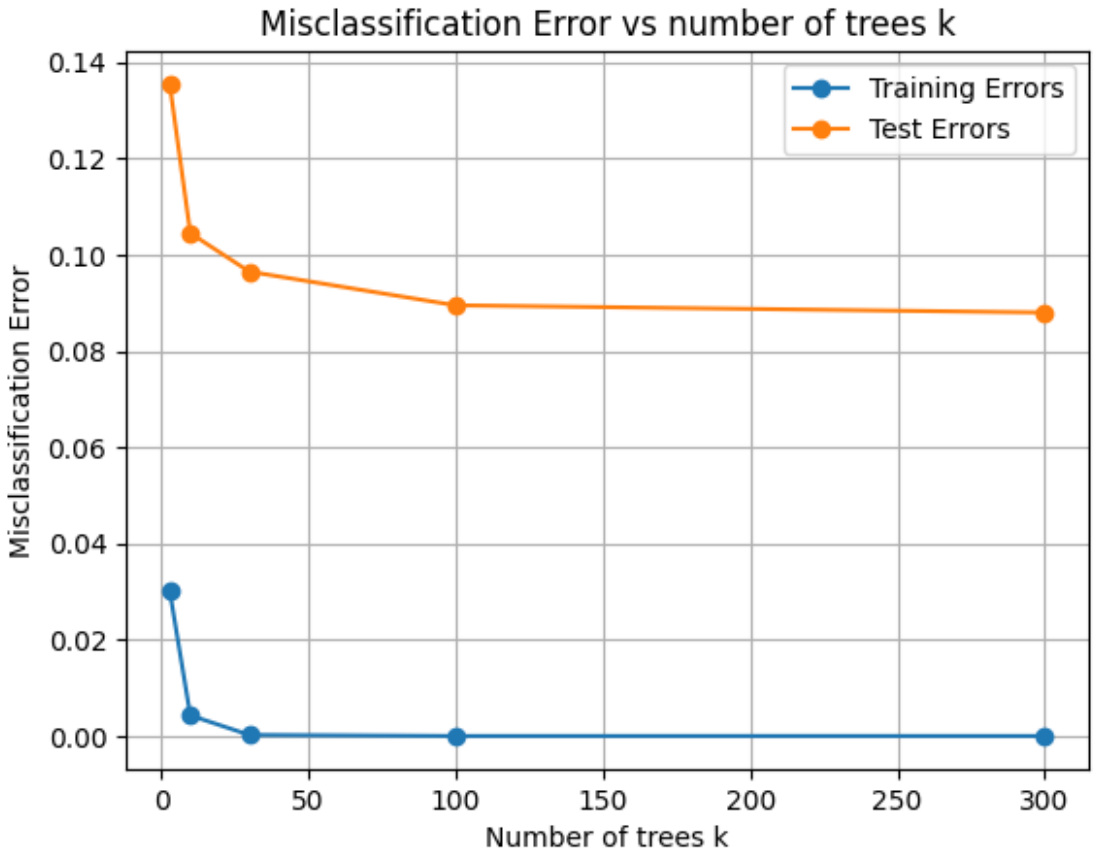
```
rf_train_misclass_errors = []
rf_test_misclass_errors = []

k_values = np.array([3,10,30,100,300])

for i in range(k_values.shape[0]):

    rf = RandomForestClassifier(n_estimators=k_values[i],
criterion='entropy', max_features='sqrt', random_state=1000)
    rf.fit(X, Y)

    rf_y_predict = rf.predict(X)
    rf_train_error = 1 - accuracy_score(Y, rf_y_predict)
    rf_train_misclass_errors.append(rf_train_error)
    print(f"train error: {rf_train_error}")

    rf_yTest_predict = rf.predict(Xtest)
    rf_test_error = 1 - accuracy_score(Ytest, rf_yTest_predict)
    rf_test_misclass_errors.append(rf_test_error)
    print(f"test error: {rf_test_error}")
```

```
train error: 0.03021420518602025
test error: 0.13549999999999995
train error: 0.004284103720405885
test error: 0.10450000000000004
train error: 0.00022547914317927464
test error: 0.09650000000000003
train error: 0.0
```

```
test error: 0.08950000000000002
train error: 0.0
test error: 0.08799999999999997
```

## Plot the Graph

```
plt.plot(k_values, rf_train_misclass_errors, marker='o',
label="Training Errors")
plt.plot(k_values, rf_test_misclass_errors, marker='o', label="Test
Errors")
plt.xlabel('Number of trees k')
plt.ylabel('Misclassification Error')
plt.title('Misclassification Error vs number of trees k')
plt.legend()
plt.grid(True)
plt.show()
```




## Showcase the errors in a table

```
table_results = pd.DataFrame({
    'K Values': k_values,
    'Training Error': rf_train_misclass_errors,
```

```
    'Testing Error': rf_test_misclass_errors
})

train_min_error_row = table_results.loc[table_results['Training
Error'].idxmin()]
test_min_error_row = table_results.loc[table_results['Testing
Error'].idxmin()]
print(table_results)
print("\nTraining Minimum Error: ")
print(train_min_error_row)
print("\nTesting Minimum Error: ")
print(test_min_error_row)
```

```
   K Values  Training Error  Testing Error
0         3        0.030214         0.1355
1        10        0.004284         0.1045
2        30        0.000225         0.0965
3       100        0.000000         0.0895
4       300        0.000000         0.0880

Training Minimum Error:
K Values          100.0000
Training Error      0.0000
Testing Error       0.0895
Name: 3, dtype: float64

Testing Minimum Error:
K Values          300.000
Training Error      0.000
Testing Error       0.088
Name: 4, dtype: float64
```

# d) essentially just copy&paste c) .... just change max features value from sqrt to log2

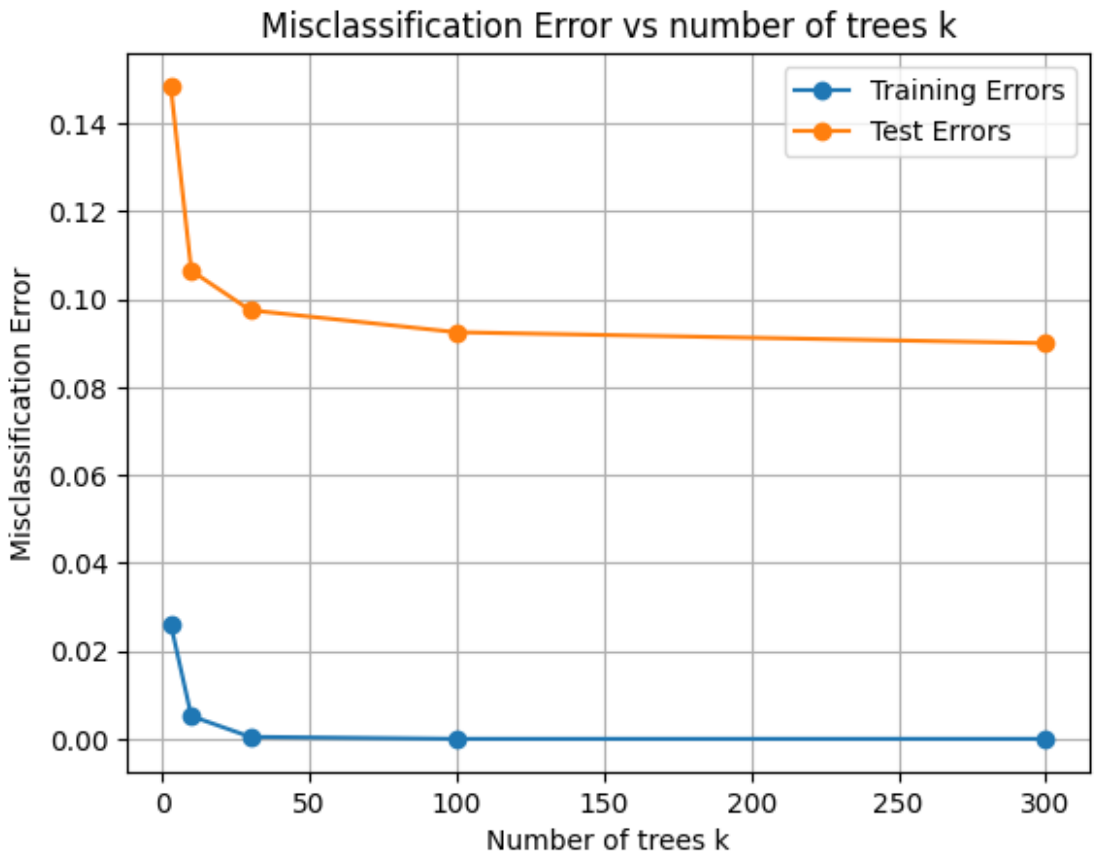
```
rf_train_misclass_errors = []
rf_test_misclass_errors = []

k_values = np.array([3,10,30,100,300])

for i in range(k_values.shape[0]):

    rf = RandomForestClassifier(n_estimators=k_values[i],
criterion='entropy', max_features='log2', random_state=1000)
    rf.fit(X, Y)

    rf_y_predict = rf.predict(X)
```

```
    rf_train_error = 1 - accuracy_score(Y, rf_y_predict)
    rf_train_misclass_errors.append(rf_train_error)
    print(f"train error: {rf_train_error}")

    rf_yTest_predict = rf.predict(Xtest)
    rf_test_error = 1 - accuracy_score(Ytest, rf_yTest_predict)
    rf_test_misclass_errors.append(rf_test_error)
    print(f"test error: {rf_test_error}")
```

```
train error: 0.026155580608793638
test error: 0.14849999999999997
train error: 0.005186020293122873
test error: 0.10650000000000004
train error: 0.0004509582863585493
test error: 0.09750000000000003
train error: 0.0
test error: 0.09250000000000003
train error: 0.0
test error: 0.08999999999999997
```

## Plot the graph

```
plt.plot(k_values, rf_train_misclass_errors, marker='o',
label="Training Errors")
plt.plot(k_values, rf_test_misclass_errors, marker='o', label="Test
Errors")
plt.xlabel('Number of trees k')
plt.ylabel('Misclassification Error')
plt.title('Misclassification Error vs number of trees k')
plt.legend()
plt.grid(True)
plt.show()
```




## Showcase the errors in a table

```
table_results = pd.DataFrame({
    'K Values': k_values,
    'Training Error': rf_train_misclass_errors,
    'Testing Error': rf_test_misclass_errors
})

train_min_error_row = table_results.loc[table_results['Training
Error'].idxmin()]
test_min_error_row = table_results.loc[table_results['Testing
Error'].idxmin()]
print(table_results)
print("\nTraining Minimum Error: ")
print(train_min_error_row)
print("\nTesting Minimum Error: ")
print(test_min_error_row)
```

```
   K Values  Training Error  Testing Error
0         3        0.026156         0.1485
1        10        0.005186         0.1065
2        30        0.000451         0.0975
3       100        0.000000         0.0925
```

```
4       300         0.000000          0.0900

Training Minimum Error:
K Values          100.0000
Training Error      0.0000
Testing Error       0.0925
Name: 3, dtype: float64

Testing Minimum Error:
K Values          300.00
Training Error      0.00
Testing Error       0.09
Name: 4, dtype: float64
```

# e) essentially just copy&paste c) and d) .... just change max features value equal None to ensure the split attribute at each node is chosen from all 500 features.

```
rf_train_misclass_errors = []
rf_test_misclass_errors = []

k_values = np.array([3,10,30,100,300])

for i in range(k_values.shape[0]):

    rf = RandomForestClassifier(n_estimators=k_values[i],
criterion='entropy', max_features=None, random_state=1000)
    rf.fit(X, Y)

    rf_y_predict = rf.predict(X)
    rf_train_error = 1 - accuracy_score(Y, rf_y_predict)
    rf_train_misclass_errors.append(rf_train_error)
    print(f"train error: {rf_train_error}")

    rf_yTest_predict = rf.predict(Xtest)
    rf_test_error = 1 - accuracy_score(Ytest, rf_yTest_predict)
    rf_test_misclass_errors.append(rf_test_error)
    print(f"test error: {rf_test_error}")
```

```
train error: 0.0248027057497181
test error: 0.127
train error: 0.005186020293122873
test error: 0.10699999999999998
train error: 0.0004509582863585493
```

```
test error: 0.10150000000000003
train error: 0.0
test error: 0.10050000000000003
train error: 0.0
test error: 0.09699999999999998
```

## Plot the graph

```
plt.plot(k_values, rf_train_misclass_errors, marker='o',
label="Training Errors")
plt.plot(k_values, rf_test_misclass_errors, marker='o', label="Test
Errors")
plt.xlabel('Number of trees k')
plt.ylabel('Misclassification Error')
plt.title('Misclassification Error vs number of trees k')
plt.legend()
plt.grid(True)
plt.show()
```

## Showcase the errors in a table

```
table_results = pd.DataFrame({
    'K Values': k_values,
    'Training Error': rf_train_misclass_errors,
    'Testing Error': rf_test_misclass_errors
})

train_min_error_row = table_results.loc[table_results['Training
Error'].idxmin()]
test_min_error_row = table_results.loc[table_results['Testing
Error'].idxmin()]
print(table_results)
print("\nTraining Minimum Error: ")
print(train_min_error_row)
print("\nTesting Minimum Error: ")
print(test_min_error_row)
```

```
   K Values  Training Error  Testing Error
0         3        0.024803         0.1270
1        10        0.005186         0.1070
2        30        0.000451         0.1015
3       100        0.000000         0.1005
4       300        0.000000         0.0970

Training Minimum Error:
K Values          100.0000
Training Error      0.0000
Testing Error       0.1005
Name: 3, dtype: float64

Testing Minimum Error:
K Values          300.000
Training Error      0.000
Testing Error       0.097
Name: 4, dtype: float64
```