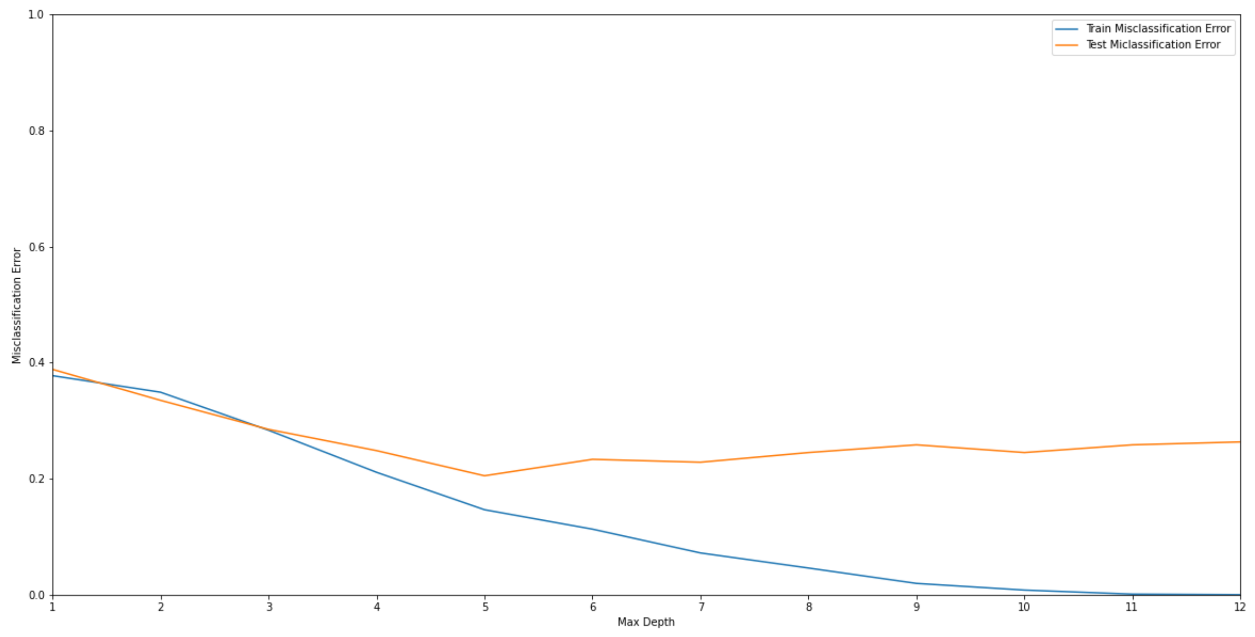
Abelardo Riojas
Jeret McCoy
Gustavo Flores
STA 5635 Homework 1
Professor Barbu, Fall 2022

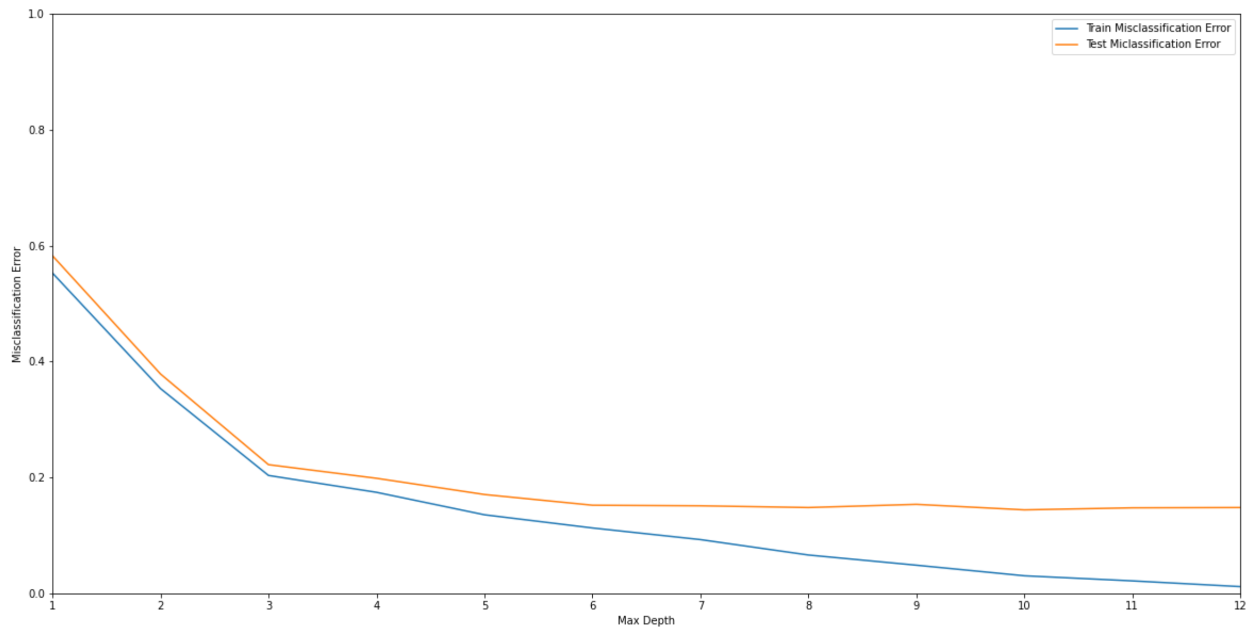
Homework 1 Report

a) On the madelon dataset, train decision trees of maximum depth 1, 2, ... up to 12, for a total of 12 decision trees.Use the trained trees to predict the class labels on the training and test sets, and obtain the training and test misclassification errors. Plot on the same graph the training and test misclassification errors vs tree depth (or log2 of nodes) as two separate curves. Report in a table the minimum test error and the tree depth (number of nodes or splits) for which the minimum was attained. (2 points)



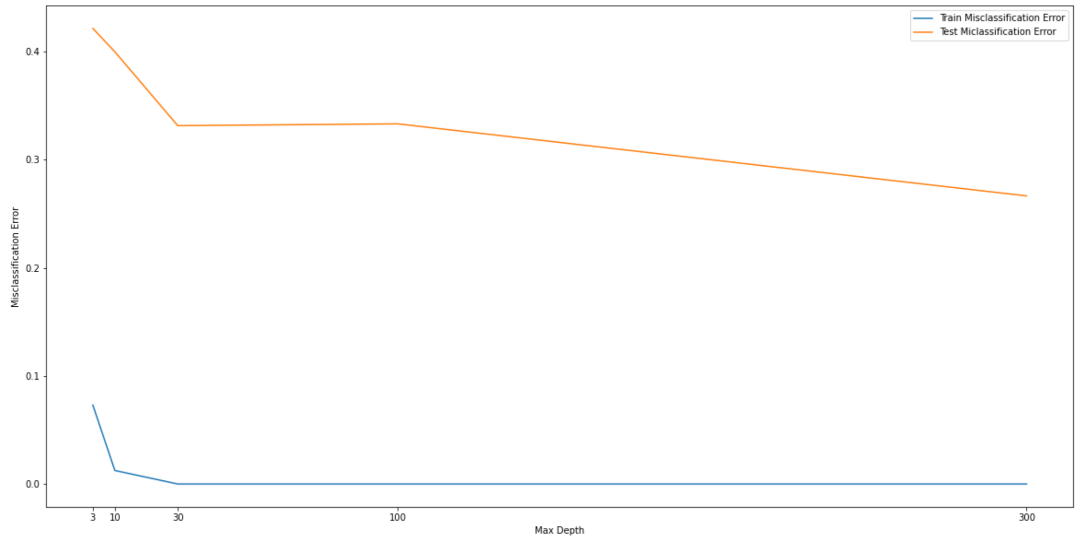
Minimum Test Error	.205
Tree Depth	5

b) Repeat point a) on the satimage dataset.



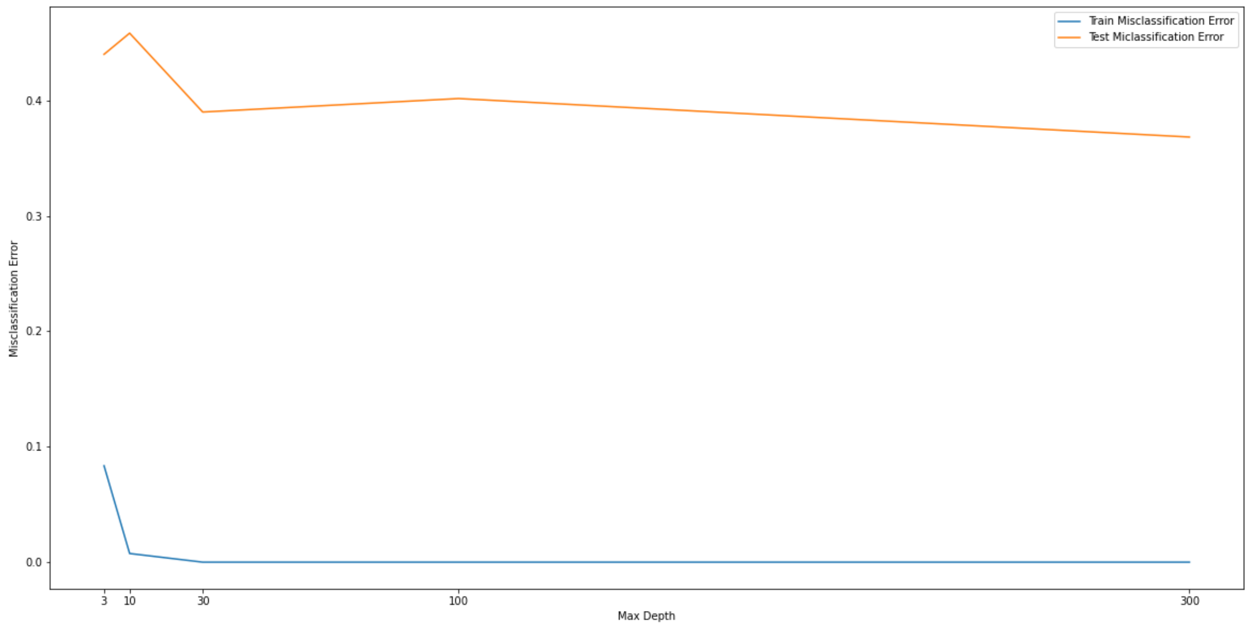
Minimum Test Error	.144
Tree Depth	10

c) On the madelon dataset, for each of k ∈ {3, 10, 30, 100, 300} train a random forest with k trees where the split attribute at each node is chosen from a random subset of ~ √500 features. Use the trained trees to predict the class labels on the training and test sets, and obtain the training and test misclassification errors. Plot on the same graph the training and test errors vs number of trees k as two separate curves. Report the training and test misclassification errors in a table.



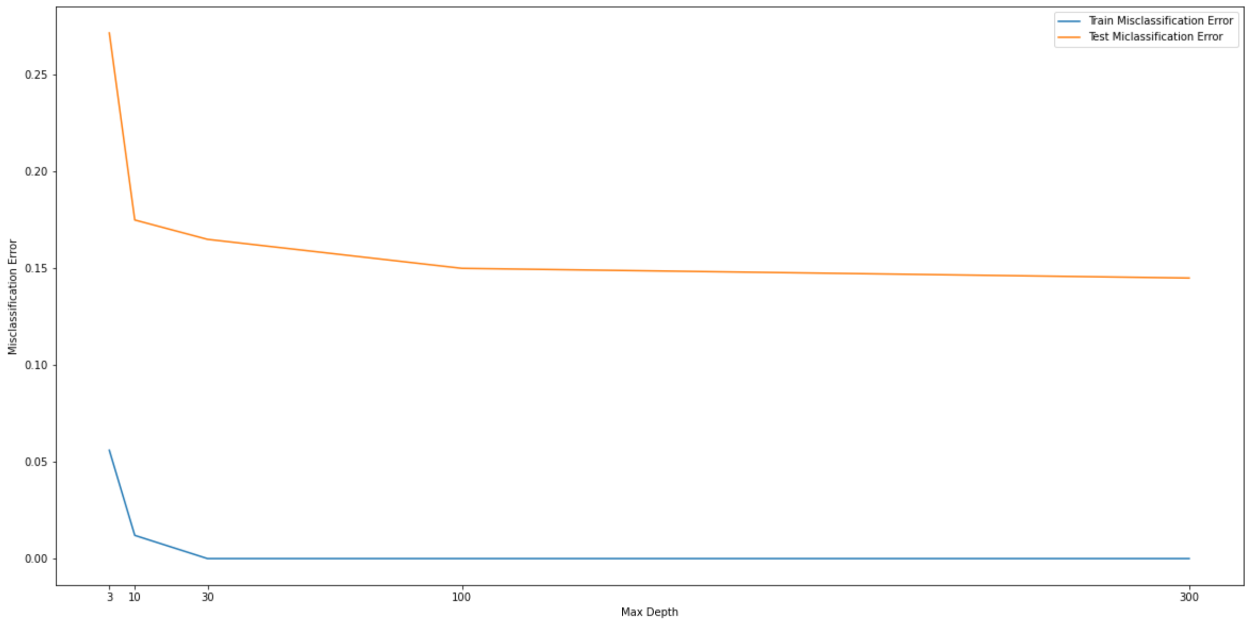
Number of trees	Training Misclassification Error	Test Misclassification Error
3	0.073	0.4216666666666667
10	0.0125	0.4
30	0	0.33166666666666667
100	0	0.33333333333333334
300	0	0.26666666666666667

d) Repeat point c) on the madelon dataset where the split attribute at each node is chosen from a random subset of ln(500) ~ 6 features.



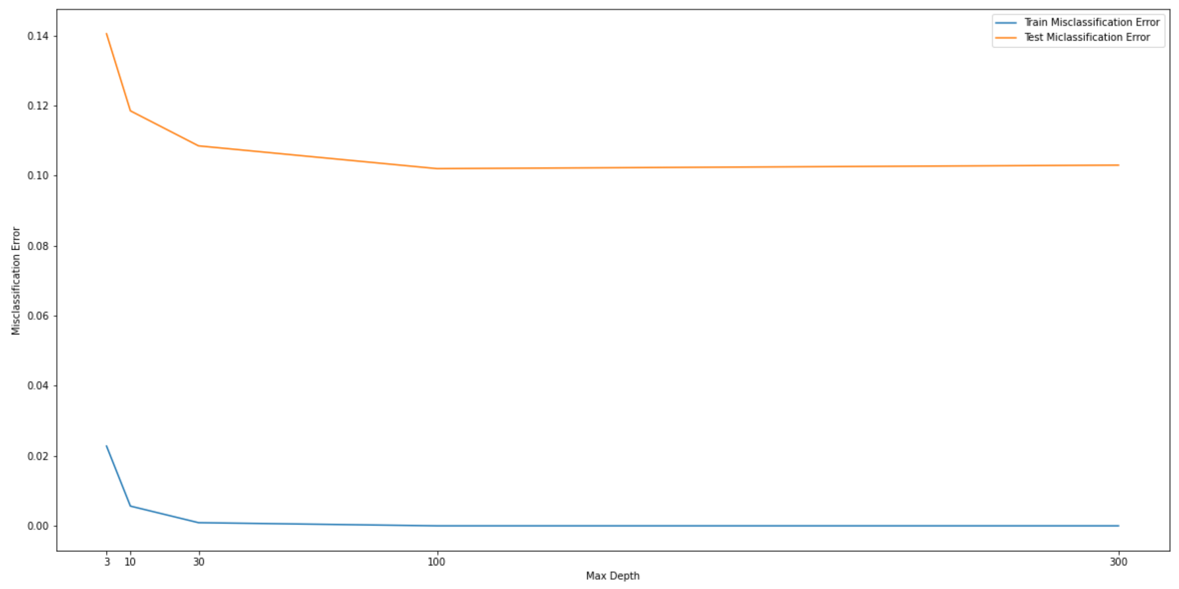
Number of trees	Training Misclassification Error	Test Misclassification Error
3	0.0835	0.44
10	0.0075	0.4583333333333333
30	0	0.39
100	0	0.40166666666666667
300	0	0.36833333333333335

e) Repeat point c) on the madelon dataset where the split attribute at each node is chosen from all 500 features.



Number of trees	Training Misclassification Error	Test Misclassification Error
3	0.056	0.27166666666666667
10	0.012	0.175
30	0	0.165
100	0	0.15
300	0	0.145

f) Repeat point c) on the satimage dataset where the split attribute at each node is chosen from all 36 features. (2 points)



Number of trees	Training Misclassification Error	Test Misclassification Error
3	0.022773393461104848	0.1405
10	0.005636978579481398	0.1185
30	0.0009019165727170237	0.1085
100	0	0.102
300	0	0.103

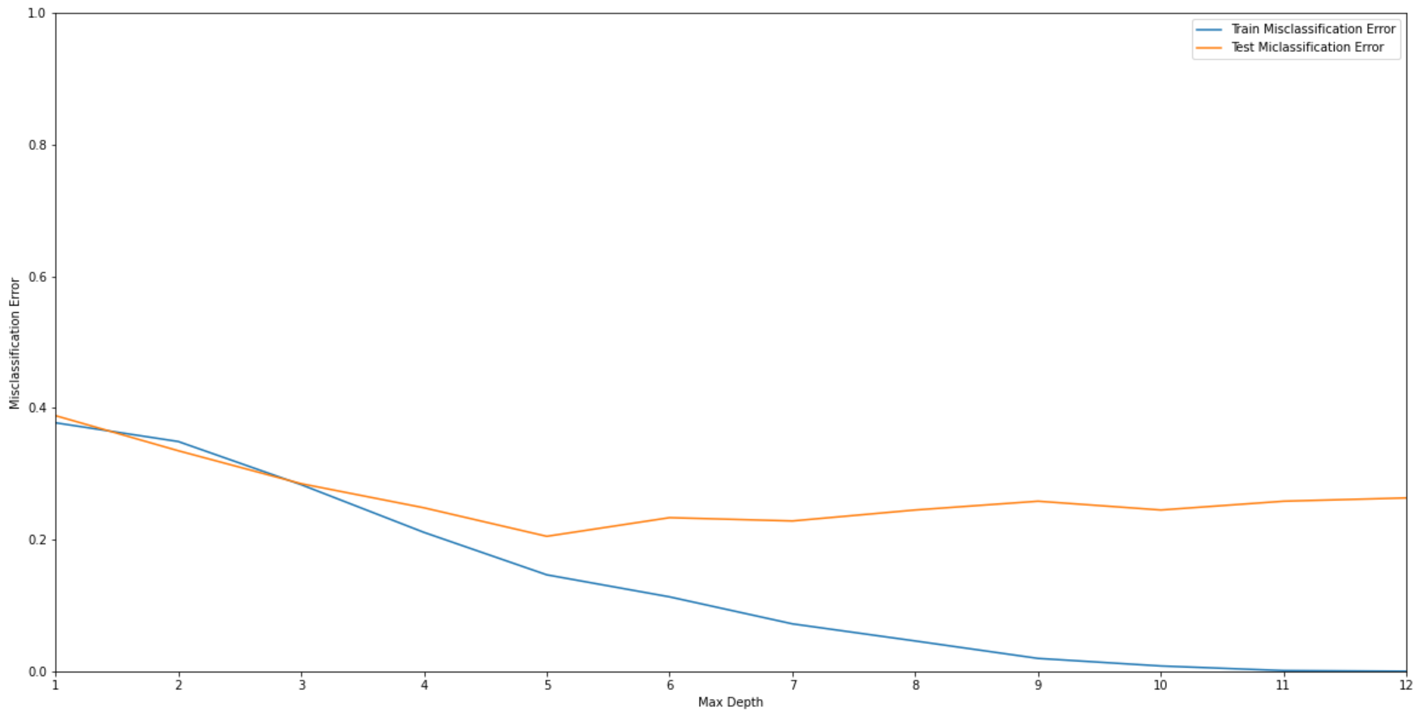
In [99]:
```
#imports
from sklearn import tree
import os
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier as forest
from math import sqrt
```

In [104…
```
#getting training and testing data

#paths
base = 'MADELON'
train_data_path = 'madelon_train.data'
train_labels_path = 'madelon_train.labels'
test_data_path = 'madelon_valid.data'
test_labels_path = 'madelon_valid.labels'
#training
path = os.path.join(base, train_data_path)
rf = open(path, 'r')
Lines = rf.readlines()
X = []
for line in Lines:
    line = line[:-2]
    line = line.split(' ')
    X.append([int(x) for x in line])

#testing
path = os.path.join(base, test_data_path)
rf = open(path, 'r')
Lines = rf.readlines()
X_test = []
for line in Lines:
    line = line[:-2]
    line = line.split(' ')
    X_test.append([int(x) for x in line])

#train labels
path = os.path.join(base, train_labels_path)
rf = open(path, 'r')
Lines = rf.readlines()
X_labels = []
for line in Lines:
    X_labels.append(int(line))

#test labels
path = os.path.join(base, test_labels_path)
rf = open(path, 'r')
Lines = rf.readlines()
X_test_labels = []
for line in Lines:
    X_test_labels.append(int(line))
```

In [67]:
```
train_misclass = []
test_misclass = []
for i in range(1,13):
    clf = tree.DecisionTreeClassifier(criterion = 'entropy', max_depth = i)
    clf = clf.fit(X, X_labels)
    count_test = 0
    for x, y in zip(X_test, X_test_labels):
        pred = clf.predict([x])
        if pred != y:
            count_test += 1
    count_train = 0
    for x, y in zip(X, X_labels):
        pred = clf.predict([x])
        if pred != y:
            count_train += 1
    train_misclass.append(count_train/len(X))
    test_misclass.append(count_test/len(X_test))
```

In [80]:

```
x = list(range(1,13))
plt.plot(x, train_misclass)
plt.plot(x, test_misclass)
plt.xlabel('Max Depth')
plt.ylabel('Misclassification Error')
plt.xticks(ticks=x)
plt.xlim([1, 12])
plt.ylim([0, 1])
plt.legend(['Train Misclassification Error','Test Miclassification Error'])
plt.show()
```



In [85]:

```
mini = min(test_misclass)

depth = x[test_misclass.index(mini)]

print(mini, depth)
```

```
0.205 5
```

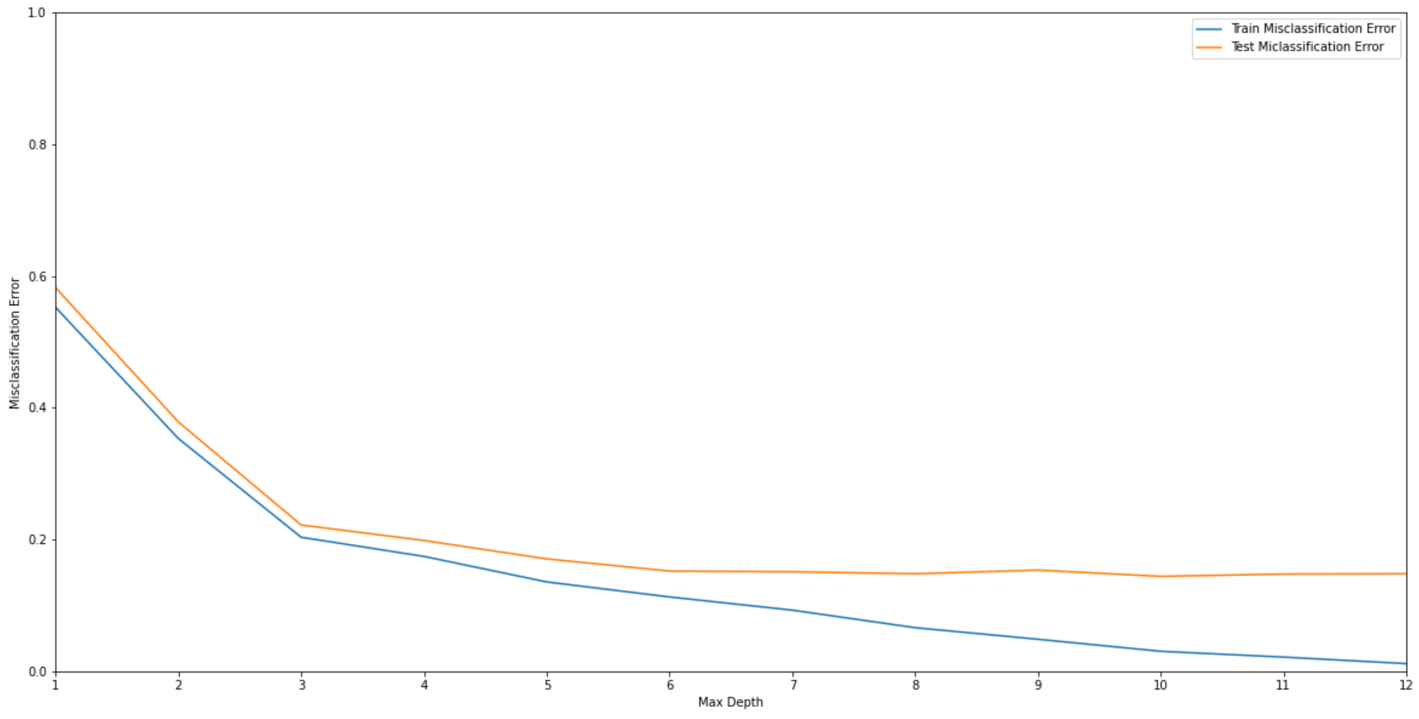
In [118…

```
#now with the satimage dataset
#getting training and testing data

#paths
base = 'satimage'
train_data_path = 'X.dat'
train_labels_path = 'Y.dat'
test_data_path = 'Xtest.dat'
test_labels_path = 'Ytest.dat'
#training
path = os.path.join(base, train_data_path)
rf = open(path, 'r')
```

```
Lines = rf.readlines()
X = []
for line in Lines:
    line = line[:-2]
    line = line.split(' ')
    X.append([int(x) for x in line])

#testing
path = os.path.join(base, test_data_path)
rf = open(path, 'r')
Lines = rf.readlines()
X_test = []
for line in Lines:
    line = line[:-2]
    line = line.split(' ')
    X_test.append([int(x) for x in line])

#train labels
path = os.path.join(base, train_labels_path)
rf = open(path, 'r')
Lines = rf.readlines()
X_labels = []
for line in Lines:
    X_labels.append(int(line))

#test labels
path = os.path.join(base, test_labels_path)
rf = open(path, 'r')
Lines = rf.readlines()
X_test_labels = []
for line in Lines:
    X_test_labels.append(int(line))
```

In [93]:

```
train_misclass = []
test_misclass = []
for i in range(1,13):
    clf = tree.DecisionTreeClassifier(criterion = 'entropy', max_depth = i)
    clf = clf.fit(X, X_labels)
    count_test = 0
    for x, y in zip(X_test, X_test_labels):
        pred = clf.predict([x])
        if pred != y:
            count_test += 1
    count_train = 0
    for x, y in zip(X, X_labels):
        pred = clf.predict([x])
        if pred != y:
            count_train += 1
    train_misclass.append(count_train/len(X))
    test_misclass.append(count_test/len(X_test))

x = list(range(1,13))
plt.plot(x, train_misclass)
plt.plot(x, test_misclass)
plt.xlabel('Max Depth')
plt.ylabel('Misclassification Error')
plt.xticks(ticks=x)
plt.xlim([1, 12])
plt.ylim([0, 1])
plt.legend(['Train Misclassification Error','Test Miclassification Error'])
plt.show()
```



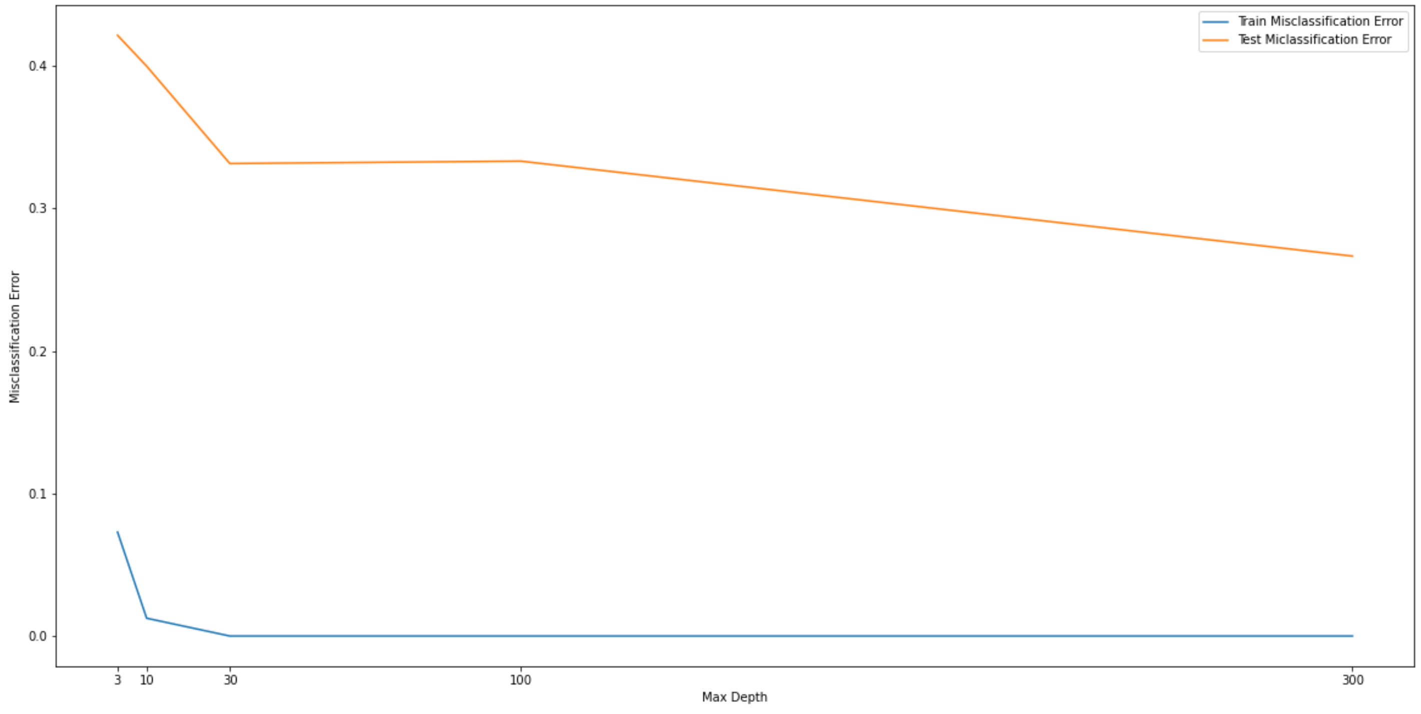
In [94]:

```
mini = min(test_misclass)

depth = x[test_misclass.index(mini)]

print(mini, depth)
```

```
0.144 10
```

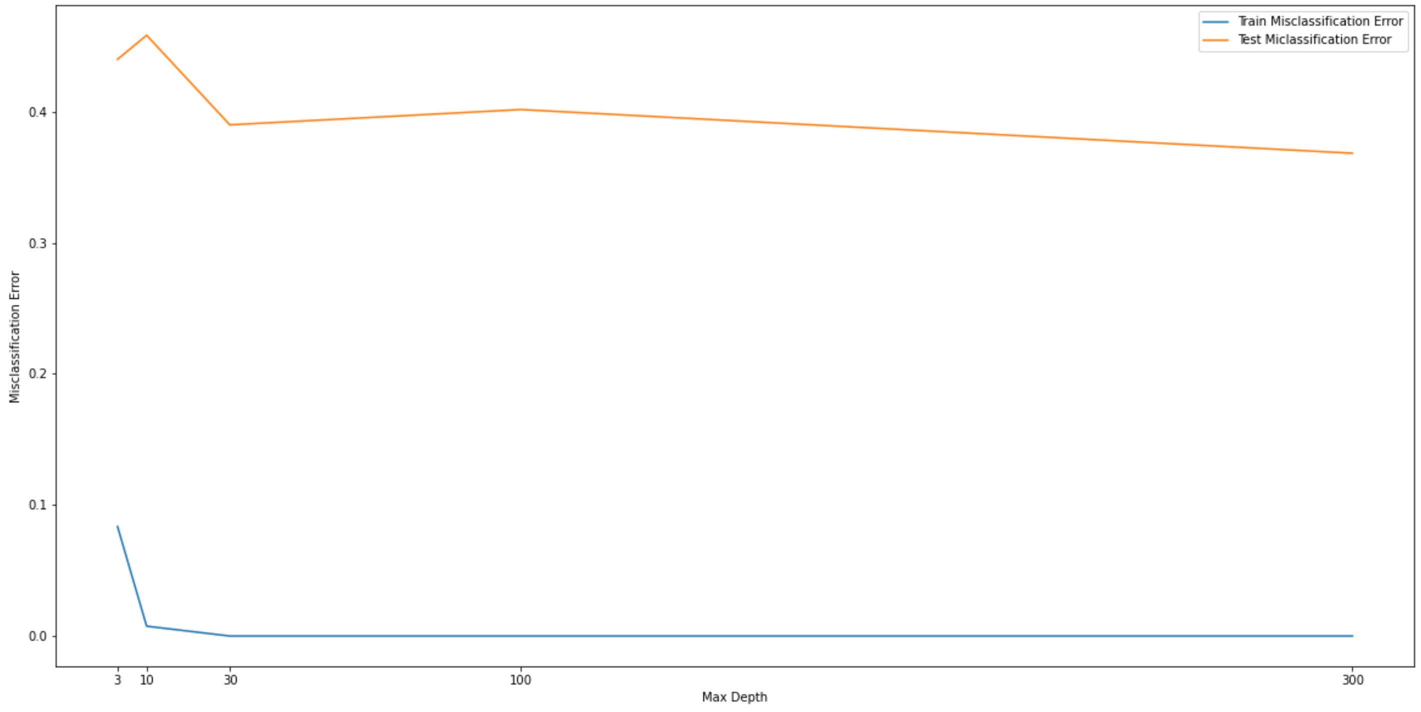
In [105…

```
#with sqrt(500) ~ 22 features
K = [3, 10, 30, 100, 300]

train_misclass = []
test_misclass = []
for k in K:
    clf = forest(n_estimators=k, max_features=round(sqrt(500)))
    clf = clf.fit(X, X_labels)
    count_test = 0
    for x, y in zip(X_test, X_test_labels):
        pred = clf.predict([x])
        if pred != y:
            count_test += 1
    count_train = 0
    for x, y in zip(X, X_labels):
        pred = clf.predict([x])
        if pred != y:
            count_train += 1
    train_misclass.append(count_train/len(X))
    test_misclass.append(count_test/len(X_test))

x = K
plt.plot(x, train_misclass)
plt.plot(x, test_misclass)
plt.xlabel('Max Depth')
plt.ylabel('Misclassification Error')
plt.xticks(ticks=x)

plt.legend(['Train Misclassification Error','Test Miclassification Error'])
plt.show()
print(train_misclass, test_misclass)
```



In [111…

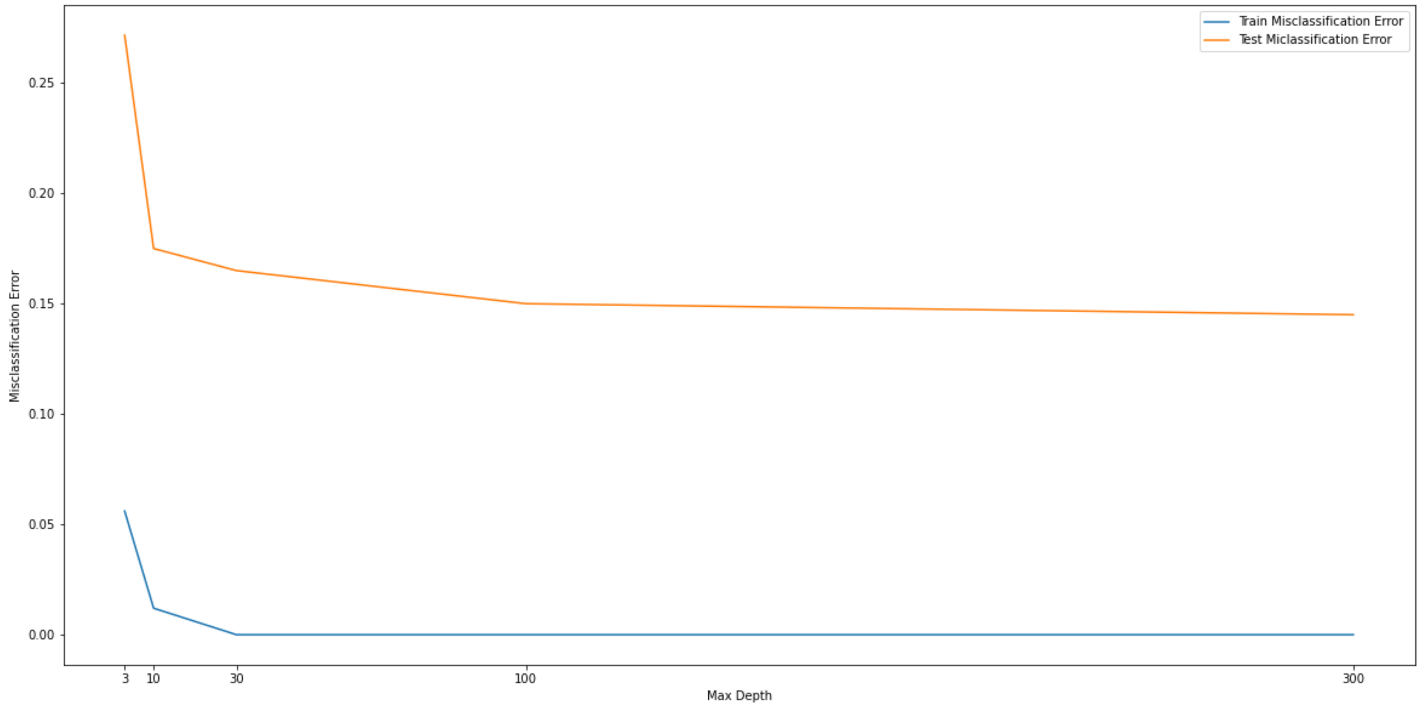
```
#now with six features
K = [3, 10, 30, 100, 300]

train_misclass = []
test_misclass = []
for k in K:
    clf = forest(n_estimators=k, max_features=6)
    clf = clf.fit(X, X_labels)
    count_test = 0
    for x, y in zip(X_test, X_test_labels):
        pred = clf.predict([x])
        if pred != y:
            count_test += 1
    count_train = 0
    for x, y in zip(X, X_labels):
        pred = clf.predict([x])
        if pred != y:
            count_train += 1
    train_misclass.append(count_train/len(X))
    test_misclass.append(count_test/len(X_test))

x = K
plt.plot(x, train_misclass)
plt.plot(x, test_misclass)
plt.xlabel('Max Depth')
plt.ylabel('Misclassification Error')
plt.xticks(ticks=x)

plt.legend(['Train Misclassification Error','Test Miclassification Error'])
plt.show()

print(train_misclass, test_misclass)
```



```
[0.0835, 0.0075, 0.0, 0.0, 0.0] [0.44, 0.4583333333333333, 0.39, 0.40166666666666667, 0.36
833333333333335]
```

In [117…

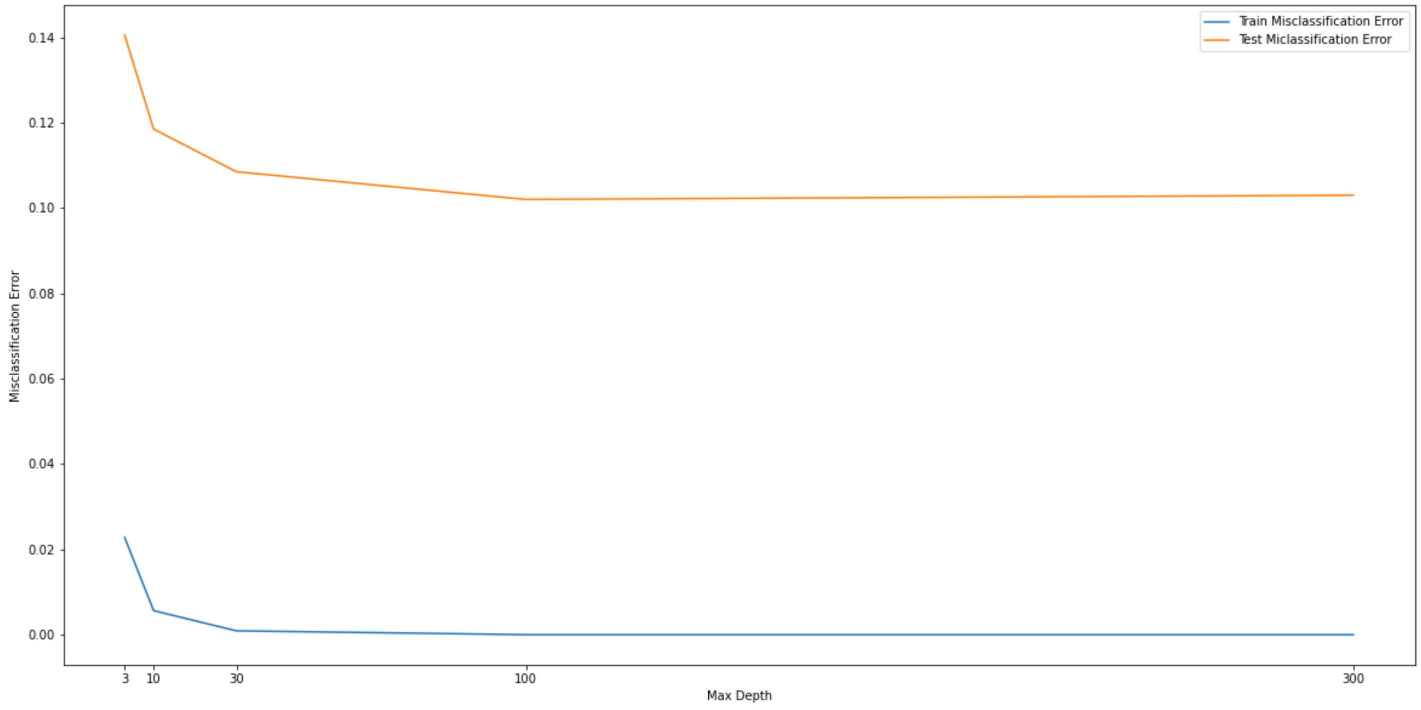
```python
#now with all 500 features
K = [3, 10, 30, 100, 300]

train_misclass = []
test_misclass = []
for k in K:
    clf = forest(n_estimators=k, max_features=500)
    clf = clf.fit(X, X_labels)
    count_test = 0
    for x, y in zip(X_test, X_test_labels):
        pred = clf.predict([x])
        if pred != y:
            count_test += 1
    count_train = 0
    for x, y in zip(X, X_labels):
        pred = clf.predict([x])
        if pred != y:
            count_train += 1
    train_misclass.append(count_train/len(X))
    test_misclass.append(count_test/len(X_test))

x = K
plt.plot(x, train_misclass)
plt.plot(x, test_misclass)
plt.xlabel('Max Depth')
plt.ylabel('Misclassification Error')
plt.xticks(ticks=x)

plt.legend(['Train Misclassification Error','Test Miclassification Error'])
plt.show()

print(train_misclass, test_misclass)
```



```
[0.056, 0.012, 0.0, 0.0, 0.0] [0.27166666666666667, 0.175, 0.165, 0.15, 0.145]
```

In [119…

```python
#now with satimage data, and all 36 features
K = [3, 10, 30, 100, 300]

train_misclass = []
test_misclass = []
for k in K:
    clf = forest(n_estimators=k, max_features=36)
    clf = clf.fit(X, X_labels)
    count_test = 0
    for x, y in zip(X_test, X_test_labels):
        pred = clf.predict([x])
        if pred != y:
            count_test += 1
    count_train = 0
    for x, y in zip(X, X_labels):
        pred = clf.predict([x])
        if pred != y:
            count_train += 1
    train_misclass.append(count_train/len(X))
    test_misclass.append(count_test/len(X_test))

x = K
plt.plot(x, train_misclass)
plt.plot(x, test_misclass)
plt.xlabel('Max Depth')
plt.ylabel('Misclassification Error')
plt.xticks(ticks=x)

plt.legend(['Train Misclassification Error','Test Miclassification Error'])
plt.show()

print(train_misclass, test_misclass)
```



```
[0.022773393461104848, 0.005636978579481398, 0.0009019165727170237, 0.0, 0.0] [0.1405, 0.1
185, 0.1085, 0.102, 0.103]
```