STA 5635 Hw #1

Alejandro Rojas

September 1, 2021

Exercise 1

Using the training and test sets specified in the syllabus, perform the following tasks:

a. On the *madelon* dataset, train decision trees of maximum depth 1, 2, ... up to 12, for a total of 12 decision trees. If your package does not allow the max depth as a parameter, train trees with 2^1 , 2^2 , ..., 2^{12} nodes, again a total of 12 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. (3 points)

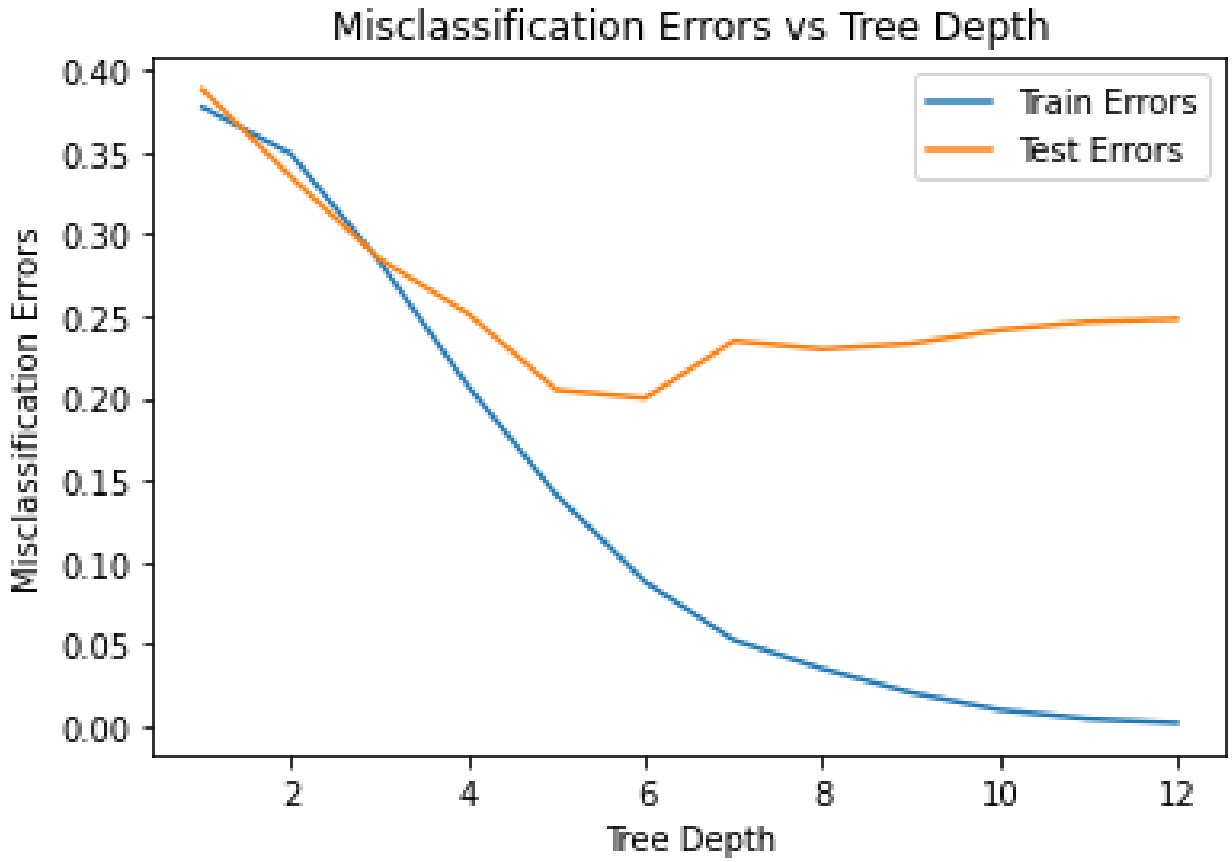
b. Repeat point a) on the *wilt* dataset, with maximum tree depths d from 1 to 10 (i.e. 2^1 , ..., 2^{10} nodes). (2 points)

c. On the *madelon* dataset, for each of $k \in \{3, 10, 30, 100, 300\}$ train a random forest with k trees where the split attribute at each node is chosen from a random subset of 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. (3 points)

d. Repeat point c) on the *madelon* dataset where the split attribute at each node is chosen from a random subset of ln(500) features. (1 point)

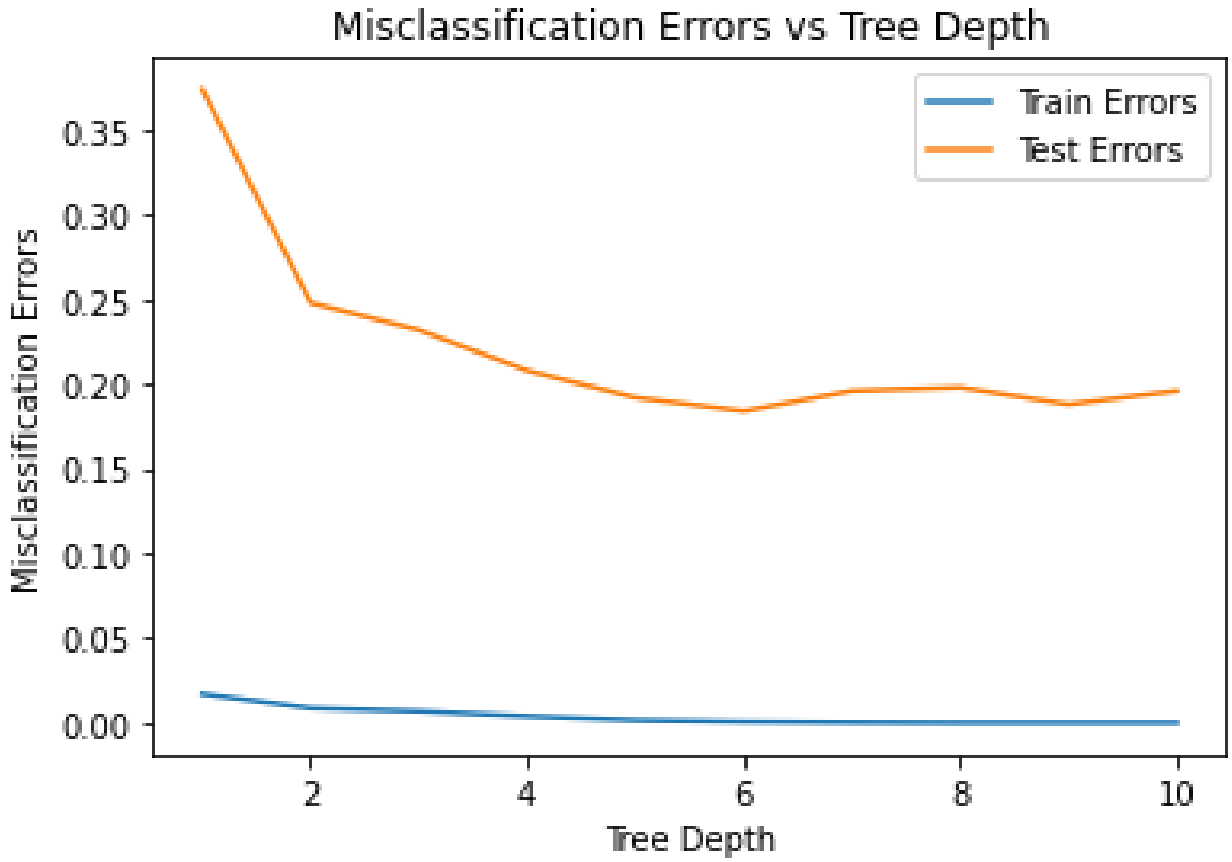
e. Repeat point c) on the *madelon* dataset where the split attribute at each node is chosen from all 500 features. (1 point)

Solutions to part A:



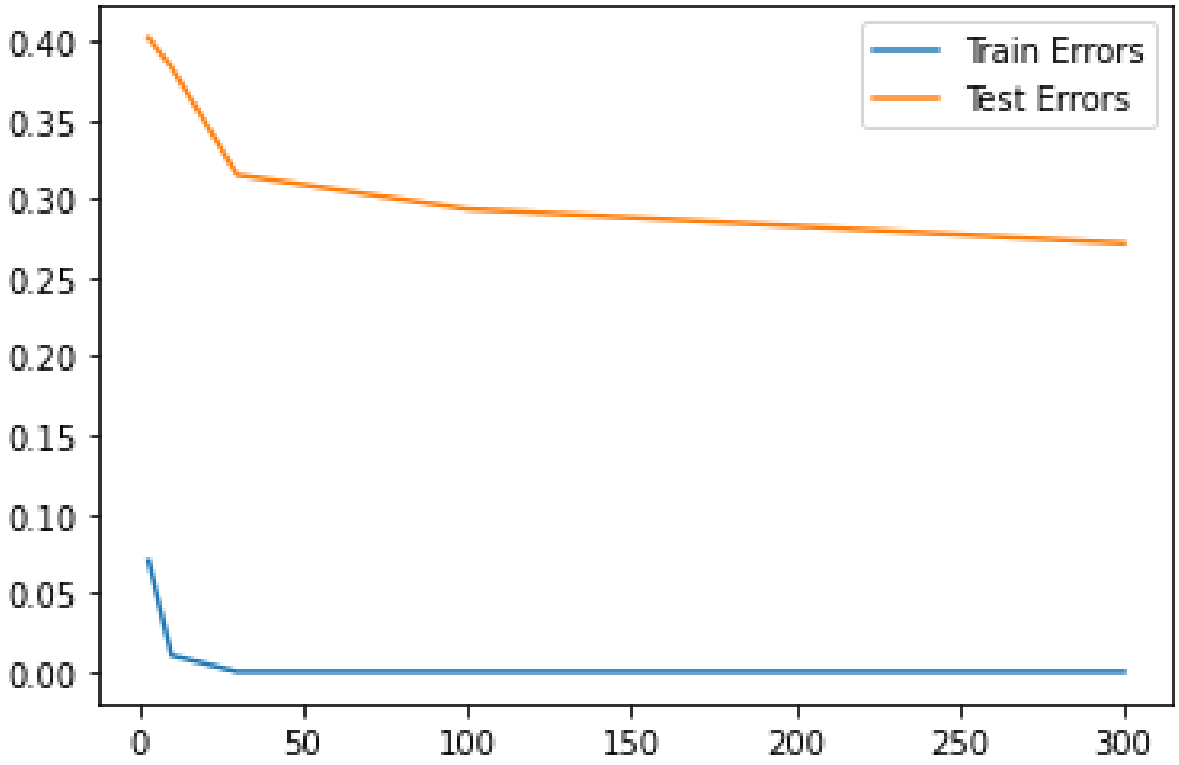
Minimum Test Error	0.1983
Depth	6

Solutions to part B:



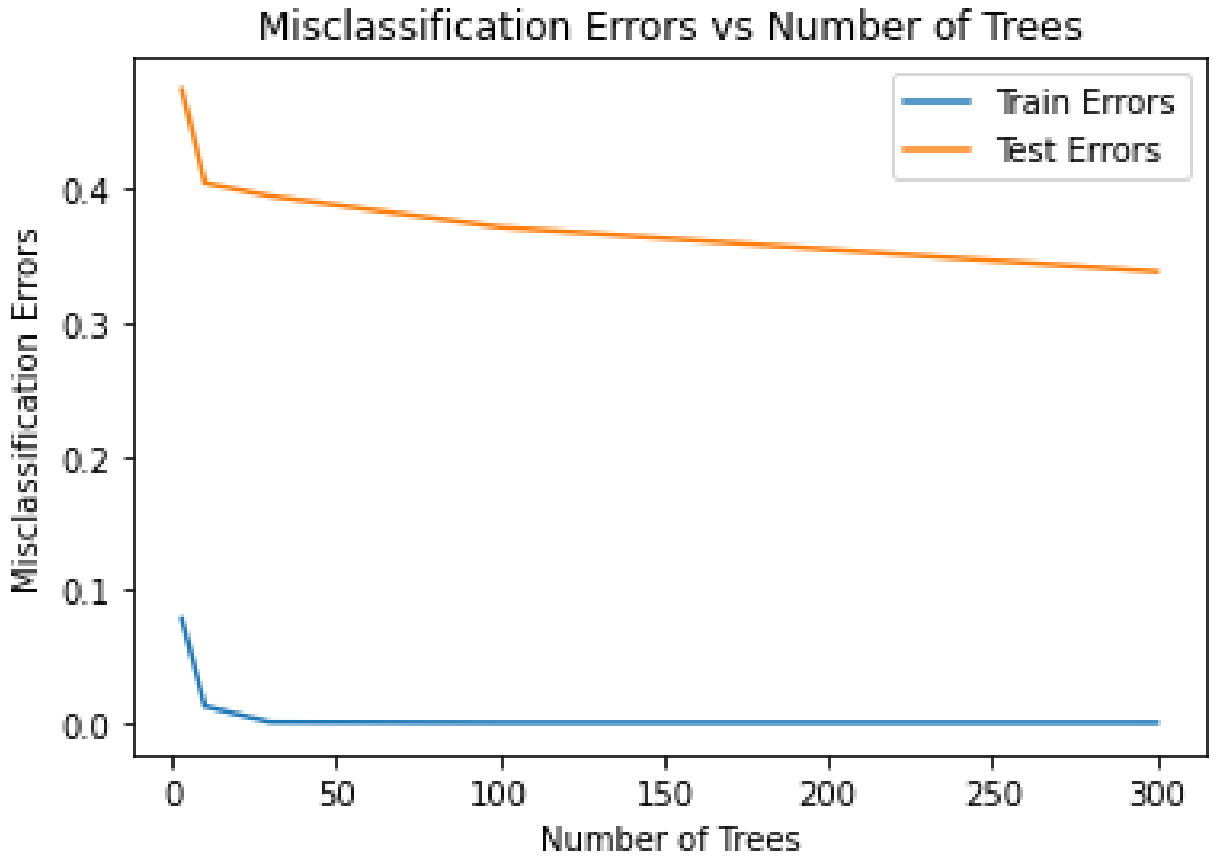
Minimum Test Error	0.1840
Depth	6

Solutions to part C:



Trees	3	10	30	100	300
Training Errors	0.0735	0.013	0.0	0.0	0.0
Test Errors	0.38	0.35	0.3249	0.2833	0.2933

Solutions to part D:



Trees	3	10	30	100	300
Training Errors	0.0779	0.01249	0.0005	0.0	0.0
Test Errors	0.475	0.405	0.395	0.3716	0.3383

Solutions to part E:

Trees	3	10	30	100	300
Training Errors	0.0535	0.008	0.001	0.0	0.0
Test Errors	0.25	0.1933	0.17	0.15	0.146

Appendix:

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

# Part A
# Madelon data input
X_train_m = pd.read_csv("data/MADELON/madelon_train.data", sep=' ', header=None)
X_train_m = X_train_m.drop(500, axis = 1) # Double space at the end of line addi
y_train_m = pd.read_csv("data/MADELON/madelon_train.labels", sep=' ', header=None

X_test_m = pd.read_csv("data/MADELON/madelon_valid.data", sep=' ', header=None)
X_test_m = X_test_m.drop(500, axis = 1)
y_test_m = pd.read_csv("data/MADELON/madelon_valid.labels", sep=' ', header=None)

# Fit, predict and derive errors
errors_train_m = []    # Store misclassification errors of the train set
errors_test_m = []     # Store misclassification errors of the test set

for i in range(1,13):
    clf_m = DecisionTreeClassifier(max_depth=i)      # Create decision tree of dep
    fit_m = clf_m.fit(X_train_m, y_train_m)          # Fit unique tree to train da
    preds_train_m = clf_m.predict(X_train_m)         # Predict target of train set
    preds_test_m = clf_m.predict(X_test_m)           # Predict target of test set

    # Store Errors
    errors_train_m.append(1 - metrics.accuracy_score(y_train_m, preds_train_m))
    errors_test_m.append(1 - metrics.accuracy_score(y_test_m, preds_test_m))

# Quick summary of the tree depth and both training and test errors
for i in range(0,12):
    print("\nTree of depth " + str(i+1) + ": ")
    print("Training error: ", errors_train_m[i])
    print("Test error: ", errors_test_m[i])

# Not surprising to see the training error approach 0 as depth increases as the m

# Plot both train and test errors vs tree depth
depth_m = list(range(1,13))
plt.plot(depth_m, errors_train_m)
plt.plot(depth_m, errors_test_m)
plt.title('Misclassification Errors vs Tree Depth')
plt.xlabel('Tree Depth')
```

```
plt.ylabel('Misclassification Errors')
plt.legend(['Train Errors', 'Test Errors'])
plt.show

# Minimum Error and depth obtained in
print('Minimum Error:', min(errors_test_m))
print('At depth:', errors_test_m.index(min(errors_test_m)) + 1) # Add 1 as first

# Part B
# Wilt data input
X_train_w = pd.read_csv("data/wilt/wilt_train.csv",header=None)
y_train_w = pd.read_csv("data/wilt/wilt_train.labels", header=None)

X_test_w = pd.read_csv("data/wilt/wilt_test.csv", header=None)
y_test_w = pd.read_csv("data/wilt/wilt_test.labels", header=None)

# Fit, predict and derive errors
errors_train_w = []    # Store misclassification errors of the train set
errors_test_w = []     # Store misclassification errors of the test set

for i in range(1,11):
    clf_w = DecisionTreeClassifier(max_depth=i)      # Create decision tree of dep
    fit_w = clf_w.fit(X_train_w, y_train_w)          # Fit unique tree to train da
    preds_train_w = clf_w.predict(X_train_w)         # Predict target of train set
    preds_test_w = clf_w.predict(X_test_w)           # Predict target of test set

    # Store Errors
    errors_train_w.append(1 - metrics.accuracy_score(y_train_w, preds_train_w))
    errors_test_w.append(1 - metrics.accuracy_score(y_test_w, preds_test_w))

# Quick summary of the tree depth and both training and test errors
for i in range(0,10):
    print("\nTree of depth " + str(i+1) + ": ")
    print("Training error: ", errors_train_w[i])
    print("Test error: ", errors_test_w[i])

# Unlike the madelon dataset, the trees overfit the training data right off the l

# Plot both train and test errors vs tree depth
depth_w = list(range(1,11))
plt.plot(depth_w, errors_train_w)
plt.plot(depth_w, errors_test_w)
plt.title('Misclassification Errors vs Tree Depth')
plt.xlabel('Tree Depth')
plt.ylabel('Misclassification Errors')
plt.legend(['Train Errors', 'Test Errors'])
plt.show()
```

```
# Minimum Error and depth obtained in
print('Minimum Error:', min(errors_test_w))
print('At depth:', errors_test_w.index(min(errors_test_w)) + 1) # Add 1 as first

# Part C
k = [3, 10, 30, 100, 300]
errors_train_rnd_c = []
errors_test_rnd_c = []

for i in k:
    rnd_clf = RandomForestClassifier(n_estimators=i, max_features='sqrt')
    rnd_clf.fit(X_train_m, y_train_m.values.ravel())
    preds_train_rnd = rnd_clf.predict(X_train_m)
    preds_test_rnd = rnd_clf.predict(X_test_m)

    # Store Errors
    errors_train_rnd_c.append(1 - metrics.accuracy_score(y_train_m, preds_train_r
    errors_test_rnd_c.append(1 - metrics.accuracy_score(y_test_m, preds_test_rnd)

# Quick summary of the tree depth and both training and test errors
for i in range(0,5):
    print("\nNumber of trees in forest: " + str(k[i]) + ": ")
    print("Training error: ", errors_train_rnd_c[i])
    print("Test error: ", errors_test_rnd_c[i])

# Plot both train and test errors vs tree depth
plt.plot(k, errors_train_rnd_c)
plt.plot(k, errors_test_rnd_c)
plt.title('Misclassification Errors vs Number of Trees')
plt.xlabel('Number of Trees')
plt.ylabel('Misclassification Errors')
plt.legend(['Train Errors', 'Test Errors'])
plt.show()

# Part D
k = [3, 10, 30, 100, 300]
errors_train_rnd_d = []
errors_test_rnd_d = []

for i in k:
    rnd_clf = RandomForestClassifier(n_estimators=i, max_features=int(np.log(500)
                                                    # max_features = 6 as ln(500)
    rnd_clf.fit(X_train_m, y_train_m.values.ravel())
    preds_train_rnd = rnd_clf.predict(X_train_m)
    preds_test_rnd = rnd_clf.predict(X_test_m)

    # Store Errors
    errors_train_rnd_d.append(1 - metrics.accuracy_score(y_train_m, preds_train_r
```

```
    errors_test_rnd_d.append(1 - metrics.accuracy_score(y_test_m, preds_test_rnd)

# Quick summary of the tree depth and both training and test errors
for i in range(0,5):
    print("\nNumber of trees in forest: " + str(k[i]) + ": ")
    print("Training error: ", errors_train_rnd_d[i])
    print("Test error: ", errors_test_rnd_d[i])

# Plot both train and test errors vs tree depth
plt.plot(k, errors_train_rnd_d)
plt.plot(k, errors_test_rnd_d)
plt.title('Misclassification Errors vs Number of Trees')
plt.xlabel('Number of Trees')
plt.ylabel('Misclassification Errors')
plt.legend(['Train Errors', 'Test Errors'])
plt.show()

# Part E
k = [3, 10, 30, 100, 300]
errors_train_rnd_e = []
errors_test_rnd_e = []

for i in k:
    rnd_clf = RandomForestClassifier(n_estimators=i, max_features=500)
    rnd_clf.fit(X_train_m, y_train_m.values.ravel())
    preds_train_rnd = rnd_clf.predict(X_train_m)
    preds_test_rnd = rnd_clf.predict(X_test_m)

    # Store Errors
    errors_train_rnd_e.append(1 - metrics.accuracy_score(y_train_m, preds_train_r
    errors_test_rnd_e.append(1 - metrics.accuracy_score(y_test_m, preds_test_rnd)

# Quick summary of the tree depth and both training and test errors
for i in range(0,5):
    print("\nNumber of trees in forest: " + str(k[i]) + ": ")
    print("Training error: ", errors_train_rnd_e[i])
    print("Test error: ", errors_test_rnd_e[i])

# Plot both train and test errors vs tree depth
plt.plot(k, errors_train_rnd_e)
plt.plot(k, errors_test_rnd_e)
plt.title('Misclassification Errors vs Number of Trees')
plt.xlabel('Number of Trees')
plt.ylabel('Misclassification Errors')
plt.legend(['Train Errors', 'Test Errors'])
plt.show()
```