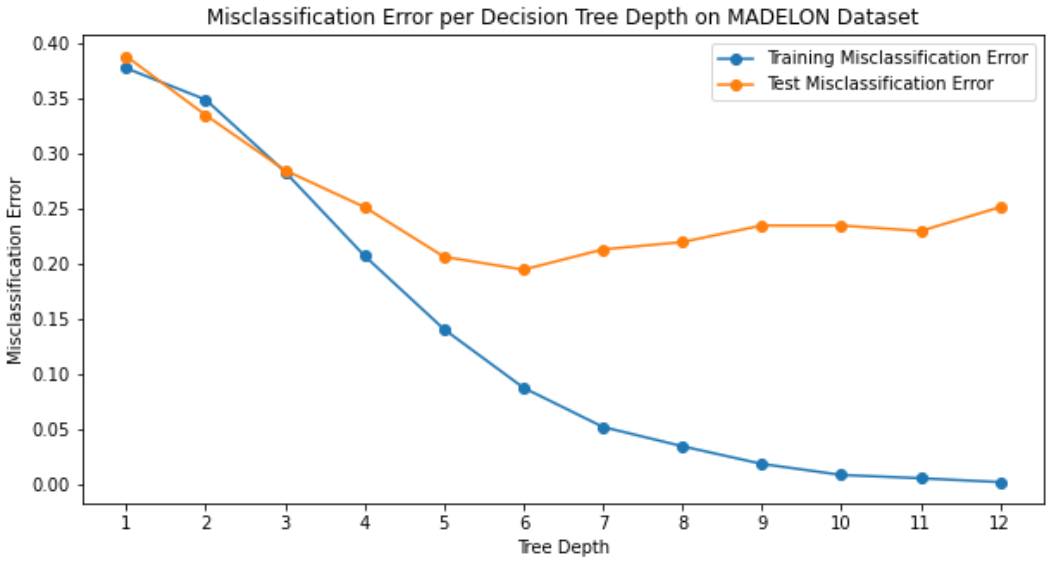
Homework 1

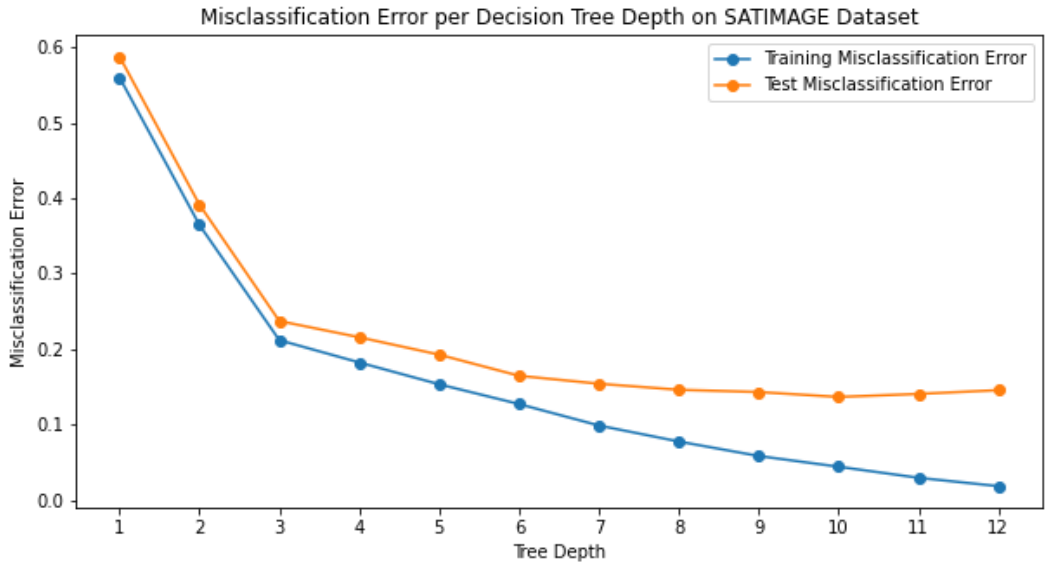
Jarod Klion

a)



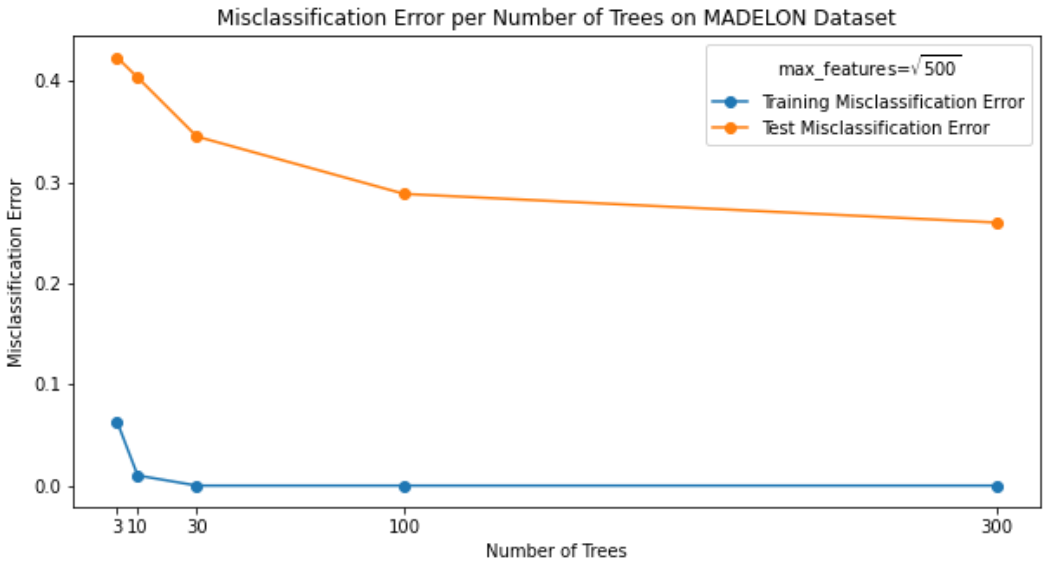
Tree Depth	Minimum Test Error
6	19.33%

b)



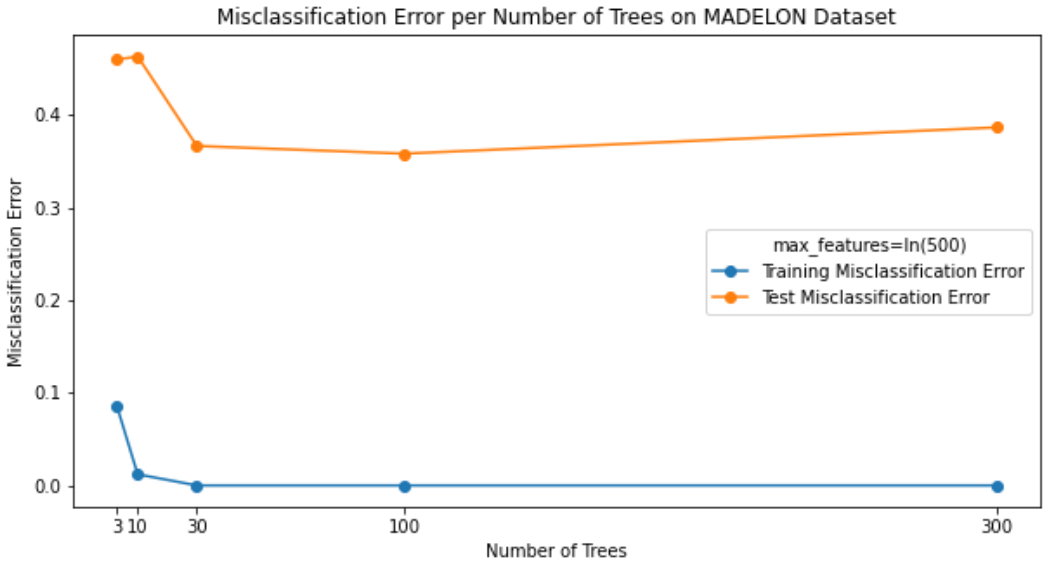
Tree Depth	Minimum Test Error
10	13.70%

c)



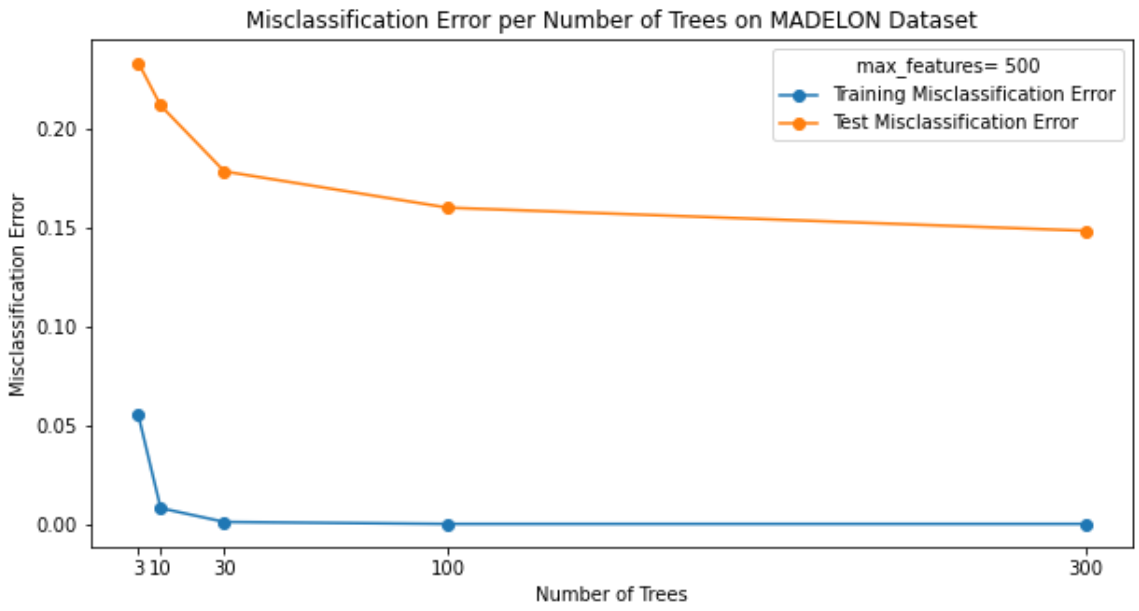
Number of trees (k)	Training Error	Test Error
3	6.35%	42.33%
10	1.0%	40.33%
30	0.0%	34.50%
100	0.0%	28.83%
300	0.0%	26.00%

d)



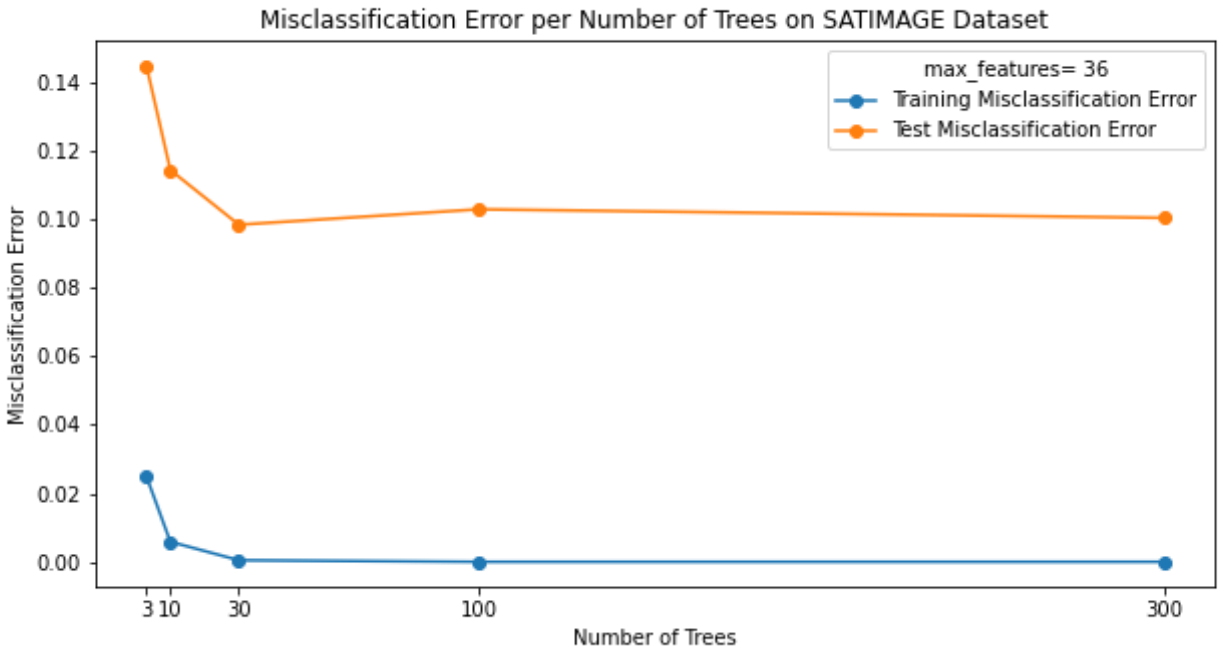
Number of trees (k)	Training Error	Test Error
3	9.0%	46.00%
10	1.0%	46.33%
30	0.0%	36.67%
100	0.0%	35.83%
300	0.0%	38.67%

e)



Number of trees (k)	Training Error	Test Error
3	5.5%	23.33%
10	0.8%	21.17%
30	0.1%	17.83%
100	0.0%	16.00%
300	0.0%	14.84%

f)



Number of trees (k)	Training Error	Test Error
3	2.50%	14.50%
10	0.586%	11.45%
30	0.045%	9.85%
100	0.00%	10.30%
300	0.00%	10.05%

++In[ ]:++

[source, ipython3]

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

++In[ ]:++

[source, ipython3]

```
#Read in the MADELON dataset
md_train = pd.read_csv("madelon_train.data", sep=' ',
header=None).drop(500, axis=1) #last column is NaN
md_train_labels = pd.read_csv("madelon_train.labels", sep=' ',
header=None)
md_test = pd.read_csv("madelon_valid.data", sep = ' ',
header=None).drop(500, axis=1) #Last column is NaN
md_test_labels = pd.read_csv("madelon_valid.labels", sep=' ',
header=None)
#X_train, X_test, y_train, y_test = md_train, md_test,
md_train_labels[0], md_test_labels[0] #typical syntax
```

++In[ ]:++

[source, ipython3]

```
#Define a function to more easily run the classifier

def run_tree(X_train, y_train, X_test, y_test, depth):

    tree_clf = DecisionTreeClassifier(max_depth=depth).fit(X_train,
y_train)

    accuracy_train = tree_clf.score(X_train, y_train)

    accuracy_test = tree_clf.score(X_test, y_test)

    print("Tree depth: ", depth)

    print("Training set accuracy: ", round(accuracy_train*100,4), "%")

    print("Testing set accuracy: ", round(accuracy_test*100,4), "%")

    return accuracy_train, accuracy_test
```

+*In[ ]:*+

[source, ipython3]

```
#Run the decision tree for the MADELON dataset

md_train_acc, md_test_acc = np.empty(12, dtype=float), np.empty(12,
dtype=float)

for k in range(1,13):

    md_train_acc[k-1], md_test_acc[k-1] = run_tree(md_train,
md_train_labels, md_test, md_test_labels, k)
```

+*In[ ]:*+

[source, ipython3]

```
#Plot both train and test error on one graph

depth = np.arange(1, 13, 1)

fig, ax = plt.subplots(1, 1, figsize=(10,5))
```

```
ax.plot(depth, 1 - md_train_acc, '-o', label = "Training
Misclassification Error")

ax.plot(depth, 1 - md_test_acc, '-o', label = "Test Misclassification
Error")

ax.set_title("Misclassification Error per Decision Tree Depth on MADELON
Dataset")

ax.set_xlabel("Tree Depth")

ax.set_ylabel("Misclassification Error")

ax.set_xticks(depth)

ax.legend()

fig.savefig("MADELON Decision Tree Error.png")

----
```

++In[ ]:++

[source, ipython3]

```
----

1- max(md_test_acc)

----
```

++In[ ]:++

[source, ipython3]

```
----

#Read in the SATIMAGE dataset

sat_X_train = pd.read_csv("X.dat", sep = ' ', header=None)

sat_y_train = pd.read_csv("Y.dat", sep = ' ', header=None)

sat_X_test = pd.read_csv("Xtest.dat", sep = ' ', header=None)

sat_y_test = pd.read_csv("Ytest.dat", sep = ' ', header=None)

----
```

++In[ ]:++

[source, ipython3]

```
#Run the decision tree for the SATIMAGE dataset
sat_train_acc, sat_test_acc = np.empty(12, dtype=float), np.empty(12, dtype=float)
for k in range(1,13):
    sat_train_acc[k-1], sat_test_acc[k-1] = run_tree(sat_X_train, sat_y_train, sat_X_test, sat_y_test, k)
```

++In[ ]:++

[source, ipython3]

```
#Plot both train and test error on one graph
depth = np.arange(1, 13, 1)
fig, ax = plt.subplots(1, 1, figsize=(10,5))
ax.plot(depth, 1 - sat_train_acc, '-o', label = "Training Misclassification Error")
ax.plot(depth, 1 - sat_test_acc, '-o', label = "Test Misclassification Error")
ax.set_title("Misclassification Error per Decision Tree Depth on SATIMAGE Dataset")
ax.set_xlabel("Tree Depth")
ax.set_ylabel("Misclassification Error")
ax.set_xticks(depth)
ax.legend()
fig.savefig("SATIMAGE Decision Tree Error.png")
```

++In[ ]:++

[source, ipython3]

1-max(sat_test_acc)

++In[ ]:++

[source, ipython3]

#Define a function to more easily run random forest

def run_random_forest(X_train, y_train, X_test, y_test, trees, feature):

    forest_clf = RandomForestClassifier(n_estimators=trees,
max_features=feature).fit(X_train, y_train)

    accuracy_train = forest_clf.score(X_train, y_train)

    accuracy_test = forest_clf.score(X_test, y_test)

    print("Number of trees: ", trees)

    print("Training set accuracy: ", round(accuracy_train*100,4), "%")

    print("Testing set accuracy: ", round(accuracy_test*100,4), "%")

    return accuracy_train, accuracy_test

++In[ ]:++

[source, ipython3]

#Run the random forest for the MADELON dataset using max_features =
sqrt(500)

```
md_train_acc1 = []

md_test_acc1 = []

trees=[3,10,30,100,300]

for i in trees:

    train, test = run_random_forest(md_train, md_train_labels[0],
    md_test, md_test_labels[0], i, "sqrt")

    md_train_acc1.append(train)

    md_test_acc1.append(test)

md_train_acc1 = np.array(md_train_acc1)

md_test_acc1 = np.array(md_test_acc1)
```

+*In[ ]:*+

[source, ipython3]

```
#Plot both train and test error on one graph

fig, ax = plt.subplots(1, 1, figsize=(10,5))

ax.plot(trees, 1 - md_train_acc1, '-o', label = "Training
Misclassification Error")

ax.plot(trees, 1 - md_test_acc1, '-o', label = "Test Misclassification
Error")

ax.set_title("Misclassification Error per Number of Trees on MADELON
Dataset")

ax.set_xlabel("Number of Trees")

ax.set_ylabel("Misclassification Error")

ax.set_xticks(trees)

ax.legend(title="max_features=" r'$\sqrt{500}$')

fig.savefig("MADELON Random Forest Sqrt Error.png")
```

++In[ ]:++

[source, ipython3]

```
print("Training Error:", [100*(1-md_train_acc1[i]) for i in range(5)])
print("Training Error:",  [100*(1-md_test_acc1[i]) for i in range(5)])
```

++In[ ]:++

[source, ipython3]

```
#Run the random forest for the MADELON dataset using max_features =
ln(500) ~ 6
md_train_acc2 = []
md_test_acc2 = []
trees=[3,10,30,100,300]
for i in trees:
    train, test = run_random_forest(md_train, md_train_labels[0],
    md_test, md_test_labels[0], i, 6)
    md_train_acc2.append(train)
    md_test_acc2.append(test)
md_train_acc2 = np.array(md_train_acc2)
md_test_acc2 = np.array(md_test_acc2)
```

++In[ ]:++

[source, ipython3]

```
#Plot both train and test error on one graph

fig, ax = plt.subplots(1, 1, figsize=(10,5))

ax.plot(trees, 1 - md_train_acc2, '-o', label = "Training
Misclassification Error")

ax.plot(trees, 1 - md_test_acc2, '-o', label = "Test Misclassification
Error")

ax.set_title("Misclassification Error per Number of Trees on MADELON
Dataset")

ax.set_xlabel("Number of Trees")

ax.set_ylabel("Misclassification Error")

ax.set_xticks(trees)

ax.legend(title="max_features=ln(500)")

fig.savefig("MADELON Random Forest Ln Error.png")

----
```

```
++In[ ]:*+

[source, ipython3]

----

print("Training Error:", [100*round((1-md_train_acc2[i]),4) for i in
range(5)])

print("Training Error:",  [100*round((1-md_test_acc2[i]),4) for i in
range(5)])

----
```

```
++In[ ]:*+

[source, ipython3]

----

#Run the random forest for the MADELON dataset using max_features = None

md_train_acc3 = []
```

```
md_test_acc3 = []

trees=[3,10,30,100,300]

for i in trees:

    train, test = run_random_forest(md_train, md_train_labels[0],
    md_test, md_test_labels[0], i, None)

    md_train_acc3.append(train)

    md_test_acc3.append(test)

md_train_acc3 = np.array(md_train_acc3)

md_test_acc3 = np.array(md_test_acc3)

----
```

++In[ ]:++

```
[source, ipython3]

----

#Plot both train and test error on one graph

fig, ax = plt.subplots(1, 1, figsize=(10,5))

ax.plot(trees, 1 - md_train_acc3, '-o', label = "Training
Misclassification Error")

ax.plot(trees, 1 - md_test_acc3, '-o', label = "Test Misclassification
Error")

ax.set_title("Misclassification Error per Number of Trees on MADELON
Dataset")

ax.set_xlabel("Number of Trees")

ax.set_ylabel("Misclassification Error")

ax.set_xticks(trees)

ax.legend(title="max_features= 500")

fig.savefig("MADELON Random Forest All Error.png")

----
```

++In[ ]:++

[source, ipython3]

```
print("Training Error:", [100*round((1-md_train_acc3[i]),4) for i in
range(5)])
print("Training Error:",  [100*round((1-md_test_acc3[i]),4) for i in
range(5)])
```

++In[ ]:++

[source, ipython3]

```
#Run the random forest for the SATIMAGE dataset using max_features = all
features (None)
sat_train_acc1 = []
sat_test_acc1 = []
trees=[3,10,30,100,300]
for i in trees:
    train, test = run_random_forest(sat_X_train, sat_y_train[0],
    sat_X_test, sat_y_test[0], i, None)
    sat_train_acc1.append(train)
    sat_test_acc1.append(test)
sat_train_acc1 = np.array(sat_train_acc1)
sat_test_acc1 = np.array(sat_test_acc1)
```

++In[ ]:++

[source, ipython3]

```
#Plot both train and test error on one graph

fig, ax = plt.subplots(1, 1, figsize=(10,5))

ax.plot(trees, 1 - sat_train_acc1, '-o', label = "Training
Misclassification Error")

ax.plot(trees, 1 - sat_test_acc1, '-o', label = "Test Misclassification
Error")

ax.set_title("Misclassification Error per Number of Trees on SATIMAGE
Dataset")

ax.set_xlabel("Number of Trees")

ax.set_ylabel("Misclassification Error")

ax.set_xticks(trees)

ax.legend(title="max_features= 36")

fig.savefig("SATIMAGE Random Forest All Error.png")

----



+*In[ ]:*+

[source, ipython3]

----

print("Training Error:", [100*(1-sat_train_acc1[i]) for i in range(5)])

print("Training Error:",  [100*(1-sat_test_acc1[i]) for i in range(5)])

----
```