Homework 1

Jarod Klion

1. Chart, line chart

   Description automatically generated

|  |  |
| --- | --- |
| Tree Depth | Minimum Test Error |
| 6 | 19.33% |

1. Chart, line chart

   Description automatically generated

|  |  |
| --- | --- |
| Tree Depth | Minimum Test Error |
| 10 | 13.70% |

1. Chart, line chart

   Description automatically generated

|  |  |  |
| --- | --- | --- |
| 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% |

1. Chart, line chart

   Description automatically generated

|  |  |  |
| --- | --- | --- |
| 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% |

1. Chart, line chart

   Description automatically generated

|  |  |  |
| --- | --- | --- |
| 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% |

1. Chart, line chart

   Description automatically generated

|  |  |  |
| --- | --- | --- |
| 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)])

----