CSCE 633 Homework 4

April 9, 2020

Submitted by Rizu Jain (430000753)

[1]:

```
import math

import pandas as pd
import numpy as np

from sklearn import tree
from sklearn.metrics import mean_squared_error
```

[2]:

```
# Load Dataset

train_data = pd.read_csv('OnlineNewsPopularityTrain.csv')
test_data = pd.read_csv('OnlineNewsPopularityTest.csv')

# Remove label column
train_arr = train_data.values[:,1:]
test_arr = test_data.values[:,1:]
```

1 Decision tree regression

[3]:

```
# Prepare randomly shuffled data for decision tree

np.random.shuffle(train_arr)
np.random.shuffle(test_arr)

x_train = train_arr[:,:-1]
y_train = train_arr[:,-1]
x_test = test_arr[:,:-1]
y_test = test_arr[:,-1]
```

[4]:

```
samples_per_fold= x_train.shape[0]//5

kfold_x_train=[]
kfold_y_train=[]
for i in range(5):
    kfold_x_train.append(x_train[samples_per_fold*i:samples_per_fold*(i+1),:])
    kfold_y_train.append(y_train[samples_per_fold*i:samples_per_fold*(i+1)])
```

[5]:

```
depths= [2,5,3,7,10,20]
depth_err = []

for curr_depth in depths:
    crossval_err = []
    for i in range(5):
        clf = tree.DecisionTreeRegressor(max_depth=curr_depth,random_state=0)

        train_xsets = []
        train_ysets = []
        curr_x_test = []
        curr_y_test = []

        for j in range(5):
            if j==i:
                curr_x_test = kfold_x_train[i]
                curr_y_test = kfold_y_train[i]
            else:
                train_xsets.append(kfold_x_train[j])
                train_ysets.append(kfold_y_train[j])


        curr_x_train = np.
 ↪concatenate((train_xsets[0],train_xsets[1],train_xsets[2],train_xsets[3]))
        curr_y_train = np.
 ↪concatenate((train_ysets[0],train_ysets[1],train_ysets[2],train_ysets[3]))

        clf = clf.fit(curr_x_train,curr_y_train)
        curr_y_pred = clf.predict(curr_x_test)

        err = mean_squared_error(curr_y_pred,curr_y_test)
        crossval_err.append(math.sqrt(err))

    print("Cross validation errors for max depth : ",curr_depth)
    print(crossval_err)
    depth_err.append(np.mean(crossval_err))
    print()


print("-----------------------------------------------------")
for k in range(len(depths)):
    print("Max depth: ",depths[k],"\tCrossVal average RSS: ",depth_err[k])
```

```
Cross validation errors for max depth :  2
[9224.11604620015, 12065.381162979264, 8510.082656879917, 15842.621659452852,
11418.842315740498]

Cross validation errors for max depth :  5
[14306.989870526284, 12179.479621233402, 8405.625688241142, 17406.20798278216,
11360.913452065786]

Cross validation errors for max depth :  3
[14030.46728927809, 12196.151465152392, 8717.023898819332, 17403.776402822554,
11463.558051403981]

Cross validation errors for max depth :  7
[14929.8187040767, 14386.506099851586, 15001.675912877789, 19062.406462796498,
11606.225755964482]

Cross validation errors for max depth :  10
[15841.729347189994, 13388.82413275757, 17386.360791621697, 20290.145438720225,
15758.206878103683]

Cross validation errors for max depth :  20
[16978.09970117997, 16327.387296521709, 17816.150254216573, 21102.489465270446,
13923.153016805458]

----------------------------------------------------
Max depth:  2   CrossVal average RSS:  11412.208768250535
Max depth:  5   CrossVal average RSS:  12731.843322969755
Max depth:  3   CrossVal average RSS:  12762.19542149527
Max depth:  7   CrossVal average RSS:  14997.32658711341
Max depth:  10  CrossVal average RSS:  16533.053317678634
Max depth:  20  CrossVal average RSS:  17229.45594679883
```

[6]:

```python
# Performance on Test data

optimal_depth=depths[depth_err.index(min(depth_err))]
clf = tree.DecisionTreeRegressor(max_depth=optimal_depth,random_state=0)
clf = clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)

test_err = mean_squared_error(y_pred,y_test)
test_err = math.sqrt(test_err)

print("Optimal Depth: ",optimal_depth)
print("Test Error: ",test_err)
```

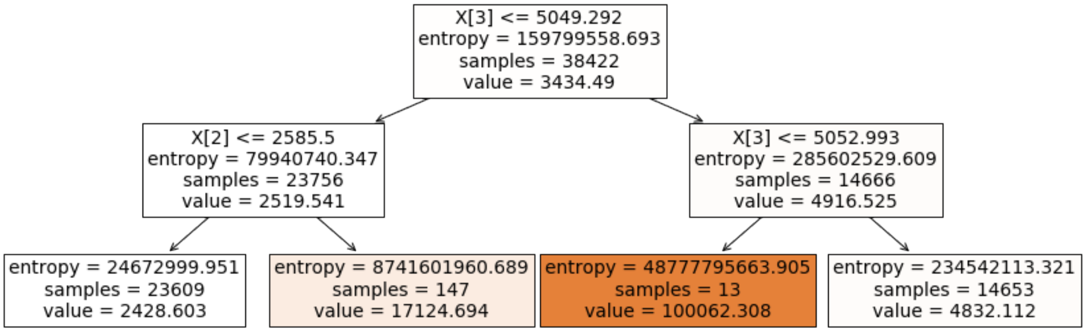
```
Optimal Depth:  2
Test Error:  8296.439356745595
```

2 Feature Exploration

[12]:

```python
import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = [15, 5]
tree.plot_tree(clf,filled = True);
```



[8]:

```python
# we dumped the 1st feature from our original dataframe
imp_features = [27,30,28]

print("The most important features are :")
for feature in imp_features:
    print(train_data.columns[feature])
```

```
The most important features are :
 kw_avg_avg
 self_reference_avg_sharess
 self_reference_min_shares
```

```
27. kw_avg_avg:                    Avg. keyword (avg. shares)
28. self_reference_min_shares:     Min. shares of referenced articles in
                                   Mashable
30. self_reference_avg_sharess:    Avg. shares of referenced articles in
                                   Mashable
```

Intution

1. The use of keywords is an important factor in predicting number of shares
2. The popularity of referenced articles is another important factor

3 Random Forest

[9]:
```
# hyperparamters to tune
# No. of trees
# depth of trees

num_trees_options = [10,20,40,80]
tree_depth = [2,4,8,16]
tree_depth_err = []


for num_trees in num_trees_options:
    depth_err = []
    for curr_depth in tree_depth:

        crossval_err = []

        for i in range(5):
            train_xsets = []
            train_ysets = []
            curr_x_test = []
            curr_y_test = []

            for j in range(5):
                if j==i:
                    curr_x_test = kfold_x_train[i]
                    curr_y_test = kfold_y_train[i]
                else:
                    train_xsets.append(kfold_x_train[j])
                    train_ysets.append(kfold_y_train[j])


            curr_x_train = np.
 ↪concatenate((train_xsets[0],train_xsets[1],train_xsets[2],train_xsets[3]))
            curr_y_train = np.
 ↪concatenate((train_ysets[0],train_ysets[1],train_ysets[2],train_ysets[3]))

            predictions= np.zeros(curr_y_test.shape)

            for _ in range(num_trees):
                # select data samples randomly
                # we select as many samples as orignal but we allow repeats
                # source: https://stats.stackexchange.com/questions/347818/
 ↪number-of-samples-per-tree-in-a-random-forest

                sample_ids = np.random.choice(curr_x_train.shape[0],curr_x_train.
 ↪shape[0])
```

```
                x_train_rf = curr_x_train[sample_ids]
                y_train_rf = curr_y_train[sample_ids]

                # randomly select 30% features for this tree
                sel_columns = np.random.choice(x_train_rf.
 ↪shape[1],int(x_train_rf.shape[1]*0.3))
                x_train_rf = x_train_rf[:,sel_columns]

                clf = tree.DecisionTreeRegressor(max_depth=curr_depth)
                clf = clf.fit(x_train_rf,y_train_rf)
                curr_y_pred = clf.predict(curr_x_test[:,sel_columns])
                predictions += curr_y_pred


            predictions = predictions/num_trees
            err = mean_squared_error(predictions,curr_y_test)
            crossval_err.append(math.sqrt(err))

        print("----------------------------")
        print("max depth : ",curr_depth)
        print("num trees : ",num_trees)
        print("crossval_err : ",crossval_err)
        depth_err.append(np.mean(crossval_err))
        print()

    tree_depth_err.append(depth_err)
```

```
----------------------------
max depth :  2
num trees :  10
crossval_err :  [9185.340878314053, 12070.276586605261, 8412.881754990376,
15799.307737160054, 11387.441569869328]

----------------------------
max depth :  4
num trees :  10
crossval_err :  [9437.13326781748, 12136.258396977764, 8586.780269075025,
15791.31208518285, 11379.1349516635]

----------------------------
max depth :  8
num trees :  10
crossval_err :  [9883.822881673846, 12305.610623246203, 8987.271770430261,
16259.022486803888, 11750.999839408036]

----------------------------
max depth :  16
```

```
num trees :  10
crossval_err :  [9970.775065148686, 12753.854540515482, 9190.986467418059,
16120.390661007328, 12304.681234486488]

---------------------------
max depth :  2
num trees :  20
crossval_err :  [9198.277123395585, 12085.858670854968, 8372.829357713827,
15748.975697556178, 11351.79005153414]

---------------------------
max depth :  4
num trees :  20
crossval_err :  [9224.842218391972, 12141.02336328227, 8380.66491990371,
15773.471087204663, 11380.044039262653]

---------------------------
max depth :  8
num trees :  20
crossval_err :  [9410.835143283497, 12171.924615451335, 8547.666218286065,
15766.101157368608, 11504.103902963549]

---------------------------
max depth :  16
num trees :  20
crossval_err :  [9530.990573779967, 12337.639951767642, 8821.508866663818,
15981.598200205788, 11948.794700090133]

---------------------------
max depth :  2
num trees :  40
crossval_err :  [9211.763895639724, 12068.790062841606, 8370.885648698306,
15752.059079752073, 11349.154047842516]

---------------------------
max depth :  4
num trees :  40
crossval_err :  [9251.393064615833, 12098.39339174754, 8399.667006008976,
15783.759791873154, 11346.256497292554]

---------------------------
max depth :  8
num trees :  40
crossval_err :  [9377.762785446423, 12117.381635336376, 8501.161040856203,
15807.077836383245, 11442.9907904012]

---------------------------
max depth :  16
```

```
num trees :  40
crossval_err :  [9489.313675408115, 12244.529619787018, 8618.082989713932,
15842.955591464131, 11513.829050615977]

---------------------------
max depth :  2
num trees :  80
crossval_err :  [9199.377005245928, 12075.617294139556, 8377.0216163679,
15781.51826091811, 11343.19267337088]

---------------------------
max depth :  4
num trees :  80
crossval_err :  [9216.044340993618, 12082.352916696347, 8375.696474494987,
15752.374612246558, 11347.832993223]

---------------------------
max depth :  8
num trees :  80
crossval_err :  [9339.208114168956, 12094.450458670784, 8419.930332885371,
15759.189305968373, 11380.786447905291]

---------------------------
max depth :  16
num trees :  80
crossval_err :  [9273.454408219484, 12112.916673038406, 8495.880566165053,
15810.977699166513, 11444.682154373224]
```

[14]:

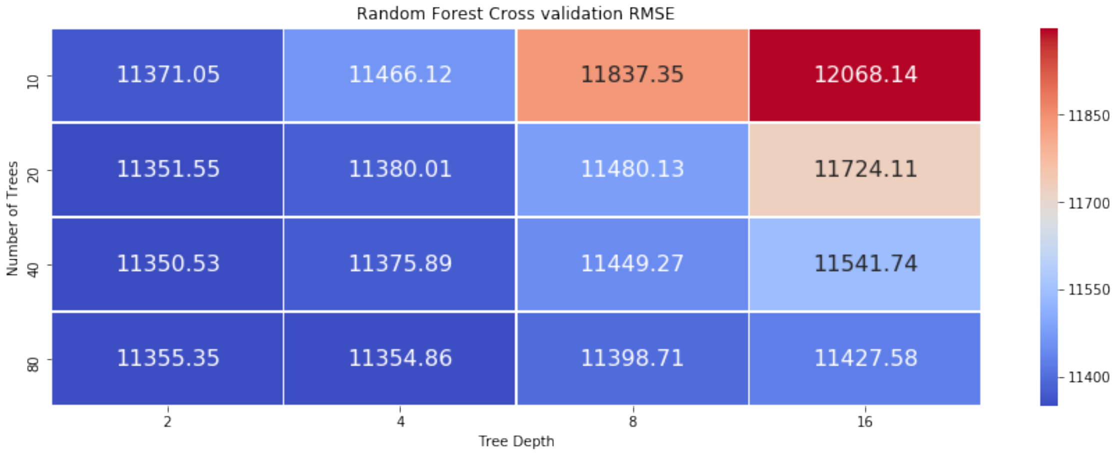
```python
# 2-dimensional color-coded matrix
# x/y dimensions are the number of trees and tree depth, and
# the color-coding reflects the average error over all folds

import seaborn as sns

xlabels = [str(i) for i in tree_depth ]
ylabels = [str(j) for j in num_trees_options ]

ax = sns.heatmap(tree_depth_err,
                 annot=True,
                 fmt = "0.2f",
                 cmap= "coolwarm",
                 annot_kws={'size':16},
                 xticklabels=xlabels,
                 yticklabels=ylabels,
                 robust=True,
                 linewidths=.5)
```

```
ax.set(xlabel='Tree Depth', ylabel='Number of Trees',title="Random Forest Cross␣
 ↪validation RMSE")
plt.show()
```



[11]:
```
# Performance on Test Set

err_random_forest = np.asarray(tree_depth_err)
ind = np.unravel_index(np.argmin(err_random_forest, axis=None),␣
 ↪err_random_forest.shape)
best_num_trees = num_trees_options[ind[0]]
best_tree_depth = tree_depth[ind[1]]

final_predictions= np.zeros(y_test.shape)

for _ in range(best_num_trees):
    # select data samples randomly
    # we select as many samples as orignal but we allow repeats
    # source: https://stats.stackexchange.com/questions/347818/
 ↪number-of-samples-per-tree-in-a-random-forest

    sample_ids = np.random.choice(x_train.shape[0],x_train.shape[0])
    features_rf = x_train[sample_ids]
    labels_rf = y_train[sample_ids]

    # randomly select 30% features for this tree
    sel_columns = np.random.choice(features_rf.shape[1],int(features_rf.
 ↪shape[1]*0.3))
    features_rf = features_rf[:,sel_columns]

    clf = tree.DecisionTreeRegressor(max_depth=best_tree_depth)
    clf = clf.fit(features_rf,labels_rf)
    y_pred = clf.predict(x_test[:,sel_columns])
    final_predictions += y_pred


final_predictions = final_predictions/best_num_trees
test_err = mean_squared_error(final_predictions,y_test)
test_err = math.sqrt(test_err)

print("---------------------------")
print("Max depth : ",curr_depth)
print("Bum trees : ",num_trees)
print("Test Error : ",test_err)
print("---------------------------")
```

```
---------------------------
Max depth :  16
Bum trees :  80
Test Error :  8337.195445959338
---------------------------
```

Test Error Random Forest : 8337.2