Machine Learning - Homework 4
Name: Guru Sarath Thangamani
UIN: 829009551

a)
Decision tree regression

Optimal Depth of decision tree = 6

Average error (Train)

```
Depth = 1 -------------
Errors in all the 5 folds: [12704.522646716448, 12451.351711495125, 15982.790465630835, 6893.117134657534, 7827.603531315466]
Avg error - 11171.877097963083
Depth = 2 -------------
Errors in all the 5 folds: [12706.361503601018, 12409.630673317362, 15939.939116389296, 6866.115132557895, 7837.578632601012]
Avg error - 11151.925011693316
Depth = 3 -------------
Errors in all the 5 folds: [12692.374104878132, 12388.103599184531, 15927.67289966213, 6855.073213100711, 7827.648071444609]
Avg error - 11138.174377654022
Depth = 4 -------------
Errors in all the 5 folds: [12694.239972166159, 12397.997397379619, 15922.120784496236, 6852.498502277804, 7825.210225582775]
Avg error - 11138.413376380518
Depth = 5 -------------
Errors in all the 5 folds: [12694.503892741612, 12398.321141261533, 15923.260313107681, 6850.399814243473, 7826.081852938912]
Avg error - 11138.513402858644
Depth = 6 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 7 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 8 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 9 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 10 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 11 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 12 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 13 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 14 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 15 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 16 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 17 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 18 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 19 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 20 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533, 15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
```

Average error (Test)

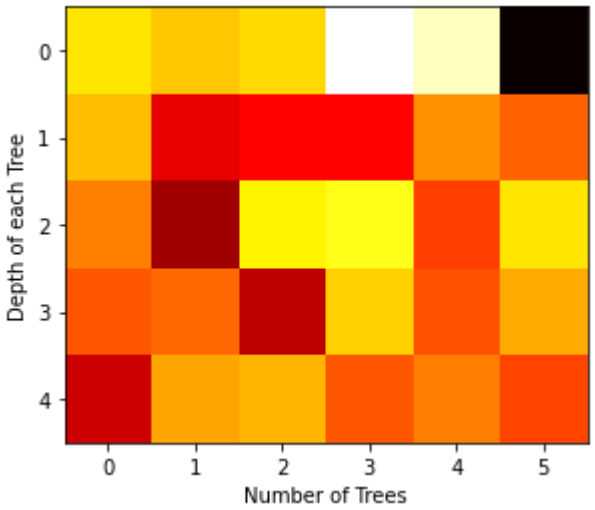
```
Testset error = 8329.687133026458
```

b)

Random forest regression

Average error (Train)

```
Num Trees --- 3    Depth --- 2
Errors in all the 5 folds: [12724.046569362174, 12498.377628849306, 16062.147746609337, 6984.6714602121365, 7868.297952735844]
Avg error - 11227.508271553761
Num Trees --- 3    Depth --- 5
Errors in all the 5 folds: [12717.421134833665, 12509.636647998374, 16062.100516001983, 6978.993940040527, 7865.722467818247]
Avg error - 11226.774941338559
Num Trees --- 3    Depth --- 7
Errors in all the 5 folds: [12729.960888094924, 12508.229812025074, 16057.208917594005, 6957.839438893067, 7882.767327728526]
Avg error - 11227.201276867121
Num Trees --- 3    Depth --- 10
Errors in all the 5 folds: [12720.974951614733, 12506.914848596572, 16068.924199212375, 6980.530195042351, 7882.1058403258]
Avg error - 11231.890006958367
Num Trees --- 3    Depth --- 15
Errors in all the 5 folds: [12736.580210990749, 12504.063949978168, 16055.340801016533, 6977.988207399826, 7880.6873418044]
Avg error - 11230.932102237934
Num Trees --- 3    Depth --- 20
Errors in all the 5 folds: [12713.103941631105, 12476.988658826376, 16070.386553947252, 6962.440116396896, 7860.761246478012]
Avg error - 11216.736103455927
Num Trees --- 5    Depth --- 2
Errors in all the 5 folds: [12717.374210813774, 12512.121654084862, 16053.308646219277, 6972.76813092451, 7877.241304158912]
Avg error - 11226.562789240266
Num Trees --- 5    Depth --- 5
Errors in all the 5 folds: [12716.395747578255, 12491.140513181628, 16060.160396596337, 6978.908565968395, 7862.041564234389]
Avg error - 11221.7293575118
Num Trees --- 5    Depth --- 7
Errors in all the 5 folds: [12721.110450531985, 12484.11353808428, 16061.025250846578, 6968.017413515005, 7877.477406867283]
Avg error - 11222.348811969026
Num Trees --- 5    Depth --- 10
Errors in all the 5 folds: [12719.294892272896, 12486.876999201417, 16056.381162213202, 6983.351793961469, 7865.7678772885965]
Avg error - 11222.334544987516
Num Trees --- 5    Depth --- 15
Errors in all the 5 folds: [12722.913979844565, 12488.194092994105, 16059.488148218212, 6975.070780648936, 7881.906879184975]
Avg error - 11225.514776178159
Num Trees --- 5    Depth --- 20
Errors in all the 5 folds: [12722.129675240943, 12497.336716576261, 16064.848800685859, 6971.981176858201, 7865.955876178577]
Avg error - 11224.450449107968
Num Trees --- 10    Depth --- 2
Errors in all the 5 folds: [12712.155928706066, 12499.075304895754, 16059.07557619714, 6969.964870291187, 7885.653685318607]
Avg error - 11225.18507308175
Num Trees --- 10    Depth --- 5
Errors in all the 5 folds: [12718.368770488354, 12483.096121417097, 16049.944694799666, 6972.7534252401065, 7876.435979623928]
Avg error - 11220.119798313832
Num Trees --- 10    Depth --- 7
Errors in all the 5 folds: [12721.751417708138, 12498.775189647033, 16062.883799251036, 6972.922732022012, 7882.587823066019]
Avg error - 11227.784192338846
Num Trees --- 10    Depth --- 10
Errors in all the 5 folds: [12719.613119945994, 12503.198175096115, 16050.665764498137, 6978.679598306325, 7890.039502673226]
Avg error - 11228.43923210396
Num Trees --- 10    Depth --- 15
Errors in all the 5 folds: [12719.45960353457, 12500.719391478093, 16062.305381510532, 6968.148624367102, 7867.787784135403]
Avg error - 11223.68415700514
Num Trees --- 10    Depth --- 20
Errors in all the 5 folds: [12723.94728336258, 12490.716573580954, 16053.427223150667, 6982.504525903613, 7886.940676450378]
Avg error - 11227.507256489638
Num Trees --- 15    Depth --- 2
Errors in all the 5 folds: [12716.93934393705, 12496.046536089332, 16058.525006050677, 6967.461147559711, 7882.281855413725]
Avg error - 11224.250777810099
Num Trees --- 15    Depth --- 5
Errors in all the 5 folds: [12717.566024429429, 12498.87459405616, 16053.607219195323, 6973.538557210984, 7879.616933885491]
Avg error - 11224.640665755478
Num Trees --- 15    Depth --- 7
Errors in all the 5 folds: [12716.668702587649, 12496.45809411756, 16054.708087434208, 6957.445636004115, 7878.70179103626]
Avg error - 11220.79646223596
Num Trees --- 15    Depth --- 10
Errors in all the 5 folds: [12718.395024299842, 12506.286113297676, 16058.977792226473, 6971.594808633657, 7879.747239941456]
Avg error - 11227.000195679819
Num Trees --- 15    Depth --- 15
Errors in all the 5 folds: [12717.149540976648, 12503.531656445613, 16057.310916671524, 6963.696049501141, 7878.871653543204]
Avg error - 11224.111963427626
Num Trees --- 15    Depth --- 20
Errors in all the 5 folds: [12721.430288511228, 12492.536991063756, 16056.569943910641, 6979.329897635477, 7880.675956165997]
Avg error - 11226.10861545742
Num Trees --- 20    Depth --- 2
Errors in all the 5 folds: [12716.481463200564, 12491.441258965213, 16053.177599722942, 6965.1137557676375, 7879.436004464423]
Avg error - 11221.130016424157
Num Trees --- 20    Depth --- 5
Errors in all the 5 folds: [12720.392157915272, 12502.98880937526, 16056.245417239461, 6971.256399091026, 7879.1580506096617]
Avg error - 11226.008166846128
Num Trees --- 20    Depth --- 7
Errors in all the 5 folds: [12717.666635633237, 12498.388067513815, 16055.77209645244, 6980.723179126257, 7879.3260758148435]
Avg error - 11226.375210908118
Num Trees --- 20    Depth --- 10
Errors in all the 5 folds: [12721.343569578332, 12497.849925980438, 16056.161900208152, 6973.468953047092, 7872.2055731953397]
Avg error - 11224.205984401882
Num Trees --- 20    Depth --- 15
Errors in all the 5 folds: [12716.291724964734, 12501.580756484827, 16057.622875030178, 6975.466075426473, 7874.51458395017]
Avg error - 11225.095203171277
Num Trees --- 20    Depth --- 20
Errors in all the 5 folds: [12718.673877358071, 12493.89365513387, 16055.62782408774, 6973.471953051058, 7877.4408435627]
Avg error - 11223.821630638688
```



The x and y axis values are the indices of number of trees and Depth of tree array.

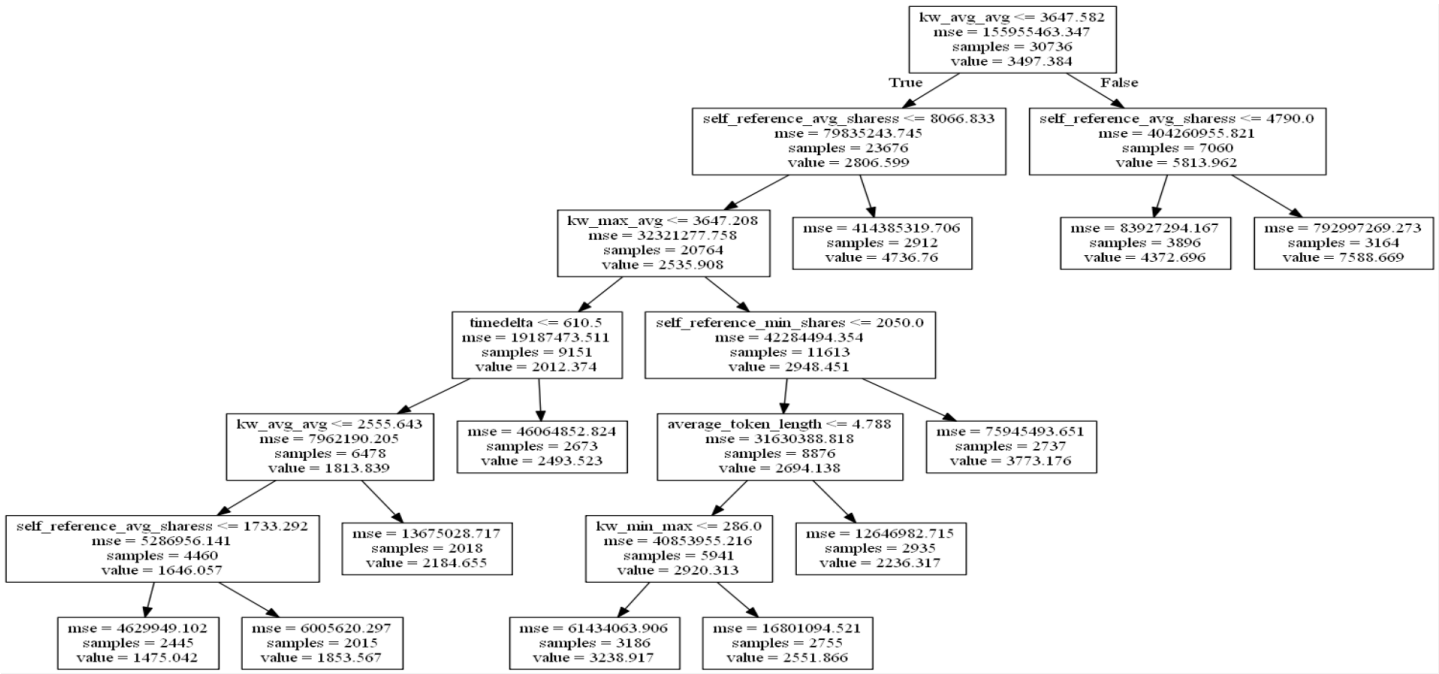
```
max_depth = [2,5,7,10,15,20]
max_trees = [3,5,10,15,20]
```

Best Combination:
Number of trees =3
Depth of trees =20

Average error (Train)

```
Error on Test set - 8448.342738846994
```

(c) *Feature exploration*



Most important feature: kw_avg_avg

The important features in this tree are (In the order of decreasing importance) –
kw_avg_avg, self_reference_avg_sharess, kw_max_avg, time_delta, average_token_length, kw_min_max ,etc.

Feature with the least entropy is used by the decision tree algorithm to split the dataset. Features used in the decision tree as the depth of the tree increases are less important compared to the features closer to the root.

(d)
NLP Feature extraction

Urls are converted to vectors and the regression tree model is again trained to get the following testset accuracy.
Each word in the last section of the Url was converted to a vector and all the vectors for a particular Url was averaged to get the final feature vector of the Url.
This vector was augmented to the training set and then the model was trained.

Testset error = `8329.687133026458`

HW4

April 8, 2020

1 Machine Learning Homework 4

2 Name: Guru Sarath Thangamani

3 UIN: 829009551

[0]:
```
from sklearn import tree
import numpy as np
import matplotlib.pyplot as plt
from random import randint
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error
import pandas
```

[0]:
```
from google.colab import files
import io

import warnings
warnings.filterwarnings('ignore')
```

[0]:
```
# Code to read csv file into Colaboratory:
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
# Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

[0]:
```
# https://drive.google.com/open?id=1QKwHh3FkJ2Lznu522hp0crzJOCvUcHoj
downloaded = drive.CreateFile({'id':'1QKwHh3FkJ2Lznu522hp0crzJOCvUcHoj'})
downloaded.GetContentFile('OnlineNewsPopularityTest.csv')

# https://drive.google.com/open?id=17a2nHMF7CZGzc73EUBThal3jbhDCLGsE
```

```
downloaded = drive.CreateFile({'id':'17a2nHMF7CZGzc73EUBThal3jbhDCLGsE'})
downloaded.GetContentFile('OnlineNewsPopularityTrain.csv')
```

[271]:
```
!ls
```

```
adc.json                        sample_data     Vectors2.bin
MyHWWords.txt                   text8           Vectors.bin
OnlineNewsPopularityTest.csv    text8.bin       word2vec.model
OnlineNewsPopularityTrain.csv   text8-phrases
```

[0]:
```
X_train = np.genfromtxt('OnlineNewsPopularityTrain.csv', delimiter=',',␣
 ↪skip_header = 1, usecols = (i+1 for i in range(59)))
y_train = np.genfromtxt('OnlineNewsPopularityTrain.csv', delimiter=',',␣
 ↪skip_header = 1, usecols = (60) )

X_test = np.genfromtxt('OnlineNewsPopularityTest.csv', delimiter=',',␣
 ↪skip_header = 1, usecols = (i+1 for i in range(59)))
y_test = np.genfromtxt('OnlineNewsPopularityTest.csv', delimiter=',',␣
 ↪skip_header = 1, usecols = (60) )
```

[273]:
```
print('X train', X_train.shape)
print('y train', y_train.shape)

print('X test', X_test.shape)
print('X test', y_test.shape)
```

```
X train (38422, 59)
y train (38422,)
X test (1222, 59)
X test (1222,)
```

[0]:
```
def split_data_set(X,y,split=2,shuffle=False):

    num_samples = X.shape[0]
    samples_per_split = num_samples // split

    if y.ndim == 1:
        y = np.array([y])
        y = y.T

    if X.ndim == 1:
        X = np.array([X])
        X = X.T

    if shuffle:

        print(X.shape, y.shape)
```

```
        FullFrame = np.hstack( (X,y) )
        print(FullFrame.shape)
        np.random.shuffle(FullFrame)

        X = FullFrame[:,0:-1]
        y = np.array( [ FullFrame[:,59] ] )
        print(X.shape, y.shape)

    X_splits = []
    y_splits = []

    for i in range(split):

        if i != split-1:
            X_splits.append( X[ i*samples_per_split:(i+1)*samples_per_split, :
↪] )
            y_splits.append( np.array( [y[ i*samples_per_split:
↪(i+1)*samples_per_split, 0 ]]).T )
        else:
            X_splits.append( X[ i*samples_per_split:, : ] )
            y_splits.append( np.array( [y[ i*samples_per_split:, 0 ]]).T )

    return [X_splits , y_splits]
```

[0]:
```
folds = 5
Splits = split_data_set(X_train,y_train,split=folds, shuffle=False)

X_train_splits = Splits[0]
y_train_splits = Splits[1]
```

4 Decision Tree regression

[279]:
```
max_depth = [i+1 for i in range(20)]
avg_error = []

for d in max_depth:
    reg_model = tree.DecisionTreeRegressor(criterion='mse', splitter='best',
↪max_depth=d, min_samples_leaf=2000, random_state=50)

    errors = []
    for fold_test_i in range(folds):

        TestSet_X = X_train_splits[fold_test_i]

        TrainSet_X = []
        TrainSet_y = []
```

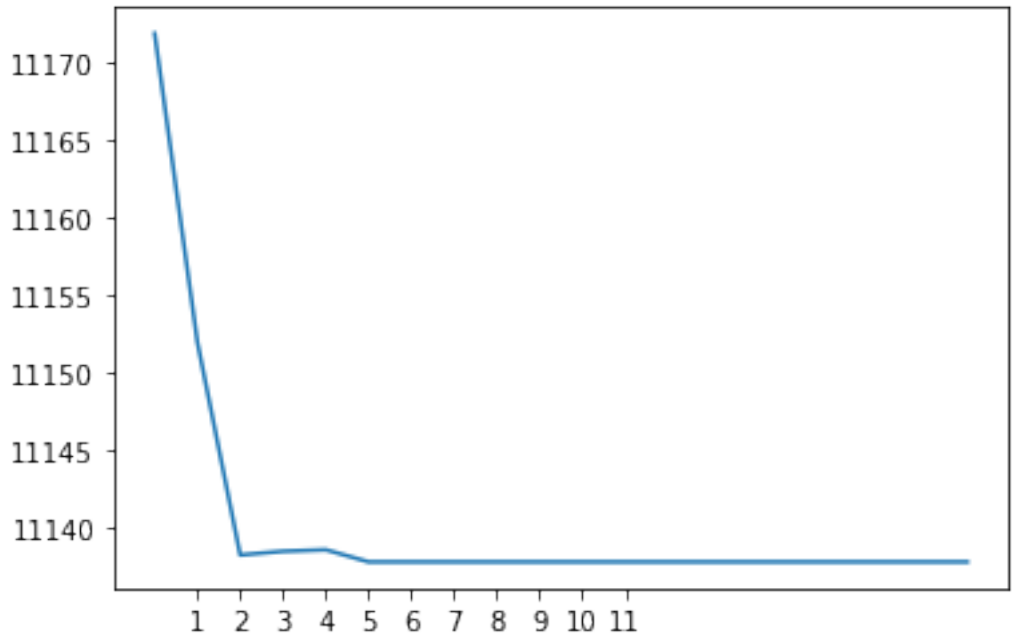
```
        for fold_train_i in range(folds):
            if fold_train_i != fold_test_i:
                TrainSet_X.append(X_train_splits[fold_train_i])
                TrainSet_y.append(y_train_splits[fold_train_i])

        TrainSet_X = np.vstack(TrainSet_X)
        TrainSet_y = np.vstack(TrainSet_y)

        reg_model.fit(TrainSet_X, TrainSet_y)

        predict_y = reg_model.predict(TestSet_X)

        act_y = y_train_splits[fold_test_i]
        rmse_error = mean_squared_error(predict_y, act_y, squared=False)

        errors.append(rmse_error)

    print('Depth =', d, '-------------')
    print('Errors in all the 5 folds:', errors)
    avg_error.append(np.mean(errors))
    print('Avg error -', np.mean(errors))

print(avg_error)
print(min(avg_error))
plt.xticks([1,2,3,4,5,6,7,8,9,10,11])
plt.plot(avg_error)
```

```
Depth = 1 -------------
Errors in all the 5 folds: [12704.522646716448, 12451.351711495125,
15982.790465630835, 6893.117134657534, 7827.603531315466]
Avg error - 11171.877097963083
Depth = 2 -------------
Errors in all the 5 folds: [12706.361503601018, 12409.630673317362,
15939.939116389296, 6866.115132557895, 7837.578632601012]
Avg error - 11151.925011693316
Depth = 3 -------------
Errors in all the 5 folds: [12692.374104878132, 12388.103599184531,
15927.67289966213, 6855.073213100711, 7827.648071444609]
Avg error - 11138.174377654022
Depth = 4 -------------
Errors in all the 5 folds: [12694.239972166159, 12397.997397379619,
15922.120784496236, 6852.498502277804, 7825.210225582775]
Avg error - 11138.413376380518
Depth = 5 -------------
Errors in all the 5 folds: [12694.503892741612, 12398.321141261533,
15923.260313107681, 6850.399814243473, 7826.081852938912]
Avg error - 11138.513402858644
```

```
Depth = 6 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 7 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 8 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 9 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 10 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 11 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 12 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 13 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 14 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 15 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 16 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 17 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
```

```
Depth = 18 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 19 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
Depth = 20 -------------
Errors in all the 5 folds: [12694.040672404715, 12398.321141261533,
15922.251794519083, 6849.378332033277, 7824.653897654285]
Avg error - 11137.729167574578
[11171.877097963083, 11151.925011693316, 11138.174377654022, 11138.413376380518,
11138.513402858644, 11137.729167574578, 11137.729167574578, 11137.729167574578,
11137.729167574578, 11137.729167574578, 11137.729167574578, 11137.729167574578,
11137.729167574578, 11137.729167574578, 11137.729167574578, 11137.729167574578,
11137.729167574578, 11137.729167574578, 11137.729167574578, 11137.729167574578]
11137.729167574578
```

[279]:
```
[<matplotlib.lines.Line2D at 0x7fcc6f8cb470>]
```



[280]:
```
# Best depth = 6
reg_model = tree.DecisionTreeRegressor(criterion='mse', splitter='best',␣
 ↪max_depth=6, min_samples_leaf=2000, random_state=50)
reg_model.fit(X_train, y_train)
predictions = reg_model.predict(X_test)
```

```
errorOnTest = mean_squared_error(predictions, y_test)
print('Testset error =', error_on_testSet)
```

```
Testset error = 8329.687133026458
```

5 Random forest regression

[0]:
```
class Random_Forest:

    def __init__(self, Train_file_loc):
        self.Train_file_loc = Train_file_loc

    def create_tree_parameters(self, num_features_per_tree, Max_features,␣
↪num_trees = 10, depth = 2):

        self.num_trees = num_trees
        self.Max_features = Max_features
        self.num_features_per_tree = num_features_per_tree
        self.depth = depth

        self.trees_features_to_use = []
        for i in range(num_trees):

            treeX_params = []
            for j in range(num_features_per_tree):
                feature = randint(1, Max_features)
                treeX_params.append(feature)

            self.trees_features_to_use.append( tuple(treeX_params) )

    def create_trees(self):

        self.tree_models = []

        for t in range(self.num_trees):

            reg_model = tree.DecisionTreeRegressor(criterion='mse',␣
↪splitter='best', max_depth=self.depth, min_samples_leaf=2000,␣
↪random_state=30)
            self.tree_models.append(reg_model)

    def fit_rf(self, Full_TrainSet_X, y_train):

        for t in range(self.num_trees):
            X_train = []
            for col in self.trees_features_to_use[t]:
```

```
                colX = Full_TrainSet_X[:,col]
                colX = np.array([colX]).T
                X_train.append(colX)

            X_train = np.hstack(X_train)

            np.random.shuffle(X_train)
            numDataToTake = X_train.shape[0] // 2
            X_train = X_train[0:numDataToTake,:]

            self.tree_models[t].fit(X_train, y_train[0:numDataToTake])


    def predict_rf(self, Full_TestSet_X):

        predictions = []

        for t in range(self.num_trees):

            X_test = []
            for col in self.trees_features_to_use[t]:
                colX = Full_TestSet_X[:,col]
                colX = np.array([colX]).T
                X_test.append(colX)

            X_test = np.hstack(X_test)

            prediction = self.tree_models[t].predict(X_test)
            predictions.append(prediction)

        mean_predictions = predictions[0]
        for i,pred in enumerate(predictions):
            if i == 0:
                continue

            mean_predictions = mean_predictions + pred

        mean_predictions = mean_predictions / self.num_trees

        return mean_predictions
```

[282]:
```
max_depth = [2,5,7,10,15,20]
max_trees = [3,5,10,15,20]

avg_errors_2D = np.zeros( ( len(max_trees) , len(max_depth) ))
avg_errors = []
loc_i = 0
```

```
for num_trees in max_trees:

    loc_j = 0
    for d in max_depth:

        print('Num Trees ---', num_trees, '   Depth ---', d)

        r_forest_model = Random_Forest('OnlineNewsPopularityTrain.csv')
        r_forest_model.create_tree_parameters(17,58, num_trees = num_trees,␣
 ↪depth=d)
        r_forest_model.create_trees()

        errors = []
        for fold_test_i in range(folds):

            TestSet_X = X_train_splits[fold_test_i]

            TrainSet_X = []
            TrainSet_y = []
            for fold_train_i in range(folds):
                if fold_train_i != fold_test_i:
                    TrainSet_X.append(X_train_splits[fold_train_i])
                    TrainSet_y.append(y_train_splits[fold_train_i])

            TrainSet_X = np.vstack(TrainSet_X)
            TrainSet_y = np.vstack(TrainSet_y)

            r_forest_model.fit_rf(TrainSet_X, TrainSet_y)

            predict_y = r_forest_model.predict_rf(TestSet_X)

            act_y = y_train_splits[fold_test_i]
            rmse_error = mean_squared_error(predict_y, act_y, squared= False)

            errors.append(rmse_error)

        print('Errors in all the 5 folds:', errors)
        MeanError_One_Full_CrossVal = np.mean(errors)
        avg_errors.append(MeanError_One_Full_CrossVal)
        avg_errors_2D[loc_i][loc_j] = MeanError_One_Full_CrossVal
        print('Avg error -', MeanError_One_Full_CrossVal)

        loc_j += 1

    loc_i += 1
```

```
Num Trees --- 3    Depth --- 2
```

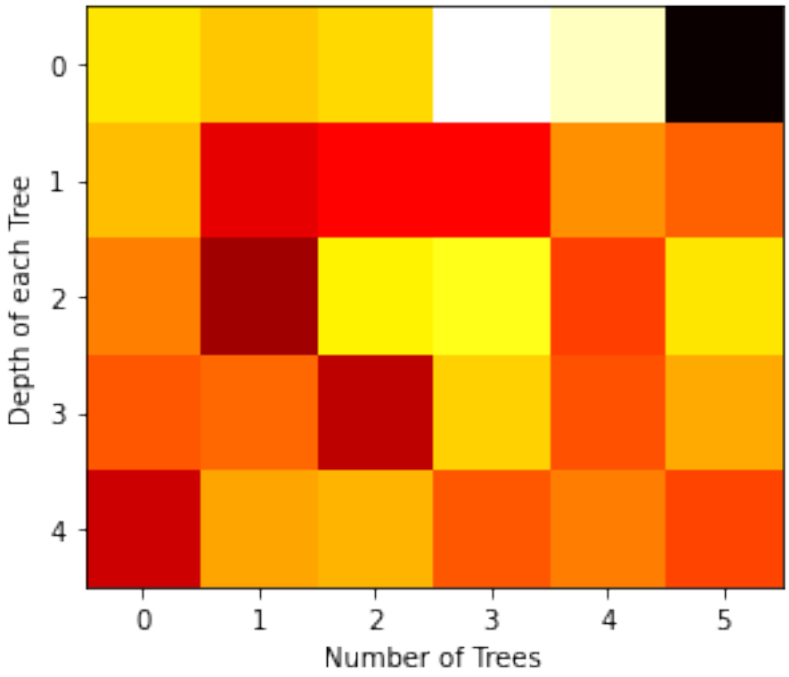
```
Errors in all the 5 folds: [12724.046569362174, 12498.377628849306,
16062.147746609337, 6984.6714602121365, 7868.297952735844]
Avg error - 11227.508271553761
Num Trees --- 3    Depth --- 5
Errors in all the 5 folds: [12717.421134833665, 12509.636647998374,
16062.100516001983, 6978.993940040527, 7865.722467818247]
Avg error - 11226.774941338559
Num Trees --- 3    Depth --- 7
Errors in all the 5 folds: [12729.960888094924, 12508.229812025074,
16057.208917594005, 6957.839438893067, 7882.767327728526]
Avg error - 11227.201276867121
Num Trees --- 3    Depth --- 10
Errors in all the 5 folds: [12720.974951614733, 12506.914848596572,
16068.924199212375, 6980.530195042351, 7882.1058403258]
Avg error - 11231.890006958367
Num Trees --- 3    Depth --- 15
Errors in all the 5 folds: [12736.580210990749, 12504.063949978168,
16055.340801016533, 6977.988207399826, 7880.6873418044]
Avg error - 11230.932102237934
Num Trees --- 3    Depth --- 20
Errors in all the 5 folds: [12713.103941631105, 12476.988658826376,
16070.386553947252, 6962.440116396896, 7860.761246478012]
Avg error - 11216.736103455927
Num Trees --- 5    Depth --- 2
Errors in all the 5 folds: [12717.374210813774, 12512.121654084862,
16053.308646219277, 6972.76813092451, 7877.241304158912]
Avg error - 11226.562789240266
Num Trees --- 5    Depth --- 5
Errors in all the 5 folds: [12716.395747578255, 12491.140513181628,
16060.160396596337, 6978.908565968395, 7862.041564234389]
Avg error - 11221.7293575118
Num Trees --- 5    Depth --- 7
Errors in all the 5 folds: [12721.110450531985, 12484.11353808428,
16061.025250846578, 6968.017413515005, 7877.477406867283]
Avg error - 11222.348811969026
Num Trees --- 5    Depth --- 10
Errors in all the 5 folds: [12719.294892272896, 12486.876999201417,
16056.381162213202, 6983.351793961469, 7865.7678772885965]
Avg error - 11222.334544987516
Num Trees --- 5    Depth --- 15
Errors in all the 5 folds: [12722.913979844565, 12488.194092994105,
16059.488148218212, 6975.070780648936, 7881.906879184975]
Avg error - 11225.514776178159
Num Trees --- 5    Depth --- 20
Errors in all the 5 folds: [12722.129675240943, 12497.336716576261,
16064.848800685859, 6971.981176858201, 7865.955876178577]
Avg error - 11224.450449107968
Num Trees --- 10    Depth --- 2
```

```
Errors in all the 5 folds: [12712.155928706066, 12499.075304895754,
16059.07557619714, 6969.964870291187, 7885.653685318607]
Avg error - 11225.18507308175
Num Trees --- 10    Depth --- 5
Errors in all the 5 folds: [12718.368770488354, 12483.096121417097,
16049.944694799666, 6972.7534252401065, 7876.435979623928]
Avg error - 11220.119798313832
Num Trees --- 10    Depth --- 7
Errors in all the 5 folds: [12721.751417708138, 12498.775189647033,
16062.883799251036, 6972.922732022012, 7882.587823066019]
Avg error - 11227.784192338846
Num Trees --- 10    Depth --- 10
Errors in all the 5 folds: [12719.613119945994, 12503.198175096115,
16050.665764498137, 6978.679598306325, 7890.039502673226]
Avg error - 11228.43923210396
Num Trees --- 10    Depth --- 15
Errors in all the 5 folds: [12719.45960353457, 12500.719391478093,
16062.305381510532, 6968.148624367102, 7867.787784135403]
Avg error - 11223.68415700514
Num Trees --- 10    Depth --- 20
Errors in all the 5 folds: [12723.94728336258, 12490.716573580954,
16053.427223150667, 6982.504525903613, 7886.940676450378]
Avg error - 11227.507256489638
Num Trees --- 15    Depth --- 2
Errors in all the 5 folds: [12716.93934393705, 12496.046536089332,
16058.525006050677, 6967.461147559711, 7882.281855413725]
Avg error - 11224.250777810099
Num Trees --- 15    Depth --- 5
Errors in all the 5 folds: [12717.566024429429, 12498.87459405616,
16053.607219195323, 6973.538557210984, 7879.616933885491]
Avg error - 11224.640665755478
Num Trees --- 15    Depth --- 7
Errors in all the 5 folds: [12716.668702587649, 12496.45809411756,
16054.708087434208, 6957.445636004115, 7878.70179103626]
Avg error - 11220.79646223596
Num Trees --- 15    Depth --- 10
Errors in all the 5 folds: [12718.395024299842, 12506.286113297676,
16058.977792226473, 6971.594808633657, 7879.747239941456]
Avg error - 11227.000195679819
Num Trees --- 15    Depth --- 15
Errors in all the 5 folds: [12717.149540976648, 12503.531656445613,
16057.310916671524, 6963.696049501141, 7878.871653543204]
Avg error - 11224.111963427626
Num Trees --- 15    Depth --- 20
Errors in all the 5 folds: [12721.430288511228, 12492.536991063756,
16056.569943910641, 6979.329897635477, 7880.675956165997]
Avg error - 11226.10861545742
Num Trees --- 20    Depth --- 2
```

```
Errors in all the 5 folds: [12716.481463200564, 12491.441258965213,
16053.177599722942, 6965.1137557676375, 7879.436004464423]
Avg error - 11221.130016424157
Num Trees --- 20    Depth --- 5
Errors in all the 5 folds: [12720.392157915272, 12502.98880937526,
16056.245417239461, 6971.256399091026, 7879.158050609617]
Avg error - 11226.008166846128
Num Trees --- 20    Depth --- 7
Errors in all the 5 folds: [12717.666635633237, 12498.388067513815,
16055.77209645244, 6980.723179126257, 7879.3260758148435]
Avg error - 11226.375210908118
Num Trees --- 20    Depth --- 10
Errors in all the 5 folds: [12721.343569578332, 12497.849925980438,
16056.161900208152, 6973.468953047092, 7872.205573195397]
Avg error - 11224.205984401882
Num Trees --- 20    Depth --- 15
Errors in all the 5 folds: [12716.291724964734, 12501.580756484827,
16057.622875030178, 6975.466075426473, 7874.51458395017]
Avg error - 11225.095203171277
Num Trees --- 20    Depth --- 20
Errors in all the 5 folds: [12718.673877358071, 12493.89365513387,
16055.62782408774, 6973.471953051058, 7877.4408435627]
Avg error - 11223.821630638688
```

[284]:
```
plt.xlabel('Number of Trees')
plt.ylabel('Depth of each Tree')
plt.imshow(avg_errors_2D, cmap='hot')

np.min(avg_errors_2D)
```

[284]: 11216.736103455927



[286]:
```
# Number of Trees 3
# Depth 20

r_forest_Bestmodel = Random_Forest('OnlineNewsPopularityTrain.csv')
r_forest_Bestmodel.create_tree_parameters(17,58, num_trees = 3, depth=20)
r_forest_Bestmodel.create_trees()
r_forest_Bestmodel.fit_rf(X_train, y_train)

predictions = r_forest_Bestmodel.predict_rf(X_test)
error_on_testSet = mean_squared_error(predictions, y_test, squared=False)

print('Error on Test set -', error_on_testSet)
```

```
Error on Test set - 8448.342738846994
```

6 Feature exploration

[287]:
```
Best_Model_Q1 = tree.DecisionTreeRegressor(criterion='mse', splitter='best',␣
↪max_depth=6, min_samples_leaf=2000, random_state=42)
Best_Model_Q1.fit(TrainSet_X, TrainSet_y)
```

[287]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=6,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
```

```
                       min_samples_leaf=2000, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=42, splitter='best')
```

[288]:
```
n_nodes = Best_Model_Q1.tree_.node_count
children_left = Best_Model_Q1.tree_.children_left
children_right = Best_Model_Q1.tree_.children_right
feature = Best_Model_Q1.tree_.feature
threshold = Best_Model_Q1.tree_.threshold

print('Number of nodes in the decision tree = ', n_nodes)
```

```
Number of nodes in the decision tree =  21
```

[0]:
```
Features = ['timedelta', 'n_tokens_title', 'n_tokens_content',␣
 ↪'n_unique_tokens', 'n_non_stop_words', 'n_non_stop_unique_tokens',␣
 ↪'num_hrefs', 'num_self_hrefs', 'num_imgs', 'num_videos',␣
 ↪'average_token_length', 'num_keywords', 'data_channel_is_lifestyle',␣
 ↪'data_channel_is_entertainment', 'data_channel_is_bus',␣
 ↪'data_channel_is_socmed', 'data_channel_is_tech', 'data_channel_is_world',␣
 ↪'kw_min_min', 'kw_max_min', 'kw_avg_min', 'kw_min_max', 'kw_max_max',␣
 ↪'kw_avg_max', 'kw_min_avg', 'kw_max_avg', 'kw_avg_avg',␣
 ↪'self_reference_min_shares', 'self_reference_max_shares',␣
 ↪'self_reference_avg_sharess', 'weekday_is_monday', 'weekday_is_tuesday',␣
 ↪'weekday_is_wednesday', 'weekday_is_thursday', 'weekday_is_friday',␣
 ↪'weekday_is_saturday', 'weekday_is_sunday', 'is_weekend', 'LDA_00',␣
 ↪'LDA_01', 'LDA_02', 'LDA_03', 'LDA_04', 'global_subjectivity',␣
 ↪'global_sentiment_polarity', 'global_rate_positive_words',␣
 ↪'global_rate_negative_words', 'rate_positive_words', 'rate_negative_words',␣
 ↪'avg_positive_polarity', 'min_positive_polarity', 'max_positive_polarity',␣
 ↪'avg_negative_polarity', 'min_negative_polarity', 'max_negative_polarity',␣
 ↪'title_subjectivity', 'title_sentiment_polarity', 'abs_title_subjectivity',␣
 ↪'abs_title_sentiment_polarity']
```

[0]:
```
dotfile = open("dt.dot", 'w')
graphic = tree.export_graphviz(Best_Model_Q1, out_file=dotfile,␣
 ↪feature_names=Features)
dotfile.close()
```

7 NLP feature extraction

[0]:
```
urls = pandas.read_csv('OnlineNewsPopularityTrain.csv', usecols=[0])
urls_test = pandas.read_csv('OnlineNewsPopularityTest.csv', usecols=[0])
```

[0]:
```
AllUrls = []
for i in range(urls.shape[0]):
```

```
    urlX = urls['url'][i]
    AllUrls.append(urlX)

All_LastWords_train = []
for i in range(len(AllUrls)):
    lastword = AllUrls[i].split('/')[-2]
    lastwords = lastword.split('-')
    All_LastWords_train.append(lastwords)

AllUrls = []
for i in range(urls_test.shape[0]):
    urlX = urls_test['url'][i]
    AllUrls.append(urlX)

All_LastWords_test = []
for i in range(len(AllUrls)):
    lastword = AllUrls[i].split('/')[-2]
    lastwords = lastword.split('-')
    All_LastWords_test.append(lastwords)
```

[254]:
```
print(len(All_LastWords_train))
print(len(All_LastWords_test))
print(All_LastWords_test[0:3])
print(All_LastWords_train[0:3])
```

```
38422
1222
[['amazon', 'instant', 'video', 'browser'], ['ap', 'samsung', 'sponsored',
'tweets'], ['apple', '40', 'billion', 'app', 'downloads']]
[['entrepreneur', 'trends', '2013'], ['facebook', 'sick', 'app'], ['felt',
'audio', 'pulse', 'speaker']]
```

[0]:
```
from gensim.test.utils import common_texts, get_tmpfile
from gensim.models import Word2Vec

for w in All_LastWords_train:
  common_texts.append(w)

for w in All_LastWords_test:
  common_texts.append(w)

path = get_tmpfile("word2vec.model")
model = Word2Vec(common_texts, size=10, window=5, min_count=1, workers=4)
model.save("word2vec.model")
```

[0]:
```
model = Word2Vec.load("word2vec.model")
```

[0]:
```
vectors = []
for words in All_LastWords_train:
  vector = model.wv[words[0]]
  numWords = 1
  for i,word in enumerate(words):
    if i==0:
      continue
    vector != model.wv[word]
    numWords += 1

  meanVector = vector/numWords
  vectors.append(np.array([meanVector]))
```

[259]:
```
allUrlVectors = np.vstack(vectors)
X_train_with_Url = np.hstack( (allUrlVectors , X_train) )
print(X_train_with_Url.shape)
```

```
(38422, 69)
```

[0]:
```
vectors = []
for words in All_LastWords_test:
  vector = model.wv[words[0]]
  numWords = 1
  for i,word in enumerate(words):
    if i==0:
      continue
    vector != model.wv[word]
    numWords += 1

  meanVector = vector/numWords
  vectors.append(np.array([meanVector]))
```

[261]:
```
allUrlVectors = np.vstack(vectors)
X_test_with_Url = np.hstack( (allUrlVectors , X_test) )
print(X_test_with_Url.shape)
```

```
(1222, 69)
```

[265]:
```
dTreeModel = tree.DecisionTreeRegressor(criterion='mse', splitter='best',␣
↪max_depth=50, min_samples_leaf=2000, random_state=50)
dTreeModel.fit(X_train_with_Url, y_train)
```

[265]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=50,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=2000, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
```

```
                       random_state=50, splitter='best')
```

[266]:
```
predictions = dTreeModel.predict(X_test_with_Url)
error_on_testSet = mean_squared_error(predictions, y_test, squared=False)
print(error_on_testSet)
```

```
8329.687133026458
```