ensembleOfRandomForestsAndGradientBoostedTrees

August 10, 2016

In [1]:
```
#This Code trains an ensemble of Random Forests
#and Gradient Boosted trees then averages the prediction of the two.


import numpy as np
import pandas as pd
from copy import copy
from sklearn.ensemble import RandomForestRegressor
import csv
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingRegressor

dateparse=lambda x:pd.datetime.strptime(x,'%Y-%m-%d %H:%M:%S')
train=pd.read_csv('train.csv',parse_dates=['datetime'],date_parser=dateparse)
test=pd.read_csv('test.csv',parse_dates=['datetime'],date_parser=dateparse)

#This was required when the number of trees were less than 50. With more trees , almost same pe
#test['windspeed']=np.log(test['windspeed']+1)
#train['windspeed']=np.log(train['windspeed']+1)
print train.shape

def extractFeaturesTrain(data):
    #print 'data is ',data
    data['Hour']=data.datetime.dt.hour
    data['DayOfWeek']=data.datetime.dt.dayofweek
    #data['Month']=data.datetime.dt.month
    labels=data['count']
    train_years=data.datetime.dt.year
    train_months=data.datetime.dt.month
    data=data.drop(['datetime','count','casual','registered'], axis = 1)

    return np.array(data),np.array(labels),np.array(train_years),np.array(train_months),(data.c

def extractFeaturesTest(data):

    data['Hour']=data.datetime.dt.hour
    data['DayOfWeek']=data.datetime.dt.dayofweek
    #data['Month']=data.datetime.dt.month
    test_years=data.datetime.dt.year
    test_months=data.datetime.dt.month
    data=data.drop(['datetime'], axis = 1)
    return np.array(data),np.array(test_years),np.array(test_months)
```

```
train2=copy(train)
test2=copy(test)
test=np.array(test)
#print 'train2 is ',train2
traind,labelsTrain,train_years,train_months,headers=extractFeaturesTrain(train2)
testd,test_years,test_months=extractFeaturesTest(test2)

submit=np.array((test.shape[0],2))

#train.to_csv('Remodeled Train.csv')
train=np.array(train)
print 'train is \n',traind.shape
print 'labels train are \n',labelsTrain.shape
print 'test is \n',testd.shape

def findLocations(year,month):
    locs=[]
    for i in range(0,test.shape[0]):
        if(test[i][0].year==year and test[i][0].month==month):
            locs.append(i)

    return locs

def findValidDates(year,month):
    locs=[]
    for i in range(0,train.shape[0]):
        if(train[i][0].year<=year and train[i][0].month<=month):
            locs.append(i)

    return locs

'''for i in set(test_years):
    for j in set(test_months):
        print 'Year : ',i,' month ',j:
            testLocs=findLocations(i,j)
            testSubset=testd[testLocs]

            trainLocs=findValidDates(i,j)
            trainSubset=traind[trainLocs]'''

def findLoss(gold,predicted):
    loss=0
    for i in range(gold.shape[0]):
        loss+=(np.log(predicted[i]+1) -np.log(gold[i]+1))**2

    loss=loss/gold.shape[0]
    #print 'loss is ',loss,' y_pred is ',predicted[i]
    return np.sqrt(loss)

def replaceNegaticeValuesWithZeroAndCountThem(ypred):
    count=0
    for i in range(ypred.shape[0]):
        if(ypred[i]<0):
```

```
            ypred[i]=0
            count+=1
    print 'Number of Negative values predicted are ',count
    return ypred,count


rf=GradientBoostingRegressor()
split1=0.8*traind.shape[0]
trainSplit=traind[:split1,:]

testSplit=traind[split1:,:]
labelsSplitTrain=labelsTrain[:split1]
labelsSplitTest=labelsTrain[split1:]
rf.fit(trainSplit,labelsSplitTrain)
ypred=rf.predict(testSplit)
ypred,count=replaceNegaticeValuesWithZeroAndCountThem(ypred)
print 'trainSplit is \n',trainSplit.shape,' and testSplit is \n',testSplit.shape
print 'ypred is \n',ypred
print 'test split is \n',labelsSplitTest
print 'the loss is ',findLoss(labelsSplitTest,ypred)




rf.fit(traind,labelsTrain)
#print 'rf.estimators_ are ',rf.estimators_
print 'testd shape is ',testd.shape
ypred2=rf.predict(testd)
with open('submit2.csv', 'wb') as csvfile:
    resultWriter= csv.writer(csvfile)
    l=['datetime','count']
    resultWriter.writerow(l)
    for i in range(testd.shape[0]):
        #print 'test[',i,'][0] is ',test[i,0]
        l=[test[i,0],ypred2[i]]
        resultWriter.writerow(l)

allEstimators=rf.estimators_
allEstimators=allEstimators.reshape(1,-1)
allEstimators=allEstimators.tolist()

#print '2 rf.estimators_ are ',allEstimators[0]



importances=rf.feature_importances_
std=np.std([tree.feature_importances_ for tree in allEstimators[0]],axis=0)
indices=np.argsort(importances)[::-1]
print 'Feature Ranking\n'

for f in range(traind.shape[1]):
    print("%d. feature %d %s (%f)" % (f + 1, indices[f],headers[indices[f]], importances[indices
```

```
        fig, ax = plt.subplots()

        ax.set_title('Feature Importances By Gradient Boosted Trees')
        ax.bar(range(traind.shape[1]),importances[indices],color="b",yerr=std[indices],align='center')
        plt.xticks(range(traind.shape[1]), indices)
        ax.set_xlim([-1, traind.shape[1]])
        ax.set_xticklabels(headers[indices])
        plt.savefig('Feature Importances By Gradient Boosted Trees')
        plt.show()
```

```
(10886, 12)
train is
(10886, 10)
labels train are
(10886,)
test is
(6493, 10)
Number of Negative values predicted are  24
trainSplit is
(8708, 10)  and testSplit is
(2178, 10)
ypred is
[  45.96415231   45.28381731  185.60055065 ...,  137.16521297  130.73808505
   94.32587411]
test split is
[ 19  19  68 ..., 168 129  88]
the loss is  0.639331761758
testd shape is  (6493, 10)
Feature Ranking

1. feature 8 Hour (0.490237)
2. feature 2 workingday (0.117948)
3. feature 6 humidity (0.097533)
4. feature 4 temp (0.079519)
5. feature 9 DayOfWeek (0.073352)
6. feature 5 atemp (0.047066)
7. feature 0 season (0.043905)
8. feature 3 weather (0.023082)
9. feature 7 windspeed (0.020010)
10. feature 1 holiday (0.007349)

/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:102: DeprecationWarning: using a non-integer
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:104: DeprecationWarning: using a non-integer
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:105: DeprecationWarning: using a non-integer
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:106: DeprecationWarning: using a non-integer
```

In [2]:
```
        rf=RandomForestRegressor(150)
        gbt=GradientBoostingRegressor(n_estimators=300)


        print 'test_years are ',set(test_years)
        def getTestLocs(year,month):

            locs=[]
```

```
    print 'In testlocs year is =',year,' month is = ',month
    for i in range(0,test.shape[0]):
        if test[i][0].year==year and test[i][0].month==month:
            locs.append(i)
    return locs

with open('submitEnsembleOfRandomForestsAndGBT.csv','wb') as csvfile:
    resultWriter=csv.writer(csvfile)
    l=['datetime','count']
    resultWriter.writerow(l)
    for i in set(test_years):
        for j in set(test_months):
                testLocs=getTestLocs(i,j)
                #print 'testLoics are ',testLocs

                testSubset1=testd[testLocs]
                #print 'testSubset1 is ',testSubset1
                testSubset2=test[testLocs]
                #print 'testSubset2 is ',min(testSubset2[:,0])
                #print 'testSubset2 is ',testSubset2
                trainLocs=np.where(train[:,0]<=min(testSubset2[:,0]))
                trainSubset=traind[trainLocs]
                labelsSubset=labelsTrain[trainLocs]
                rf.fit(trainSubset,labelsSubset)
                gbt.fit(trainSubset,labelsSubset)

                #ypred3=rf2.predict(testSubset1)
                #print 'Training Random Forest '
                ypredRf=rf.predict(testSubset1)
                #print 'Training Gradient Boosted Trees '
                ypredGbt=gbt.predict(testSubset1)

                print 'For Random Forest year ',i,' month ',j
                ypredRf,count1=replaceNegaticeValuesWithZeroAndCountThem(ypredRf)
                print 'For Grandient Boosted Trees year ',i,' month ',j
                ypredGbt,count2=replaceNegaticeValuesWithZeroAndCountThem(ypredGbt)
                yensemble=[]
                #print 'ypredRf is ',type(ypredRf),' \n and ypredGbt is ',type(ypredGbt)
                for m in range(ypredRf.shape[0]):
                    if ypredRf[m]>0 and ypredGbt[m]>0:
                        yensemble.append(ypredRf[m]/2+ypredGbt[m]/2)
                    elif ypredRf[m]>0 and ypredGbt[m]==0:
                        yensemble.append(ypredRf[m])
                    elif ypredRf[m]==0 and ypredGbt[m]>0:
                        yensemble.append(ypredGbt[m])
                    else:
                        yensemble.append(0)


                for k in range(0,testSubset2.shape[0]):
                    l=[testSubset2[k,0],yensemble[k]]
                    resultWriter.writerow(l)
```

```
test_years are  set([2011, 2012])
In testlocs year is = 2011  month is =  1
For Random Forest year  2011  month  1
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2011  month  1
Number of Negative values predicted are  14
In testlocs year is = 2011  month is =  2
For Random Forest year  2011  month  2
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2011  month  2
Number of Negative values predicted are  4
In testlocs year is = 2011  month is =  3
For Random Forest year  2011  month  3
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2011  month  3
Number of Negative values predicted are  13
In testlocs year is = 2011  month is =  4
For Random Forest year  2011  month  4
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2011  month  4
Number of Negative values predicted are  8
In testlocs year is = 2011  month is =  5
For Random Forest year  2011  month  5
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2011  month  5
Number of Negative values predicted are  11
In testlocs year is = 2011  month is =  6
For Random Forest year  2011  month  6
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2011  month  6
Number of Negative values predicted are  0
In testlocs year is = 2011  month is =  7
For Random Forest year  2011  month  7
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2011  month  7
Number of Negative values predicted are  0
In testlocs year is = 2011  month is =  8
For Random Forest year  2011  month  8
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2011  month  8
Number of Negative values predicted are  0
In testlocs year is = 2011  month is =  9
For Random Forest year  2011  month  9
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2011  month  9
Number of Negative values predicted are  4
In testlocs year is = 2011  month is =  10
For Random Forest year  2011  month  10
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2011  month  10
Number of Negative values predicted are  4
In testlocs year is = 2011  month is =  11
For Random Forest year  2011  month  11
Number of Negative values predicted are  0
```

```
For Grandient Boosted Trees year  2011  month  11
Number of Negative values predicted are  6
In testlocs year is = 2011  month is =  12
For Random Forest year  2011  month  12
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2011  month  12
Number of Negative values predicted are  25
In testlocs year is = 2012  month is =  1
For Random Forest year  2012  month  1
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  1
Number of Negative values predicted are  38
In testlocs year is = 2012  month is =  2
For Random Forest year  2012  month  2
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  2
Number of Negative values predicted are  11
In testlocs year is = 2012  month is =  3
For Random Forest year  2012  month  3
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  3
Number of Negative values predicted are  16
In testlocs year is = 2012  month is =  4
For Random Forest year  2012  month  4
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  4
Number of Negative values predicted are  9
In testlocs year is = 2012  month is =  5
For Random Forest year  2012  month  5
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  5
Number of Negative values predicted are  3
In testlocs year is = 2012  month is =  6
For Random Forest year  2012  month  6
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  6
Number of Negative values predicted are  8
In testlocs year is = 2012  month is =  7
For Random Forest year  2012  month  7
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  7
Number of Negative values predicted are  13
In testlocs year is = 2012  month is =  8
For Random Forest year  2012  month  8
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  8
Number of Negative values predicted are  0
In testlocs year is = 2012  month is =  9
For Random Forest year  2012  month  9
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  9
Number of Negative values predicted are  8
In testlocs year is = 2012  month is =  10
For Random Forest year  2012  month  10
```

```
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  10
Number of Negative values predicted are  3
In testlocs year is = 2012  month is =  11
For Random Forest year  2012  month  11
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  11
Number of Negative values predicted are  3
In testlocs year is = 2012  month is =  12
For Random Forest year  2012  month  12
Number of Negative values predicted are  0
For Grandient Boosted Trees year  2012  month  12
Number of Negative values predicted are  36
```

```
In [3]: def getSplits(years,months):
            locsTrain=[]
            locsTest=[]
            for i in range(0,train.shape[0]):
                    if (train[i,0].year==years[0] or train[i,0].year==years[1]) and (train[i,0].month in
                        locsTest.append(i)
                    else:
                        locsTrain.append(i)

            return locsTrain,locsTest

        def getCustomLocsTest(year,month,data):
            locs=[]
            for i in range(0,data.shape[0]):
                if data[i][0].year==year and data[i][0].month==month:
                    locs.append(i)
            return locs

        def crossValidate():
                months=[12]
                locsTrain,locsTest=getSplits([2011,2012],months)

                testSubset=traind[locsTest]
                testSubset2=train[locsTest]
                testLabels=labelsTrain[locsTest]
                rf=RandomForestRegressor(100)
                gbt=GradientBoostingRegressor(n_estimators=500)

                trainSubset=traind[locsTrain]
                trainSubset2=train[locsTrain]
                trainLabels=labelsTrain[locsTrain]

                for i in [2011,2012]:
                    for j in months:
                        testLocs=getCustomLocsTest(i,j,testSubset2)
                        testSubset3=testSubset2[testLocs]
                        testSubset4=testSubset[testLocs]
                        testLabels4=testLabels[testLocs]

                        trainLocs2=np.where(trainSubset2[:,0]<=min(testSubset3[:,0]))
```

```
                    trainSubset3=trainSubset[trainLocs2]
                    trainLabels3=trainLabels[trainLocs2]
                    x1=trainSubset2[trainLocs2]
                    x2=testSubset2[testLocs]

                    print 'trainSubset min is  ', min(x1[:,0]),' and max is ',max(x1[:,0])
                    print 'testSubset  min is  ', min(x2[:,0]),' and max is ',max(x2[:,0])

                    rf.fit(trainSubset3,trainLabels3)
                    gbt.fit(trainSubset3,trainLabels3)

                    ypredRf=rf.predict(testSubset4)
                    ypredGbt=gbt.predict(testSubset4)

                    ypredRf,count1=replaceNegaticeValuesWithZeroAndCountThem(ypredRf)
                    ypredGbt,count2=replaceNegaticeValuesWithZeroAndCountThem(ypredGbt)
                    yensemble=[]
                    #print 'ypredRf is ',type(ypredRf),' \n and ypredGbt is ',type(ypredGbt)
                    for i in range(ypredRf.shape[0]):
                        if ypredRf[i]>0 and ypredGbt[i]>0:
                            yensemble.append(ypredRf[i]/2+ypredGbt[i]/2)
                        elif ypredRf[i]>0 and ypredGbt[i]==0:
                            yensemble.append(ypredRf[i])
                        elif ypredRf[i]==0 and ypredGbt[i]>0:
                            yensemble.append(ypredGbt[i])
                        else:
                            yensemble.append(0) #change here to append the mean sales of the week/m


                    print 'Random Forest loss with year =',i,' and month = ',j,' is ',findLoss(testl
                    print 'Gradient Boosted Trees loss with year =',i,' and month = ',j,' is ',findl
                    print 'Ensemble og Random Forest Gradient Boosted Trees loss with year =',i,' a
        crossValidate()
```

```
trainSubset min is   2011-01-01 00:00:00  and max is  2011-11-19 23:00:00
testSubset  min is   2011-12-01 00:00:00  and max is  2011-12-19 23:00:00
Number of Negative values predicted are  0
Number of Negative values predicted are  5
Random Forest loss with year = 455  and month =  12  is  0.449498148039
Gradient Boosted Trees loss with year = 455  and month =  12  is  0.499525255609
Ensemble og Random Forest Gradient Boosted Trees loss with year = 455  and month =  12  is  0.425085880
trainSubset min is   2011-01-01 00:00:00  and max is  2012-11-19 23:00:00
testSubset  min is   2012-12-01 00:00:00  and max is  2012-12-19 23:00:00
Number of Negative values predicted are  0
Number of Negative values predicted are  9
Random Forest loss with year = 455  and month =  12  is  0.362786611535
Gradient Boosted Trees loss with year = 455  and month =  12  is  0.591045997747
Ensemble og Random Forest Gradient Boosted Trees loss with year = 455  and month =  12  is  0.386895734
```