gbtrees

August 10, 2016

```
In [20]: #This code trains the  Gradient Boosted Trees for predicting the bike sales
         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 p

         #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.

         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(n_estimators=200)
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[indic



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  32
trainSplit is
(8708, 10)  and testSplit is
(2178, 10)
ypred is
[  34.27567496   35.7852371   158.05560733 ...,  135.21527701  131.96349257
   89.58574943]
test split is
[ 19  19  68 ..., 168 129  88]
the loss is  0.659359778897
testd shape is  (6493, 10)
Feature Ranking

1. feature 8 Hour (0.417246)
2. feature 9 DayOfWeek (0.116821)
3. feature 6 humidity (0.096584)
4. feature 2 workingday (0.093433)
5. feature 4 temp (0.085545)
6. feature 5 atemp (0.065820)
7. feature 0 season (0.047824)
8. feature 7 windspeed (0.039049)
9. feature 3 weather (0.024028)
10. feature 1 holiday (0.013651)

/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 [3]: 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
```

```
In [4]: set(test_years)
```

```
Out[4]: {2011, 2012}
```

```
In [19]: rf2=GradientBoostingRegressor(n_estimators=300)

         with open('submit3.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]
                         testSubset2=test[testLocs]
                         #print 'testSubset2 is ',testSubset2
                         trainLocs=np.where(train[:,0]<=min(testSubset2[:,0]))
                         trainSubset=traind[trainLocs]
                         labelsSubset=labelsTrain[trainLocs]
                         rf2.fit(trainSubset,labelsSubset)
                         ypred3=rf2.predict(testSubset1)
                         ypred3,count=replaceNegaticeValuesWithZeroAndCountThem(ypred3)

                         for k in range(0,testSubset2.shape[0]):
                             l=[testSubset2[k,0],ypred3[k]]
                             resultWriter.writerow(l)
```

```
In testlocs year is = 2011  month is =  1
Number of Negative values predicted are  14
In testlocs year is = 2011  month is =  2
Number of Negative values predicted are  4
In testlocs year is = 2011  month is =  3
Number of Negative values predicted are  13
In testlocs year is = 2011  month is =  4
Number of Negative values predicted are  9
In testlocs year is = 2011  month is =  5
Number of Negative values predicted are  13
In testlocs year is = 2011  month is =  6
Number of Negative values predicted are  0
In testlocs year is = 2011  month is =  7
Number of Negative values predicted are  0
In testlocs year is = 2011  month is =  8
Number of Negative values predicted are  0
In testlocs year is = 2011  month is =  9
Number of Negative values predicted are  4
In testlocs year is = 2011  month is =  10
Number of Negative values predicted are  4
In testlocs year is = 2011  month is =  11
Number of Negative values predicted are  6
In testlocs year is = 2011  month is =  12
Number of Negative values predicted are  25
In testlocs year is = 2012  month is =  1
Number of Negative values predicted are  38
In testlocs year is = 2012  month is =  2
```

```
Number of Negative values predicted are  11
In testlocs year is = 2012  month is =  3
Number of Negative values predicted are  16
In testlocs year is = 2012  month is =  4
Number of Negative values predicted are  9
In testlocs year is = 2012  month is =  5
Number of Negative values predicted are  3
In testlocs year is = 2012  month is =  6
Number of Negative values predicted are  8
In testlocs year is = 2012  month is =  7
Number of Negative values predicted are  13
In testlocs year is = 2012  month is =  8
Number of Negative values predicted are  0
In testlocs year is = 2012  month is =  9
Number of Negative values predicted are  8
In testlocs year is = 2012  month is =  10
Number of Negative values predicted are  3
In testlocs year is = 2012  month is =  11
Number of Negative values predicted are  3
In testlocs year is = 2012  month is =  12
Number of Negative values predicted are  36
```

```
In [18]: 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
                         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]
                 rf3=GradientBoostingRegressor(n_estimators=300)
                 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])

                #print 'trainSubset3 is ',trainSubset3,'\ntrainLabels3 are  ',trainLabels3
                rf3.fit(trainSubset3,trainLabels3)
                #print 'trained'
                ypred=rf3.predict(testSubset4)
                ypred,count=replaceNegaticeValuesWithZeroAndCountThem(ypred)
                print 'loss with year =',i,' and month = ',j,' is ',findLoss(testLabels4,ypred


crossValidate()
```

```
Number of Negative values predicted are  4
loss with year = 2011  and month =  12  is  0.498786860892
Number of Negative values predicted are  7
loss with year = 2012  and month =  12  is  0.552550269169
```

In [ ]:

```
dataTrain=np.array(train)
dataTrain.shape
plt.plot(dataTrain[:,1],dataTrain[:,11],'*')
plt.show()
```