plotPredictions

August 10, 2016

```
In [1]: #This code plots the  training predictions vs the actual predictions

        import numpy as np
        from mpl_toolkits.mplot3d import Axes3D
        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.preprocessing import OneHotEncoder
        import itertools
        from sklearn.svm import SVR
        from sklearn.decomposition import PCA
        import matplotlib.patches as mpatches

        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)

        #test['windspeed']=np.log(test['windspeed']+1)
        #train['windspeed']=np.log(train['windspeed']+1)
        print 'train.shape is ',train.shape,' and test.shape is ',test.shape


        def extractFeaturesTrain(data):

            data['Hour']=data.datetime.dt.hour
            data['DayOfWeek']=data.datetime.dt.dayofweek

            labels=data['count']
            train_years=data.datetime.dt.year
            train_months=data.datetime.dt.month
            data=data.drop(['datetime','count','casual','registered'], axis = 1)
            #print 'Training data is \n',data
            return np.array(data),np.array(labels),np.array(train_years),np.array(train_months),(data.c

        def extractFeaturesTest(data):
            #print 'data is \n',data
            data['Hour']=data.datetime.dt.hour
            data['DayOfWeek']=data.datetime.dt.dayofweek
            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,originalTrain,seasonTrain,hoursTrain=extrac
        testd,test_years,test_months=extractFeaturesTest(test2)

        cov1=np.cov(traind.T)


        print 'traind.shape is ',traind.shape,' and testd.shape is ',testd.shape
```

```
train.shape is  (10886, 12)  and test.shape is  (6493, 9)
traind.shape is  (10886, 10)  and testd.shape is  (6493, 10)
```

```
In [4]: predicted=pd.read_csv('submitrf150.csv',parse_dates=['datetime'],date_parser=dateparse)
        #print predicted
        predicted=np.array(predicted)
        #predicted[0,0].year
```

```
In [22]: trainA=np.array(train)

         palette=np.array(sns.color_palette("Set2", 2))
         LabelsName=['Train','PredictedTest']
         print 'predicted[0,0] is ',predicted[0,0]


         patches=[mpatches.Patch(color=palette[i]) for i in range(0,2)]
         def getCustomLocs(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


         #now my task is to plot the predicted values and the actual values for each month
         def plots():
             for i in [2011,2012]:
                 fig=plt.figure()
                 fig.suptitle('Year '+str(i)+' Monthly plots by season')
                 fig.subplots_adjust(hspace=0.5)
                 fig.legend(handles=patches,labels=LabelsName,loc='upper right')
                 fig.text(0.5,0.04,'-------------------------------------------------------------------
                 fig.text(0.08,0.5,'-----------------------------------------------------Count ( Number

                 for j in range(1,13):
                     trainLocs=getCustomLocs(i,j,trainA)
                     testLocs=getCustomLocs(i,j,predicted)
                     actualTrain=labelsTrain[trainLocs]
                     predictedTest=predicted[testLocs,1]
                     colorArray=np.hstack((np.zeros(len(trainLocs)),np.ones(len(testLocs))))
```

```
            time_series=np.hstack((actualTrain,predictedTest))
            #print colorArray
            ax=fig.add_subplot(4,3,j)
            ax.scatter(range(0,time_series.shape[0]),time_series,c=palette[colorArray.astype(np
            title=’ Month ’+str(j)
            #print ’title is ’,title
            ax.set_title(title)
        plt.savefig(’ Year ’+str(i) +’ Monthly plots by season (Train and Predicted)’)
        plt.show()

plots()
```

```
predicted[0,0] is  2011-01-20 00:00:00
```

In [ ]:

```
trainA=np.array(train)

flatui = [ "#e74c3c",  "#2ecc71"]

predictedTrain=pd.read_csv(’rf150predictedTrain.csv’,parse_dates=[’datetime’],date_parser=datepa
predictedTrain
predictedTrain=np.array(predictedTrain)
palette=np.array(sns.color_palette(flatui))

LabelsName=[’Train’,’PredictedTrain’]
print ’predicted[0,0] is ’,predicted[0,0]


patches=[mpatches.Patch(color=palette[i]) for i in range(0,2)]

def getCustomLocs2(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 plots2():
    for i in [2011,2012]:
        fig=plt.figure()
        fig.suptitle(’Train Values and Actual Values Year ’+str(i)+’ Monthly plots by season’)
        fig.subplots_adjust(hspace=0.5)
        fig.legend(handles=patches,labels=LabelsName,loc=’upper right’)
        fig.text(0.5,0.04,’-------------------------------------------------------------------T
        fig.text(0.08,0.5,’-----------------------------------------------------Count ( Number

        for j in range(1,13):
            #trainLocs=getCustomLocs2(i,j,trainA)
            predictedTrainLocs=getCustomLocs2(i,j,predictedTrain)

            trainValues=predictedTrain[predictedTrainLocs,1]
            predictedValues=predictedTrain[predictedTrainLocs,2]

            #print colorArray
            ax=fig.add_subplot(4,3,j)
            ax.scatter(range(0,trainValues.shape[0]),trainValues,c=palette[0])
            ax.scatter(range(0,predictedValues.shape[0]),predictedValues,c=palette[1])

            title=' Month '+str(j)
            #print 'title is ',title
            ax.set_title(title)
        plt.savefig('Train Values and Actual Values Year '+str(i) +' Monthly plots by season (T
        plt.show()

plots2()
```

```
In [35]: trainA=np.array(train)

         flatui = [ "#9b59b6"]

         predictedTrain=pd.read_csv('rf150predictedTrain.csv',parse_dates=['datetime'],date_parser=datep
         predictedTrain
         predictedTrain=np.array(predictedTrain)
         palette=np.array(sns.color_palette(flatui))

         LabelsName=['Residuals']
         print 'predicted[0,0] is ',predicted[0,0]


         patches=[mpatches.Patch(color=palette[i]) for i in range(0,1)]

         def getCustomLocs3(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 plots3():
             for i in [2011,2012]:
                 fig=plt.figure()
                 fig.set_size_inches(18.5, 10.5)
                 fig.suptitle('Residuals Year '+str(i)+' Monthly plots by season')
                 fig.subplots_adjust(hspace=0.5)
                 fig.legend(handles=patches,labels=LabelsName,loc='upper right')
                 fig.text(0.5,0.04,'------------------------------------------------------------------
                 fig.text(0.08,0.5,'----------------------------------------------------Count ( Number

                 for j in range(1,13):
                     #trainLocs=getCustomLocs2(i,j,trainA)
                     predictedTrainLocs=getCustomLocs3(i,j,predictedTrain)

                     trainValues=predictedTrain[predictedTrainLocs,1]
                     predictedValues=predictedTrain[predictedTrainLocs,2]
                     residuals=predictedTrain[predictedTrainLocs,3]
                     #print colorArray
                     ax=fig.add_subplot(4,3,j)
                     ax.plot(range(0,trainValues.shape[0]),residuals,c=palette[0])
```

```
            #ax.scatter(range(0,predictedValues.shape[0]),predictedValues,c=palette[1])

            title=' Month '+str(j)
            #print 'title is ',title
            ax.set_title(title)
        plt.savefig('Residuals Year '+str(i) +' Monthly plots by season (Train and Predicted)'
        plt.show()

plots3()
```

```
predicted[0,0] is  2011-01-20 00:00:00
```