timeSeriesPlots

August 10, 2016

```
In [ ]: #This code plots the data for visualization purposes
        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)

        #Not required for data visualization
        #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)

eigs=np.linalg.eigvals(cov1)
print 'eigs are \n',eigs
for i in range(traind.shape[1]):
    print 'Feature : ',headers[i],' : eigenvalue : ',eigs[i]


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

```
In [ ]: 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

        palettle_seasons=np.array(sns.color_palette("hls",4))
        SeasonsName=['Spring','Summer','Fall','Winter']

        patches_season=[mpatches.Patch(color=palettle_seasons[i-1]) for i in range(1,5)]
        #first we will plot the pe rmonth chart for each year and set the seasons too

        trainArray=np.array(train)
        for i in [2011,2012]:
            fig=plt.figure()
            fig.set_size_inches(18.5, 10.5)
            fig.suptitle('Year '+str(i)+' Monthly plots by season')
            fig.subplots_adjust(hspace=0.5)
            fig.legend(handles=patches_season,labels=SeasonsName,loc='upper right')
            fig.text(0.5,0.04,'-------------------------------------------------------------------Time
            fig.text(0.08,0.5,'-----------------------------------------------------Count ( Number of b
            for j in range(1,13):
                locs=getCustomLocs(i,j,trainArray)
                print
                seasons=seasonTrain[locs]
                seasons=seasons-1;
                time_series=labelsTrain[locs]
                ax=fig.add_subplot(4,3,j)

                #plt.subplot(4,3,j)
                #time_series=pd.Series(time_series)
                print 'time_series.shape[0] is ',time_series.shape[0]
                ax.scatter(range(0,time_series.shape[0]),time_series,c=palettle_seasons[seasons.astype(
                title=' Month '+str(j)
                #print 'title is ',title
                ax.set_title(title)
                #plt.legend(handles=patches_season)
                #plt.title(title)
                #plt.show()
            #plt.legend(handles=patches_season)
            plt.savefig(' Year '+str(i) +' Monthly plots by season')
            plt.show()
```

```
In [ ]: # here we will plot the count by time scale and then cluster by hours , once per month and then

        palette_hours=np.array(sns.color_palette("hls",24))

        hourLabels=[]
        for i in range(0,24):
            hourLabels.append('Hour '+str(i))

        patches_hours=[mpatches.Patch(color=palette_hours[i]) for i in range(0,24)]

        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

        trainArray=np.array(train)
        for i in [2011,2012]:
            fig=plt.figure()
            fig.set_size_inches(18.5, 10.5)
            fig.suptitle('Year '+str(i)+' Monthly plots clustered by hour')
            fig.subplots_adjust(hspace=0.5)
            fig.legend(handles=patches_hours,labels=hourLabels,loc='upper right')
            fig.text(0.5,0.04,'-----------------------------------------------------------------Time ( 
            fig.text(0.08,0.5,'----------------------------------------------------Count ( Number of bi
            for j in range(1,13):
                locs=getCustomLocs2(i,j,trainArray)
                print
                hours=hoursTrain[locs]

                time_series=labelsTrain[locs]
                ax=fig.add_subplot(4,3,j)

                #plt.subplot(4,3,j)
                #time_series=pd.Series(time_series)
                #print 'time_series.shape[0], ',time_series.shape[0]
                ax.scatter(range(0,time_series.shape[0]),time_series,c=palette_hours[hours.astype(np.in
                title=' Month '+str(j)
                #print 'title is ',title
                ax.set_title(title)
                #plt.legend(handles=patches_season)
                #plt.title(title)
                #plt.show()
            #plt.legend(handles=patches_season)
            plt.savefig(' Year '+str(i) +' Monthly plots clustered by hour')
            plt.show()
```

```
In [ ]: #Here we plot the autocorrelation of each time step with the months

    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

    trainArray=np.array(train)
    for i in [2011,2012]:
        fig=plt.figure()
        fig.set_size_inches(18.5, 10.5)
        maxlags=10
        fig.suptitle('Year '+str(i)+' Monthly ACF plots '+' with lag of '+str(maxlags))
        fig.subplots_adjust(hspace=0.5)
        #fig.legend(handles=patches_hours,labels=hourLabels,loc='upper right')
        fig.text(0.5,0.04,'-----------------------------------------------------------------Time ( I
        fig.text(0.08,0.5,'------------------------------------------------------------------ACF---
        for j in range(1,13):
            locs=getCustomLocs3(i,j,trainArray)
            print
            hours=hoursTrain[locs]

            time_series=labelsTrain[locs]
            ax=fig.add_subplot(4,3,j)
            #plt.subplot(4,3,j)
            #plt.subplot(4,3,j)
            #time_series=pd.Series(time_series)
            time_series=np.float32(time_series)
            #ax.scatter(time_series,range(0,time_series.shape[0]),c=palette_hours[hours.astype(np.i

            ax.acorr(time_series,usevlines=True, normed=True, maxlags=maxlags, lw=2)
            title=' Month '+str(j)
            #print 'title is ',title
            ax.set_title(title)
            #plt.title(title)
            #plt.legend(handles=patches_season)
            #plt.title(title)
            #plt.show()
        #plt.legend(handles=patches_season)
        plt.savefig(' Year '+str(i) +' Monthly ACF plots' +' with lag of '+str(maxlags))
        plt.show()
```