svrkaggle2

August 10, 2016

```
In [ ]: #This code trains the SVR model using Gaussian Kernel for predicting the bike sales

        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

        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 made very little difference for the SVR
        #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):
            #print ’data is ’,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)

            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=extractFeaturesTrain(train2)
        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 [ ]: enc=OneHotEncoder(categorical_features=[0],sparse=False)
        traind2=enc.fit_transform(traind)
        print traind2.shape
        testd2=enc.fit_transform(testd)
        print testd2.shape
        ones1=np.ones((traind.shape[0],1))
        ones2=np.ones((testd.shape[0],1))
        traind2=copy(np.hstack((traind2,ones1)))
        testd2=copy(np.hstack((testd2,ones2)))
        print traind2.shape
        print testd2.shape
```

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



        def getSplits(years,months):
            locsTrain=[]
            locsTest=[]
            print 'in getSplits ,train is \n',train
            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 i
                        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 TrainFucntion(x,y,xtest,ytest):
```

```
    weights=np.random.rand(1,x.shape[1])
    Cs=[0.01,0.1,1,10,100,1000]
    epsilons=np.arange(0,2,0.2)
    losses=[]
    parameters=list(itertools.product(Cs,epsilons))
    plotx=[]
    ploty=[]
    for p in parameters:
        plotx.append(p[0])
        ploty.append(p[1])

        svr=SVR(C=p[0],epsilon=p[1],kernel=’poly’,degree=5)
        svr.fit(x,y)
        ypred=svr.predict(xtest)
        #print ’ypred is \n’,ypred
        loss=findLoss(ytest,ypred)
        losses.append(loss)
        print ’Loss with C=’,p[0],’ and epsilon= ’,p[1],’ is ’,loss

    fig = plt.figure()
    ax=fig.add_subplot(111,projection=’3d’)
    ax.scatter(plotx,ploty,losses)
    plt.title(’Loss of SVR with C and epsilon’)
    plt.xtitle(’C’)
    plt.ytitle(’epsilon’)
    plt.show()


def Predict(weights,test):

    return np.dot(test,weights.T)

def findLoss(gold,predicted):
    loss=0

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

    loss=loss/gold.shape[0]
    return np.sqrt(loss)

def crossValidate():
        months=[10]
        locsTrain,locsTest=getSplits([2011,2012],months)

        testSubset=traind2[locsTest]
        testSubset2=train[locsTest]
        testLabels=labelsTrain[locsTest]
        #rf3=RandomForestRegressor(20)

        trainSubset=traind2[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])
                for i in range(trainSubset3.shape[1]):
                    if max(trainSubset3[:,i])!=0:
                        trainSubset3[:,i]=trainSubset3[:,i]/max(trainSubset3[:,i])
                for i in range(testSubset4.shape[1]):
                    if max(testSubset4[:,i])!=0:
                        testSubset4[:,i]=testSubset4[:,i]/max(testSubset4[:,i])
                #rf3.fit(trainSubset3,trainLabels3)change here to program new function to train
                TrainFucntion(trainSubset3,trainLabels3,testSubset4,testLabels4)



crossValidate()
```