**import pandas as pd**

import numpy as np

from datetime import datetime

pd.set\_option('display.max\_columns',50)

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import roc\_curve,precision\_score,auc,roc\_auc\_score

from sklearn.metrics import confusion\_matrix

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import GridSearchCV,cross\_val\_score

from sklearn.svm import SVC

def blight\_model():

train\_data=pd.read\_csv("train.csv",encoding='ISO-8859-1')

test\_data=pd.read\_csv("test.csv",encoding='ISO-8859-1')

train\_data.set\_index('ticket\_id',inplace=True)

total\_null=train\_data.isnull().sum()

null\_per=(train\_data.isnull().sum()/train\_data.isnull().count())

null\_data=pd.concat([total\_null,null\_per],axis=1,keys=['total','percent'])

train\_data.drop(null\_data[null\_data['percent']>0.5].index,axis=1,inplace=True)

address\_data=pd.read\_csv('addresses.csv')

latlons\_data=pd.read\_csv('latlons.csv')

address\_lat=pd.merge(address\_data,latlons\_data)

address\_lat.set\_index('ticket\_id',inplace=True)

train\_data2=train\_data.join(address\_lat)

test\_data2=test\_data.set\_index('ticket\_id').join(address\_lat)

columns=[ 'violator\_name','inspector\_name','clean\_up\_cost',

'violation\_street\_number', 'violation\_street\_name',

'mailing\_address\_str\_number', 'mailing\_address\_str\_name', 'city',

'state', 'zip\_code', 'country']

train\_data2.drop(columns,inplace=True,axis=1)

test\_data2.drop(columns,inplace=True,axis=1)

train\_data2['total\_payment']=train\_data2['fine\_amount']+train\_data2['admin\_fee']+train\_data2['state\_fee']+train\_data2['late\_fee']-train\_data2['discount\_amount']

test\_data2['total\_payment']=test\_data2['fine\_amount']+test\_data2['admin\_fee']+test\_data2['state\_fee']+test\_data2['late\_fee']-test\_data2['discount\_amount']

def time\_gap(hearing\_date\_str,ticket\_date\_str):

if not hearing\_date\_str or type(hearing\_date\_str)!=str:return 73

hearing\_date=datetime.strptime(hearing\_date\_str,"%Y-%m-%d %H:%M:%S")

ticket\_date=datetime.strptime(ticket\_date\_str, "%Y-%m-%d %H:%M:%S")

diff=hearing\_date-ticket\_date

return diff.days

train\_data2['time\_gap']=train\_data2.apply(lambda x:time\_gap(x['hearing\_date'],x['ticket\_issued\_date']),axis=1)

test\_data2['time\_gap']=test\_data2.apply(lambda x:time\_gap(x['hearing\_date'],x['ticket\_issued\_date']),axis=1)

column\_list=['hearing\_date','ticket\_issued\_date','fine\_amount','admin\_fee','state\_fee','late\_fee','discount\_amount']

train\_data2.drop(column\_list,axis=1,inplace=True)

test\_data2.drop(column\_list,axis=1,inplace=True)

for i in range(len(train\_data2.columns)):

if len(train\_data2[train\_data2.columns[i]].unique()) < 250:

train\_data2[train\_data2.columns[i]] = train\_data2[train\_data2.columns[i]] .astype('category')

string\_name=['agency\_name','disposition']

train\_cat=pd.get\_dummies(train\_data2,columns=string\_name)

test\_cat=pd.get\_dummies(test\_data2,columns=string\_name)

final\_clean= [

'balance\_due',

'compliance\_detail',

'payment\_amount',

'payment\_status',

'violation\_code',

'violation\_description',

'address']

train\_cat.drop(final\_clean,axis=1,inplace=True)

# test\_cat.drop(final\_clean,axis=1,inplace=True)

train\_cat['lat'].fillna(73,inplace='true')

train\_cat['lon'].fillna(73,inplace='true')

test\_cat['lon'].fillna(73,inplace='true')

test\_cat['lat'].fillna(73,inplace='true')

train\_set=train\_cat[(train\_cat['compliance']==0)|(train\_cat['compliance']==1)]

list\_remove=['violation\_zip\_code','non\_us\_str\_code','violation\_code','violation\_description','grafitti\_status','address']

test\_cat.drop(list\_remove,axis=1,inplace=True)

#training the model

y\_train=train\_set['compliance']

X\_train=train\_set.drop('compliance',axis=1)

X\_test=test\_cat

train\_features=X\_train

train\_features\_set=set(train\_features)

for features in set(train\_features):

if features not in X\_test:

train\_features\_set.remove(features)

train\_features\_list=list(train\_features\_set)

print(train\_features\_list)

X\_train=X\_train[train\_features\_list]

X\_test=X\_test[train\_features\_list]

print(X\_train)

blight\_model()