Assignment 4

July 20, 2019

---

*You are currently looking at **version 1.1** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ course resource.*

0.1 Assignment 4 - Understanding and Predicting Property Maintenance Fines

This assignment is based on a data challenge from the Michigan Data Science Team (MDST).

The Michigan Data Science Team (MDST) and the Michigan Student Symposium for Interdisciplinary Statistical Sciences (MSSISS) have partnered with the City of Detroit to help solve one of the most pressing problems facing Detroit - blight. Blight violations are issued by the city to individuals who allow their properties to remain in a deteriorated condition. Every year, the city of Detroit issues millions of dollars in fines to residents and every year, many of these fines remain unpaid. Enforcing unpaid blight fines is a costly and tedious process, so the city wants to know: how can we increase blight ticket compliance?

The first step in answering this question is understanding when and why a resident might fail to comply with a blight ticket. This is where predictive modeling comes in. For this assignment, your task is to predict whether a given blight ticket will be paid on time.

All data for this assignment has been provided to us through the Detroit Open Data Portal. **Only the data already included in your Coursera directory can be used for training the model for this assignment.** Nonetheless, we encourage you to look into data from other Detroit datasets to help inform feature creation and model selection. We recommend taking a look at the following related datasets:

- Building Permits
- Trades Permits
- Improve Detroit: Submitted Issues
- DPD: Citizen Complaints
- Parcel Map

We provide you with two data files for use in training and validating your models: train.csv and test.csv. Each row in these two files corresponds to a single blight ticket, and includes information about when, why, and to whom each ticket was issued. The target variable is compliance, which is True if the ticket was paid early, on time, or within one month of the hearing data, False if the ticket was paid after the hearing date or not at all, and Null if the violator was found not responsible. Compliance, as well as a handful of other variables that will not be available at test-time, are only included in train.csv.

Note: All tickets where the violators were found not responsible are not considered during evaluation. They are included in the training set as an additional source of data for visualization, and to enable unsupervised and semi-supervised approaches. However, they are not included in the test set.

File descriptions (Use only this data for training your model!)

```
readonly/train.csv - the training set (all tickets issued 2004-2011)
readonly/test.csv - the test set (all tickets issued 2012-2016)
readonly/addresses.csv & readonly/latlons.csv - mapping from ticket id to addresses
 Note: misspelled addresses may be incorrectly geolocated.
```

Data fields

train.csv & test.csv

```
ticket_id - unique identifier for tickets
agency_name - Agency that issued the ticket
inspector_name - Name of inspector that issued the ticket
violator_name - Name of the person/organization that the ticket was issued to
violation_street_number, violation_street_name, violation_zip_code - Address where
mailing_address_str_number, mailing_address_str_name, city, state, zip_code, non_us
ticket_issued_date - Date and time the ticket was issued
hearing_date - Date and time the violator's hearing was scheduled
violation_code, violation_description - Type of violation
disposition - Judgment and judgement type
fine_amount - Violation fine amount, excluding fees
admin_fee - $20 fee assigned to responsible judgments
```

state_fee - $10 fee assigned to responsible judgments late_fee - 10% fee assigned to responsible judgments discount_amount - discount applied, if any clean_up_cost - DPW clean-up or graffiti removal cost judgment_amount - Sum of all fines and fees grafitti_status - Flag for graffiti violations

train.csv only

```
payment_amount - Amount paid, if any
payment_date - Date payment was made, if it was received
payment_status - Current payment status as of Feb 1 2017
balance_due - Fines and fees still owed
collection_status - Flag for payments in collections
compliance [target variable for prediction]
 Null = Not responsible
 0 = Responsible, non-compliant
 1 = Responsible, compliant
compliance_detail - More information on why each ticket was marked compliant or non
```

0.2 Evaluation

Your predictions will be given as the probability that the corresponding blight ticket will be paid on time.

The evaluation metric for this assignment is the Area Under the ROC Curve (AUC).

Your grade will be based on the AUC score computed for your classifier. A model which with an AUROC of 0.7 passes this assignment, over 0.75 will recieve full points. ___

For this assignment, create a function that trains a model to predict blight ticket compliance in Detroit using `readonly/train.csv`. Using this model, return a series of length 61001 with the data being the probability that each corresponding ticket from `readonly/test.csv` will be paid, and the index being the ticket_id.

Example:

```
ticket_id
   284932    0.531842
   285362    0.401958
   285361    0.105928
   285338    0.018572
             ...
   376499    0.208567
   376500    0.818759
   369851    0.018528
   Name: compliance, dtype: float32
```

0.2.1 Hints

- Make sure your code is working before submitting it to the autograder.
- Print out your result to see whether there is anything weird (e.g., all probabilities are the same).
- Generally the total runtime should be less than 10 mins. You should NOT use Neural Network related classifiers (e.g., MLPClassifier) in this question.
- Try to avoid global variables. If you have other functions besides blight_model, you should move those functions inside the scope of blight_model.
- Refer to the pinned threads in Week 4's discussion forum when there is something you could not figure it out.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestRegressor
pd.set_option('display.max_columns', 999)

def blight_model():
```

```
    #load data
#     train=pd.read_csv('readonly/train.csv',encoding="ISO-8859-1")
#     test=pd.read_csv('readonly/test.csv',encoding="ISO-8859-1")
#     add=pd.read_csv('readonly/addresses.csv')
#     lalon=pd.read_csv('readonly/latlons.csv')

    train=pd.read_csv('train.csv',encoding="ISO-8859-1")
    test=pd.read_csv('test.csv',encoding="ISO-8859-1")
    add=pd.read_csv('addresses.csv')
    lalon=pd.read_csv('latlons.csv')
    #merge location and address data
    train=pd.merge(train,pd.merge(add,lalon,on='address'),on='ticket_id')
    test=pd.merge(test,pd.merge(add,lalon,on='address'),on='ticket_id')

    #drop null compliance
    train=train.dropna(axis=0,subset=['compliance'])

    #create X_train and y_train
    y_train=train['compliance']

    #drop data leakage causing columns
    drop_train=['balance_due','collection_status','compliance_detail','paym
    drop_col=['agency_name','inspector_name','violator_name','violation_zip
              'violation_street_number', 'violation_street_name','mailing_ad
              'hearing_date','ticket_issued_date','city','state','zip_code']

    X_train=train.drop(drop_col+drop_train+['compliance'],axis=1)
    test=test.drop(drop_col,axis=1)
    X_train.set_index(['ticket_id','address'],inplace=True)
    test.set_index(['ticket_id','address'],inplace=True)

    #preprocessing
    le=LabelEncoder()
    X_train['disposition']=le.fit_transform(X_train['disposition'])
    X_train['violation_code']=le.fit_transform(X_train['violation_code'])
    test['disposition']=le.fit_transform(test['disposition'])
    test['violation_code']=le.fit_transform(test['violation_code'])
#     X_train = pd.get_dummies(X_train, columns=['disposition'])#categorica
    X_train['lat'] = X_train['lat'].fillna(X_train['lat'].mean())
    X_train['lon'] = X_train['lon'].fillna(X_train['lon'].mean())
    test['lat'] = test['lat'].fillna(test['lat'].mean())
    test['lon'] = test['lon'].fillna(test['lon'].mean())


    clf =RandomForestRegressor(n_estimators=100, max_depth=30).fit(X_train,
    return pd.DataFrame(clf.predict(test), test.index.get_level_values('tic
```

In [ ]:
```
blight_model()
```

```
/opt/conda/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2827: Dtype
  if self.run_code(code, result):
```

Out[ ]:

	0
ticket_id	
284932	0.057765
285362	0.186098
285361	0.297372
285338	0.391828
285346	0.180000
285345	0.210000
285347	0.180000
285342	0.990000
285530	0.020000
284989	0.062636
285344	0.360000
285343	0.160000
285340	0.380000
285341	0.101828
285349	0.166828
285348	0.200161
284991	0.052636
285532	0.080000
285406	0.099314
285001	0.410000
285006	0.300000
285405	0.271478
285337	0.266428
285496	0.242915
285497	0.208037
285378	0.230477
285589	0.361478
285585	0.099510
285501	0.097469
285581	0.213002
285583	0.059129
285372	0.141478
285470	0.150314
285475	0.130314
285370	0.311478
285503	0.086863
285502	0.010000
285411	0.152734
285498	0.990000
285414	0.045206
285484	0.181762
285499	0.086685

285419	0.215330
288210	0.930000
285351	1.000000
288200	0.062667
285353	0.695000
285364	0.491324
288206	1.000000
288135	0.168898
285366	0.310639
288174	0.170714
288068	0.168898
288140	1.000000
285434	0.126744
288072	0.168898
288213	0.106521
285428	0.030000
288167	0.990000
288145	0.168898
285367	0.081042
285422	0.084309
285505	0.088863
285425	0.171478
292482	0.052549
292484	0.060000
292481	0.052549
292483	0.060000
288137	0.168898
285408	0.094309
285427	0.320000
285359	0.136548
288216	0.000000
288214	1.000000
285546	0.110000
285549	0.250000
285595	0.530000
285594	0.180000
285357	0.350505
288163	0.840000
288220	0.000000
285416	0.079965
285421	0.006576
285678	0.210247
288148	0.142667
285358	0.269840
285573	0.400000
285543	0.646495
285571	0.760000
288157	0.850000

285591	0.188527
285128	0.050428
285679	0.990000
288224	0.980000
285384	0.940000
285671	0.075966
285672	0.041485
285560	0.090000
285558	0.080000
285857	0.064444
285356	0.071681
285431	0.219348
285382	0.010000
285355	0.145129
286295	0.880000
285429	0.059500
285554	0.593333
285552	0.578333
285948	0.585250
285446	0.170000
285441	1.000000
285391	0.003000
285859	0.042636
285371	0.490000
285388	0.003000
285858	0.302636
285951	0.241006
286073	0.080000
286068	0.120000
285436	0.131875
286079	0.272500
285953	0.047328
285439	0.112675
285731	0.055041
288096	0.168898
285461	0.980000
285468	0.091478
285868	0.062769
285415	0.132506
285963	0.072474
285438	0.291478
285698	0.051512
285700	0.178689
285711	0.178689
285709	0.051512
285390	0.040000
285854	0.231052
285873	0.920000

288077	0.142667
285437	0.083158
288082	0.142667
285600	0.394753
285598	0.071512
285510	1.000000
286462	0.980000
285603	0.535486
285667	0.422500
285721	0.514753
285719	0.011627
285511	0.474342
285506	0.152675
285602	0.472143
285601	0.131512
288090	0.142667
285398	0.060000
288094	0.168898
285599	0.100000
285409	0.030000
285507	0.121875
285728	0.990000
285855	0.211820
285989	0.205684
285703	0.051512
285706	0.178689
285397	0.177000
285686	0.101512
285682	0.101512
285688	0.491616
285404	0.120000
285417	0.091084
285413	0.043617
285435	0.397770
285407	0.030000
286290	0.101828
286291	0.130000
285410	0.103617
285509	0.297875
285376	0.240000
285891	0.730000
285964	0.700000
286088	0.140909
288289	0.000000
285794	0.510000
286143	0.370000
288097	0.071521
286145	0.510000

285471	0.092122
286455	0.134368
288247	0.980000
286097	0.421175
285789	0.745714
285400	0.050166
285788	0.530952
285856	0.114224
286096	0.504491
288296	0.125161
285401	0.182565
288098	1.000000
288299	0.141521
288099	0.254631
288105	0.142667
285792	0.280000
285791	0.430000
288139	0.344327
285984	0.072231
285793	0.395714
285786	0.455000
288102	0.168898
286075	0.066589
286069	0.167388
286066	0.150000
286078	0.044021
285941	0.453593
286070	0.100000
285402	0.070000
285979	0.103645
285433	0.380000
285805	0.041828
286028	0.046524
285383	0.074029
286071	0.060000
285395	0.043000
285334	0.077170
285399	0.223000
288136	0.980000
286296	0.880000
286297	0.620000
286299	0.660000
288152	0.133381
288302	0.010000
286067	0.200000
288146	0.980000
285876	0.485094
288310	1.000000

285944	0.020000
285500	0.129285
288191	0.323774
288306	0.168898
288176	0.220036
288185	0.413774
286150	0.059768
288308	0.447520
286157	0.010370
288168	0.374920
286148	0.483729
285971	0.650000
288209	0.338948
285871	0.920000
285870	0.920000
286203	0.150000
...	...
375858	0.455932
375287	0.049201
375843	0.064874
375922	0.122393
375920	0.122393
375916	0.076289
375237	0.049344
376065	0.120000
376067	0.357143
376066	0.541828
375244	0.104996
375894	0.920000
376053	0.397321
375210	0.022689
375248	0.062637
375955	0.207143
375289	0.102609
375286	0.053332
376024	0.199745
376020	0.189745
376018	0.259745
376021	0.189745
376022	0.189745
375220	0.254327
375236	0.320450
375240	0.186246
375231	0.150471
375290	0.193753
376105	0.130000
376103	0.090000
376100	0.210000

376056	0.524464
375896	0.890000
375958	0.070811
375954	0.608667
376044	0.600000
376043	0.720000
375380	0.128084
375437	0.392321
375365	0.440000
375366	0.420000
375764	0.940000
375391	0.990000
375484	0.041226
375421	0.231066
375367	0.890000
375368	0.940000
375960	0.047349
375372	0.255867
375766	0.330000
375768	0.060000
375487	0.095775
375376	0.173935
375374	0.129051
375389	0.180000
375770	0.010000
375762	0.080000
375771	0.950000
375761	0.030000
376112	0.052634
375918	0.249328
375411	0.660000
375373	0.940000
376048	0.169866
375363	0.377143
375283	0.101446
375294	0.011667
375769	0.010000
376616	0.078337
375369	0.181828
376201	0.920000
375295	0.305232
375763	0.120000
375772	0.147500
375760	0.010000
375371	0.055982
375370	0.202648
375759	0.010000
375409	0.293192

375895	0.362383
375396	0.122172
375993	0.010000
376033	1.000000
375546	0.550000
375745	1.000000
375748	0.079246
375525	0.091980
375995	0.050000
375548	0.660000
375392	0.324757
375534	0.337498
375495	0.020000
375492	0.040000
376028	0.207070
375647	0.120000
375649	0.040000
376025	0.560000
375927	0.058569
375991	0.080000
375439	0.541397
376027	0.187070
376031	0.940000
375996	0.890000
376038	0.130000
376040	0.130000
376042	0.285119
375485	0.175433
375486	0.175433
375488	0.062500
376030	0.060000
376208	0.128559
376209	0.184921
375962	0.830000
375500	0.131319
376203	0.082264
375545	0.660000
375544	0.550000
375965	0.320000
375552	0.305923
375572	0.230637
375910	0.017357
376026	0.294707
375557	0.230536
376617	0.053107
375575	0.367851
376287	0.106593
375578	0.380498

376618	0.980000
376288	0.780000
375576	0.357851
375567	0.294497
375565	0.055372
375577	0.394184
376146	0.034766
375902	0.031042
375905	0.365855
375571	0.051240
375569	0.210575
375573	0.118797
375913	0.342786
375561	0.265947
376147	0.091042
375551	0.000000
375903	0.535624
375566	0.040651
376167	0.242714
376168	0.108083
375558	0.275694
375574	0.162734
376621	0.050000
375688	0.288085
375700	0.250202
376537	0.203333
376531	0.193333
376532	0.135536
375680	0.370536
375928	0.087647
376036	0.062428
376037	0.142200
376032	0.047274
376008	1.000000
375684	0.065429
375710	0.351320
375666	0.245400
376115	0.058120
376057	0.170000
376622	0.113362
376123	0.052548
376173	0.251076
376535	0.035556
376534	0.086814
376620	0.240000
375751	0.055238
375735	0.063308
376039	0.122221

376071	0.085576
375893	0.325066
376307	0.235232
375801	0.440800
375794	0.980000
376158	1.000000
376160	0.970000
375863	0.416716
375987	0.481828
375988	0.481828
375379	0.167956
376623	1.000000
375994	0.031042
376624	0.910000
376002	0.157923
375999	0.103554
376000	0.103554
375998	0.095143
375997	0.030950
376247	0.990000
375992	0.087647
376187	0.300000
376012	0.500000
376323	0.153321
376340	0.990000
376104	0.247500
376191	0.057029
376190	0.069062
374953	0.515400
374958	0.053108
376223	0.113528
376230	0.930000
376360	0.120820
376361	0.232485
376221	0.862539
376358	0.245452
376359	0.223785
376227	0.400000
376226	0.014630
376276	0.060428
376218	0.093738
376368	0.142639
376369	0.132639
376225	0.012739
376222	0.177818
376367	0.142200
376366	0.203728
376362	0.043283

```
376363     0.302050
376365     0.142200
376364     0.203728
376228     0.430000
376265     0.156946
376286     0.990000
376320     0.285212
376314     0.083986
376327     0.970000
376385     0.990000
376435     0.940000
376370     0.940000
376434     0.127810
376459     0.090000
376478     0.000000
376473     0.092783
376484     0.190000
376482     0.325755
376480     0.325755
376479     0.325755
376481     0.325755
376483     0.175085
376496     0.040000
376497     0.040000
376499     0.085577
376500     0.065577
369851     1.000000

[61001 rows x 1 columns]
```