---

*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 (https://www.coursera.org/learn/python-machine-learning/resources/bANLa)](https://www.coursera.org/learn/python-machine-learning/resources/bANLa) course resource.*

Assignment 4 - Understanding and Predicting Property Maintenance Fines

This assignment is based on a data challenge from the Michigan Data Science Team (MDST (http://midas.umich.edu/mdst/)).

The Michigan Data Science Team (MDST (http://midas.umich.edu/mdst/)) and the Michigan Student Symposium for Interdisciplinary Statistical Sciences (MSSISS (https://sites.lsa.umich.edu/mssiss/)) have partnered with the City of Detroit to help solve one of the most pressing problems facing Detroit - blight. Blight violations (http://www.detroitmi.gov/How-Do-I/Report/Blight-Complaint-FAQs) 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 (https://data.detroitmi.gov/). **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 (https://data.detroitmi.gov/Property-Parcels/Building-Permits/xw2a-a7tf)
- Trades Permits (https://data.detroitmi.gov/Property-Parcels/Trades-Permits/635b-dsgv)
- Improve Detroit: Submitted Issues (https://data.detroitmi.gov/Government/Improve-Detroit-Submitted-Issues/fwz3-w3yn)
- DPD: Citizen Complaints (https://data.detroitmi.gov/Public-Safety/DPD-Citizen-Complaints-2016/kahe-efs3)
- Parcel Map (https://data.detroitmi.gov/Property-Parcels/Parcel-Map/fxkw-udwf)

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, and from addresses to lat/lon coordinates.
 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 the violation occurred
mailing_address_str_number, mailing_address_str_name, city, state, zip_code, non_us_str_code, country - Mailing address of the violator
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-compliant
```

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
```

import pandas as pd import numpy as np

def blight_model_not used(): import subprocess raise ValueError("".join(map(lambda f:f.decode("utf-8"), subprocess.Popen(["ls", "-l", "test.csv"], stdout=subprocess.PIPE).stdout))) return

Importing libraries

In [1]:
```
%load_ext autoreload
%autoreload 2

import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.preprocessing import LabelEncoder

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.dummy import DummyClassifier

from sklearn.metrics import recall_score, precision_score, accuracy_score, classification_report
from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_curve, auc

# Hide warnings
import warnings
warnings.filterwarnings('ignore')

# The following lines adjust the granularity of reporting
pd.options.display.max_rows = 10
pd.options.display.float_format = '{:.2f}'.format
```

Loading the data files

In [2]:
```
train = pd.read_csv('train.csv', encoding='ISO-8859-1')
train.index = train['ticket_id']
train.shape
```

Out[2]: (250306, 34)

In [3]:
```
# ! cat readonly/test.csv > test.csv
# ! chmod 664 test.csv
test = pd.read_csv('test.csv', encoding='ISO-8859-1')
test.index = test['ticket_id']
test.shape
```

Out[3]: (61001, 27)

Data processing

In [4]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 250306 entries, 22056 to 325561
Data columns (total 34 columns):
ticket_id                     250306 non-null int64
agency_name                   250306 non-null object
inspector_name                250306 non-null object
violator_name                 250272 non-null object
violation_street_number       250306 non-null float64
violation_street_name         250306 non-null object
violation_zip_code            0 non-null float64
mailing_address_str_number    246704 non-null float64
mailing_address_str_name      250302 non-null object
city                          250306 non-null object
state                         250213 non-null object
zip_code                      250305 non-null object
non_us_str_code               3 non-null object
country                       250306 non-null object
ticket_issued_date            250306 non-null object
hearing_date                  237815 non-null object
violation_code                250306 non-null object
violation_description         250306 non-null object
disposition                   250306 non-null object
fine_amount                   250305 non-null float64
admin_fee                     250306 non-null float64
state_fee                     250306 non-null float64
late_fee                      250306 non-null float64
discount_amount               250306 non-null float64
clean_up_cost                 250306 non-null float64
judgment_amount               250306 non-null float64
payment_amount                250306 non-null float64
balance_due                   250306 non-null float64
payment_date                  41113 non-null object
payment_status                250306 non-null object
collection_status             36897 non-null object
grafitti_status               1 non-null object
compliance_detail             250306 non-null object
compliance                    159880 non-null float64
dtypes: float64(13), int64(1), object(20)
memory usage: 66.8+ MB
```

In [5]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 61001 entries, 284932 to 369851
Data columns (total 27 columns):
ticket_id                     61001 non-null int64
agency_name                   61001 non-null object
inspector_name                61001 non-null object
violator_name                 60973 non-null object
violation_street_number       61001 non-null float64
violation_street_name         61001 non-null object
violation_zip_code            24024 non-null object
mailing_address_str_number    59987 non-null object
mailing_address_str_name      60998 non-null object
city                          61000 non-null object
state                         60670 non-null object
zip_code                      60998 non-null object
non_us_str_code               0 non-null float64
country                       61001 non-null object
ticket_issued_date            61001 non-null object
hearing_date                  58804 non-null object
violation_code                61001 non-null object
violation_description         61001 non-null object
disposition                   61001 non-null object
fine_amount                   61001 non-null float64
admin_fee                     61001 non-null float64
state_fee                     61001 non-null float64
late_fee                      61001 non-null float64
discount_amount               61001 non-null float64
clean_up_cost                 61001 non-null float64
judgment_amount               61001 non-null float64
grafitti_status               2221 non-null object
dtypes: float64(9), int64(1), object(17)
memory usage: 13.0+ MB
```

In [6]:
```
# Drop columns that are not present in the test dataset

train.drop(['payment_amount',
         'balance_due',
         'payment_date',
         'payment_status',
         'collection_status',
         'compliance_detail'], axis=1, inplace=True)
```

In [7]:
```
# Drop unnecessary columns

train.drop(['agency_name',
            'inspector_name',
            'violator_name',
            'non_us_str_code',
            'ticket_issued_date',
            'violation_description',
            'grafitti_status',
            'hearing_date'], axis=1, inplace=True)

test.drop(['agency_name',
            'inspector_name',
            'violator_name',
            'non_us_str_code',
            'ticket_issued_date',
            'violation_description',
            'grafitti_status',
            'hearing_date'], axis=1, inplace=True)
```

In [8]:
```
# Drop all nan instances based on the compliance column (target)

train = train[train['compliance'].notnull()]

# selecting rows corresponding t0 country USA and state MI

train = train.loc[(train['country'] == 'USA') & (train['state'] == 'MI')]
test = test.loc[(test['state'] == 'MI')]
```

In [9]:
```
# Adding new column to the df

train['total_amount'] = (train['fine_amount'] +
                         train['admin_fee'] +
                         train['state_fee'] +
                         train['late_fee'] +
                         train['clean_up_cost'] +
                         train['judgment_amount']) - train['discount_amount']

test['total_amount'] = (test['fine_amount'] +
                         test['admin_fee'] +
                         test['state_fee'] +
                         test['late_fee'] +
                         test['clean_up_cost'] +
                         test['judgment_amount']) - test['discount_amount']
```

In [10]:
```
# Convert string to integer (Label Encoding)

encoder = LabelEncoder()
cols = ['disposition', 'violation_code']

for col in cols:
    train[col] = encoder.fit_transform(train[col])
    test[col] = encoder.fit_transform(test[col])
```

In [11]:
```
print('train: ', train.shape)
print('test: ', test.shape)
```

```
train:  (143655, 21)
test:  (51866, 20)
```

Splitting the data

In [12]:
```
important_feature_names  = ['total_amount', 'disposition', 'violation_code']

X = train[important_feature_names]
y = train.iloc[:,-2]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

print('Train Features Shape:', X_train.shape)
print('Train Labels Shape:', y_train.shape)
print('Test Features Shape:', X_test.shape)
print('Test Labels Shape:', y_test.shape)
```

```
Train Features Shape: (96248, 3)
Train Labels Shape: (96248,)
Test Features Shape: (47407, 3)
Test Labels Shape: (47407,)
```

In [13]:
```
# check for class imbalance
'''
0 = Responsible, non-compliant
1 = Responsible, compliant
'''

y.value_counts()
```

Out[13]:
```
0.00    133060
1.00     10595
Name: compliance, dtype: int64
```

Model training and selection (not optimized)

In [14]:
```
'''
@author:  Steven Ponce
Date:     June 2021
'''
def evaluate(model, X, y):
    X_train, X_test, y_train, y_test  = train_test_split(X, y, test_size=0.33, random_state=42)

    # train the model
    model.fit(X_train, y_train)
    print(f'Accuracy: {model.score(X_test, y_test) * 100:.2f} %')

    # cross-validation
    score = cross_val_score(model, X, y, cv=5)
    print(f'CV score: {np.mean(score) * 100:.2f} %')

def dummy(dummy, X, y):

    # most frequent
    dummy = DummyClassifier(strategy = 'most_frequent').fit(X_train, y_train)
    print(f'Dummy Score: {dummy.score(X_test, y_test) * 100:.2f} %')
    print('-'*20)
```

Logistic Regression

In [15]:
```
# from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
dummy(dummy, X, y)
evaluate(model, X, y)
```

```
Dummy Score: 92.64 %
--------------------
Accuracy: 92.64 %
CV score: 92.62 %
```

Due to the class imbalance, accuracy is not a good metric

In [16]:
```
'''
@author:  Steven Ponce
Date:     June 2021
'''
def evaluate2(model, X, y):
    X_train, X_test, y_train, y_test  = train_test_split(X, y, test_size=0.33, random_state=42)

    # train the model
    model.fit(X_train, y_train)
    model_pred = model.predict(X_test)
    confusion = confusion_matrix(y_test, model_pred)
    print(confusion)
    print('-'*55)

    print(classification_report(y_test, model_pred, target_names = ['0', '1']))
    #  0 = Responsible, non-compliant
    #  1 = Responsible, compliant

def dummy2(dummy, X, y):

    # most frequent
    dummy = DummyClassifier(strategy = 'most_frequent').fit(X_train, y_train)
    y_dummy_pred = dummy.predict(X_test)
    confusion = confusion_matrix(y_test, y_dummy_pred)
    print(f'Dummy Most Frequent Class:\n', confusion)
```

Dummy Classifier - Most Frequent

In [17]:
```
dummy2(dummy, X, y)
```

```
Dummy Most Frequent Class:
 [[43919     0]
 [ 3488     0]]
```

Logistic Regression

In [18]:
```
model = LogisticRegression()
evaluate2(model, X, y)
```

```
[[43919     0]
 [ 3488     0]]
-------------------------------------------------------
             precision    recall  f1-score   support

          0       0.93      1.00      0.96     43919
          1       0.00      0.00      0.00      3488

avg / total       0.86      0.93      0.89     47407
```

SVM

In [19]:
```
model = SVC()
evaluate2(model, X, y)
```

```
[[43816   103]
 [ 2620   868]]
-------------------------------------------------------
             precision    recall  f1-score   support

          0       0.94      1.00      0.97     43919
          1       0.89      0.25      0.39      3488

avg / total       0.94      0.94      0.93     47407
```

Random Forest Classifier

In [20]:
```
model = RandomForestClassifier()
evaluate2(model, X, y)
```

```
[[43795   124]
 [ 2600   888]]
-------------------------------------------------------
             precision    recall  f1-score   support

          0       0.94      1.00      0.97     43919
          1       0.88      0.25      0.39      3488

avg / total       0.94      0.94      0.93     47407
```

Model optimization - Random Forest Classifier

Grid Search with Cross Validation

In [21]:
```
param_grid = {
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 4, 6],
    'n_estimators': [5, 10, 15]}

model = RandomForestClassifier()

grid_search = GridSearchCV(estimator = model, param_grid = param_grid, scoring='roc_auc',
                           cv = 3, n_jobs = -1, verbose = 2)

grid_search.fit(X_train, y_train);
```

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
[CV] max_depth=5, min_samples_split=2, n_estimators=5 ...............
[CV] max_depth=5, min_samples_split=2, n_estimators=5 ...............
[CV] max_depth=5, min_samples_split=2, n_estimators=5 ...............
[CV] max_depth=5, min_samples_split=2, n_estimators=10 ..............
[CV] max_depth=5, min_samples_split=2, n_estimators=10 ..............
[CV] max_depth=5, min_samples_split=2, n_estimators=10 ..............
[CV] max_depth=5, min_samples_split=2, n_estimators=15 ..............
[CV] . max_depth=5, min_samples_split=2, n_estimators=5, total=   1.4s
[CV] . max_depth=5, min_samples_split=2, n_estimators=5, total=   1.5s
[CV] . max_depth=5, min_samples_split=2, n_estimators=5, total=   1.5s
[CV] max_depth=5, min_samples_split=2, n_estimators=15 ..............
[CV] max_depth=5, min_samples_split=2, n_estimators=15 ..............
[CV] max_depth=5, min_samples_split=4, n_estimators=5 ...............
[CV] max_depth=5, min_samples_split=4, n_estimators=5 ...............
[CV]  max_depth=5, min_samples_split=2, n_estimators=10, total=   2.9s
[CV]  max_depth=5, min_samples_split=2, n_estimators=10, total=   3.0s
[CV]  max_depth=5, min_samples_split=2, n_estimators=10, total=   2.7s
[CV] max_depth=5, min_samples_split=4, n_estimators=5 ...............
[CV] max_depth=5, min_samples_split=4, n_estimators=10 ..............
[CV] . max_depth=5, min_samples_split=4, n_estimators=5, total=   2.1s
[CV] max_depth=5, min_samples_split=4, n_estimators=10 ..............
[CV] . max_depth=5, min_samples_split=4, n_estimators=5, total=   2.3s
[CV]  max_depth=5, min_samples_split=2, n_estimators=15, total=   4.0s
[CV] max_depth=5, min_samples_split=4, n_estimators=10 ..............
[CV] . max_depth=5, min_samples_split=4, n_estimators=5, total=   2.0s
[CV] max_depth=5, min_samples_split=4, n_estimators=15 ..............
[CV] max_depth=5, min_samples_split=4, n_estimators=15 ..............
[CV]  max_depth=5, min_samples_split=2, n_estimators=15, total=   4.1s
[CV]  max_depth=5, min_samples_split=2, n_estimators=15, total=   5.1s
[CV] max_depth=5, min_samples_split=4, n_estimators=15 ..............
[CV] max_depth=5, min_samples_split=6, n_estimators=5 ...............
[CV]  max_depth=5, min_samples_split=4, n_estimators=10, total=   3.0s
[CV] max_depth=5, min_samples_split=6, n_estimators=5 ...............
[CV] max_depth=5, min_samples_split=6, n_estimators=5 ...............
[CV] max_depth=5, min_samples_split=6, n_estimators=10 ..............
[CV] . max_depth=5, min_samples_split=6, n_estimators=5, total=   2.1s
[CV]  max_depth=5, min_samples_split=4, n_estimators=10, total=   4.0s
[CV] max_depth=5, min_samples_split=6, n_estimators=10 ..............
[CV]  max_depth=5, min_samples_split=4, n_estimators=10, total=   3.4s
[CV] . max_depth=5, min_samples_split=6, n_estimators=5, total=   1.8s
[CV] max_depth=5, min_samples_split=6, n_estimators=10 ..............
[CV]  max_depth=5, min_samples_split=4, n_estimators=15, total=   4.1s
[CV]  max_depth=5, min_samples_split=4, n_estimators=15, total=   5.1s
[CV] max_depth=5, min_samples_split=6, n_estimators=15 ..............
[CV] . max_depth=5, min_samples_split=6, n_estimators=5, total=   3.8s
[CV] max_depth=5, min_samples_split=6, n_estimators=15 ..............
[CV]  max_depth=5, min_samples_split=4, n_estimators=15, total=   4.8s
[CV] max_depth=5, min_samples_split=6, n_estimators=15 ..............
[CV]  max_depth=5, min_samples_split=6, n_estimators=10, total=   2.7s
[CV] max_depth=10, min_samples_split=2, n_estimators=5 ..............
[CV]  max_depth=5, min_samples_split=6, n_estimators=10, total=   3.2s
[CV] max_depth=10, min_samples_split=2, n_estimators=5 ..............
[CV] max_depth=10, min_samples_split=2, n_estimators=5 ..............
[CV]  max_depth=5, min_samples_split=6, n_estimators=10, total=   2.7s
[CV] max_depth=10, min_samples_split=2, n_estimators=10 .............
[CV] max_depth=10, min_samples_split=2, n_estimators=10 .............
[CV] max_depth=10, min_samples_split=2, n_estimators=10 .............
[CV]  max_depth=10, min_samples_split=2, n_estimators=5, total=   1.9s
[CV]  max_depth=10, min_samples_split=2, n_estimators=5, total=   2.1s
[CV] max_depth=10, min_samples_split=2, n_estimators=15 .............
[CV] max_depth=10, min_samples_split=2, n_estimators=15 .............
[CV]  max_depth=10, min_samples_split=2, n_estimators=5, total=   2.6s
[CV] max_depth=10, min_samples_split=2, n_estimators=15 .............
[CV]  max_depth=5, min_samples_split=6, n_estimators=15, total=   4.5s
[CV] max_depth=10, min_samples_split=4, n_estimators=5 ..............
[CV] max_depth=10, min_samples_split=4, n_estimators=5 ..............
[CV]  max_depth=10, min_samples_split=2, n_estimators=10, total=   3.4s
[CV] max_depth=10, min_samples_split=4, n_estimators=5 ..............
[CV]  max_depth=5, min_samples_split=6, n_estimators=15, total=   5.0s
[CV]  max_depth=10, min_samples_split=2, n_estimators=10, total=   3.6s
[CV]  max_depth=5, min_samples_split=6, n_estimators=15, total=   5.5s
[CV] max_depth=10, min_samples_split=4, n_estimators=10 .............
[CV]  max_depth=10, min_samples_split=2, n_estimators=10, total=   4.4s
[CV] max_depth=10, min_samples_split=4, n_estimators=10 .............
[CV] max_depth=10, min_samples_split=4, n_estimators=10 .............
[CV]  max_depth=10, min_samples_split=4, n_estimators=5, total=   2.2s
[CV] max_depth=10, min_samples_split=4, n_estimators=15 .............
[CV]  max_depth=10, min_samples_split=4, n_estimators=5, total=   3.3s
[CV] max_depth=10, min_samples_split=4, n_estimators=15 .............
[CV]  max_depth=10, min_samples_split=2, n_estimators=15, total=   5.4s
[CV] max_depth=10, min_samples_split=4, n_estimators=15 .............
[CV]  max_depth=10, min_samples_split=4, n_estimators=5, total=   3.2s
[CV] max_depth=10, min_samples_split=6, n_estimators=5 ..............
[CV] max_depth=10, min_samples_split=6, n_estimators=5 ..............
```

```
[CV]  max_depth=10, min_samples_split=2, n_estimators=15, total=   6.3s
[CV] max_depth=10, min_samples_split=6, n_estimators=5 ...............
[CV]  max_depth=10, min_samples_split=2, n_estimators=15, total=   6.5s
[CV] max_depth=10, min_samples_split=6, n_estimators=10 ..............
[CV] max_depth=10, min_samples_split=6, n_estimators=10 ..............
[CV]  max_depth=10, min_samples_split=4, n_estimators=10, total=   5.2s
[CV]  max_depth=10, min_samples_split=6, n_estimators=5, total=   3.2s
[CV]  max_depth=10, min_samples_split=4, n_estimators=10, total=   5.7s
[CV] max_depth=10, min_samples_split=6, n_estimators=10 ..............
[CV]  max_depth=10, min_samples_split=4, n_estimators=10, total=   6.4s
[CV] max_depth=10, min_samples_split=6, n_estimators=15 ..............
[CV]  max_depth=10, min_samples_split=6, n_estimators=5, total=   3.5s
[CV]  max_depth=10, min_samples_split=4, n_estimators=15, total=   4.9s
[CV]  max_depth=10, min_samples_split=4, n_estimators=15, total=   5.9s
[CV] max_depth=10, min_samples_split=6, n_estimators=15 ..............
[CV]  max_depth=10, min_samples_split=6, n_estimators=5, total=   4.6s
[CV] max_depth=10, min_samples_split=6, n_estimators=15 ..............
[CV]  max_depth=10, min_samples_split=4, n_estimators=15, total=   5.9s
[CV] max_depth=15, min_samples_split=2, n_estimators=5 ...............
[CV]  max_depth=10, min_samples_split=6, n_estimators=10, total=   3.9s
[CV] max_depth=15, min_samples_split=2, n_estimators=5 ...............
[CV] max_depth=15, min_samples_split=2, n_estimators=5 ...............
[CV] max_depth=15, min_samples_split=2, n_estimators=10 ..............
[CV]  max_depth=10, min_samples_split=6, n_estimators=10, total=   5.4s
[CV]  max_depth=15, min_samples_split=2, n_estimators=5, total=   2.3s
[CV] max_depth=15, min_samples_split=2, n_estimators=10 ..............
[CV]  max_depth=10, min_samples_split=6, n_estimators=10, total=   4.2s
[CV] max_depth=15, min_samples_split=2, n_estimators=10 ..............
[CV] max_depth=15, min_samples_split=2, n_estimators=15 ..............
[CV]  max_depth=15, min_samples_split=2, n_estimators=5, total=   2.9s
[CV]  max_depth=15, min_samples_split=2, n_estimators=5, total=   3.1s
[CV] max_depth=15, min_samples_split=2, n_estimators=15 ..............
[CV] max_depth=15, min_samples_split=2, n_estimators=15 ..............
[CV] max_depth=15, min_samples_split=4, n_estimators=5 ...............
[CV]  max_depth=10, min_samples_split=6, n_estimators=15, total=   6.3s
[CV] max_depth=15, min_samples_split=4, n_estimators=5 ...............
[CV]  max_depth=10, min_samples_split=6, n_estimators=15, total=   5.7s
[CV] max_depth=15, min_samples_split=4, n_estimators=5 ...............
[CV] max_depth=15, min_samples_split=4, n_estimators=10 ..............
[CV]  max_depth=10, min_samples_split=6, n_estimators=15, total=   6.9s
[CV]  max_depth=15, min_samples_split=2, n_estimators=10, total=   4.3s
[CV] max_depth=15, min_samples_split=4, n_estimators=10 ..............
[CV] max_depth=15, min_samples_split=4, n_estimators=10 ..............
[CV]  max_depth=15, min_samples_split=2, n_estimators=10, total=   4.9s

[Parallel(n_jobs=-1)]: Done  59 out of  81 | elapsed:   42.4s remaining:   15.8s

[CV] max_depth=15, min_samples_split=4, n_estimators=15 ..............
[CV]  max_depth=15, min_samples_split=4, n_estimators=5, total=   2.5s
[CV]  max_depth=15, min_samples_split=4, n_estimators=5, total=   3.8s
[CV] max_depth=15, min_samples_split=4, n_estimators=15 ..............
[CV]  max_depth=15, min_samples_split=4, n_estimators=5, total=   3.5s
[CV] max_depth=15, min_samples_split=4, n_estimators=15 ..............
[CV] max_depth=15, min_samples_split=6, n_estimators=5 ...............
[CV] max_depth=15, min_samples_split=6, n_estimators=5 ...............
[CV]  max_depth=15, min_samples_split=2, n_estimators=10, total=   7.6s
[CV]  max_depth=15, min_samples_split=2, n_estimators=15, total=   7.0s
[CV] max_depth=15, min_samples_split=6, n_estimators=5 ...............
[CV]  max_depth=15, min_samples_split=2, n_estimators=15, total=   7.3s
[CV] max_depth=15, min_samples_split=6, n_estimators=10 ..............
[CV]  max_depth=15, min_samples_split=4, n_estimators=10, total=   4.8s
[CV] max_depth=15, min_samples_split=6, n_estimators=10 ..............
[CV]  max_depth=15, min_samples_split=4, n_estimators=10, total=   5.3s
[CV] max_depth=15, min_samples_split=6, n_estimators=10 ..............
[CV]  max_depth=15, min_samples_split=6, n_estimators=5, total=   3.7s
[CV] max_depth=15, min_samples_split=6, n_estimators=15 ..............
[CV] max_depth=15, min_samples_split=6, n_estimators=15 ..............
[CV]  max_depth=15, min_samples_split=6, n_estimators=5, total=   3.9s
[CV]  max_depth=15, min_samples_split=2, n_estimators=15, total=  10.0s
[CV]  max_depth=15, min_samples_split=4, n_estimators=10, total=   6.8s
[CV]  max_depth=15, min_samples_split=4, n_estimators=15, total=   5.9s
[CV]  max_depth=15, min_samples_split=6, n_estimators=5, total=   4.7s
[CV] max_depth=15, min_samples_split=6, n_estimators=15 ..............
[CV]  max_depth=15, min_samples_split=6, n_estimators=10, total=   4.4s
[CV]  max_depth=15, min_samples_split=4, n_estimators=15, total=   6.4s
[CV]  max_depth=15, min_samples_split=6, n_estimators=10, total=   3.8s
[CV]  max_depth=15, min_samples_split=6, n_estimators=10, total=   5.1s
[CV]  max_depth=15, min_samples_split=4, n_estimators=15, total=   9.7s
[CV]  max_depth=15, min_samples_split=6, n_estimators=15, total=   3.3s
[CV]  max_depth=15, min_samples_split=6, n_estimators=15, total=   4.6s
[CV]  max_depth=15, min_samples_split=6, n_estimators=15, total=   4.1s

[Parallel(n_jobs=-1)]: Done  81 out of  81 | elapsed:   55.1s finished
```

In [22]:
```
grid_search.best_params_
```
Out[22]:
```
{'max_depth': 10, 'min_samples_split': 4, 'n_estimators': 15}
```
In [23]:
```
# optimized model
model = RandomForestClassifier(max_depth=10,random_state=42,
                               min_samples_split=6, n_estimators=15)
evaluate2(model, X, y)

print('-'*55)
print('Grid best parameter: ', grid_search.best_params_)
print('Grid best score (roc_auc): ', grid_search.best_score_)
```
```
[[43812   107]
 [ 2662   826]]
-------------------------------------------------------
             precision    recall  f1-score   support

          0       0.94      1.00      0.97     43919
          1       0.89      0.24      0.37      3488

avg / total       0.94      0.94      0.93     47407

-------------------------------------------------------
Grid best parameter:  {'max_depth': 10, 'min_samples_split': 4, 'n_estimators': 15}
Grid best score (roc_auc):  0.806422449531
```
In [ ]:

Testing the model

In [24]:
```
important_feature_names  = ['total_amount', 'disposition', 'violation_code']

X = test[important_feature_names]

X_pred = model.predict_proba(test[important_feature_names])

print('-'*55)
print('Grid best parameter: ', grid_search.best_params_)
print('Grid best score (roc_auc): ', grid_search.best_score_)
```
```
-------------------------------------------------------
Grid best parameter:  {'max_depth': 10, 'min_samples_split': 4, 'n_estimators': 15}
Grid best score (roc_auc):  0.806422449531
```
In [25]:
```
results = pd.Series(data = X_pred[:,1], index = test['ticket_id'], dtype='float32')

results
```
Out[25]:
```
ticket_id
284932   0.41
285362   0.21
285361   0.01
285338   0.27
285346   0.32
         ...
376496   0.03
376497   0.03
376499   0.32
376500   0.32
369851   0.71
dtype: float32
```
In [ ]:

In [27]:

```
import pandas as pd
import numpy as np

def blight_model():

    # Your code here

    important_feature_names  = ['total_amount', 'disposition', 'violation_code']

    X = test[important_feature_names]

    X_pred = model.predict_proba(test[important_feature_names])

    print('-'*55)
    print('Grid best parameter: ', grid_search.best_params_)
    print('Grid best score (roc_auc): ', grid_search.best_score_)

    results = pd.Series(data = X_pred[:,1], index = test['ticket_id'], dtype='float32')


    return results
```

In [28]:

```
blight_model()
```

```
-------------------------------------------------------
Grid best parameter:  {'max_depth': 10, 'min_samples_split': 4, 'n_estimators': 15}
Grid best score (roc_auc):  0.806422449531
```

Out[28]:

```
ticket_id
284932   0.41
285362   0.21
285361   0.01
285338   0.27
285346   0.32
         ...
376496   0.03
376497   0.03
376499   0.32
376500   0.32
369851   0.71
dtype: float32
```