HouseEasy Final Report: Investment Potential of NYC Real Estate Market

Chun-Yi Yang(cyy292), Haozheng Zhu(hz1448), Xiaoxuan Tang(xt470)

September 13, 2017

1 Business Understanding

"Big Apple" is one of the most prosperous and beloved places around the world. Since 2009, with the recovery of US economy and financial market, New York has again become a hot spot for residing or investing. When we are talking about investment potential, it is always a very abstract and diversified definition. While in our report, unlike investment potential defined by voluble estate agents that equated with perfect location and fancy price tag, we recognize that we should set up a guide for those who have interested in investing real estate in New York City but are not willing to pay a lot fortune for Manhattan luxurious condos and penthouses. Instead, some high potential areas with heavy rental demand but with relatively lower sales price are in our consideration. Therefore, the definition here is determined by the comparison between rental and ownership.

We realize that the ever increasing quantity of data in real estate industry is a valuable asset for extract useful information, especially by various machine learning methods. For instance, some characteristics like price or inventory trend can be aligned with external economic and environmental factors by supervised and unsupervised learning and time series analysis. Since the 1990s there has been considerable research that enormously expanded data application in real estates and shaped the whole industry. We hope our report can be delivered as a well-organized reference as well to better understanding those undervalued communities, fueling a new way to look at the future of these neighborhoods. We expect that we can achieve results with persuasive accuracy to locate the high potential area in NYC.

2 Data Understanding

Understanding the present New York real estate market by updated data collected from professional agencies will be extremely important for us to find the perfect way to design and analyze our model. Our team decides to utilize open neighborhood-leveled data and real estate trading data to create a new thinking of New York real estate markets. Current available data mainly comes from two resources: StreetEasy.com and CoreData.nyc. Our primary resource of house price and inventory is StreetEasy, which is an online residential real-estate shopper in the New York region launching from 2006. From StreetEasy we have acquired median sales price, median ask price, median rental price,

sales and rental inventory monthly data for all types of houses in New York City from 2009 to 2017. The data contains records from more than 130 neighborhoods from 5 boroughs in NYC[1], covering more than 80% areas. Each dataset averagely contains more than 9000 samples from different months and locations.

The term investment potential we define in this report comes from median rental price divided by median sales price by monthly. All ratio with multipliers constitutes the housing investment potential index as desired output values in the data analysis section. CoreData.nyc is New York City's housing and neighborhoods data hub, presented by the NYU Furman Center. It has provided neighborhood-level information on housing markets, home affordability, land use, demographics, and neighborhood conditions[1]. The detailed introduction of selected features will be present in Section 3.1.

All data are representative and informative for research in NYC housing market yet some concerns need to be scrutinized. Despite most residential areas have been covered, the sample size in each area differentiate a lot which might damp the data quality. For example, some very popular neighborhood in Manhattan with low sample volume overall decrease the average price and influence the data effectiveness. Meanwhile, lack of professional industry prospects for outliers filtration makes us hard to discover flaws in the dataset before utilization. Overall, the dataset is satisfied and well fit to our model.

3 Data Preparation

3.1 Feature Definition

CoreData.nyc has provided neighborhood-level data in different perspectives. However, not all of them can satisfy the feature requirement in our model. When we want to seek out "true" value of a neighborhood, we have concluded several selection criteria such as easy public transport access, gentle and talented neighbor and so on. As per our discussion.

We selected features as below[2],

Features	Description
income	The total income of all members of a household aged 15 years or older
local_pop	Residents born in New York State
foreign_pop	The share of the population that is born outside the United States
loan	The number of first-lien home purchase loan originations for owner-occupied 1-4 family buildings
vacancy	The number of vacant, habitable, for-rent units divided by the number of renter-occupied units
travel_time	The mean commute time in minutes for commuters residing in the geographic area
park	The percentage of residential units that are within a $\frac{1}{4}$ mile of a park.
sub	The percentage of residential units that are within a $\frac{1}{2}$ mile walk of a station entrance for the New York City Subway
crime	Serious crime rate (per 1,000 residents)
supply	The number of sales inventory in each neighborhood

Table 1: Selected Features

[1]The neighborhoods are defined by StreetEasy

3.2 Target Definition

The dataset on StreetEasy is per month per area; on the other hand, the dataset from CoreData.nyc is per year per area. The different scale of these two datasets leads to a combination issue. To align this two datasets, we will take the arithmetic mean of monthly data to represent the corresponding annual figure.

Next step goes to define the label for training data. The price-to-rent ratio is a well-established economic principle used for real estate valuation. Generally it is used to evaluate whether it is better to buy or rent the property. Considering our target is to find out most valuable rental area with lower sales price, reverse of this measurement makes it a perfect indicator for our study. Thus, an investment index is defined as:

$$Index = \frac{rental\ \ price}{house\ \ price} * 100$$

The bigger the number, values more investment potential of the house; vise versa if smaller the number. If either rental price or house price is missing, we ignore this data sample. Finally, there are totally 470 data samples in our dataset. Afterward, we define 3 different investing potential from low to high by sorted investing index percentile 3:4:3. A supervised multi-classes classification algorithm can be applied.

3.3 Data Preprocessing

Due to unclear definition of neighborhood, all selected data is measured either by 59 Community Districts(CDs)[2] or by 55 Sub-borough Areas(SBAs)[3] on yearly basis from CoreData.nyc. CoreData.nyc provides data for CDs where available, but otherwise employs data at the SBA level. Hence, the term neighborhood is different from previous one from StreetEasy, which has 131 areas. To align all datasets, we design transition matrix to convert datasets based on CDs, SBAs and StreetEasy.

The data from either StreetEasy and CoreData.nyc is incomplete. We noted any missing value as zero and replace it as the average of the rest values of corresponding feature.

3.4 Data Normalization

The number of house supply is correlated to the total house number in that area. We need to find a way to offset the difference between "supply power" in each area. Due to lacking data of overall house number, we normalized the *supply* feature by row in same area by the **feature scaling**.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} + 1$$

[2]determined by NYC Community Board

[3]defined by groups of census tracts summing to at least 100,000 residents, determined by the NYC Department of Housing Preservation and Development

We add one because we think even the smallest supply has supply power and we take it as the base supply of the region. $X \in [1, 2]$. After the feature table is established, the whole dataset is normalized by **Standard Score**

$$X' = \frac{X - \mu}{\alpha}$$

where μ is the mean of the feature and α is the standard deviation of feature.

4 Modeling and Evaluation

4.1 Data Preparation

We first replaced all the 'null' factors with average value of their specific columns. We also normalized data to increase the accuracy of our results. In order to split test and training data, we split our original data frame randomly by 80/20. 80 percent of data are training data, and 20 percent are testing data.

4.2 Model Comparison

There are different models for multi-classes classification. We decided to test Decision Tree, SVM, and used linear SVM as our baseline model. As we were doing multi-class classification, the "one-against-one" approach was not enough. We tried SVM with "one-vs-the-rest" approach.

Those different models have different pros and cons. Both Decision Tree and SVM can handle multi-class classifications and non-linear feature interactions. For Decision Trees, the model outputs are very straight forward and intuitive. We can easily explain our business problem with the Decision Tree model such as analyzing the important features. However, the biggest problem is over-fitting as the models can be highly biased. The SVM doesn't face big problem of over-fitting, and it doesn't rely on the entire data. The problem of SVM is that it's kind of difficult to find out the appropriate Kernel[3]. To solve this problem, we tested SVM with Kernel = RBF and Kernel = POLY to find out the better model.

To test the performance of these models, we decided to use precision and see which model has greatest precision in class 3, which is the class that has highest investment potential. We used precision after considered our business problem. We aimed to find the areas with highest investment potential, so we expected our model to place new inputs in correct classes. In addition, as those features in Class 3 has highest investment potential, we only considered the precision score of Class 3.

Linear SVM is suitable for our multi-class data and we tested its performance by using precision. The precision score of Linear SVM is 0.833, we wanted to improve our performance, therefore we tested the precision score for each model.

As we want to find out the real estate with highest investment potential–which drops in Class 3, we only compare the precision score of Class 3. In addition, to analyze different algorithms, we needed to find parameters that have best performance. For SVM(RBF), we made changes of C. For SVM[Poly], the degree matters, so we tested degrees within [2,3,4,5,6] as well.

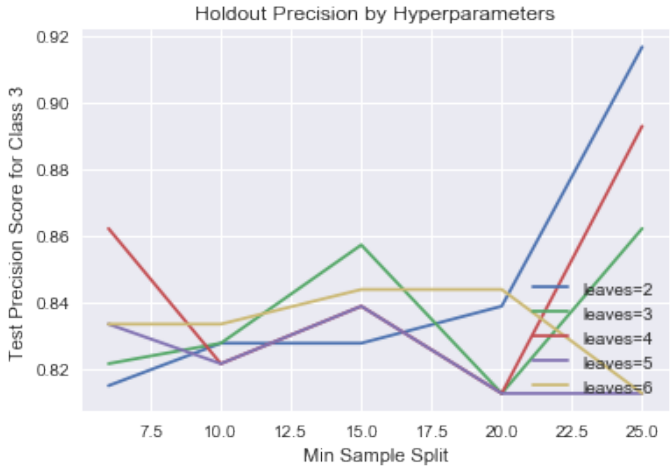

Figure 1: Decision Tree Precision

For decision tree, we tested min samples split and min samples leaf. We plotted the changes of precision score with different parameters to find the parameters with best precision score. After testing we got best parameters and precision score of these models. The precision score of decision tree was 0.884 with min samples split=25,min samples leaf=2, and the SVM models had lower scores, which was 0.827 for SVM[RBF] with C=10.

4.3 Precision and Recall Testing

After comparing different model we could find that the Decision Tree has the best performance. That is, after setting parameters, our model could do classification well and 88.4 percents of features which belongs to Class 3 were placed in the right class. To have detailed view of how many features were placed in the right class, we plotted the confusion matrix. From this confusion matrix we identified that 23 out of 26 features in Class 3 were placed in Class 3, while only 3 was placed in Class 2.

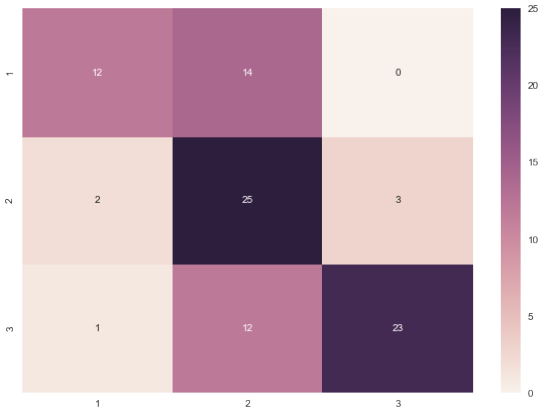

Figure 2: Confusion Matrix

We also wanted to know recall scores. We wanted to know how well our model could retrieve relevant instances-those belong to Class 3-over total relevant instances in our input data. We therefore plotted the precision-recall-curve. The precision score and recall score were related negatively. From the curve we could identify that, the more instances we retrieved, the lower precision we got. However, considering our business problem was to predict the investment potential, especially for high potential real estates, we cared more about precision of Class 3. As we had good precision

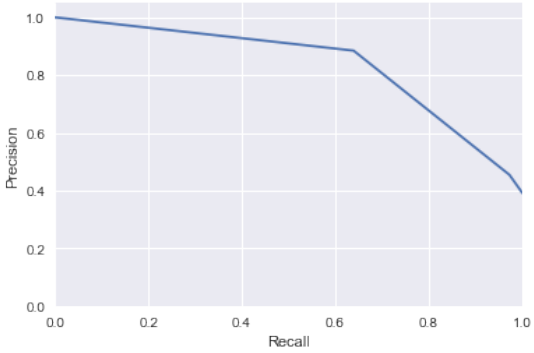

Figure 3: Precision-recall-curve

when we retrieved smaller amount of features, our model was suitable and we successfully improved the performance.

4.4 Further Analysis and Outputs

Because the Decision Tree model provided the best results, we did deeper analysis on it. We evaluated the important features of this model.

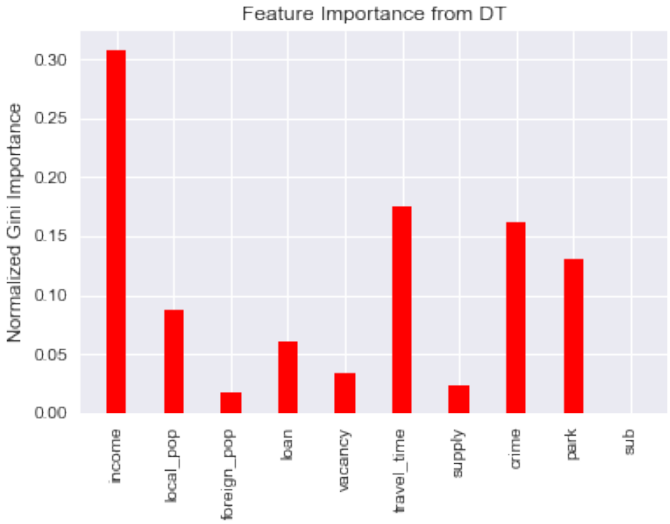

Figure 4: Important features

From figure 4 we could identify that, income was the most important factor to evaluate the value of the investment potential of houses and apartments. The second important factor was travel time and third was crime rate. This observation matches reality. Income is always the first and foremost factor considered by investors. Then people will take the neighborhood environment into account. They prefer real estates that locate near their working places and have safe and beautiful surroundings.

4.5 Data Mapping

To better understand recent change of NYC housing investment potential, we mapped all available data in Tableau shown the trend,

The map is based on Zip Codes areas in NYC and the color shows different level of investment potential. The scale arranges from 1 to 3. Deeper colors represent higher rental-to-sales ratios which consider as higher investment

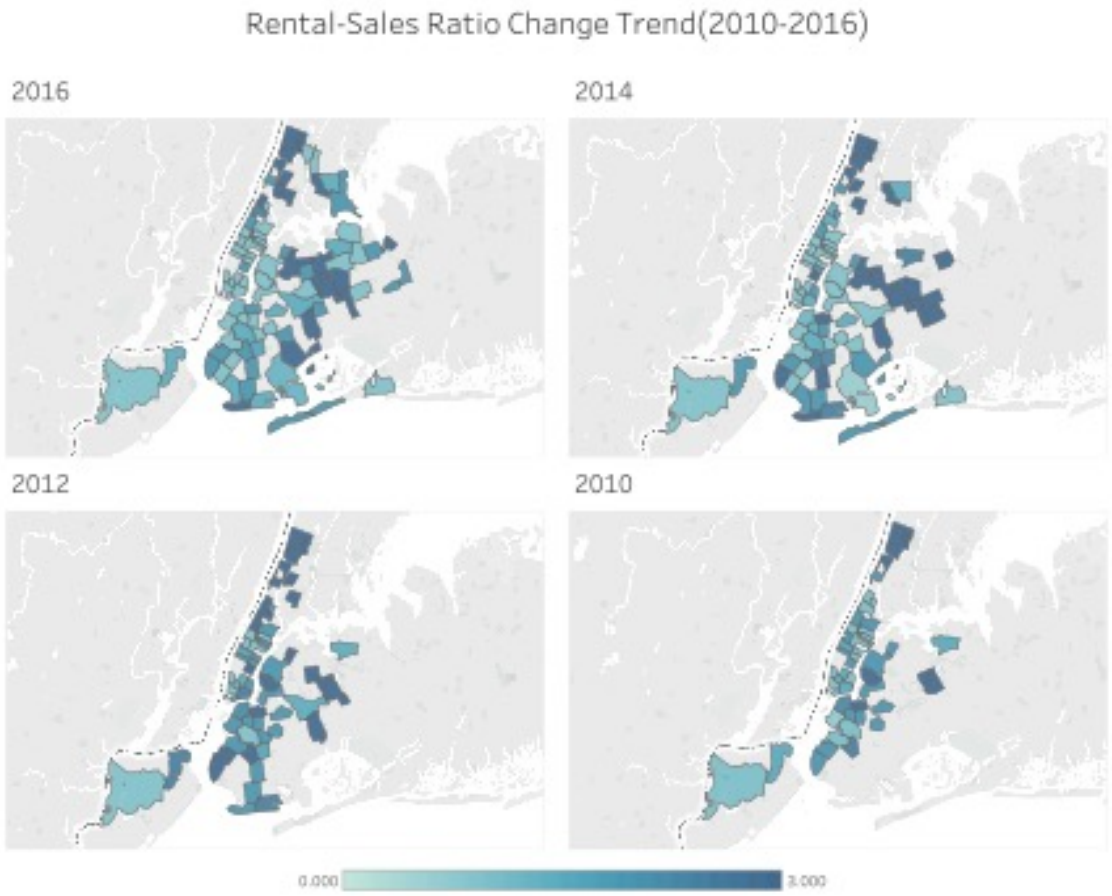

Figure 5: Rental-Sales Ratio Change Trend

potential in this area. All neighborhoods have been remediated to correct Zip Codes areas. The target time interval ranges from 2009 to 2016. Here we select four years as representatives of the whole trend. Several key findings are as below,

- **Investment potential has increased but edged down recently**. The existing data sample presented on the graph shows deeper filled areas in NYC from 2010 to 2014. But in 2016 the number of deep filled areas dropped which means relative rental price had an increasing trend
- **Outside-of-center has higher potential**. Most deep filled areas appear in boroughs except Manhattan which has proved that suburb areas in NYC has higher rental/sales ratio than the center.
- **Queens and Bronx have most deep filled areas**. On East Queens and North Bronx the ratios have remained high for several years on record.

We believe that for the trend there are several reasons behind. The first one is the high sales price in city center gives lighter colors in Manhattan. Manhattan is probably the most expensive place to buy a home in the country. According to official house price data provided by U.S. Census Bureau, the investment potential index for the five New York boroughs individually

Considering relatively high sales price in Manhattan, the new residents moving in in recent years have less demand

Table 2: Investmentd Potential table

	Manhattan	Brooklyn	Queens	Bronx	Staten Island
Index	0.164	0.198	0.285	0.269	0.226

of buying or renting houses in city center. The new resident, in contrast, would prefer a neighborhood where is away from Manhattan but with lower rent and convenient public transport access. It has also proved that why several suburb areas have higher ratios for these years.

5 Deployment

In our project we have built up a model by supervised learning method to discover relation between neighborhood features and high-potential area in New York real estate market. We used Linear SVM as baseline model, then improved the performance by testing precision scores of highest investment potential class of Decision Tree and SVM with different parameters. Then we plotted the important features to see how those features were related to investment potential.

One concern about our model is the over-fitting problem. To improve the future performance and resolve this problem, we can implement cross validation for small amount of data and use random forest if we increase our input features.

The outcome from our study can be applied as an independent reference for those who want to invest property by renting out to tenants to generate reliable cash flow. Most of such kinds of investors have already had their first property and look for a "cash cow" in their investment portfolio. The area in NYC with high and steady demand of rental indeed perfectly satisfy their requirement.

At least, our study doesn't ignore the fact that those shown as "low" potential areas might have better investment value from different perspectives. For example, Upper East Side and Upper West Side have always been the expensive neighborhoods in New York for a long time. Their attractions might generated from unparalleled location, upper-class social circle and high return from sales. We are only devoting to find out those affordable place with better matching facilities. Hence, the scope of our study should be limited.

References

[1] About coredata.nyc - coredata.nyc user guide. http://furmancenter.org/coredata/userguide/about/, November 2016.

[2] Data dictionary – Coredata.nyc User Guide. http://furmancenter.org/coredata/userguide/dictionary/, May 2017.

[3] Lalit Sachan. Logistic regression vs decision trees vs svm: Part ii. http://http://www.edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part2/, June 2015.

Appendix

Member Contribution

- Chun-Yi Yang: preliminary research, data cleaning, data analysis
- Xiaoxuan Tang: preliminary research, model testing
- Haozheng Zhu: preliminary research, data visualization

Model Testing Code

```
# model test
import pickle
with open('p_file/input.p','rb') as f:
     data_input = pickle.load(f)


# replace missing value
import pandas as pd
import numpy as np
# np.random.seed(5500)
df_mv = data_input.replace(0, np.nan)
df_replace = df_mv.replace(np.nan, df_mv.dropna().mean())


# split the testing and training data
s = np.random.rand(len(data_input)) < 0.8
train_df = df_replace[s]
test_df =  df_replace[~s]


# normalization and build the tree
import sklearn
import os
from sklearn import tree
from os import system
from sklearn.tree import DecisionTreeClassifier


Y = train_df['label']
X = train_df.drop('label', 1)


B = test_df['label']
```

```
A = test_df.drop('label', 1)


A_norm = (A - A.mean()) / (A.max() - A.min())
X_norm = (X - X.mean()) / (X.max() - X.min())

# set baseline model-linear SVM and test precision
from sklearn.metrics import precision_score, confusion_matrix, precision_recall_curve
from sklearn import svm,linear_model
import seaborn as sn
import matplotlib.pyplot as plt
from itertools import cycle

lin_clf = svm.LinearSVC()
lin_clf.fit(X_norm, Y)
B_pred_linear = lin_clf.fit(X_norm, Y).predict(A_norm)
cm_linear = confusion_matrix(B, B_pred_linear)
precision_linear = precision_score(B, B_pred_linear,labels=[3], pos_label=3,average=None)
print ('confusion_matrix_linear={}'.format(cm_linear),'precision_linear={}'.format(precision_linear))

# test different models
# first test SVM-rbf
import math
c = []
for i in range(-8,2,1):
    c.append(math.pow(10,i))

prec_svm = 0
for i in c:
    clf_best = svm.SVC(C=i,decision_function_shape='ovr',probability=True)
    clf_best = clf_best.fit(X_norm, Y)
    B_predict_svm = clf_best.fit(X_norm, Y).predict(A_norm)
    prec_rbf = precision_score(B, B_predict_svm,labels=[3], pos_label=3,average=None)
    if prec_rbf > prec_svm:
        prec_svm = prec_rbf
        C_best = i
        clf_rbf=clf_best
print (clf_rbf)
```

```
# then test the svm[ploy], with degree and c that provide best precision
Degree = [2,3,4,5,6]
prec_svmpoly = 0
for m in c:
    for f in Degree:
        clf2 = svm.SVC(C=i,kernel='poly', degree=f, decision_function_shape='ovr',probability=True)
        clf2 = clf2.fit(X_norm, Y)
        B_predict_poly = clf2.predict(A_norm)
        precpoly = precision_score(B, B_predict_poly,labels=[3], pos_label=3,average=None)
        if precpoly > prec_svmpoly:
            prec_svmpoly = precpoly
            C_poly_best = m
            clf_poly=clf2
        else:
            clf_poly=clf2
print (clf_poly)
precision_score(B, B_predict_poly,average=None)

# test decision tree, find the best precision for decision tree
min_samples_split_values = [6,10,15,20,25]
min_samples_leaf_values = [2,3,4,5,6]

splits = min_samples_split_values
leaves = min_samples_leaf_values

def testTrees2(X_train, y_train, X_test, y_test,split,leaf, auc):
    clf = DecisionTreeClassifier(criterion='entropy',min_samples_split=split,min_samples_leaf=leaf)
    clf = clf.fit(X_train, y_train)
    if (auc==0):
        cm = confusion_matrix(y_test, clf.predict(X_test))
        accuracy= (cm[0,0]+cm[1,1]+cm[2,2])/cm.sum()
        return accuracy
    else:
        return precision_score(y_test, clf.predict(X_test),labels=[3], pos_label=3,average=None)

%matplotlib inline
```

```
run=1
if (run == 1):
    res =dict()
    for leaf in leaves:
        res[leaf]= list()
    for leaf in leaves:
        for split in splits:
            res[leaf].append(testTrees2(X_norm,Y,A_norm,B,split,leaf,1))

fig = plt.figure()
ax=fig.add_subplot(111)
for i in range(0,5):
    plt.plot(splits,res[leaves[i]], label = 'leaves={}'.format(leaves[i]))

plt.legend(loc = 4)
ax.set_xlabel('Min Sample Split')
ax.set_ylabel('Test Precision Score for Class 3')
plt.title('Holdout Precision by Hyperparameters')

prec_dt = 0

for i in splits:
    for j in leaves:
        dt_best = DecisionTreeClassifier(criterion='entropy',min_samples_split=i,min_samples_leaf=j)
        dt_best = dt_best.fit(X_norm, Y)
        t_precision= precision_score(B, dt_best.predict(A_norm),labels=[3], pos_label=3,average=None)
        if t_precision > prec_dt:
            prec_dt = t_precision
            best_split=i
            best_leaves=j
            dt2=dt_best
print (dt2)
prec_dt

# plot precision_recall curve for decision tree
precision, recall, thresholds = precision_recall_curve(B, dt2.predict(A_norm), pos_label=3)
```

```
plt.plot(recall, precision, label='Precision-Recall curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.show()

# plot confusion matrix for decision tree
B_pred_best = dt2.predict(A_norm)
cm_best = confusion_matrix(B, B_pred_best)

import seaborn as sn
df_cm_best = pd.DataFrame(cm_best, index = [i for i in range(1,4)],columns = [i for i in range(1,4)] )


plt.figure(figsize = (10,7))
sn.heatmap(df_cm_best, annot=True)

# plot important features
fig, ax = plt.subplots()
width = 0.35
ax.bar(np.arange(10), dt2.feature_importances_, width, color = 'r')
ax.set_xticks(np.arange(len(dt2.feature_importances_)))
ax.set_xticklabels(X_norm.columns.values, rotation = 90)
plt.title('Feature Importance from DT')
ax.set_ylabel('Normalized Gini Importance')

# plot the tree
from IPython.display import Image, display
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from pydotplus import *

dot_data = tree.export_graphviz(dt2, out_file=None)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```