DSC 540
Instructor: Casey Bennett, PhD
Authors: Chaonan Shi
ID: 1901412

Analysis of New York City Airbnb open Data

Abstract:

Since 2008, guests and hosts have used Airbnb to expand on traveling possibilities and present more unique, personalized way of experiencing the world. Moreover, Airbnb has seen a meteoric growth since its inception in 2008 with the number of rentals listed on its website growing exponentially each year. Airbnb has successfully disrupted the traditional hospitality industry as more and more travellers, not just the ones who are looking for a bang for their buck but also business travellers resort to Airbnb as their premier accommodation provider.

Introduction:

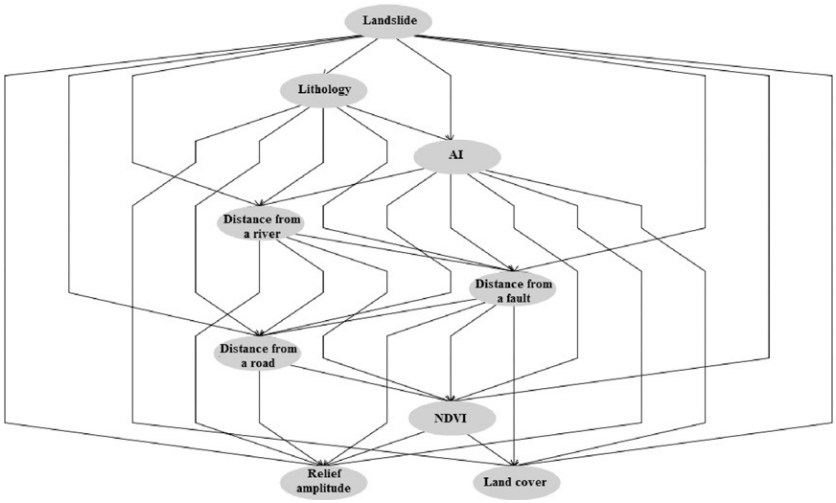
New York City has been one of the hottest markets for Airbnb, with over 52,000 listings as of November 2018. This means there are over 40 homes being rented out per square km. in NYC on Airbnb! One can perhaps attribute the success of Airbnb in NYC to the high rates charged by the hotels, which are primarily driven by the exorbitant rental prices in the city.

In this paper, I will perform an exploratory analysis of the Airbnb dataset sourced from the Kaggle (https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data) to understand the rental landscape and room type in NYC through various static and machine learning algorithms.

Literature review:

From <*Susceptibility assessment of earthquake-induced landslides using Bayesian network: A case study in Beichuan, China, Yiquan Song a,b, Jianhua Gong a,n, Sheng Gao c, Dongchuan Wang d, Tiejun Cui b, Yi Li a, Baoquan Wei e* > the article illustrate that a BN approach was adopted for modeling the occurrence of earthquake-induced landslides. Also, A BN model was developed through data training to determine the TAN structure, and the dependent relations between variables were quantified. In this paper, authors get the conclusion that the occurrence of a landslide is highly sensitive to relief amplitudes above 116.5 m. AI values between 109.3 and 109.5 m/s and the sandstone–shale and limestone rock types also have significant influences on landslides in the study area.

Resulting structure of Bayesian networks for the landslide-susceptibility assessment.

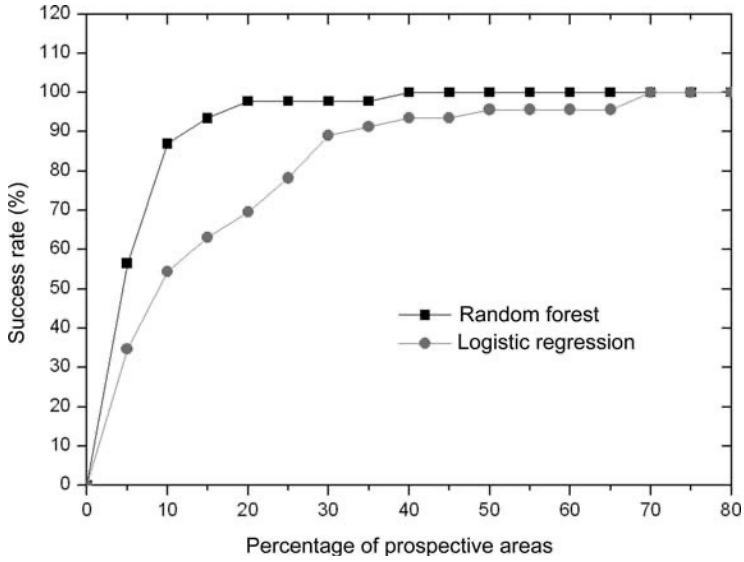


Resulting structure of Bayesian networks for the landslide-susceptibility assessment.

In another paper, <Predictive modelling of gold potential with the integration of multisource information based on random forest: a case study on the Rodalquilar area, Southern Spain, Rodriguez-Galiano (2014)>

The purpose of authors want to increase probability of finding mineral deposits in certain place, so that they introduced machine data-driven predictive method named Random Forest (RF). Then step by step, they went through the detail of RF first. Since objective of this study is to predict gold potential, so they can only apply regression method in this case. And introduced two mean algorithms to split nodes and calculate impurity: GINI index and Entropy. Second, using bagging process to avoid the correlation of different trees, since RF increases the diversity of the trees by making them grow from different training data subsets created through a procedure.

Throughout the whole study, the authors found that performance of RF algorithm is pretty good for the context of gold potential mapping from a suite of geochemical, geophysical, geological and remotely sensed GIS data.



Methodology:

In this paper, I will go through both NB (Gaussian & MultinomialNB) and Random Forest algorithm to predict the 'Private_room' from [room_type].

Preprocessing:
First, the dataset including 16 columns which including: numeric (ID, host id, price, etc), category (name, host_name, neighbourhood_group, etc), and geographic (latitude and longtitude) with 48896 records. The first thing is that we need to check missing value, since problem of missing value not only will lower our accuracy score, but also let us risk about overfitting problem. In this case, the feature of [last_review] and [reviews_per_month] lost over 20% record, considering that the 20% is also acceptable for missing case, I just filled all the missing blank by mode.

Figure 1: (before)

Out[4]:

	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365
	48895.000000	48895	48895.000000	48895.000000	48895.000000	38843	38843.000000	48895.000000	48895.000000
	NaN	3	NaN	NaN	NaN	1764	NaN	NaN	NaN
	NaN	Entire home/apt	NaN	NaN	NaN	2019/6/23	NaN	NaN	NaN
	NaN	25409	NaN	NaN	NaN	1413	NaN	NaN	NaN
	-73.952170	NaN	152.720687	7.029962	23.274466	NaN	1.373221	7.143982	112.781327
	0.046157	NaN	240.154170	20.510550	44.550582	NaN	1.680442	32.952519	131.622289
	-74.244420	NaN	0.000000	1.000000	0.000000	NaN	0.010000	1.000000	0.000000
	-73.983070	NaN	69.000000	1.000000	1.000000	NaN	0.190000	1.000000	0.000000
	-73.955680	NaN	106.000000	3.000000	5.000000	NaN	0.720000	1.000000	45.000000
	-73.936275	NaN	175.000000	5.000000	24.000000	NaN	2.020000	2.000000	227.000000
	-73.712990	NaN	10000.000000	1250.000000	629.000000	NaN	58.500000	327.000000	365.000000

Figure 2: (after)

```
: 1 arbnb['last_review'].fillna(arbnb['last_review'].mode()[0], inplace=True)
  2 arbnb['reviews_per_month'].fillna(arbnb['reviews_per_month'].mode()[0], inplace=True)
```

```
: 1 arbnb.describe(include="all")
```

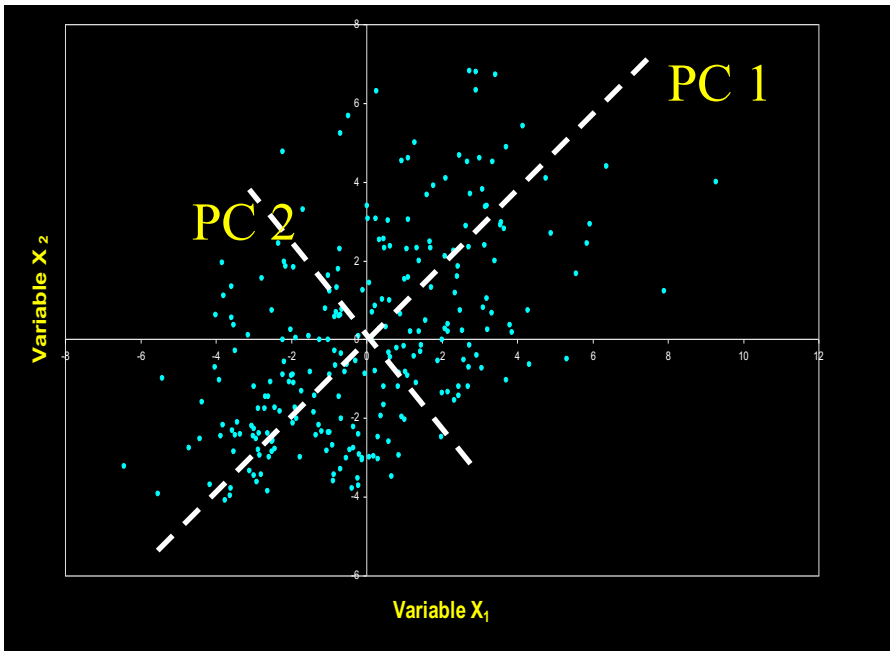
:

longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365
48895.000000	48895	48895.000000	48895.000000	48895.000000	48895	48895.000000	48895.000000	48895.000000
NaN	3	NaN	NaN	NaN	1764	NaN	NaN	NaN
NaN	Entire home/apt	NaN	NaN	NaN	2019/6/23	NaN	NaN	NaN
NaN	25409	NaN	NaN	NaN	11465	NaN	NaN	NaN
-73.952170	NaN	152.720687	7.029962	23.274466	NaN	1.095022	7.143982	112.781327
0.046157	NaN	240.154170	20.510550	44.550582	NaN	1.594493	32.952519	131.622289

Moreover, I also dropped several unhelpful or redundant features which may increase computational cost for us. in this case the feature ID and corresponding name is unique but duplicated , instead, we can use only ID to replace name and make our prediction more meaningful.

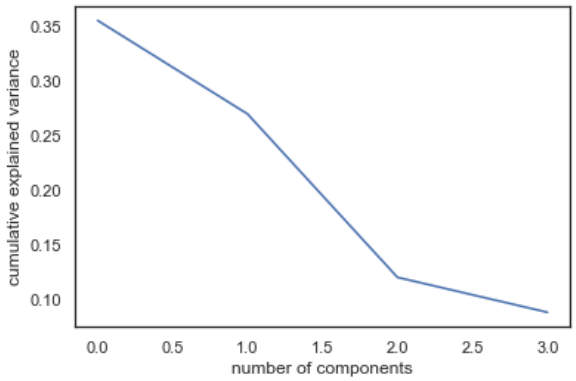
Feature Extraction:
PCA (principal component analysis) is a widely used data compression and dimensionality reduction technique. When we suffer the high dimensionality problem, PCA would be one of best approach to reduce the dimensional of dataset. Specifically, PCA takes a data matrix, *A*, of *n* objects by *p* variables, which may be correlated, and summarizes it by uncorrelated axes (principal components or principal axes) that are linear combinations of the original *p* variables.

Figure 3: (PCA)



From Airbnb dataset, the whole dataset can be grouped as four components, PCA1, PCA2, PCA3, and PCA4. The first component explained 35% variance, second component explained 27% variance, third component explained 12% variance, and last component explained 9% variance, so that first four components explained total 83% variance. Also from the scree plot we can clearly to observe this:

Figure 4: (Scree Plot)



Naïve Bayes:
In term of Bayes theorem, it plays a critical role in probabilistic learning and classification, uses prior probability of each class given no information about an item, also the models are incremental in the sense that each training example can incrementally increase or decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.

$$\Pr(H \mid X) = \frac{\Pr(X \mid H)\Pr(H)}{\Pr(X)}$$

On the top of that, Bayes theorem also has different approaches for example, the Gaussian and MultinomialNB. The Gaussian refers the strong independence assumption in the model, rather than the particular distribution of each features. For MultinomialNB, it focuses on the classification with discrete features (e.g., word counts for text classification)

$$P(E \mid c_i) = P(e_1 \wedge e_2 \wedge \cdots \wedge e_m \mid c_i) = \prod_{j=1}^{m} P(e_j \mid c_i)$$

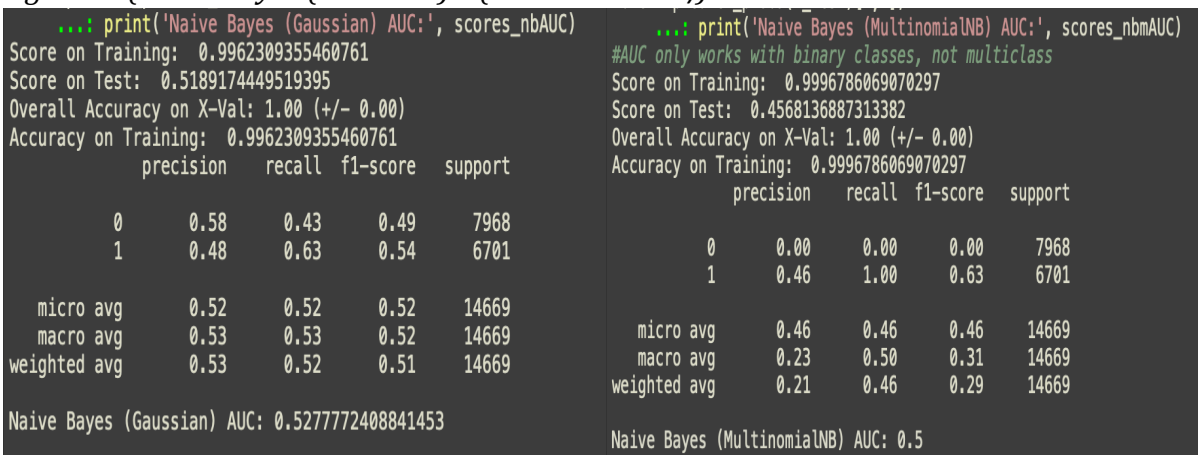
Random Forest Classifier:
Another powerful algorithm which I applied for this dataset which is Random Forest Classifier. Random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. For instance, single decision tree would only available for limited dataset, the performance may vary depending on the root node. In this case, for generalizing our model from one dataset to others, random forest would be one of best algorithm, not only for the model generalization, but also for overcoming the overfitting problem.

$$I(p,n) = -\Pr(P)\log_2 \Pr(P) - \Pr(N)\log_2 \Pr(N)$$

Results:

For the Airbnb dataset, I apply both version of Bayes theorem and to compare the performance from them. For the Random Forest Classifier, just using default setting with ‘entropy’.

Figure 5: (Naive Bayes (Gaussian)& (MultinomialNB)):



```
...: print('Naive Bayes (Gaussian) AUC:', scores_nbAUC)
Score on Training:  0.9962309355460761
Score on Test:  0.5189174449519395
Overall Accuracy on X-Val: 1.00 (+/- 0.00)
Accuracy on Training:  0.9962309355460761
              precision    recall  f1-score   support

           0       0.58      0.43      0.49      7968
           1       0.48      0.63      0.54      6701

   micro avg       0.52      0.52      0.52     14669
   macro avg       0.53      0.53      0.52     14669
weighted avg       0.53      0.52      0.51     14669

Naive Bayes (Gaussian) AUC: 0.5277772408841453
```

```
...: print('Naive Bayes (MultinomialNB) AUC:', scores_nbmAUC)
#AUC only works with binary classes, not multiclass
Score on Training:  0.9996786069070297
Score on Test:  0.4568136887313382
Overall Accuracy on X-Val: 1.00 (+/- 0.00)
Accuracy on Training:  0.9996786069070297
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      7968
           1       0.46      1.00      0.63      6701

   micro avg       0.46      0.46      0.46     14669
   macro avg       0.23      0.50      0.31     14669
weighted avg       0.21      0.46      0.29     14669

Naive Bayes (MultinomialNB) AUC: 0.5
```

Figure 7: Random Forest Classifier:

```
#AUC only works with binary classes, not multiclass
/Users/appobs/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:
246: FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
Score on Training:  1.0
Score on Test:  0.9967277933056105
Overall Accuracy on X-Val: 1.00 (+/- 0.00)
Accuracy on Training:  1.0
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      7968
           1       0.99      1.00      1.00      6701

   micro avg       1.00      1.00      1.00     14669
   macro avg       1.00      1.00      1.00     14669
weighted avg       1.00      1.00      1.00     14669

Random Forest AUC: 1.0
```

Discussion:

From above results, we can summarize that the random forest classifier performed way better that Naïve Bayes for both *Gaussian and MultinomialNB version.*
For Naïve Bayes, the results of both *Gaussian* and *Multinomial is similar*. Accuracy on training is close to 1.00, but testing score stay at 0.46 (*MultinomialNB*) and 0.52 (*Gaussian*). The AUC score for both is stay at 0.5.
In this case, it is apparently that we get problem about overfitting. In term of overfitting, the model is too hard to study from error term. Overfitting can be caused by several reasons, for example: not enough records, model structure is too complex, multicollinearity, etc.

Surprisingly, the random forest performed perfect. Accuracy on both training and testing is close to 1.00. Also the AUC score is exactly equal to1.

Conclusion:

Throughout the whole analysis, the random forest perform perfectly in [Airbnb open data] dataset. However, the fancy results from random forest may not reliable in real world. For instance, it possible that we overfitted in both train and test. From my point of view, the perfect results we get from random forest may because of multicollinearity. Since we create

too many dummies from original dataset and features, thus, model may study from each feature even the errors.

Future work:

Since we get serious overfitting problem in this analysis. I believe we may get wrong at data cleaning and feature engineer stage. For instance, it may not a good idea that create many dummies from original features, instead, we can discrete level from original feature into binary.

Moreover, we can also use the geographical features to analysis, for example, What can we learn about different hosts and areas, different hosts and areas, is there any noticeable difference of traffic among different areas and what could be the reason for it?

Reference:

https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html#sphx-glr-auto-examples-model-selection-plot-roc-py