Airbnb: Predicting Yield Through Text Mining and Data Analysis

CIS 4680

Text Mining Project

22 October 2019

Minjae S

Stephen A

Dylan A

Data

http://insideairbnb.com/get-the-data.html

Project Introduction:

Airbnb is an online marketplace that offers a platform where users can rent out whole private homes, to private rooms, to hotel suites, etc. While the user can view and book listings, the "host" finds challenges when trying to produce a profit. The Airbnb system allows for users to leave reviews for other users to view and use as a reference. However, new, or future Airbnb "hosts" are afraid that their listing might not product profit because of various challenges. These challenges are present in the Los Angeles area, as there are new, competitive listings everyday. With other factors and variables that are important, we want to calculate a yield, future earnings, for "hosts", so we can analyze and discover what variables produce to most yield. This can also help "hosts" calculate if they would want to enter the Airbnb market because one may be unsure if they want to invest unless they know they can gain profit. By taking in text descriptions of the listings, we will build a predictive model that will calculate future earnings. In order to fully understand our data, we will be doing a descriptive analysis, and we will use the linear regression, decision trees, random forest methods to see which model produces the best R-2 score. For this specific project, we will be using Airbnb's dataset from Inside_Airbnb for listings in the Los Angeles area from the past 6 months.

Project Statement:

To provide Airbnb "hosts" with a powerful prediction, analysis model, that will use "listing description" to predict a yield, possible future earnings. Yield will be calculated using important variables such as price, average length of stay, review rate. By creating a prediction model, we can identify what features (within the "listing description") are crucial for a healthy yield. The use of text mining will be used to analyze the "listing description", and target our calculated "yield" and compare the R-2 Score between the methods.

Questions: Can we create an accurate predictive model that can predict yield through text mining of Airbnb hosts' listing description? Between Linear Regression, Decision Trees, and Random Forest Regression, which is the most accurate?

Project Workflow:

The workflow and steps for this project are presented:

1. Import data from http://insideairbnb.com/get-the-data.html for Los Angeles area. (compiled Sept. 14, 2019)
2. Load data and needed modules
3. Clean, preprocess data
    a. Remove unnecessary features (dataset had 44,620 rows and 105 columns)
    b. Kept important variables

```
cols_to_keep = [
    'description',
    'property_type', 'room_type', 'accommodates',
    'bathrooms', 'bedrooms', 'beds', 'square_feet',
    'price', 'cleaning_fee', 'guests_included', 'extra_people', 'minimum_nights',
    'availability_365', 'reviews_per_month', 'latitude', 'longitude', 'bed_type'
]
```

   c. Removed part-time listings or new listings (available 300+ days) because we wanted to analyze full-time listings that have had at least a few reviews before.
   d. Removed listings that could accomodate 10+ people, as well as homes that were priced at $1,000+. This is because it is obvious that biggers homes will generate a larger yield, therefore, we don't want to include obvious information. We also don't want to fit our model to those listings.

4. Calculate yield.
   a. To calculate yield, we will use the following formula

```
# Calculate yield
# Average length of stay (by city) X Price ('price') X No. of reviews('reviews_per_month') / Review rate('50%')
# got information from inside.airbnb los angeles data
avg_length_of_stay_losangeles = 3
review_rate = 0.5
df['price'] = df['price'] + df['cleaning_fee']
df['yield'] = avg_length_of_stay_losangeles * df['price'] * (df['reviews_per_month'] / review_rate) * 12
```

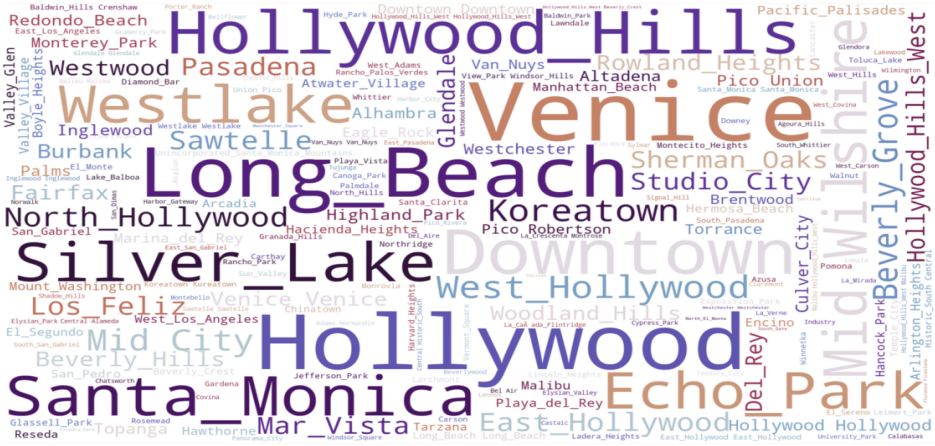
   b. Yield, possible future earnings, is calculated using average length of stay, price, reviews per month, and review rate.
5. Feature engineering - NLP, LDA
   a. We have our target variable, "yield", so we will use the NLP pipeline model to create a dictionary and matrix for our "listing description"
   b. We will also use the LDA topic model to find the important features within the descriptions and find their respective weight.
6. Test and compare results
   a. Linear regression, decision trees, and random forest regression will be used to improve the model
   b. We used linear regression because with text mining we can see if there is a binary relationship between the "listing description" and our calculated "yield".
   c. Decision Trees can help us find the important features and its importance.

Descriptive Analysis:

In order to properly categorize the listing descriptions we pulled the most frequent terms used in the description of the listings to give them the proper topics. This process was to make it easier to identify the terms we should use in our descriptions in order to increase yield. We also created a word cloud to better understand the impact location has on yield and the most popular locations within our dataset.

[{"metadata":{"trusted":true},"cell_type":"code","source":"import heapq\nmost_freq = heapq.nlargest(20, wordfreq, key=wordfreq.get)\nmost_freq","execution_count":137,"outputs":[{"data":{"text/plain":"['room',\n 'private',\n 'bedroom',\n 'house',\n 'beach',\n 'bed',\n 'kitchen',\n 'home',\n 'hollywood',\n 'one',\n

'bathroom',\n 'large',\n 'located',\n 'full',\n 'parking',\n 'la',\n 'apartment',\n 'living',\n 'venice',\n 'restaurants']"},"execution_count":137,"metadata":{},"output_type":"execute_result"}]}]



Predictive Analysis:

Linear Regression:

For the linear regression model, we had to change a few of the variables into dummy variables such as property_type, room_type.

In [34]:
```
# Dummy encoding
categorical_feats = ['property_type', 'room_type', 'bed_type']
df = pd.get_dummies(df, columns=categorical_feats, drop_first=False)
print("Dataset has {} rows, {} columns.".format(*df.shape))
```

```
Dataset has 13049 rows, 24 columns.
```

After, we declare our target variable as ‘yield’, and we split our dataset into a training set and test set.

```
# Create response and target variable
target = 'yield'
X = df.drop(target, axis=1)
y = df[target]
```

```
# Train test split
test_size = 0.3

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=seed)
```

```
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

Although, linear regression isn't the most powerful tool, it helped us identify the variables we needed. With this model, we were able to model a relationship between the "yield" and important features in the description.

In [202]:

```
# Make predictions using the testing set
regr_y_pred = regr.predict(X_test)

# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test, regr_y_pred))
# Explained R-2 score: 1 is perfect prediction
print('R-2 score: %.2f' % r2_score(y_test, regr_y_pred))
```

```
Mean squared error: 925952989.41
R-2 score: 0.28
```

With our regression model, we were able to get a R-2 score of 0.28, while 1 is for perfect prediction. It isn't very effective but it is only a baseline model for us at the moment.

Decision Trees:

Originally, we were going to compare the methods to one another and choose the most accurate. However, because our linear regression model wasn't very accurate, we decided to improve the existing model by fitting the data onto the decision tree regressor.

```
from sklearn.tree import DecisionTreeRegressor
```

```
# Fit regression model
dt_regr = DecisionTreeRegressor(random_state=seed, max_depth=5)
dt_regr.fit(X_train, y_train)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=5, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=42, splitter='best')
```

```
# Make prediction
dt_y_pred = dt_regr.predict(X_test)

# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test, dt_y_pred))
# Explained variance score: 1 is perfect prediction
print('R-2 score: %.2f' % r2_score(y_test, dt_y_pred))
```

```
Mean squared error: 824012882.50
R-2 score: 0.32
```

With the Decision Tree regressor, we were able to improve our model to a 0.32 R-2 score.

Random Forest Regressor:

With online sources, we were also able to fit the Random Forest regressor after fitting decision trees. With the Random Forest regressor, our R-2 Score slightly increased. We also learned that it is the most complex form of decision trees and could make individual predictions as well.

```
# %%time
rf_regr = RandomForestRegressor(random_state=seed, bootstrap=True, criterion='mse', max_depth=10,
                            max_features='auto', min_samples_split=4, n_estimators=100)

rf_regr.fit(X_train, y_train)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=10,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=4,
           min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
           oob_score=False, random_state=42, verbose=0, warm_start=False)
```

```
# Make prediction
rf_regr.fit(X_train, y_train)
rf_y_pred = rf_regr.predict(X_test)

# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test, rf_y_pred))
# Explained variance score: 1 is perfect prediction
print('R-2 score: %.2f' % r2_score(y_test, rf_y_pred))
```

```
Mean squared error: 749229817.56
R-2 score: 0.39
```

With the Random Forest regressor, we got our R-2 score to 0.39

Model Evaluation:

While linear regression isn't the best predictive model, we learned that it serves as a very good baseline model to improve on. Linear regression builds a relationship between variables, and that is what we needed for our model. Once we built a relationship between the yield and description, we were able to improve our model with decision trees and random forests. In order to make a more accurate model, we needed to toggle with the random state as well as testing with other methods. In order to evaluate our model further, we created a function for feature_importance to see the difference in weight in different features. We expected to see the top 3 features as most important because location and house/room type is what most people look for. After viewing the important features, we realized that people don't really read the "listing description" and rather just compare important features like the ones below. Therefore, in order to maximize yield, especially for new "hosts", one must have relative competitiveness in the following features rather than a strong description.

Different methods could be used to maximize the accuracy of this model, and toggling of the numbers may also help. However, this project provides insight on important information for new Airbnb hosts. Topic modeling can be hard for listing descriptions because there isn't much sentiment like a review. To further analyze the Los Angeles Airbnb area, we could join listings and reviews to further understand the important features to maximize yield.

```python
#Display feature importance
def feature_importance(model, trainData, display_n_rows):
    """Display feature importance & weighting for tree based model"""
    fi = model.feature_importances_*100
    feat_imp = pd.DataFrame(list(zip(fi,trainData.columns.values)))
    feat_imp = feat_imp.sort_values(by=0, axis=0, ascending=False)
    feat_imp.columns = ['importance %', 'feature']
    print(feat_imp[:display_n_rows])
```

```python
#Display features & weighting
rf_best = rf_cv.best_estimator_
feature_importance(rf_best, X_train, 20)
```

	importance %	feature
19	19.460896	room_type_Entire home/apt
8	16.723040	longitude
7	15.776929	latitude
6	14.262518	minimum_nights
0	11.027638	accommodates
5	5.666040	extra_people
4	3.339480	guests_included
1	2.950248	bathrooms
3	2.876839	beds
2	1.854468	bedrooms
9	1.014793	property_type_Apartment
14	0.816511	property_type_House
11	0.671349	property_type_Condominium
16	0.640382	property_type_Other
10	0.608685	property_type_Bungalow
13	0.524941	property_type_Guesthouse
17	0.517746	property_type_Townhouse
12	0.413557	property_type_Guest suite
18	0.380767	property_type_Villa
15	0.318647	property_type_Loft