Machine Learning Engineer Nanodegree

Capstone Project

Joaee Chew
September 14, 2017

I. Definition

(approx. 1-2 pages)

Project Overview

AirBnb is a popular home-sharing platform, enabling people all over the world to lease or rent short-term lodgings. For potential hosts, this is a potentially lucrative option for their under-utilised vacation homes, spare rooms or even beds. However, it is difficult for new hosts to ascertain how much their property could earn them and whether it will be a worthy investment. Furthermore, how can they know the true value of their beloved penthouse adorned with designer furniture, compared to your run-of-the-mill IKEA filled apartment?

This project seeks to solve the problem by building a predictive model for the potential earnings of Airbnb listings, taking into account text descriptions to capture a rich and qualitative model of individual listings. While there are available commercial tools such as AirSorted (http://www.airsorted.uk/) that predict Airbnb value, these models are generic with minimal features used and no indication of model accuracy.

To build this model, I use the dataset provided by Inside Airbnb (http://insideairbnb.com/get-the-data.html), where publicly available information about a city's Airbnb's listings have been scraped and released for independent, non-commercial use. This includes details about the listing such as no. of rooms, guests available, description, location as well as information about the yield such as price and no. of reviews.

Specifically, I will use the detailed listings information for London listings active from 3rd October 2016 – 4th March 2017. An active listing is defined as a property that has been reviewed at least once during this time period. After data cleaning, the dataset has 9722 rows and 16 columns. Importantly, the dataset contains the listings description as well as headers in text format that I will mine for qualitative predictors.

Problem Statement

To build this model, the concept of yield is used as a proxy for potential future earnings. Yield is defined as the amount of revenue that a property will earn over a year. The calculation of yield uses Inside Airbnb's 'San Francisco Model' based on price, average length of stay and review rate. A review rate of 50% (http://insideairbnb.com/about.html) to convert reviews to estimated bookings.

> **This project will predict the yield on prospective AirBnb listings in London and test whether text mining can lead to an increase in predictive accuracy.**

The intended workflow for this project is as follows:

1. Read, clean and preprocess the data:
   a. Remove unnecessary features. This includes data that does not help with predicting yield, such as listing URL or scrape ID.
   b. Remove 'leaking' features. This includes data that should not be used to predict yield, such as ratings (which are not present at the point of listing), or the no. of other properties listed by the host (model should be built on the property only and not the host).
   c. Removing incomplete listings (e.g. listings with no price, or with availability less than 50% which indicates part-time listings that are not comparable).
   d. Impute missing values (e.g. using the most frequent or median where appropriate).
2. Build the model. I will use three models of increasing complexity, both to test for increase in predictive power but also to act as challenger models to the final model.
   a. I will begin with a simple linear regression to establish a benchmark accuracy.
   b. I propose the use of a decision tree as a next step.
   c. Finally, I will use a random forest ensemble model as the final model.
3. Feature engineering. This will be done iteratively with step 3, with new features tested for improvement in model accuracy. Potential ideas include:
   a. Using natural language techniques to extract features from the description. This involves the following NLP pipeline:
   i. Text data preparation (creating corpus, word stemming, creating document term matrix, removing sparse terms etc.)
   ii. Selection of features using bag-of-words and/or TF-IDF
   iii.Use topic modelling to discover any natural topics in the listing descriptions
   b. Bucketing categorical features to reduce dimensionality (eg. Whole apartment vs. non-whole apartment instead of entire class size).
4. Test the model. Here I will print final accuracy measures and produce insights such as feature importance.

Metrics

Mean squared error is used as the measure for model accuracy. This measures the average of the square of the errors between the actual yield and predicted yield. The formula for this is as follows:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

MSE is a commonly used error score for regression models, and allows for an intuitive measure of error i.e. an MSE of 10000 indicates the yield model is off by £100 (square root of 10000). It is always non-negative, and values closer to zero are better.

II. Analysis

(approx. 2-4 pages)

```
Dataset has 53904 rows, 94 columns.

//anaconda/lib/python3.6/site-packages/IPython/core/interactiveshel
l.py:2717: DtypeWarning: Columns (88) have mixed types. Specify dtyp
e option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Data Exploration

The data set as scraped by Inside AirBnb is extremely comprehensive, containing 53904 rows and 94 columns. However, many of the data collected are unnecessary metadata (e.g. urls, id, pictures, host data), review data that causes leaking (e.g. review_scores), or has low relevance to the yield model (e.g. security_deposit, is_location_exact, notes).

To keep the model manageable and effective, I will focus on the following features:

```
1. 'name': Listing header text for text mining.
2. 'description': Listing description text for text mining.
3. 'property_type': Type of property e.g. apartment, house etc.
4. 'room_type': Type of room e.g. private, shared etc.
5. 'accommodates': No. of people the listing can accommodate.
6. 'bathrooms': No. of bathrooms.
7. 'bedrooms': No. of bedrooms.
8. 'beds': No. of beds.
9. 'square_feet': Size of listing.
10. 'price': Price of listing.
11. 'cleaning_fee': Cleaning fee.
12. 'guests_included': No. of guests included in the base price.
13. 'extra_people': Price for extra guests not included in the base pric
e.
14. 'minimum_nights': Minimum no. of nights required for booking.
15. 'availability_365': No. of nights the listing is available for bookin
g. (Note: This is a 'leaking' feature as listings may be unavailable as i
t is already booked and leaks the popularity. However, this needs to be u
sed for filtering part-time listings and will be removed at a later stag
e.)
16. 'reviews_per_month': Average no. of reviews listing receives per mont
h. Used to calculate yield.
17. 'latitude': Latitude of listing.
18. 'longitude': Longitude of listing.
```

A sample of the reduced dataset along with summary statistics is shown below.

```
Dataset has 53904 rows, 18 columns.
```

	property_type	room_type	accommodates	bathrooms	square_feet	price	availability_365	r
id								
15896822	Apartment	Private room	1	1.0	NaN	$23.00	61	0
4836957	Apartment	Private room	2	1.0	NaN	$50.00	364	0
13355982	Apartment	Private room	2	1.0	NaN	$24.00	0	0
13472704	House	Private room	2	1.5	NaN	$50.00	0	N
17430865	House	Private room	1	1.0	NaN	$25.00	179	N

```
Dataset has 53904 rows, 18 columns.
```

	accommodates	bathrooms	bedrooms	beds	square_feet	price	square_fe
count	53904.000000	53644.000000	53811.000000	53731.000000	582.000000	53904.000000	582.00000
mean	3.036676	1.262751	1.353980	1.708027	577.508591	96.099622	577.50859
std	1.907429	0.547699	0.841912	1.201165	726.154243	117.641082	726.15424
min	1.000000	0.000000	0.000000	0.000000	0.000000	8.000000	0.000000
25%	2.000000	1.000000	1.000000	1.000000	108.000000	42.000000	108.00000

The summary statistics reveal certain interesting insights about the dataset, and these are explored individually below.

Removing part-time listings

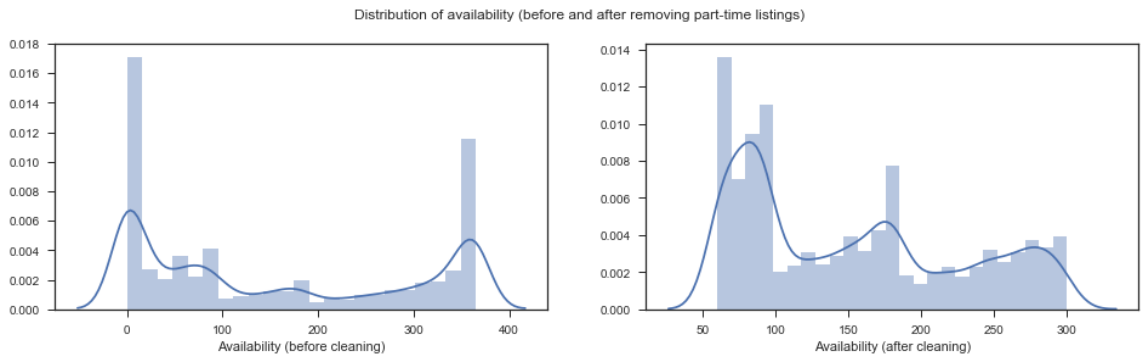
An Airbnb host can setup a calendar for their listing so that it is only available for a few days or weeks a year. Other listings are available all year round (except for when it is already booked). For consistency, I will use Inside AirBnb's definition of highly available being >60 days a year.

I will remove part-time listings by removing availability <60 days a year, and also newly-listed listings with availability >300 days (using the other tail end as approximation).

The below distribution curve shows a bi-modal distribution for availability, demonstrating two concentrations of listing types. The lower availability peaks below 50 days indicating a part-time or non-committed listing, while the higher availability peaks at almost at the maximum 365 indicating a dedicated rental property.

Once used to clean the data, I will remove availability as it is a leaking feature. It reveals how many bookings there will be in the coming year.

```
Before cleaning, dataset has 18502 rows, 18 columns.
After cleaning, dataset has 18502 rows, 17 columns.
```

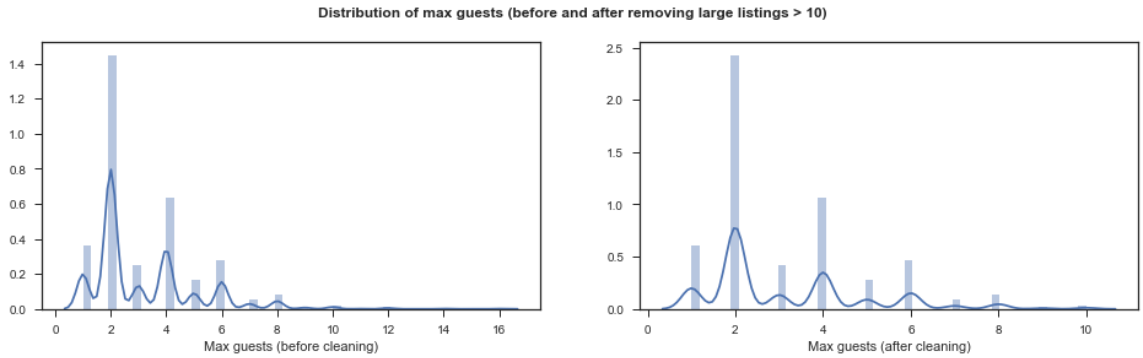


Removing large houses

For this model, I will remove extremely large rentals accommodating more than 10. The no. of persons a listing can accommodate will directly impact the calculation of yield, and whilst it is possible for a host to influence this number, this is less within their control and more an indicator of size. It is also unlikely that we have enough data to make tailored predictions for such listings.

```
You dropped 116 rows.
Dataset has 18386 rows, 17 columns.
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1323e8518>
```

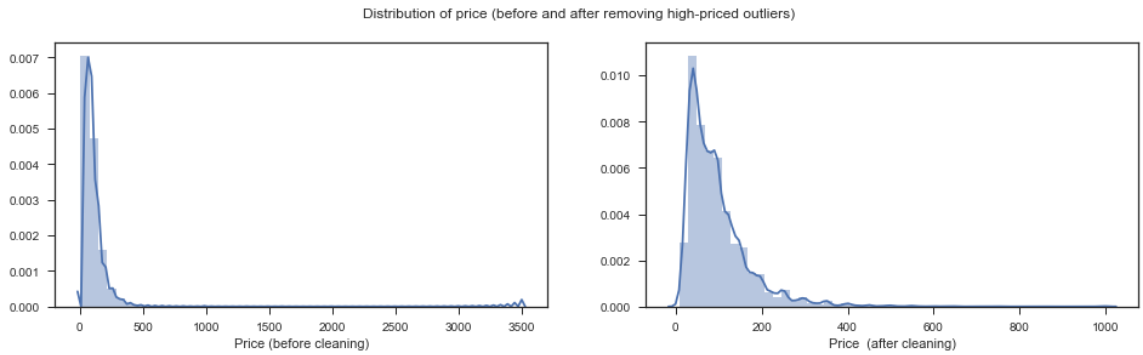


Removing high-priced rentals

The below distribution curve for price is right-skewed with a long tail of low frequency high-priced rentals. For this model, I will remove extremely high priced rentals above £1000/night to maintain comparability. It is also unlikely that we have enough data to make tailored predictions for such bespoke listings

```
You dropped 8 rows.
Dataset has 18378 rows, 17 columns.
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1207aa320>
```



Changing rare category types into 'other' bucket

For certain categories, we need to have a big enough sample set to ensure that it is statistically significant, i.e. the category is common enough to isolate correlation between that and predictor. This has an added benefit of reducing dimensionality. For property types, I create an 'Others' category for all except for the most common ones - Apartment, House and Bed & Breakfast.

```
Apartment             13478
House                  4060
Bed & Breakfast         360
Townhouse               103
Loft                     85
Other                    83
Guesthouse               37
Dorm                     33
Serviced apartment       33
Boat                     32
Bungalow                 19
Condominium              18
Cabin                    16
Boutique hotel           10
Hostel                    4
Villa                     3
Chalet                    2
Lighthouse                1
Yurt                      1
Name: property_type, dtype: int64


//anaconda/lib/python3.6/site-packages/pandas/core/indexing.py:179:
 SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)


//anaconda/lib/python3.6/site-packages/pandas/core/indexing.py:179:
 SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

Calculation of yield

As disuccsed earlier, **Yield** is defined as the amount of revenue that a property will earn over a year. This is calculated as follows:

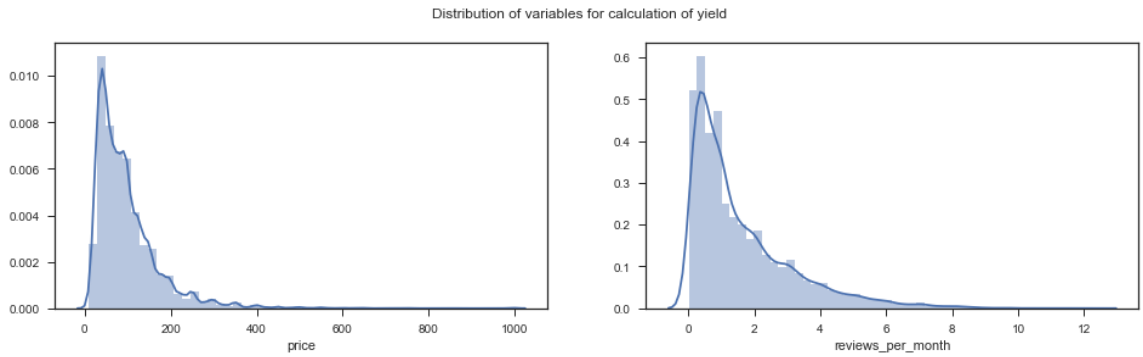
> AVERAGE LENGTH OF STAY X PRICE X NO. OF REVIEWS/MTH X REVIEW RATE X 12 MONTHS

Using Inside AirBnb's San Franciso Model, an average length of stay in London is 3 nights with a review rate of 50%. Note that this is fairly conservative, and yield in reality is likely to be higher.

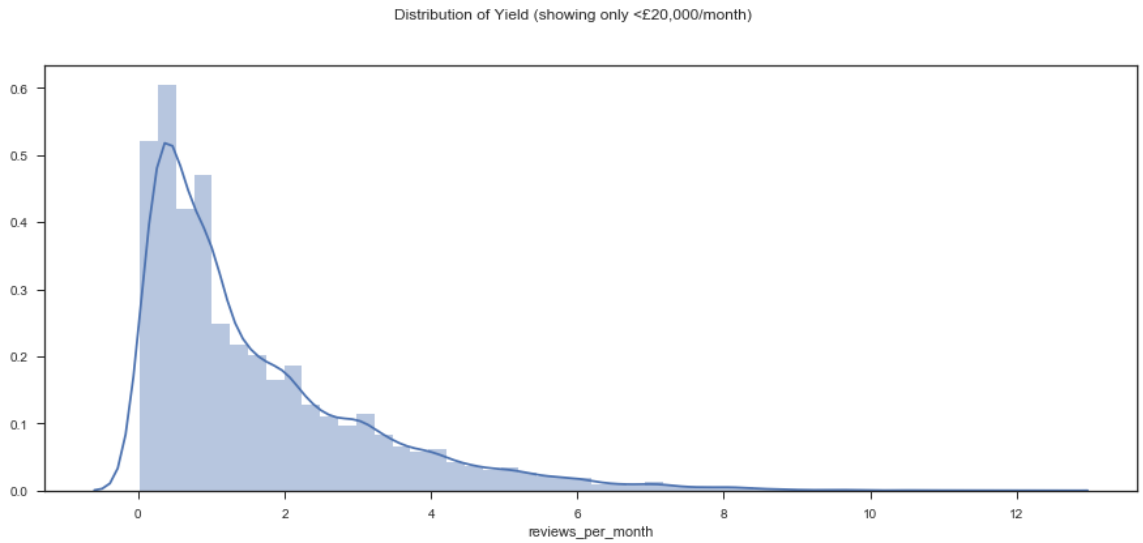
Analysing the distributions for price, reviews and resulting yield makes intuitive sense. The bulk of listings are less than £100/night probably for a private room, and receive 1-4 bookings a month for stays over the weekend. This indicates occupancy rates of 10-50%, earning the average AirBnb host £15,739 per listing in a year, going up to £21,480 for upper quartile listings.

Sense checking against the median residential rents in Central London (https://www.london.gov.uk/what-we-do/housing-and-land/renting/london-rents-map,) of £7,800 also supports the numbers.

```
<matplotlib.axes._subplots.AxesSubplot at 0x1232dcdd8>
```




```
Dataset has 18378 rows, 17 columns.
```



Missing values

A preview of the dataset above shows that there are missing values in some of the features such as 'beds', 'price' and 'reviews_per_month'. Depending on the nature of the column, we will treat missing values differently.

In 'square_feet', almost 90% of the rows are missing and I will choose to remove the feature completely

As 'yield' is an important target variable, I will completely remove any rows that have this datum missing. This will also take care of 'price' and 'reviews_per_month' as those are the required data to calculate 'yield'.

For the remaining features with a small number of missing values, I impute the categorical features with the most frequent category and numericla features with the median.

```
Dataset has 9722 rows, 16 columns.
```

Topic modelling for listing description text

For textual information, we use a Natural Language Processing pipeline to convert the corpus into a Document-Term-Matrix, whereby each listing (document) consists of a matrix of terms (processed words). With this in place, we can use Latent Dirichlet Allocation (LDA) to discover topics inherent in the corpus, classify the corpus according to the learned topics and use them as features for the regression model.

> LDA is a generative Bayesian inference model that associates each document with a probability distribution over topics, where topics are probability distributions over words. It is an efficient way to analyse large volumes of text and is a more human interpretable method of topic modelling.

In this project, I model the listing descriptions using LDA to explore emerging topics. The listing description field contains the bulk of the textual information found in a listing, and may contain important information that is not captured elsewhere. What emergent topics might we see from the rich text describing the history of a home, a host's beloved neighbourhood or it's carefully curated interiors?

```
Dataset has 9722 rows, 15 columns.
```

	description
id	
14912894	Lovely bright and sunny room overlooking the g...
14296637	My place is situated about 10-15mins walking d...
6222799	Sunny ensuite room in the newly converted loft...
7327883	Surbiton is very convenient for getting into ...
7515814	Our one bedroom apartment is a comfortable spa...

```
CPU times: user 8.08 s, sys: 126 ms, total: 8.21 s
Wall time: 8.23 s
```

```
CPU times: user 3min 15s, sys: 201 ms, total: 3min 15s
Wall time: 3min 16s
```

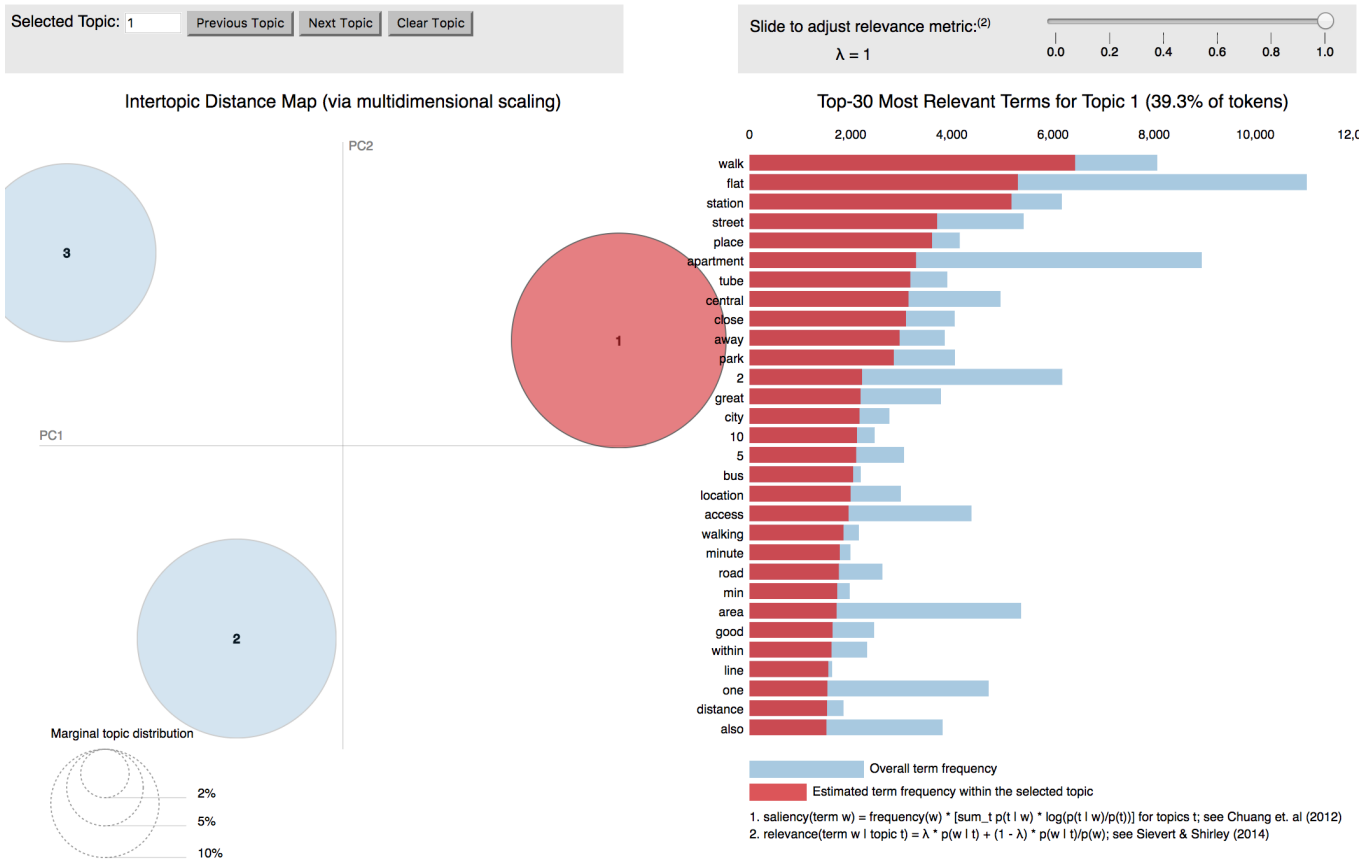
Exploratory Visualization

The topic models are visualised in an Intertopic Distance Map, which uses the brilliant gensim and pyLDAvis packages. In this visualisation, the area of the circles represent the prevalence of each topic while the length of the bars on the right represent the membership of a term in a particular topic. For example, topic 1 below has the most prevalent, and within topic 1 the term 'walk' has the highest estimated term frequency (red area) compared to the overall term frequency (blue and red area) of the term in the entire corpus.

```
//anaconda/lib/python3.6/site-packages/pyLDAvis/_prepare.py:387: Dep
recationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer
-is-deprecated
  topic_term_dists = topic_term_dists.ix[topic_order]

CPU times: user 11.4 s, sys: 103 ms, total: 11.5 s
Wall time: 12 s
```



I chose to cluster the document terms into 3 topics as it was the largest number of topics that had no overlaps with very distinct topics. The separate topics also makes sense within the context of AirBnb listings.

Based on the top terms in each topic, the topics for listings could be described as:

- Topic 1 - Location: This focuses on location with words such as 'location', 'walk', 'station' and 'central'. This might appeal to travellers who value convenience and proximity to transport links, such as first-time tourists in London who want to maximise their time and visit all the attractions in Central London.
- Topic 2 - Luxury: This seems to focus on modern apartments, with words such as 'kitchen', 'modern', 'private' and 'space' standing out. This might appeal to travellers with a bigger budget and want to stay in a comfortable place with space and privacy (which is expensive in London!).
- Topic 3 - Budget: This focuses on words associated with more basic facilities such as 'bed', 'room', 'shower' and 'clean'. This might appeal to budget travellers who don't want to spend too much and are looking for listings that assure them of the basic amenities.

Algorithms and Techniques

I intend to train 3 regression algorithms to predict the yield for the listings.

1. Linear Regression

The first algorithm used is a simple linear regression. This is a linear approach for modelling the relationship between the features in the data set (e.g. no. of rooms, no. of guests accommodated) and the target 'yield' variable.

The features are the x values, which is assigned a coefficient. There is also an additional degree of freedom (the intercept). The x values and the intercept equates to the y value. In a simple regression problem with single x and single y, the equation of the model is:

$y = B0 + B1*x$

In this scikit-learn implementation, the linear regression algorithm assigns the coefficients using the Ordinary Least Squares procedure. This works by calculating the distance from each data point to the regression line, square it and sum all of the squared errors together. This is the residual sum of squares between the x and y values that the algorithm seeks to minimise.

This algorithm is used to first establish a baseline model that we can then seek to improve on.

2. Decision Tree

The second algorithm is a Decision Tree algorithm. This is a more complex tree-based model that can capture nonlinear relationships in the dataset whilst still retaining interpretability. It is also robust to missing values, and has the added advantage of being able to help with feature selection based on feature importance.

The model uses a tree representation, where each internal node of the tree corresponds to a feature (X variable) and each leaf node is a class label. For numerical features, the classes are first discretized.

To learn the decision rules, this scikit-learn implementation uses the criterion of Information Gain. This borrows from Information Theory, and estimates the amount of information in each attribute. Information gain is calculated by first measuring the mean squared error using each attribute, and then calculated the expected reduction in error by using the attribute as a node split. The attributes with the highest information gain is placed as the root node, with subsequent nodes placed according to information gain until the final tree is formed.

3. Random Forest

This is the most complex ensemble model built from decision trees, and should be able to provide additional accuracy. While it sacrifices interpretability, sense checking can be made with the earlier decision tree model, using sensitivity analysis and analysing feature importances.

The principle behind this ensemble model is that a group of weak learners, the individual decision trees, can be used together to create one strong leaner - the Random Forest model.

To build the ensemble model of decision trees, a random sample of N cases are taken to build each tree T. At each node of tree T, a random number of predictor variables M is selected. The variable with the best split, in this instance based on MSE, is used. This process is repeated for each node split with a new random set of M predictor variables.

Once all the trees in the model have been built, new inputs are run down each of the trees with each tree having its own results. The results are then averaged to return the overall result for the random forest model.

Benchmark

To my knowledge, there are no existing Airbnb pricing models released to the public. However, I will use a hold-out set to test my model, and the mean squared error will be used to measure the accuracy of the model.

I will also build challenger models, and in particular start with a linear regression model to act as the baseline benchmark. A linear regression model is chosen as it is simple and interpretable, and the results can be easily interrogated. The methodology for building this model can be found below in the methodology section.

After building the model, an MSE of 254474109 with variance of 0.18 is established.

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

Before performing any machine learning, the dataset needs to be pre-processed. The following steps were taken for this project and discussed in the 'Data Exploration' section of the report:

1. Converting data types (e.g. string into integers)
2. Removing part-time listings
3. Removing large houses
4. Removing high-priced rentals
5. Changing minor category times into ‘Other’
6. Calculation of yield
7. Topic modelling

In addition to those steps already taken, missing values need to be imputed, features have to be selected and categorical features have to be encoded.

Feature selection

I also remove features that are not necessary for the machine learning phase.

'Price' and 'reviews_per_month' are removed as they are used to calculate 'yield' and would have a strong correlation with 'yield'.

```
Dataset has 9722 rows, 13 columns.
```

Dummy encoding

Finally, categorical features need to be encoded to be treated properly by the machine learning algorithms. As the algorithms require numerical features only, the different categories have to be given a number. However, the categorical features present do not have any order inherent in them (eg. apartment vs. house) and thus needs to be encoded where 1 is given if the category is true and 0 is not.

```
Dataset has 9722 rows, 19 columns.
```

Implementation

Once the dataset has been processed it is now ready to for the models to be built.

1. As a first step, the target variable 'yield' has to be seperated from the response variables to from the X and y data sets.
2. Both X and y data sets and respectively split into a training set and a testing set. I chose to use 30% of the data set for the testing set. As part of the splitting process, the data set is also shuffled to ensure there is no inherent or hidden order to the data set and that both the testing and training sets are a fair representative of the complete data set.
3. In the training phase, the training set is fitted to the 3 algorithms set out in the 'Algorithms and Techniques' section. During this process, the models are tuned using scikit-learn's GridSearchCV function. This allows us to set out a grid of parameters to iteratively train and set the models on and ultimately choose the best parameters. As tree-based algorithms are being applied, the parameters being tuned are the max no. of features used, the max depth of trees, the minimum sample split and the no. of trees grown (for random forest).
4. Once the 3 models have been fit, they are scored on the testing set. To do this, we predict the 'yield' variable using the trained models on the response data set. The predicted 'yield' values are scored against the actual 'yield' values in the test set using the mean squared error (MSE) as set out in the 'Metrics' section. Variance is also measured alongside MSE to have a better view of consistency of our models.
5. After each model is trained and tested, the results are compared to check that more complicated models do indeed produce a better accuracy.

Linear regression model

As outlined in the 'Algorithms and Techniques' section, I fit the data to a linear regression model. This produces an MSE of 254474109 and a variance of 0.18. While not the most powerful model, it is simple and quick and will form our baseline model.

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normaliz
e=False)
```

```
Mean squared error: 254474109.10
Variance score: 0.18
```

Decision tree model

The next step is to fit a Decision Tree model. This is more complex, and returns a lower MSE of 253282661. While this is a better result, the improvement is not significant.

```
DecisionTreeRegressor(criterion='mse', max_depth=5, max_features=Non
e,
           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')


Mean squared error: 253282661.10
Variance score: 0.18
```

Random forest model

Finally, I fit a Random Forest model. This is the most complex model, and returns the best results of 21254832 MSE and 0.31 variance. This suggests that the data set has many non-linear relationships which can only be captured by a more complex model.

```
CPU times: user 2.3 s, sys: 23.9 ms, total: 2.32 s
Wall time: 2.4 s


Mean squared error: 212548432.31
Variance score: 0.31
```

Complications

While building this pipeline, one complication that occured was the inherent stochasticity of the LDA topic modelling algorithm. As a Bayesian model, the LDA model uses a posterior probability distribution over the random vectors of topics. This causes different runs of training the LDA algorithm on the same corpus can result in different topics and term frequencies.

To reduce this variability, one method is to increase the number of training runs (which needs to be offset against increased training time) to help the model converge on the same results. By increasing the training passes to 10 from an initial 1, I found that the topics produced were much more consistent. Reducing the number of topics used also helps the model converge on a smaller set of topics.

For reproducibility, np.random.seed can be used to set the same seed each time inference is performed. However, increasing the number of passes is still important otherwise the resulting topic models will be sensitive to the seed set.

Refinement

In the 'Problem Statement' section, it was first highlighted that one of the key aspects of this project is to test the predictive value of textual information in AirBnb listings. As such, one of they key refinements made is to include the earlier topics that have been modelled using LDA (explained in the 'Data Exploration' section). Each listing is classified into one topic based on probability determined by the trained LDA model. This topic is then added in as a few feature.

Before this, the tuned random forest model had the best result with the lowest MSE of 212548432 and the highest variance of 0.31. After including the description topics, MSE reduces further to 211994401 while variance increases further to 0.32. This validates the hypothesis that there is valuable information as modelling description text as topics increases the model's predictive accuracy, although it is a very small effect.

```
Dataset has 9722 rows, 22 columns.
```

```
CPU times: user 2.46 s, sys: 15.1 ms, total: 2.48 s
Wall time: 2.55 s
```

```
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=150, n_jobs=1,
           oob_score=False, random_state=42, verbose=0, warm_start=F
alse)
```

```
Mean squared error: 211994401.07
Variance score: 0.32
```

Tuning the final model

After confirming that including NLP features helps to improve accuracy, it is included in the final model. The parameters for the final Random Forest model is then tuned using scikit-learn's GridSearchCV.

A parameters grid of hyperparameters are created for GridSearchCV to iteratively use and test for the best cross-validated accuracy. The paramters being tuned are: Number of estimators, the maximum depth and the minimum samples split. After tuning, the best parameters are a max_depth of 15, minimum samples split of 10 and number of estimators 200.

The final MSE lowers to 210835523 with a variance score of 0.32. This shows that the model is fairly complex, with a large number of trees and depth.

```
CPU times: user 28min 32s, sys: 9.37 s, total: 28min 41s
Wall time: 29min 19s
```

```
Mean squared error: 210835523.74
Variance score: 0.32
Tuned Model Parameters: {'bootstrap': True, 'criterion': 'mse', 'max
_depth': 15, 'max_features': 'auto', 'min_samples_split': 10, 'n_est
imators': 200}
```

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

To test the robustness of the final model, sensitivity analysis is done by making adjustments to the inputs and evaluating the size of changes in the outputs. The adjustments tested are:

1. Changing the random state: This will cause different data points to be randomly selected during the train/testing split and during the random forest training process. By choosing different random states, we can ensure that model training is not a fluke and can be replicated on different states. Changing the state from 42 to 49, the MSE remains in the same ballpark at 211329730 and variance score is the same.
2. Changing the split proportions between training and test set: By changing the amount of data split between training and testing, we can ensure that the model retains its accuracy on different size of testing data. A fairly conservative split of 30% for the test set had been chosen initially. By reducing this to 10%, the MSE drops to 205098824 which is to be expected. However, the change is not radicially different and the model is fairly robust.
3. Changing input data: By manually tweaking some of the input data, it is possible to see how senstive the model is. If a small change in inputs lead to an unrealistic change in outputs then the model is unlikely to be robust. As an example, I take a random data point in the test data set and change the 'accommodates' feature which is one of the most significant features. The data point originally accommodates 4 persons and the model predicts a yield of £17,400 annually. Adjusting it to accommodate for 2 reduces the predicted yield to £13,377 while increasing it to accommodate for 6 increases it to £18,106. This change is in line with expectations, as a property that can accommodate less guests should have a smaller yield and vice-versa. Importantly, increasing the number of guests leads to a smaller increase in yield than a decrease in yield for decreasing the number of guests. This makes sense as increasing the number of guests without changing any other variable indicates that each guests needs to share of the the listing facilities and hence have less yield per guest.

```
Mean squared error: 211994401.07
Variance score: 0.32
```

```
Mean squared error: 204781232.75
Variance score: 0.32
```

```
accommodates                       4.000000
bathrooms                          1.000000
bedrooms                           2.000000
beds                               3.000000
guests_included                    2.000000
extra_people                       5.000000
minimum_nights                     2.000000
latitude                          51.467320
longitude                         -0.167568
property_type_Apartment            1.000000
property_type_Bed & Breakfast      0.000000
property_type_House                0.000000
property_type_Other                0.000000
room_type_Entire home/apt          1.000000
room_type_Private room             0.000000
room_type_Shared room              0.000000
bed_type_Non-Real Bed              0.000000
bed_type_Real Bed                  1.000000
topics_description_Budget          1.000000
topics_description_Location        0.000000
topics_description_Luxury          0.000000
Name: 1400514, dtype: float64


[ 16041.37718383]


//anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:4: Futu
reWarning: reshape is deprecated and will raise in a subsequent rele
ase. Please use .values.reshape(...) instead


[ 12732.87267]
[ 17594.75291864]


//anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:6: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
//anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:7: Futu
reWarning: reshape is deprecated and will raise in a subsequent rele
ase. Please use .values.reshape(...) instead
//anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:9: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/indexing.html#indexing-view-versus-copy
//anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:10: Fut
ureWarning: reshape is deprecated and will raise in a subsequent rel
ease. Please use .values.reshape(...) instead
```

Another method of evaluating the model is to look at what the model is saying and sense checking the results. By looking at feature importance, we are able to determine what the model determines to be important features and whether that aligns with what might be expected. Features that are surprising could be further explored to check if they are genuine anomalies or if the model is inconsistent with reality.

The top features are latitude (25%), accomodates (22%) and longtitude (19%) accounting for 66% of model importance. Longitude and latitude makes intuitive sense, and proves the age-old real estate adage that it's all about 'Location, location, location!'. 'Accomodates' is the number of people a listing can accommodate which also makes sense as that is a direct influence on price. Perhaps a potential hint to future hosts that an extra sofa-bed would be well worth the investment?

Interestingly, the topics modelled from the listing descriptions have a combined importance of 2.5%. While this seems like a small number, it is higher than property type (i.e. house, apartment or other) which has 1.3%. This further validates the hypothesis that there is valuable information in the descriptions and tell you more about a listing than whether it is a house or apartment.

The features importances produced by the model seem to make sense, and further builds confidence in the final model built.

	importance %	feature
7	24.932740	latitude
0	21.772927	accommodates
8	19.621701	longitude
6	7.702141	minimum_nights
5	6.356207	extra_people
1	4.105430	bathrooms
4	3.769556	guests_included
2	2.893417	bedrooms
13	2.039581	room_type_Entire home/apt
3	1.844825	beds
20	0.956488	topics_description_Luxury
19	0.914949	topics_description_Location
18	0.900623	topics_description_Budget
12	0.477287	property_type_Other
11	0.442333	property_type_House
9	0.430975	property_type_Apartment
14	0.396504	room_type_Private room
10	0.152781	property_type_Bed & Breakfast
17	0.107925	bed_type_Real Bed
15	0.098952	room_type_Shared room

Justification

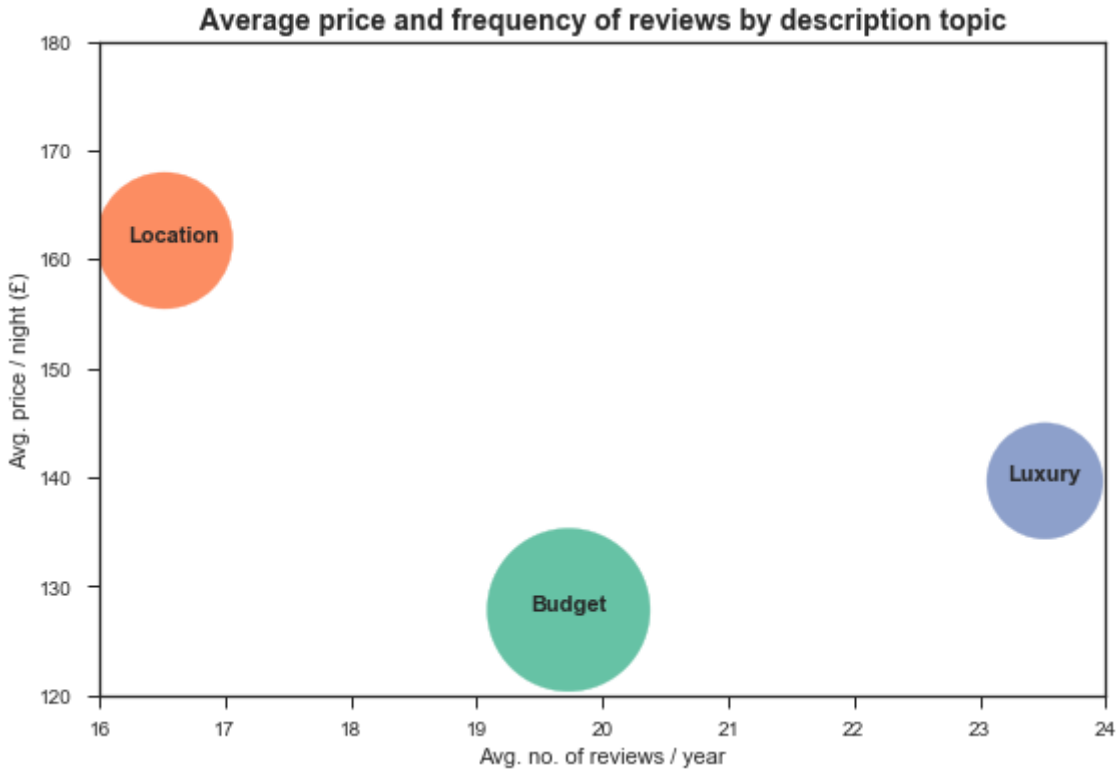
The final MSE is 200185564 with a variance score of 0.33, which is a significantly better result compared to the baseline model with an MSE of 254474109 and variance of 0.18. The final model has also been rigorously tested, cross-validated and evaluated with different inputs and been proven to be a robust solution.

This justifies the use of the final model, as it is robust and a more accurate predictor of AirBnb yield compared to a simple linear regression model. It has a lower error rate, and is able to more consistently predict a yield that is closer to the actual yield.

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization



NLP is an important feature of this project, and there are valuable insights produced from topic modelling in addition to improving predictive strength. By plotting a bubble chart of the 3 topics described in the earlier section of this report (Location, Amenities and Luxury), it is possible to segment London's Airbnb market into the different topics, each of which demonstrating a different average price and frequency which are the component factors of yield.

In the below chart, each bubble is plotted at the average of that segment while the size represents the number of listings in that segment. Some of the characteristics of this chart make intuitive sense, while others are slightly more surprising.

In terms of size, the 'Budget' segment has the most number of listings, followed by 'Location' while 'Luxury' has the least number of listings. This makes sense and one would expect there to naturally be more 'Budget' type listings available.

Looking at the average price, 'Location' is the clear leader followed by 'Luxury' and 'Budget' at just £5-10 lower. This suggests that location is still the king in property values, and that is also corroborated with the feature importances.

What is surprising is that although location commands the highest price, it is 'Luxury' stays that get the highest number of reviews. This is a proxy for the frequency of bookings and suggest that having a 'Luxury' type listing might even to lead to above-average yields. This suggests that the appeal of AirBnb is in the unique opportunity of staying in boutique places, and those are the most popular types of listings. Other segments of the market, such as business travellers who typically are more concerned with Location might prefer to stay in traditional hotels, while other segments such as budget travellers would prefer to stay in hostels for assured amenities and lower prices.

Reflection

This project has been extremely rewarding to complete, giving me a challenging and interesting problem to explore and design my own solution for. Each stage of the project presented its own unique learning points a different part of the analytical toolkit to practice.

In the design phase, the focus was on researching and understanding the problem of predicting yield in AirBnb. What data is publicly available? What research has already been done? What are the implicit assumptions in the problem statement?

This led naturally onto the exploratory data analysis phase once I had gotten my hands on the right data set. To design the right solution, I had to thoroughly understand the data and quirks in it. This included understanding what the data points meant, how it was collected, and whether there might have been any unintended biases in the data. This also laid the foundation of understanding potential insights and hypotheses to test and what pre-processing might be necessary.

Iteratively, this also led to the data cleaning and manipulation stage. After understanding the data, I was able to perform the manipulations necessary such as imputing or removing missing values, feature selection, and calculating yield values.

Once the data has been cleaned and processed, I was able to fit them on the selected machine learning algorithms. The course syllabus was most useful here, and I began to appreciate the complexity of the different algorithms and how to interpret the results of each algorithm. By comparing the results I was able to choose a final cross-validated model and fine-tune it according to best practices.

After building the final model, it had to be thoroughly tested and evaluated. This included doing sensitivity analysis the test the robustness of the final solution, and sense checking the predictions and insights produced by the model.

One interesting aspect, and also the most challenging for me, was undertaking natural language processing for the feature engineering stage. I tested various techniques such as bag-of-words, term-frequency-inverse-document-frequency, latent Dirichlet allocation and also various regexes in the preprocessing stage. Ultimately, LDA proved to be the most powerful method of reducing the no. of dimensions and was able to give more interpretable topics compared to other topic modelling methods.

The unique challenges in topic modelling stemmed from the more qualitive nature of text mining as opposed to working with numerical data. It is a lot more free-form and there is seldom a "right answer" to fall back to. For example, choosing the number of topics to model greatly affected the results. After testing several topic numbers, I decided to choose a relatively small number of 3. This kept the data manageable, the topics interpretable and also distinct enough to form insightful segments. Labelling the topics was another free-form challenge, though it was thankfully made easier by the ability to see word distributions in LDA. Though there is certainly no correct topic label, the word distributions and earlier EDA phase helped to suggest potential topic labels. I tried a few of these labels and tested them against the data set to see if they would make sense (e.g. the 'Budget' label should have a proportionaly lower price!). In the end, I was comfortable with the labels I settled on and there was a plausible story behind each topic label.

Improvement

Looking back on the project, there have been numerous learnings and further improvements that could have been made.

As topic modelling was a big part of the project, a potential improvement would be to try and implement other topic modelling algorithms. While I focussed on LDA due to its interpretability, there were other algorithms such as Non-Negative Matrix Factorization that might have also led to good results. I could also have taken a more scientific approach to choosing the number of topics, and perhaps build a pipeline to iterate through a range of numbers to choose to optimal number with the highest predictive power.

Ultimately, I am happy with the end results and am convinced that the final solution is better than the benchmark initially established. Nevertheless, there will always be further improvements that can be made.

What made this project unique was the attempt to use alternative data sources (in this case, text data) to improve predictive models of AirBnb yield. In a similar vein, AirBnb listings images could also be used as potential features. Attempting this would be quite complicated, requring an understanding of image recognition techniques but also the use of big data processing due to the amount of images one would need to analyse. Nevertheless, that would undoubtedly be an interesting next step to pursue.