Machine learning approach to Superhost Identification and Cost estimation of Airbnb listings

Vidhi Mistry (300101658) and Varun Hanabe Muralidhara (300055628)

University of Ottawa, Ontario, Canada

Abstract. Classification of novice and experienced hosts as Superhost and the accurate pricing for individual listing has been a challenge for Airbnb. In this project, these two problems are addressed by applying standard machine learning algorithm to find a pattern in the problem space and generalise the outcome.

1 Introduction

Airbnb, Inc. is an online marketplace for arranging or offering lodging, primarily homestays, or tourism experiences. InsideAirbnb[1] has collected vast amount of information about Airbnb listing, where the listing contains detailed information about hosts not limited to accommodation type, amenities offered by the host, prices, and reviews for that listing. Montreal being one of the most visited places in Canada, analysing this data could unveil patterns that can be beneficial for Airbnb to enrich host as well as user's experience. Machine learning will help to understand the patterns and commonalities in the data which are hidden otherwise. It is crucial to invest efforts and time into such approaches, which helps enhancing the experience of guests and improving the loyalty of the host, which helps Airbnb to increase revenue and stay competitive in the market.

Exploratory analysis could offer deeper insights into the short term market rental in Montreal, and can be eventually be used for prediction and classification. Getting the price right is important for Airbnb, as a listing with price too high, will lead to less bookings and a price too low will cause Airbnb to lose margin. Regression technique will help to predict the prices with a certain accuracy.

The other problem that can be addressed here is, how can a guest more confidently select a premium property? The Superhost program[2] can be taken into account to solve the problem but identifying host as a Superhost is a challenging task. In this project we have used classic machine learning algorithms to predict the price of a listing and to identify if a host can be a Superhost or not. The mentioned two problems are addressed with regression and classification machine learning approaches respectively. The report is organized in the following structure : Section 2 discusses about the experimental setup including tools used, data preprocessing steps and the algorithms used within this project. Training

and testing pardigms and evaluation metrics for classificaton and regression used within this project are discussed in Section 3 and Section 4 respectively. Experimental results for both types of machine learning tasks are summarized in the Section 5, followed by Conclusion and future work in the last section.

2 Experimental Setup

The project follows the common machine learning lifecycle. Firstly, extensive feature engineering followed by feature transformation and feature scaling were performed. Thereafter, with the processed data, models were built using hyperparameter optimization. The performance measures for each models were calculated and later Statistical Testing was performed to determine the statistical significance of the models.

2.1 Implementation Details

Sci-kit learn module in python was used to create all the machine learning models. Matplotlib library was used to generate the graphs. Pandas is an open source, BSD licensed library. It provides high performance and simple tools for data analysis which was also used in the project.[14][15]

2.2 Dataset

The raw dataset of Airbnb listing for Montreal was collected from insideAirbnb website. The dataset contained the host's information as 106 features such as `host-since`, `neighbourhood`, `accommodation type`, `amenities offered`, `superhost status`, `performance_rating`, `price` etc. The total number of examples in the dataset were 21253.

id	listing_url	scrape_id	last_scraped	maximum_minimum_nights	summary	space	description
thumbnail_url	medium_url	picture_url	xl_picture_url	host_id	host_url	host_name	host_since
host_picture_url	host_neighbourhoo	host_listings_count	host_total_listings_count	host_verifications	host_has_profile_pic	host_identity_verified	street
smart_location	country_code	country	latitude	longitude	is_location_exact	property_type	room_type
price	weekly_price	monthly_price	security_deposit	cleaning_fee	guests_included	extra_people	minimum_nights
calendar_updated	has_availability	availability_30	availability_60	availability_90	availability_365	calendar_last_scraped	number_of_reviews
review_scores_communicat	review_scores_loc	review_scores_value	requires_license	license	jurisdiction_names	instant_bookable	is_business_travel_ready
reviews_per_month	house_rules	host_thumbnail_url	square_feet	maximum_nights_avg_ntm		review_scores_checkin	calculated_host_listings_count_shared
transit	access	interaction	host_response_rate	host_acceptance_rate	host_is_superhost	city	state
minimum_maximum_nights	maximum_maximu	minimum_nights_avg_ntm	review_scores_rating	review_scores_accuracy	review_scores_cleanliness	calculated_host_listings_count	calculated_host_listings_count_entire_
neighborhood_overview	notes	host_about	host_response_time	neighbourhood_cleansed	neighbourhood_group_cleansed	bathrooms	bedrooms
first_review	last_review	require_guest_profile_picture	require_guest_phone_verification	bed_type	amenities	host_location	neighbourhood
name	summary	space	description	experiences_offered	beds	zipcode	
experiences_offered	accommodates	maximum_nights	number_of_reviews_ltm	cancellation_policy	calculated_host_listings_count	private_rooms	

Fig. 1. List of Airbnb Host features

There were many problems within this raw data such as noisy data, missing values, etc. Such data cannot be used directly for the training or testing of the

models. Extensive data cleaning and data-processing had to be done in order to use this data for model construction. The problems with the dataset and how they were addressed is mentioned as follows:

Missing and null values As the dataset was a scrapped data it had many null values. For the features which contained only null values were eliminated from the dataset such as `thumbnail_url`, `medium_url`, `xl_picture_url`, `license`, `host_acceptance_rate`, and `neighbourhood_group_cleansed`.

Unimportant features Unimportant fields that contained same values for all the samples. The features such as `requires_license`, `has_availability`, `is_business_travel_ready` were all eliminated.

Inconsistent data types Data types with generic types such as object had to be re-constructed to a discrete data types. This posed a major challenge as the contents of the objects were not consistent in few of the features. The data types for `price` and `cleaning fee` and `security_deposit` were not appropriate, they were supposed to be numbers but they were object. This problem was solved by manually curated filter scripts for such features which converted object type to a particular data type. In this case it was float.

Noisy Data and Outlier Although the city that was chosen was Montreal, the listings from other countries were also present. The filters were carefully constructed to include listings related to Montreal city thus restricting the data from other countries and other provinces. Moreover, information of some of the listing were in different language altogther. As there were just few such sample data, such data were eliminated.

Feature Transformation Features such as `room_type` and `bed_type` were categorical. The `room_type` had values such as Entire Home/Apt, and Shared Room, and Private Room which were converted into ordinal feature. Same with the `bed_type` feature which was converted to ordinal type as well.

Feature Extraction `Neighborhood` feature can offer valuable information about the locality and the price of the listing. However, there were more than 30 neighborhood in the neighbour feature, one hot encoding of all those neighborhood would be an inefficient way of feature extraction as it causes dimensionality problems. Hence, KMeans clustering algorithm was applied on `latitude` and `longitude` features to create a clusters of neighbourhood according to geographic locations. According to the inertia score of the clusters, 7 neighbourhood were created. One hot encoding were applied on those newly created neighbourhoods.

`Amenities` of each listing offer insights for the price as well as identification of superhosts. However, in the raw data, all the amenities that were offered were encapsulated in a list e.g `[TV,"Cable TV",Internet,Wifi,"Air`

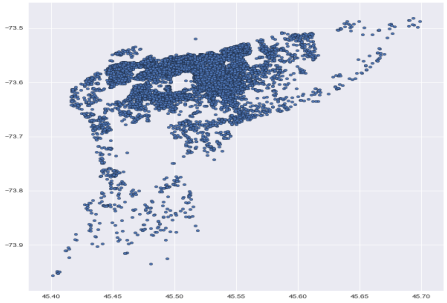

Fig. 2. Listings geographic location wise

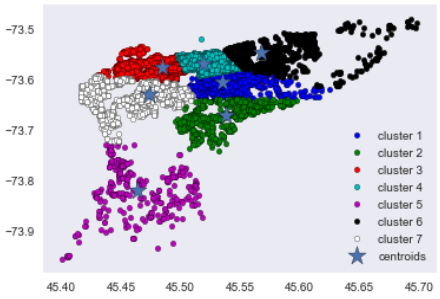

Fig. 3. Clustered Neighbourhood

conditioning", "Pets live on property"] and there were more than 19,000 such unique lists in the data. In order to use this information, bins were created. In all, 5 bins were created : `Basic`, `Facilities`, `Location`, `Parking`, `SafetyFeatures`. `Basic` feature value was be set to 1 if the basic amenities were covered, or 0 if none of them were offered. It included amenities essential amenities like TV, wifi, etc. `Facilities` featured was set to 1 for that listing if it offered any kind of extra facility like Pool, Gym, Air-Conditioning, etc. While the `Parking` ensured whether any kind of parking was offered. The `Location` was set for properties that offered advantage of location like beachfront, waterfront, etc. `SafetyFeatures` ensured whether common safety features were provided by the hosts in the listings.

Another column that needed similar feature extraction approach as `Amenities` feature was `property_type` . There were more than 20 unique values for the feature `property_type` . Binning was carried out to extract information from this feature. 3 bins were curated : `Basic Stay`, `Luxury Stay` and `Unique Stay`. `Basic Stay` included stays such as Guesthouse, Apartment, Loft, Serviced apartment, Guest suite etc. Whereas `Luxury Stay` included stays like Condominium, Boutique hotel, House, Villa, Bungalow, etc. The `Unique Stay` covered all the unique experiences listings such as Cottage, Cabin, Camper/RV, Earth house, Boat, Tiny house, Cave, Tent, Campsite, Farm stay etc.

Data Preprocessing for Superhost Identification Classification The target variable `host_is_superhost` has class imbalance. Only 4010 of the all the data are superhosts whereas 17083 data samples belong to non-superhost.

While addressing the problem of superhost identification, two types datasets were used. Firstly, the imbalanced data was used as is to build models. Since imbalanced data adds bias to the model, that is, model generalises well for the negative class more than positive, the data set was balanced using SMOTE and that data was used for building models.

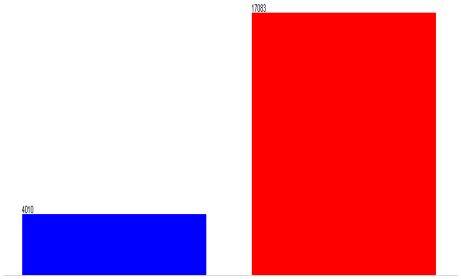

Fig. 4. Class Imbalance for host_is_superhost

Data Preprocessing for Price Prediction Regression Upon inspection it was found that the `price` feature had many outliers, for example there were many features that had values more than 1000$ and even some listings had price higher than 8000$. The histogram (Figure 5) and boxplot (Figure 6) for `price` feature are highly right skewed. Moreover, most of the listing fall in the price range from 0$ to 200$ with median being 79$ and mean being 116$.

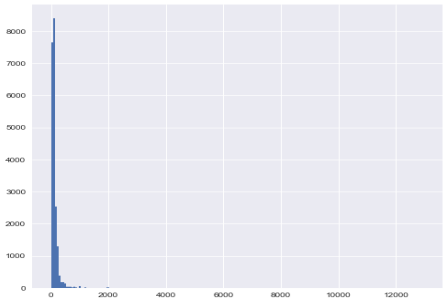

Fig. 5. Price Histogram

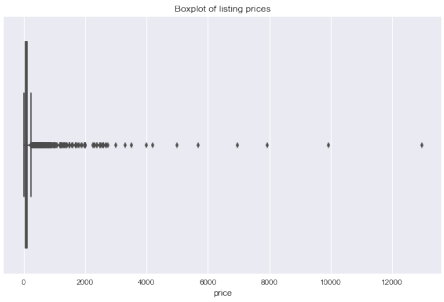

Fig. 6. Price Boxplot

Training on such data would introduce biases and model might not be able to generalise for such a wide range of values. In order for fair comparison of all models, two different datasets were created by introducing threshold prices. Only those lisitngs that had prices equal to or lower than that threshold value were included in that dataset. One of the thresholds for prices that was chosen is 700 because 99% of the data had prices less than 700. Other price threshold was 80 as approximately 50% (Median = 79$) of the data was included in this dataset. These two datasets were used for further training purposes for all the regression models.

2.3 Feature Selection

After extensive data cleaning and data preprocessing, there were 53 features left. High dimensionality can be computationally expensive, also can lead to overfitting and sparse data. The data should have only those features that contribute to the prediction. In order to gauge features to keep and features to eliminate, different feature selection approach were adopted such as Correlation Matrix, Feature Importance using Random Forest, Recursive Feature Elimination and Principal Component analysis(PCA).

Correlation Matrix Correlations among variables may impact the model negatively leading to misleading interpretations.[3] Decsision Trees and Boosted tree algorithms are immune to such multi-collinearity, but not every model. Pearson's Correlation method is one of the statistical methods to calculate the strength of correlation between two variables or attributes. Figure 7 shows the Pearson's Correlation values between two variables.

There was correlation between variables such as `bedrooms`, `beds`, `accommodates` with `guests_included` among `review_scores`, and among `availibilty` features.

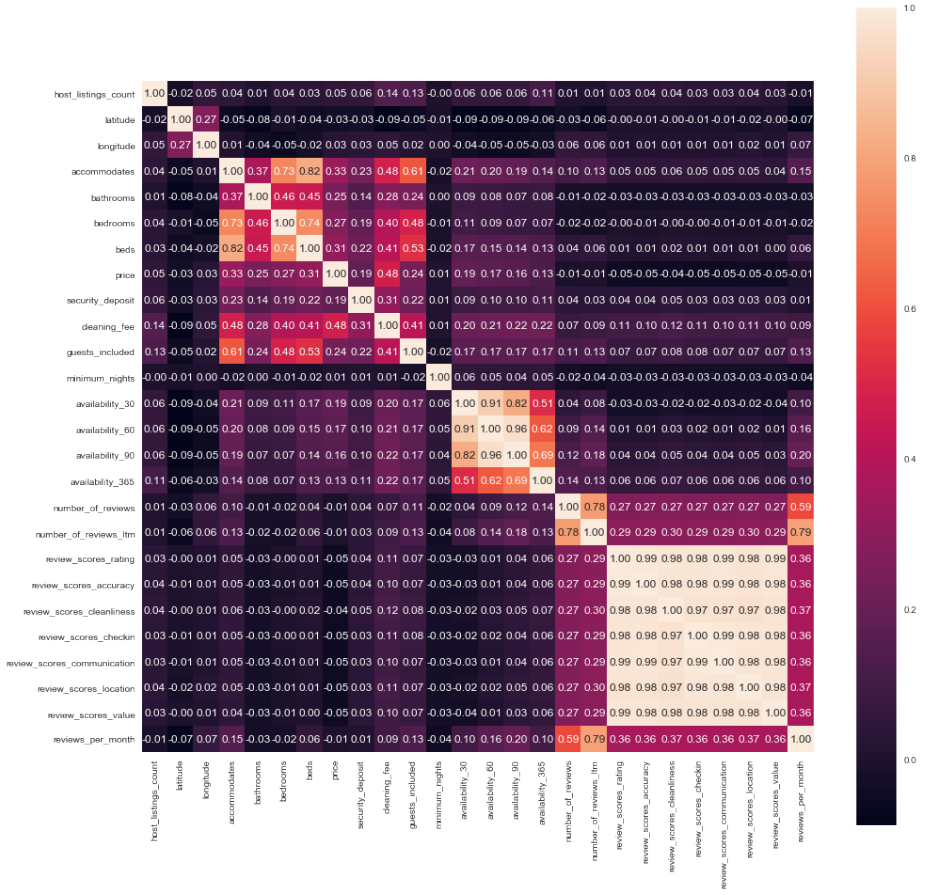

Fig. 7. Correlation Heatmap

Feature Importance The decision to drop or select the feature can be based upon the feature importance of the variable towards the target variable. A tree based classifier or regressor is used to find the feature importance of each variable. In this project, to find out the important features, Random Forest method was used. Random Forest Classifier is used for classification purposes and Random Forest Regressor is used for Regression problem.[4] For price pre-

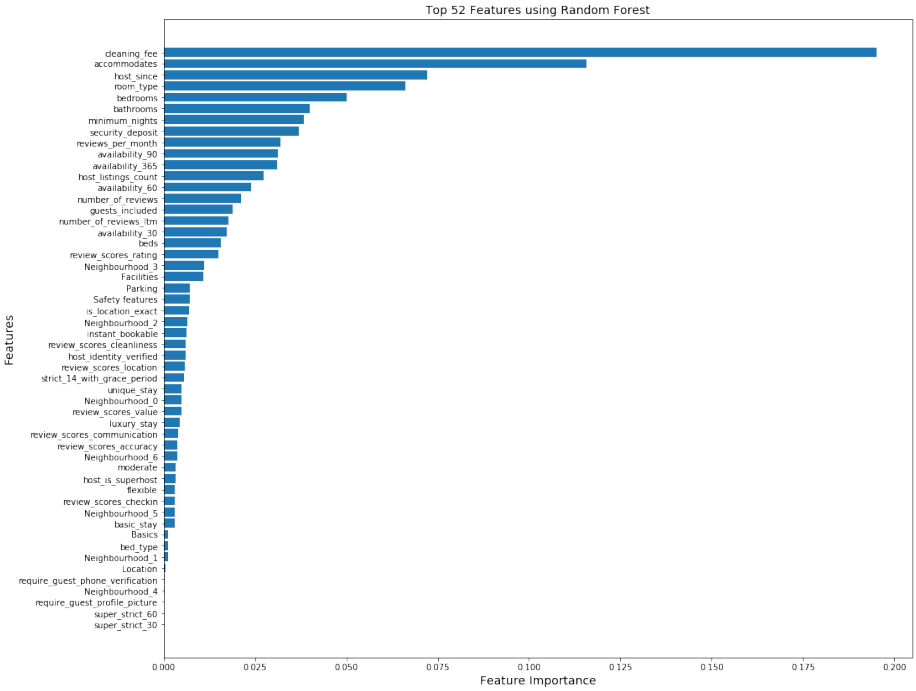

Fig. 8. Feature Importance for Price prediction - Regression

diction (Figure 8), the top features contributing to the price were `cleaning_fee`, `accommodates`, `room_type`, `minimum_nights`, etc. For classification problem, the top features include `number_of _reviews`, `rating`, price etc.

Similarly, for Superhost identification (Figure 11), the the top features contributing to the price were `number_of_reviews_ltm`, `number_of_reviews`, `host_since`, `review_score_rating`, `host_listings_count` etc.

Recursive Feature Elimination For both cases, there were few features that had low or near to zero importance , for example, `super_strict_60` and `super_strict_30` (Figure 8and 11). These features can safely be assumed to be unimportant towards the problem under consideration, they can be eliminated from the data.

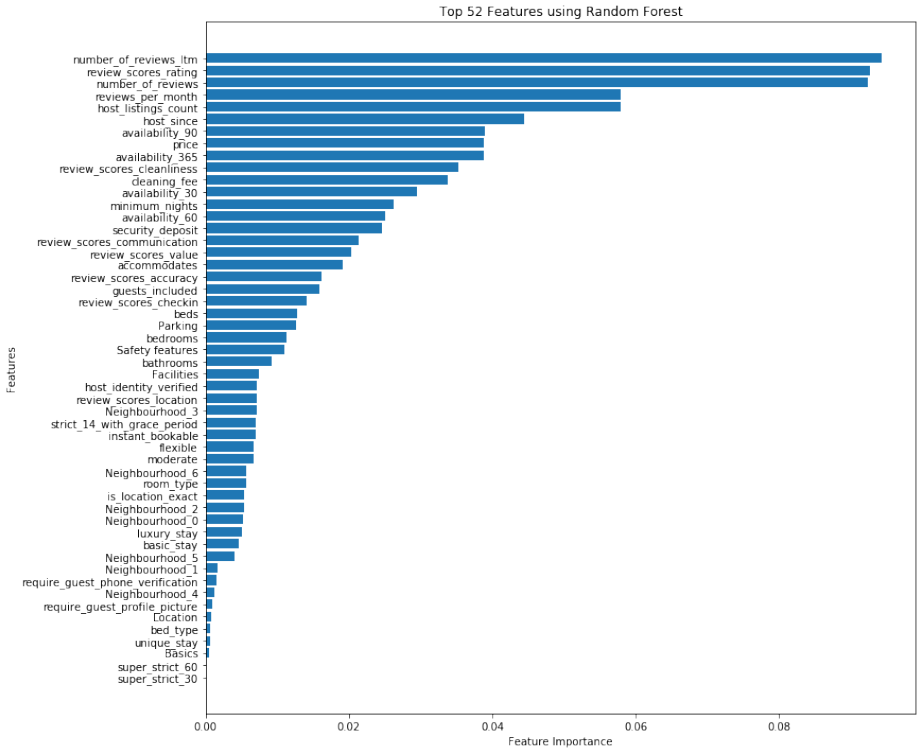

Fig. 9. Feature Importance for Superhost Identification - Classification

The Recursive Feature Elimination (RFE method) works by removing attributes recursively while building a model on the attributes that remain. It uses evaluation metrics like accuracy to rank the features based on their importance. RFE algorithm takes the model as the input and outputs ranking of the features - 1 being the most important and the support for each feature - True being relevant and False being irrelevant. It outputs the optimal number of features that are relevant and the ones that should be kept. [6]

For Regression analysis, RFE algorithm with Random Forest Regressor as the model to calculate Feature Importance gave list of 40 features that should be kept and 12 features that should be eliminated.

Similarly, for classification analysis, RFE algorithm suggested to keep 45 features and remaining 7 that could be eliminated.

Principal Component Analysis Principal component analysis is a statistical method of converting high dimensional data to lower dimensional data by selecting a set of most important features which can capture maximum information about the target data. On the basis of how much each feature causes variance in the target variable, features are selected. Feature that has the highest variance becomes the first principal component. The feature that causes the second highest variance is the second principal component, and so on. The PCA algorithm's components are linear combinations of the features. Moreover, principal components are orthogonal to each other that is they have no correlation among them. PCA algorithm doesn't knock out the features altogether, but some principal components that fail to cause required variation in the output are eliminated. Thus, unimportant features still partly remain in the preserved principal components and can serve to offer insights and build patters. Thus, dimensionality is reduced while keeping all the features either completely or partly.[5]

Normalizaton Principal components are biased towards features with high variance,so if the data isn't scaled equally, the principal components tend to be biased towards the features that have maximum range. Thus, prior to using PCA, the feature set must be normalised. Min - Max Normalization was used in this project as the normalization technique with all the features transformed into 0 to 1 scale.

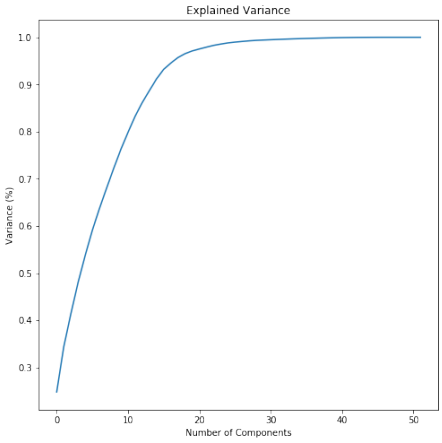

Fig. 10. PCA for Superhost Identification - Classification

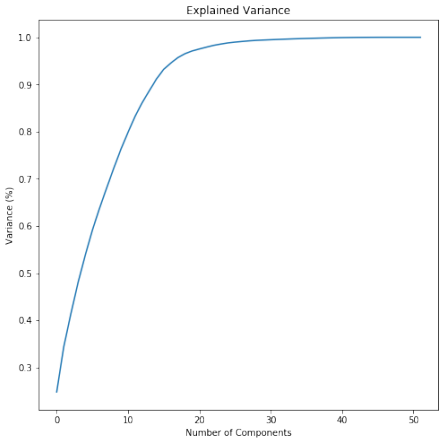

Fig. 11. PCA for Price Prediction - Regression

PCA algorithm was applied to both regression and classification task's data. In order to find the optimum number of components, plot of explained variance vs number of components was plotted for both tasks. From the Figures, 35 components were chosen as they reduced the dimensional space by 17 features and yet was able to explain more than 95% variability in the output variable.

In this project, PCA approach was chosen for dimensionality reduction over RFE for two reasons : it gave more reduced dimensional space and it gave better models than the RFE technique.

2.4 Algorithms

All algorithms that were used to construct the models for both classification and regression tasks falls under the following categories:

- Linear Models
- Distance based
- Tree based
- Ensembles
- Rule based

Linear Models The linear model work in a manner trying to define a linear decision boundary. The linear models that were used within this project:

- Logistic Regression (for Classificaton)
- Naive Bayes (for Classificaton)
- Linear Regression (for Regression)
- Support Vector Machines

Distance based Distance-based algorithms are nonparametric methods. Distance metric uses distance function which provides a relationship metric between each elements in the dataset. The distance based algorithm that were used in this project:

- KNN

Tree based Tree based algorithm are easy to interpret models. In this project, decision trees were used which is a classic tree based algorithm.

Ensembles Ensembles and boosting algorithms transform a weak learner into strong learner.[8] In this project the following ensemble algorithms were used:

- Random Forest
- XGBoost
- LightGBM (LGBM)

Rule based Rule based algorithm uses association rule learning to learn intresting pattern in the data. In this project, Rulefit algorithm was used.[7]

Hyperparameter Tuning Most of the algorithms have one or more hyperparameters to tune to get the optimum results. This process of fine tuning the parameters is laborious if done manually by trial and error. Here, two hyperparameter optimizing techniques were used : GridSearchCV and RandomizedSearchCV. For each algorithm a range of hyper-parameter values were selected. The list of hyper-parameter values were provided to the GridSearchCV or RandomizedSearchCV along with the model to test upon. Both of these techniques outputs the best parameters that resulted in the best score according to the scoring metric provided. GridSearchCV checks and compares exhaustively all the possible hyperparameter pairs, whereas RandomizedsearchCV randomly picks a combination of hyperparameter values and tries to converge to local minima within the mentioned fixed number of iterations. For computationally expensive algorithms, the latter one was used.

3 Training and Testing Paradigm

There are different training and testing paradigms used within machine learning such as train-test split, train-validation-test split, and cross-validation. Within this project, cross-validations are used to train the model and test them. K-fold cross validations are used for Regression task. For classification, stratified K fold cross validation were performed for two datasets: balanced and unbalanced. For handling the imbalance classes, the dataset was split into train, test and SMOTE was applied on each cross validation train set to make each fold represent the same number of positive and negative example to create a balanced dataset.[11]

4 Evaluation of algorithms

4.1 Classification problem

Choosing the best classification model is a challenging task.[9] Evaluation measures that are widely used :

1. **Area Under the Curve (AUC)** (Area Under The Curve) curve and ROC(Receiver Operating Characteristics) curve are the most important evaluation metrics for checking any classification model's performance. ROC is a probability curve and AUC represents degree or measure of separability. Higher the AUC, indicates the model is better at prediction.

2. **F1 score** F1 score is a vital performance measurement. F1 Score is the weighted average of Precision and Recall. F1 can be used in all the models to evaluate performance when there is uneven class distribution.

4.2 Regression Problem

Choosing the best regression model is challenging task.[10] Two regression evaluation measure that are widely used :

1. **Root Mean Square Error (RMSE)**
   It is the standard deviation of the prediction errors. The predictions errors are the measure of how far the predictions are from its actual values. Low RMSE is desired.

2. **R-Squared (R2) Score**
   It is ratio of the explained variance to the total variance in the target variable.It indicates how well the model can explain the variability of the target value around its mean. High R2 Score is desired.

The aforementioned evaluation metrics are used within this project to evaluate the performance of different price prediction and Superhost identification models. Since, all the models were trained using K-fold cross validation, metrics for each of k folds were obtained. Mean results of all the folds were used for comparing these algorithms. Individual k-fold results were used to determine the statistical significance between the algorithms using paired t-tests.

5 Results

5.1 Classification problem

For each of the models constructed, mean AUC and mean F1 score were calculated to compare the model performances. As mentioned before, the two datasets were created for classification problems. Summary tables mentioned each of the datasets.

Classical machine learning algorithms like Logistic Regression, Naive Bayes, K nearest neighbours , Random forest, Decision trees, Support vector Machine, XGBoost, LGBM , Rule based are able to classify a binary class i.e. whether a host is Superhost or not. These models were built on two datasets and the results were as follows:

Imbalanced dataset From the Figures 14 to 21, for highly imbalance dataset LGBM algorithm had a ROC of 0.87 and F1 score of 0.56. The other algorithm that performed better were Support vector machine and Random forest with the AUC of 0.85 and 0.86, the F1 score of 0.53 and 0.52 respectively. Clearly, LGBM performed the best among all the other algorithms.

	Logistic Regression	Gaussian Naive Bayes	K Nearest Neighbours	Random Forest	Decision Tree	Support Vector Machines	XGBoost	lgbm	Rule Based
Mean AUC	0.81	0.74	0.77	0.86	0.76	0.85	0.82	**0.87**	0.63
Mean F1	0.35	0.5	0.37	0.52	0.41	0.53	0.42	**0.56**	0.33

Fig. 12. Unbalanced Dataset

Balanced From the Figures 22 to 29 , for balanced dataset LGBM alogrithm had a ROC of 0.87 and F1 score of 0.62. The other algorithm that performed better were Support vector machine and Random forest with the same AUC of 0.86, the F1 score of 0.55 and 0.64 respectively. Clearly, if we were to consider F1 score then Random Forest performed slightly better than lgbm and LGBM performed better with respect to AUC. Here it can be seen that F1 score for all the models had increased expect the decision the decision trees.

	Logistic Regression	Gaussian Naive Bayes	K Nearest Neighbours	Random Forest	Decision Tree	Support Vector Machines	XGBoost	lgbm	Rule Based
Mean AUC	0.81	0.75	0.78	0.86	0.71	0.86	0.83	**0.87**	0.67
Mean F1	0.53	0.41	0.5	0.64	0.41	0.61	0.55	**0.62**	0.4

Fig. 13. Balanced Dataset

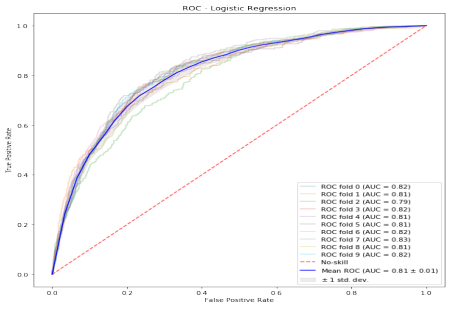

Fig. 14. Imbalanced - Logistic Regression

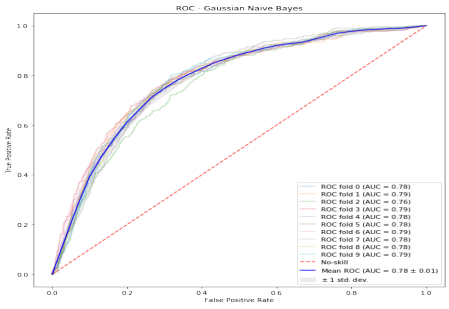

Fig. 15. Imbalanced - Naive Bayes

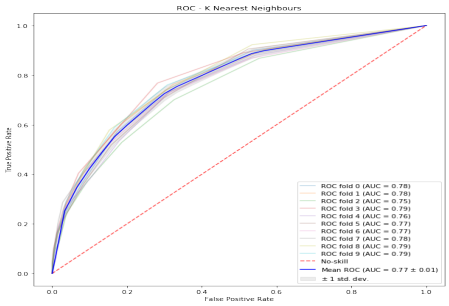

Fig. 16. Imbalanced - KNN

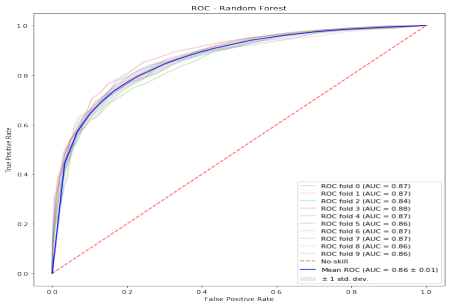

Fig. 17. Imbalanced - Random Forest

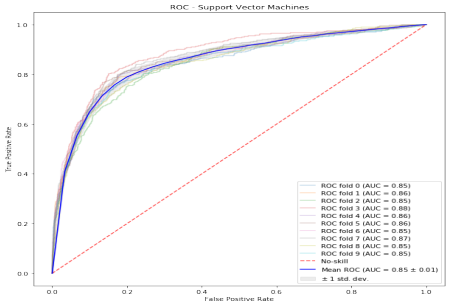

Fig. 18. Imbalanced - SVM

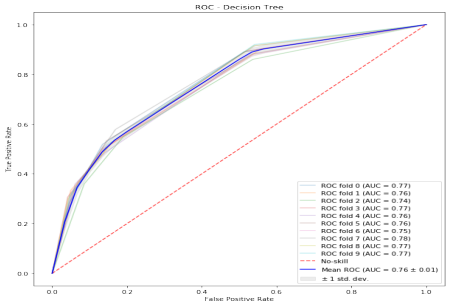

Fig. 19. Imbalanced - Decision Trees

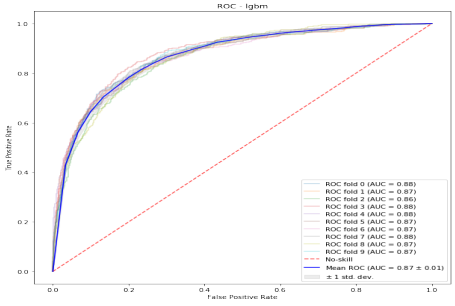

Fig. 20. Imbalanced - LGBM

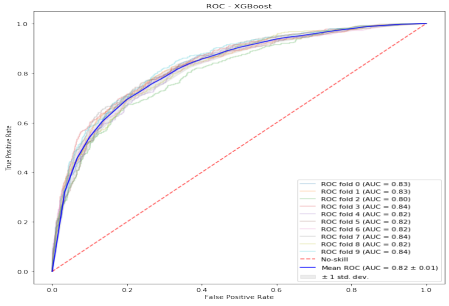

Fig. 21. Imbalanced - XGBoost

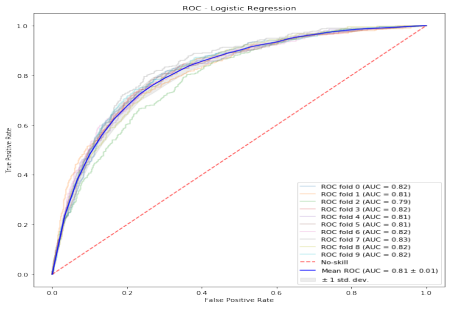

Fig. 22. Balanced - Logistic Regression

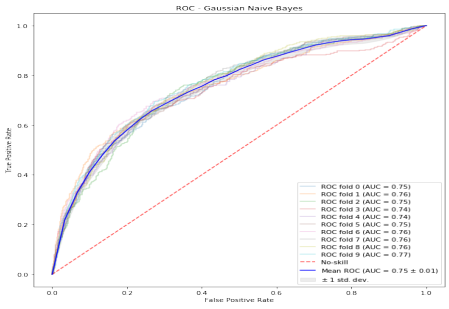

Fig. 23. Balanced - Naive Bayes

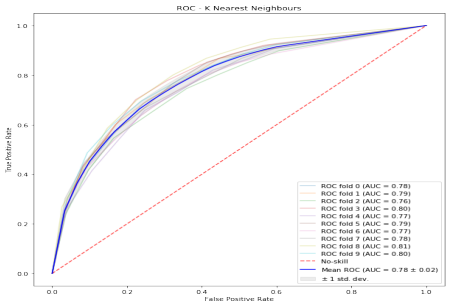

Fig. 24. Balanced - KNN

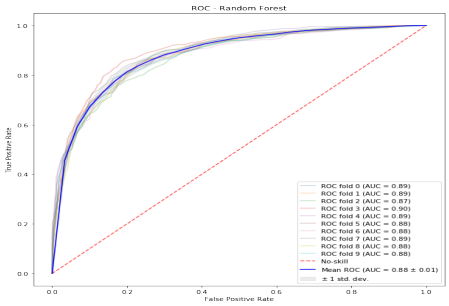

Fig. 25. Balanced - Random Forest

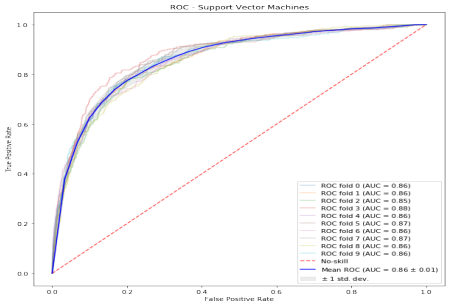

Fig. 26. Balanced - SVM

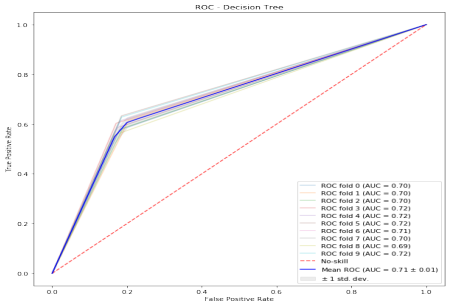

Fig. 27. Balanced - Decision Trees

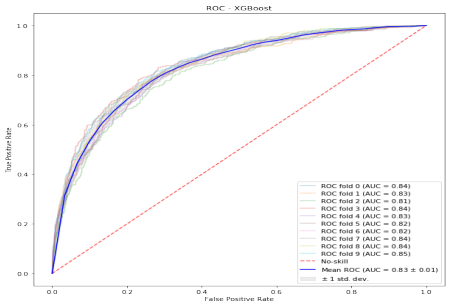

Fig. 28. Balanced - XGBoost

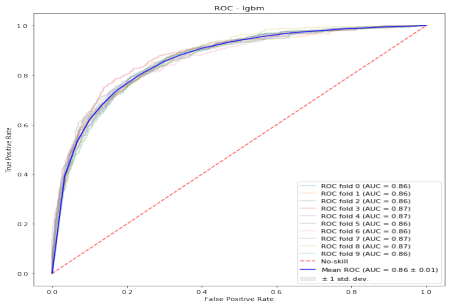

Fig. 29. Balanced - LGBM

5.2 Regression Problem

For each of the models constructed, mean RMSE and R2 scores were calculated to compare model performances.As mentioned earlier, two datasets were chosen for regression problems. Summary table is mentioned for each of the datasets.

Price threshold $\leq$ 700 As summarized in the in the Table 1, classical machine learning algorithms are able to predict the price fairly well. Among them linear regression performed the best whereas K-Nearest Neighbours performed the worst. Among ensemble methods, all of them performed almost similarly. LightGBM outperformed all the other algorithms with RMSE as low as 60.36 and R2 score of 0.5 . Rules based classifier RuleFit performed well but not as good as the ensemble methods.

Algorithm	Linear Regression	Support Vector Regressor	Decision Trees	K-Nearest Neighbours	RuleFit	Random Forest	LightGBM	XGBoost
RMSE	65.33	67.47	68.99	73.03	65.54	62.33	**60.36**	61.42
R2 Score	0.41	0.37	0.34	0.26	0.39	0.5	**0.5**	0.48

Table 1. Evaluation metrics for dataset with price threshold 700$

Price threshold $\leq$ 80 The trends were similar to that seen with the dataset having price threshold set to 700. Ensemble methods outperformed the other algorithms and all of them gave similar RMSE and R2 scores.
Due to LightGBM's way of implementation, besides giving best predictions, it was also fastest among all the other ensemble methods.

Algorithm	Linear Regression	Support Vector Regressor	Decision Trees	K-Nearest Neighbours	RuleFit	Random Forest	LightGBM	XGBoost
RMSE	14.44	14.19	14.99	16.07	14.38	13.96	**13.88**	13.93
R2 Score	0.38	0.4	0.33	0.23	0.37	0.42	**0.42**	0.42

Table 2. Evaluation metrics for dataset with price threshold 80$

6 Statistical testing

Statistical testing helps to compare the machine learning models and choosing the best model.[12] In this project, paired t-tests are conducted to find if there is any statistical significant difference between the algorithms.

t-test is an inferential statistic used to determine if two groups have significant differences between their means. This test is also used as hypothesis testing tool. Within the context of machine learning, hypothesis is that difference between the mean values of two algorithms are same.[13] By the means of t-test, we calculate p-values for a confidence interval. We reject the hypothesis if the p-value obtained is less than the confidence, stating that two algorithms are statistically significantly different. T-paired statistical test is a classic way of testing different folds within a same dataset for two algorithms.

6.1 Statistical testing for classification

The results of statistical testing for unbalanced and balanced are summarized in the table below. In the summarized Tables 3 and 4, S is indicative of the statistical significance between two algorithms.

Logistic Regression VS Gaussian Naive Bayes	Logistic Regression VS K Nearest Neighbours	Logistic Regression VS Random Forest	Logistic Regression VS Decision Tree	Logistic Regression VS Support Vector Machines	Logistic Regression VS XGBoost	Logistic Regression VS lgbm	Logistic Regression VS Rule Based	Gaussian Naive Bayes VS K Nearest Neighbours	Gaussian Naive Bayes VS Random Forest	Gaussian Naive Bayes VS Decision Tree	Gaussian Naive Bayes VS Support Vector Machines
2.40E-06	7.87E-06	2.43E-09	7.76E-09	1.01E-07	0.022793	3.21E-11	5.66E-07	0.321279	2.11E-13	0.000448	4.91E-12
S	S	S	S	S	S	S	S	-	S	S	S
Random Forest VS lgbm	Random Forest VS Rule Based	Decision Tree VS Support Vector Machines	Decision Tree VS XGBoost	Decision Tree VS lgbm	Decision Tree VS Rule Based	Support Vector Machines VS XGBoost	Support Vector Machines VS lgbm	Support Vector Machines VS Rule Based	XGBoost VS lgbm	XGBoost VS Rule Based	lgbm VS Rule Based
0.156278342	6.04E-09	4.12E-13	2.20E-10	7.08E-16	0.000109	8.94E-06	0.002894	1.29E-08	1.40E-09	1.78E-07	3.40E-09
-	S	S	S	S	S	S	S	S	S	S	S
Gaussian Naive Bayes VS XGBoost	Gaussian Naive Bayes VS lgbm	Gaussian Naive Bayes VS Rule Based	K Nearest Neighbours VS Random Forest	K Nearest Neighbours VS Decision Tree	Neighbours VS Support Vector Machines	K Nearest Neighbours VS XGBoost	K Nearest Neighbours VS lgbm	K Nearest Neighbours VS Rule Based	Random Forest VS Decision Tree	Random Forest VS Support Vector Machines	Random Forest VS XGBoost
1.81E-08	1.80E-15	1.19E-05	8.23E-12	0.030622	9.48E-11	1.46E-07	4.63E-13	2.73E-05	3.11E-14	0.089819	1.34E-07
S	S	S	S	S	S	S	S	S	S	-	S

Table 3. Statistical Test for Imbalanced dataset

Logistic Regression VS Gaussian Naive Bayes	Logistic Regression VS K Nearest Neighbours	Logistic Regression VS Random Forest	Logistic Regression VS Decision Tree	Logistic Regression VS Support Vector Machines	Logistic Regression VS XGBoost	Logistic Regression VS lgbm	Logistic Regression VS Rule Based	Gaussian Naive Bayes VS K Nearest Neighbours	Gaussian Naive Bayes VS Random Forest	Gaussian Naive Bayes VS Decision Tree	Gaussian Naive Bayes VS Support Vector Machines
5.40E-11	0.000194	9.92E-12	1.86E-14	2.82E-09	0.005278	3.87E-10	4.50E-07	4.96E-05	1.07E-17	5.38E-09	1.20E-16
S	S	S	S	S	S	S	S	S	S	S	S
Gaussian Naive Bayes VS XGBoost	Gaussian Naive Bayes VS lgbm	Gaussian Naive Bayes VS Rule Based	K Nearest Neighbours VS Random Forest	K Nearest Neighbours VS Decision Tree	K Nearest Neighbours VS Support Vector Machines	K Nearest Neighbours VS XGBoost	K Nearest Neighbours VS lgbm	K Nearest Neighbours VS Rule Based	Random Forest VS Decision Tree	Random Forest VS Support Vector Machines	Random Forest VS XGBoost
1.44E-12	1.50E-17	0.000286	3.11E-12	4.34E-10	1.43E-10	1.25E-06	4.48E-11	1.17E-05	3.00E-19	1.49E-05	1.09E-09
S	S	S	S	S	S	S	S	S	S	S	S
Random Forest VS lgbm	Random Forest VS Rule Based	Decision Tree VS Support Vector Machines	Decision Tree VS XGBoost	Decision Tree VS lgbm	Decision Tree VS Rule Based	Support Vector Machines VS XGBoost	Support Vector Machines VS lgbm	Support Vector Machines VS Rule Based	XGBoost VS lgbm	XGBoost VS Rule Based	lgbm VS Rule Based
2.62E-05	1.37E-09	1.64E-18	1.99E-15	3.16E-19	0.043009	1.82E-06	0.38018	6.91E-09	2.22E-07	1.04E-07	5.26E-09
S	S	S	S	S	S	S	-	S	S	S	S

Table 4. Statistical Test for Balanced dataset

6.2 Statistical testing for Regression

The results of statistical testing for both the datasets with price thresholds are summarized in the table below. Tables 5 and 6, S is indicative of the statistical significance between two algorithms.

7 Conclusions and Lessons Learnt

By applying Classification and Regression problems on the dataset set, it was clear that pre-processing stage of the machine learning is the most important task. Problems with dimensionality curse was well understood as the training and testing time reduced gradually as unrelated features were removed. Moreover applying concepts such as PCA to retain features with co-relations, in-spite of reducing features boosted the understanding of dimensionality reduction.The limitation of using the whole dataset was observed when the models were tested with cross validation techniques. Class imbalance pose a major problems in determining the accuracy of the model. This was realized when the performance results of balanced and unbalanced results were compared. Balanced data model showed the better results for F1 score and AUC in the classification problems. Filtering outlier was very crucial to find the best results in the regression problem of price prediction as root mean square error(RMSE) values decreased gradually after filtering the outliers. Overall, this project, helped to understand and solidify the theoretical concepts and methods of ML learning with hands-on approach.

LinearRegression VS SVR	LinearRegression VS DecisionTrees	LinearRegression VS KNearest Neighbours	LinearRegression VS RuleFit	LinearRegression VS RandomForest	LinearRegression VS LightGBM	LinearRegression VS XGBoost	SVR VS DecisionTrees	SVR VS KNearest Neighbours	SVR VS RuleFit
0.001154	0.000137	2.06E-10	0.477913	0.000167	1.75E-07	1.55E-05	0.040403	6.63E-09	0.272828
S	S	S	-	S	S	S	S	S	-
SVR VS RandomForest	SVR VS LightGBM	SVR VS XGBoost	DecisionTrees VS KNearest Neighbours	DecisionTrees VS RuleFit	DecisionTrees VS RandomForest	DecisionTrees VS LightGBM	DecisionTrees VS XGBoost	KNearest Neighbours VS RuleFit	KNearest Neighbours VS RandomForest
4.28E-08	1.04E-10	9.08E-09	3.90E-05	0.033209	1.02E-07	1.45E-09	2.61E-08	8.48E-06	1.42E-12
S	S	S	S	S	S	S	S	S	S
KNearest Neighbours VS LightGBM	KNearest Neighbours VS XGBoost	RuleFit VS RandomForest	RuleFit VS LightGBM	RuleFit VS XGBoost	RandomForest VS LightGBM	RandomForest VS XGBoost	LightGBM VS XGBoost		
4.02E-14	8.01E-13	0.004875	0.000127	0.001086	0.005607	0.20635	0.124134		
S	S	S	S	S	S	-	-		

Table 5. Evaluation metrics for dataset with price threshold 700$

LinearRegression VS SVR	LinearRegression VS DecisionTrees	LinearRegression VS KNearest Neighbours	LinearRegression VS RuleFit	LinearRegression VS RandomForest	LinearRegression VS LightGBM	LinearRegression VS XGBoost	SVR VS DecisionTrees	SVR VS KNearest Neighbours	SVR VS RuleFit
0.109801	0.002475	6.20E-10	0.857595	0.004864	0.001586	0.005578	5.74E-05	4.05E-11	0.222313
-	S	S	-	S	S	S	S	S	-
SVR VS RandomForest	SVR VS LightGBM	SVR VS XGBoost	DecisionTrees VS KNearest Neighbours	DecisionTrees VS RuleFit	DecisionTrees VS RandomForest	DecisionTrees VS LightGBM	DecisionTrees VS XGBoost	KNearest Neighbours VS RuleFit	KNearest Neighbours VS RandomForest
0.140688	0.052373	0.125568	4.44E-07	0.044334	2.52E-06	1.05E-06	4.61E-06	8.92E-07	4.89E-12
-	-	-	S	S	S	S	S	S	S
KNearest Neighbours VS LightGBM	KNearest Neighbours VS XGBoost	RuleFit VS RandomForest	RuleFit VS LightGBM	RuleFit VS XGBoost	RandomForest VS LightGBM	RandomForest VS XGBoost	LightGBM VS XGBoost		
3.21E-12	1.78E-11	0.039246	0.019827	0.035529	0.580763	0.847646	0.75095		
S	S	S	S	S	-	-	-		

Table 6. Statistical Test for dataset with price threshold 80$

8 Future Work

This project created models to classify Superhost and predict the price of a listing, regression models were built for one city Montreal, in the future work the models created in this project can be tested with other cities such as Toronto, Ottawa and extensive tests such as Wilcoxin and Friedman's test can applied to determine the best generalized model. More rigorous data pre-preprocessing techniques can be applied to further decrease the RMSE values for price prediction in regressor models. Further more, neural network can be applied for Superhost classification problem.

9 The References Section

References

1. InsideAirBnb Data Link, http://insideairbnb.com/get-the-data.html
2. What is Airbnb's Superhost Status Really Worth?, https://www.airdna.co/blog/airbnb_superhost_status
3. Why Feature Correlation Matters .... A Lot!, https://towardsdatascience.com/why-feature-correlation-matters-a-lot-847e8b
4. Feature Selection with sklearn and Pandas, https://towardsdatascience.com/feature-selection-with-pandas-e3690ad8504b
5. Implementing PCA in Python with Scikit-Learn, https://stackabuse.com/implementing-pca-in-python-with-scikit-learn/
6. Feature Selection in Python — Recursive Feature Elimination, https://towardsdatascience.com/feature-selection-in-python-recursive-feature-elimination-19f1c39b8d15
7. Rulefit repository, https://github.com/christophM/rulefit
8. Gradient Boosting Decision trees: XGBoost vs LightGBM (and catboost), https://medium.com/kaggle-nyc/gradient-boosting-decision-trees-xgboost-vs-lightgbm-and-catboost-72df6979e0bb
9. The ultimate guide to binary classification metrics, https://towardsdatascience.com/the-ultimate-guide-to-binary-classification-metrics-c25c3627dd0a
10. Regression Analysis: How Do I Interpret R-squared and Assess the Goodness-of-Fit?, https://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit
11. Model Tuning (Part 2 - Validation Cross-Validation), https://dziganto.github.io/cross-validation/data20science/machine20learning/model20tuning/python/Model-Tuning-with-Validation-and-Cross-Validation/
12. 17 Statistical Hypothesis Tests in Python, https://machinelearningmastery.com/statistical-hypothesis-tests-in-python-cheat-sheet/
13. Hypothesis testing in Machine learning using Python, https://towardsdatascience.com/hypothesis-testing-in-machine-learning-using-python-a0dc89
14. Scikit-Learn User Guide, https://scikit-learn.org/stable/user_guide.html
15. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O.,Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A.,Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine-learning in Python. Journal of Machine Learning Research12, 2825–2830 (2011)