Predicting Interest Levels for Rental Listings

Team - Alpha Beta Gamma

Srinivas Rao Gunti
M.Tech-CSE 1st Year
IIIT, Bangalore
MT2018120

Vasu Bansal
M.Tech-CSE 1st Year
IIIT, Bangalore
MT2018130

Saurabh Mehta
M.Tech-CSE 1st Year
IIIT, Bangalore
MT2018105

***Abstract*—This report focuses on prediction of an apartment listings popularity on renthop.com. We use the dataset provided by renthop on Kaggle to predict the interest level of potential customers for a specified listing. The interest levels are bucketed into 'high', 'medium', 'low' buckets and we do classification to predict probability of a listing for each bucket. Evaluation metric we use is multiclass log loss on a test dataset extracted from the training data provided.**

I. Introduction

Apartment hunting can be tough and finding a new home isn't about looking at every apartment listing, it's about looking at the best ones. Quality, not quantity. RentHop.com, an apartment listing website makes apartment search smarter by using data to sort rental listings by quality. Our data comes from the website which contains features such as price, photos, features of the apartment, number of bedroom and bathrooms etc for every apartment listing with an additional columns called 'interest_level' which measures the number of inquiries a listing has in the duration during which the listing was live on the site. Using these features we try to predict the probability of an apartment falling into one of the interest level bucket 'high', 'medium' and 'low'. The goal of this precition is to help RentHop better handle fraud control, identify potential listing quality issues, and allow owners and agents to better understand renters' needs and preferences.

II. Data Set analysis

The original dataset provided to us on the Kaggle Platform had 39481 rows with 15 features as listed below:

- bathrooms: number of bathrooms
- bedrooms: number of bathrooms
- building_id
- created
- description
- display_address
- features: list of features about the apartment
- latitude
- listing_id
- longitude
- manager_id
- photos: a list of photo links
- price: in USD
- street_address
- interest_level: this is the target variable. It has 3 categories: 'high', 'medium', 'low'

Basic Analysis on the data shows there are no missing values for any of the features. There are a few outliers in some features which we cap to the 99th percentile to avoid overfitting.

A. 'interest_level'

This is the target variable that we need to predict. Fig.1 shows how the interest level distributes across the dataset.

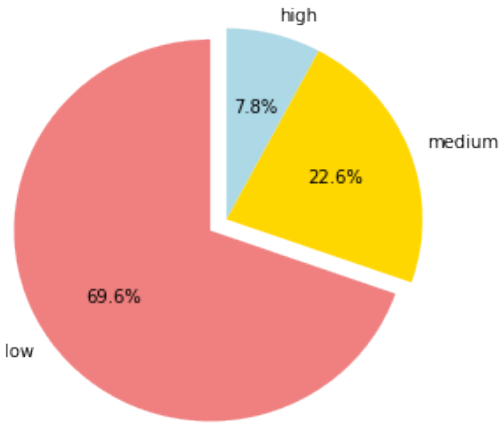

Fig. 1: 'interest_level' Distribution

B. 'bedrooms' & 'bathrooms'

People show "high" interest level when the no of bedrooms are on par with no of bathrooms. Blue diagonal dots in Fig.2 gives evidence for the same.

C. 'latitude' & 'longitude'

Fig.3 shows the plot of latitude and longitude with colouring based on interest level (red: 'high', blue: 'medium', 'green': low) to get a general idea of how the listings interest level varies based on its location. Although no general trend is directly visible but on further analysis we find that grouping listings based on location has a positive impact on predicting its popularity.

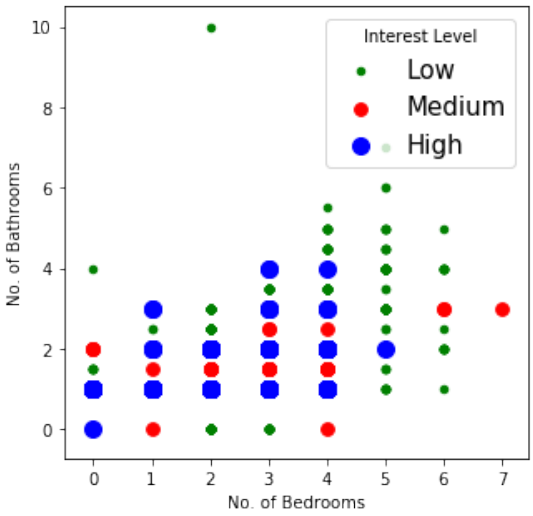

Fig. 2: Number of bedrooms vs bathrooms for different levels

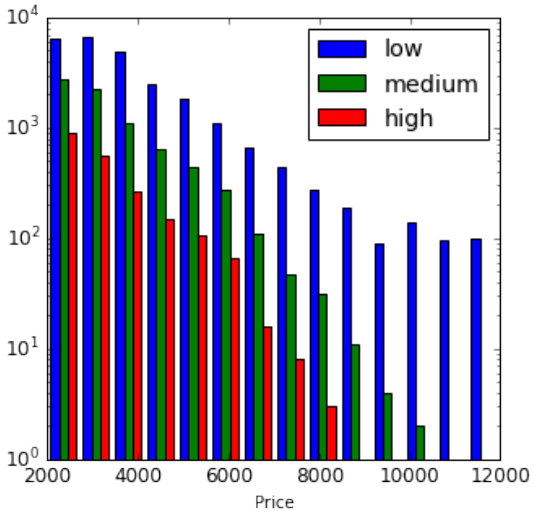

Fig. 4: Price range for different interest levels

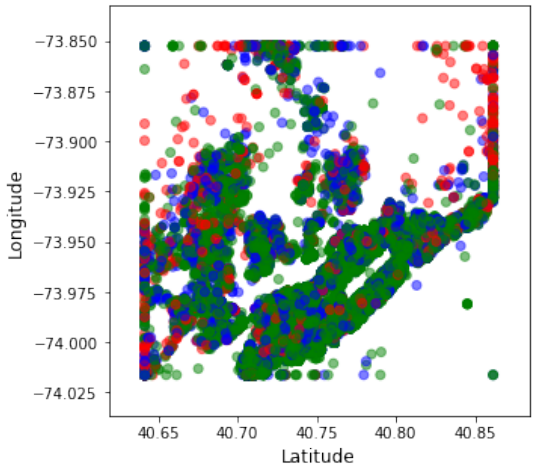

Fig. 3: Latitude vs Longitude

D. *'price'*

Fig.4 clearly shows that as the price of the listing increases the interest level drops as seen by the red and green bar plots. This can be attributed to the fact that while renting most people prefer apartments that are low priced.

III. Feature Engineering

Apart from the features provided to us in the original dataset we brainstormed for new features that could have an impact on the interest level for a listing.

New features we created are:

- Simple features:
  - 'num_photos'
  - 'num_features'
  - 'price_bed'
  - 'created_day'
  - 'num_description_words'
  - 'no_lines_desc'
  - 'manager_count'
  - 'street_count'
  - 'building_count'
  - 'total_days'
  - 'price_latitude'
  - 'num_cap_share'
  - 'diff_rank'
- 'distance_park' & 'distance_center'
- 'manager_id_skill'
- 50 features extracted from features column using vectorization
- 25 description keywords extracted from description column using vectorization
- 'cluster_label' & 'price_var'

A. *Simple Features*

We create some simple features from our original features:

- 'price_bed': Price per bed. We divide the number of bed rooms by the price of the listing to find the price per bed.
- 'created_day': The day on which the listing was created. We tried different combinations of date components like hour, day, month and year and found that the day on which the listing was created has a positive impact on the listings interest level. This may be due to the fact that listings created on weekends get more traffic.
- 'num_description_words': Number of words written in the description. This seems like not so important feature, but it had a great impact on models overall prediction. We tried to extract more features from this column like extracting most repeatable meaningful words, length of description, num of capital words etc. but they had a low impact on the models overall prediction. So we dropped them to keep the model generalizable. We cannot make a concrete observation from the Fig.5 but we can say that listing with less description has a higher probability of

having low interest level which is evident from the bulge where number of description words tend to zero.

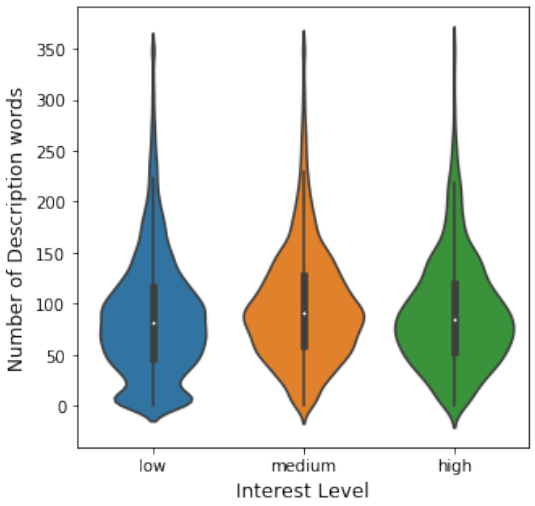

Fig. 5: num_description_words vs interest_Level

- 'no_lines_desc': Number of lines in the description written. More number of lines written in description led to a higher interest level for a listing.
- 'manager_count': The person who puts up a listing on renthop is called the manager of that listing. On analysis we found that certain managers have a very good record of putting up listing's that generate a higher interest level. Here we do a simple count based on manager_id. We explore the feature 'manager_id' further to extract 'skill' of the manager.
- 'num_features': Count of number of features provided for a listing. Fig.6 shows the same.

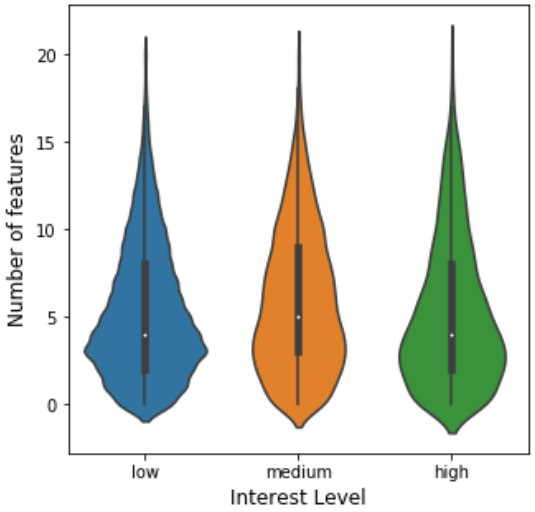

Fig. 6: no_features vs interest_level

- 'street_count': Represents the number of other listings that are in the same street as the current listing.
- 'building_count': Number of listings in the same building as the listing. Based on 'building_id'.
- 'total_days': Number of days passed since the first listing got recorded in the data.
- 'price_latitude': Ratio of the price of a listing wrt latitude. Usually price of a listing has a huge impact on interest level, so we came up with this feature which compared the relative price of a listing with respect to lattitude. We tested with longitude as well, but that did not improve our model.
- 'num_photos': Count of the number of photos provided for a listing. If we observe Fig.7, we notice that the listings which have zero photos have high chances of having low interest level. This is observed from a significant spread when number of photos are absent from that particular listing.

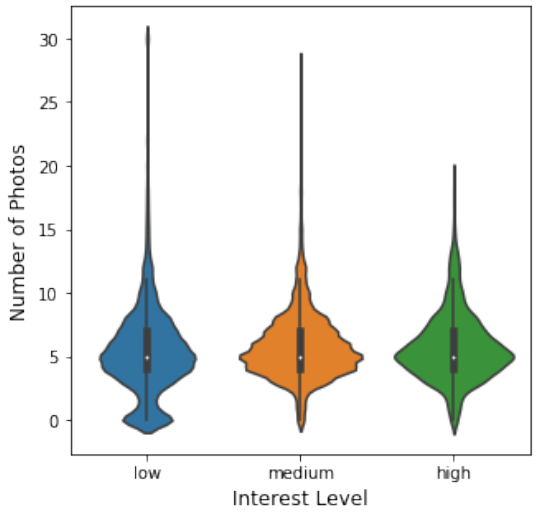

Fig. 7: no_photos vs interest_level

- 'num_cap_share': Number of capital letters in the description of the listing. This feature was helping in identifying spam in the description field.
- 'diff_rank': Ratio of 'total_days' to 'listing_id'.

B. 'manager_id_skill'

Certain managers have a very good record of putting up listing's that generate a higher interest level. This is a valid assumption since certain people are more skilled than others in marketing and thus the quality of their listings is better than the others. This assumption is proven true by our analysis as this feature has the biggest impact on the models predictions accuracy. Fig.8 shows the number of listings of each interest level when bucketed by manager skill values. We can observe that as manager skill increases listings with "low" interest_level decreases. It can be observed that most of listings corresponding to low manager skill will have "low" interest level.

We calculate this feature by grouping the data based on manager id's then calculate the fraction of listings under a

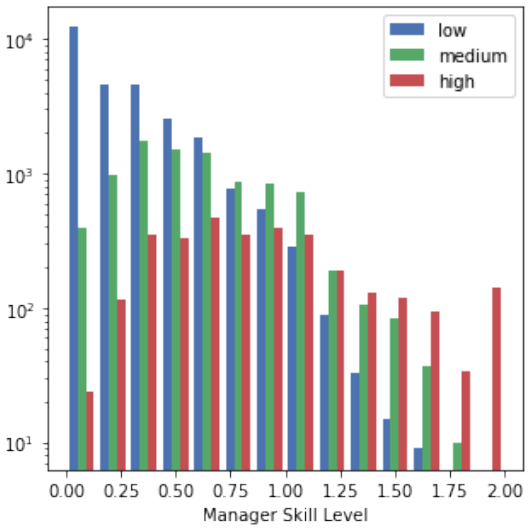

Fig. 8: manager_id_skill vs interest_level

manager based on interest levels. We cut the values beyond a certain threshold with mean value of top 10 managers since our data is skewed towards 'low' interest level and this offsets our predictions for medium and high interest level listings.

We tried doing the same analysis for calculating 'building_id_skill'. Logically it makes sense since a building could have more interest level based on its location and facilities and thus all listings belonging to that building must have a similar interest level. But on adding this feature our model accuracy surprisingly decreases. On analysis we found that a manager is usually responsible for a whole building so the building skill valuation was already captured by the manager skill feature thus a little to no improvement in performance.

C. 50 features extracted from 'feature' column using vectorization

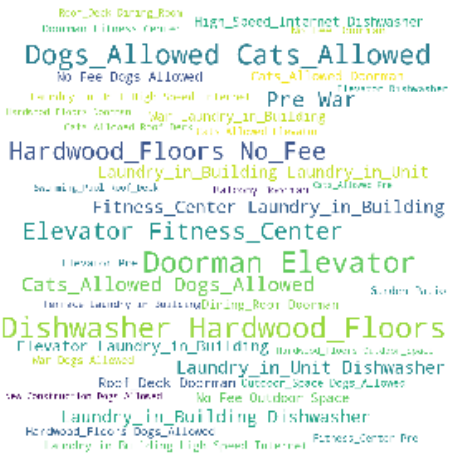

Fig. 9: Features Word Cloud

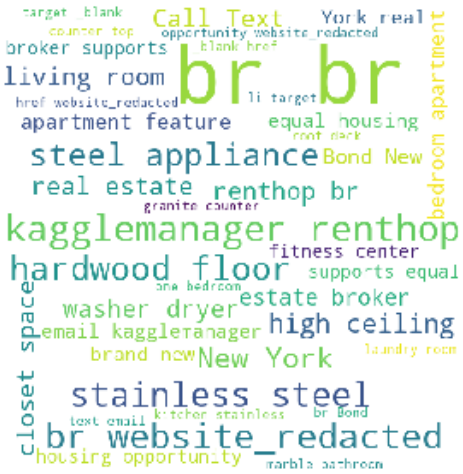

Fig. 10: Description Word Cloud

The features column list 'features' of each listing such as pets allowed or not, centralized air conditioning or separate, parking space private or common, garden present or not, apartment has hardwood floors, garage, terrace, laundry units, dishwasher, modular kitchen etc. On analysis we found that there are appx. 1200 features listed in the dataset provided to us. Many of them were synonymous, had spelling mistakes or had a special character embedded in them. We first removed special characters and then we sort these according to their frequencies after which we manually identified and corrected synonymous and spelling mistakes. We then extracted top 50 features and vectorized them to create 50 new features. We tried taking different number of features and found 50 as the best value using hit and trial.

D. 'cluster_label' & 'price_var'

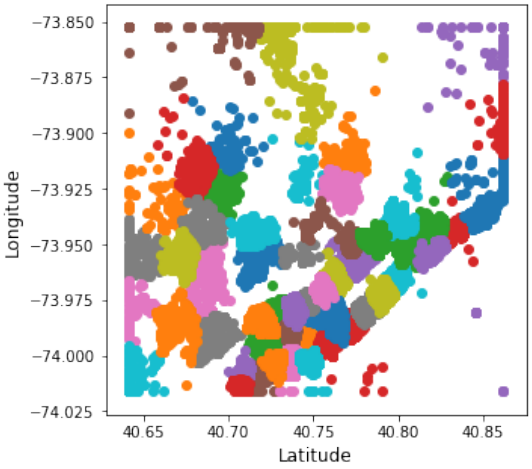

Fig. 11: clusters

These features are again based on latitude and longitude. We cluster the listings based on their locations. The idea

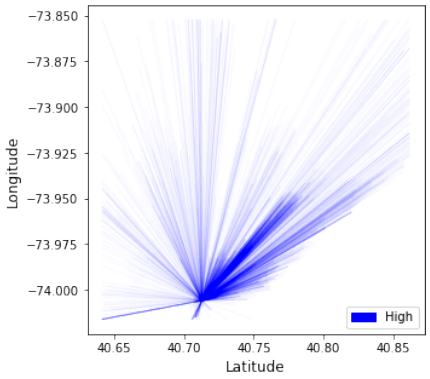

Fig. 12: city_center - ”high”

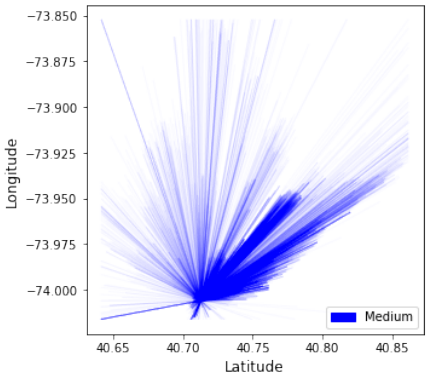

Fig. 13: city_center - ”medium”

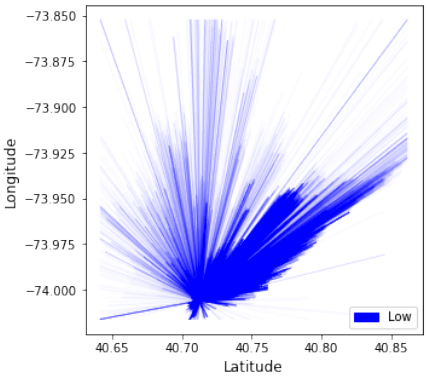

Fig. 14: city_center - ”low”

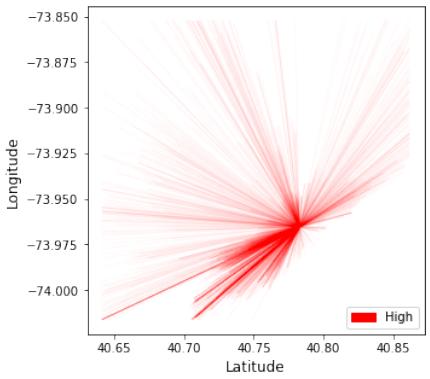

Fig. 15: central_park - ”high”

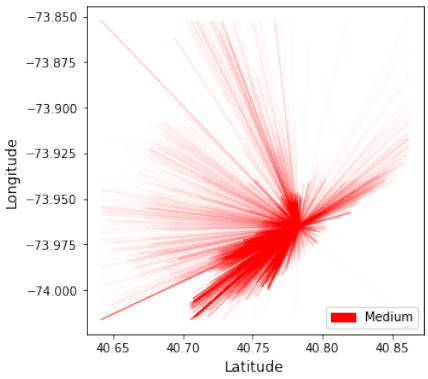

Fig. 16: central_park - ”medium”

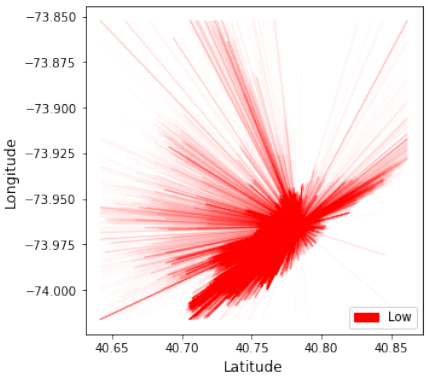

Fig. 17: central_park - ”low”

behind this is to identify neighbourhoods which have high interest level. We create 60 such clusters. We also identify the price variance for the neighbourhood to identify the price gap between the listings in a neighbourhood.

E. 25 features extracted from 'description' column using vectorization

'description' attribute has a brief description about the listing. We extracted keywords from this column based on the frequency of their occurrences ignoring stop words assuming that some key words might pique the interest of user seeing that listing. We limited number of keywords to 25 as anything more was degrading the performance of our model on previously unseen data.

F. 'distance_park' & 'distance_center'

We explore latitude and longitude columns. Adding these columns directly has a positive impact on the model's prediction. We extract more information from these columns by calculating the distance of each listing from the city's hot spots. We based this feature based on the fact that apartments close to hot spots has more value and thus generate more interest. We identified “New York City Center” and “New York Central Park” as the two of such hot spots which have an impact on the listing's interest level. Fig.12, 13, 14 shows the distance of listing from central park, we can observe that the listings which are closer are mostly of high interest and as the distance increase listings interest level drops. Similar is the case for city center.

The Fig.15, 16, 17 shows that the distance from the city center has a similar effect on the interest_level of the listings. We also observe that the listings between city center and the central park has high interest_level than other listings.

For creating these two features we find the actual distance between the coordinate values of the listing and the city's hot spots.

IV. Predictive Task

A. Task Description

Our predictive task on this dataset is to determine the probablility of interest_level's for rental listings. Since the interest level of a certain rental listing can only be low, medium and high, this problem could be viewed as a classification problem.

B. Evaluation Metric

Model Performance is evaluated by the multi-class cross-entropy or multi-class logarithmic loss on the test set.

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

where N is the number of listings in the test set, M is the number of class labels (3 classes), log is the natural logarithm, yij is 1 if observation i belongs to class j and 0 otherwise, and pij is the predicted probability that observation i belongs to class j.

C. Feature Selection

The final feature set that we confirmed after testing on different combinations is as follows:

- 'bathroom'
- 'total_rooms'
- 'latitude'
- 'longitude'
- 'num_photos'
- 'num_features'
- 'price_bed'
- 'created_day'
- 'num_description_words'
- 'no_lines_desc'
- 'manager_count'
- 'street_count'
- 'building_count'
- 'total_days'
- 'price_latitude'
- 'num_cap_share'
- 'diff_rank'
- 'distance_park '
- 'distance_center '
- 'manager_id_skill'
- 'label'
- 'price_var'
- '50 keywords extracted from feature attribute'
- '25 keywords extracted from description attribute'

V. Models

Appropriate models for this task are classifiers that are capable of making soft predictions since our outcome variable 'interest_level' has 3 categories and we have to provide probabilities for each of them. Given the fact that our data contains strongly heterogeneous features, like price, number of bedrooms, number of bathrooms are numerical data, manager_id, building_id & features columns have categorical data, descriptions and addresses are text data, latitude and longitude are geographical data, we decided that instead of training a single classifier for all the features, we will use ensemble classifiers. We use stratify aspect of train_test_split to split the training data into training and validation sets, as the data given to us was skewed. We used the random state attribute of train_test_split throughout the model building process.

For parameter tuning we used the grid search package in Python scikit-Learn. Through the cross-validation packages, we were able to optimize the algorithms to minimize log-loss. It took us a great deal of time and research to tune the hyper-parameters to churn out an improvement in the log loss.

A. Random Forest Classifier

Random Forest is an ensemble of classifiers, where all classifiers are linked by a number of decision trees and a method where the performance is adaptively improved by introducing a series of weak classifiers in each iteration.

Hyper-parameters:

- no_of_estimators: 1050
- max_depth: 30
- min_samples_leaf: 1
- min_samples_split: 10
- max_features: sqrt
- n_jobs: -1 (This parameter is just to speed up training process)

Validation Result's for Random Forest

	Accuracy(%)	**Log Loss**	**F-Score**
Training Data	94.94	0.29702	91.58
Test Data	74.15	0.58653	51.18

B. Gradient Boosting Classifier

Apart from Random Forest, one approach similar with it is Gradient Boosting Classifier. Gradient boosting is one of the state-of-art machine learning algorithms, especially for solving supervised learning problems. Gradient boosting is a combination of gradient descent and boosting. This family of algorithms can reduce bias and variance at the same time in supervised learning.

Hyper-parameters:

- no_of_estimators: 190
- learning_rate: 0.15
- max_depth: 7
- min_samples_leaf: 60
- min_samples_split:1900
- max_features: 18

Validation Result's for GBM

	Accuracy(%)	**Log Loss**	**F-Score**
Training Data	83.80	0.40664	73.97
Test Data	75.14	0.55164	58.15

C. Light GBM

Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.

Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise. So when growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms. Also, it is surprisingly very fast, hence the word 'Light'.

- Number of estimators: 280
- learning_rate : 0.07

- Max_Depth: 10
- Min_child_sample: 22
- objective : Multiclass
- Feature Fraction : 0.75
- max_bin : 265

Validation Result's for LightGBM

	Accuracy(%)	Log Loss	F-Score
Training Data	86.64	0.36501	79.98
Test Data	75.12	0.56061	58.41

D. XGBoost

XGBoost is an improved implementation of Gradient Boosting Decision Trees (GBDT). Our hypothesis for choosing this algorithm was that it may perform best as it might deal with all problems of our dataset, i.e. large feature space, non-linear decision boundary and feature interaction, potential outliers and overfitting.

- Number of estimators: 600
- learning_rate : 0.05
- Max_Depth: 3
- Min_child_weight: 1
- Gamma :1900
- Subsample : 0.7
- colsample_bytree : 18
- reg_alpha : 0.075

Validaition Result's for XGBoost

	Accuracy(%)	Log Loss	F-Score
Training Data	82.35	0.42780	71.38
Test Data	75.30	0.55725	58.51

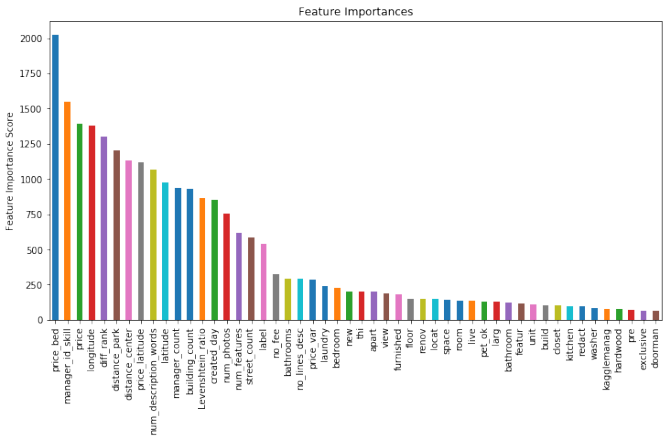

Fig. 18: Feature Importances

VI. Summary

In this project, we extracted low level and fine-designed features from the RentHop dataset, and tried improving the log loss.

Our best model was gradient boosting which gave us a logloss of 0.545 on the private leader board.

Log Loss for final models

	Private LB Score	Public LB Score
RandomForestClassifer	0.57937	0.58603
XGBoost	0.55274	0.55928
LightGBM	0.56124	0.56413
GradientBoostingClassifier	0.54668	0.55733

Compared to the performance of other models in the actual Two Sigma Rental Listing Competition leaderboard, there is clearly some scope for us to make further improvements. The future work would include building more complex models that are able to account for time series, sentiments, interest trends, etc.

References

[1] https://www.kaggle.com/c/two-sigmaconnect-rental-listing-inquiries
[2] https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/
[3] https://towardsdatascience.com/analytics-on-the-new-york-rental-market-top-17-on-kaggle-91bb139d8f4b
[4] https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html