# Machine Learning Engineer Nanodegree – Capstone Project

## *Two Sigma Rental Listings*

## I.Definition:

***Project Overview:***

This is project's goal is to find the interest level of rental listings of a startup company Renthop. Finding the interest level helps the customers not to spend much time in searching for the highly interested advertisements and also it reduces fraudulent activity on website. This project belongs to real estate domain and we are going to solve this problem using machine learning skills. Machine learning is applied in real estate domain for several applications. For ex: Zillow Inc applies machine learning to find predict the house prices and they call this feature as zestimate. Machine learning is applied further in recommender systems to find the advertisement according to our previous search. In our project we are applying machine learning to find interest level of the advertisements.

***Problem Statement:***

In the project we need to classify the rental listings based on interest level, So we need to find out the Interest level. In the rental listing advertisement we have some important data such as No.of Bedrooms, No.of Bathrooms, Features, Description, Longitude, Latitude etc. and using these we need to classify the rental advertisement. We can model this real world real estate problem into a machine learning problem by taking the advertisement data as input applying classification algorithms on this data to develop a classification model that classifies the incoming advertisements according to the interest level. Interest level is classified into 3 classes 'High' , 'Medium' and 'Low'

***Metrics:***

F1 score is used as evaluation metric in the project. F1 score is the evaluation metric to measure accuracy; it is the ratio of true positive rate (recall) to precision. i.e we recall is when actually yes , how often it is predicted yes and precision is when it is predicted yes how often it is correct. So F1 score measures accuracy using the statistics precision p and recall r. Precision is the ratio of true positives (tp) to all predicted positives (tp + fp). Recall is the ratio of true positives to all actual positives (tp + fn). F1 is a very good evaluation metric for the classification as F1 weights recall and precision equally .A good retrieval algorithm will maximize both precision and recall simultaneously. Thus moderately good performance on both will be favored over extremely good performance on one and poor performance on the other. I want to clearly classify the given listing as High or Medium or Low i.e. tagging a listing with any of these 3 labels. So, in our project, I am interested it to find the Recall and precision ratio. As F1 summarizes exactness and completeness I am going to use F1 as evaluation metric

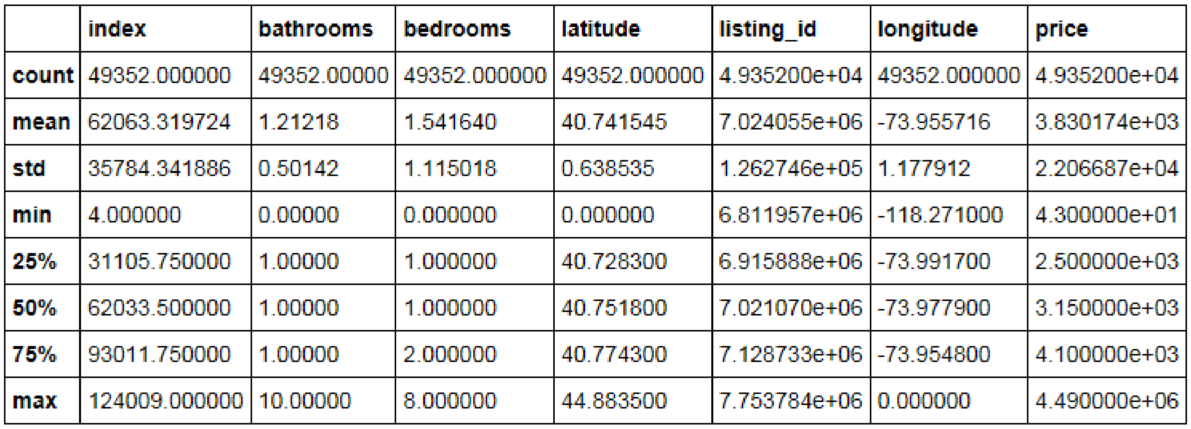
## II.Analysis

### ***Data Exploration***

Attached datasets along with this document. Datasets are in JSON format, and for our data analysis and machine learning modeling we convert JSON format data into DataFrames using Pandas Library

- Train data contains 49352 instances and it takes 100 MB of memory
- Test data contains 74659 instances and it takes 47.3 MB of memory

Insight of train data



| | index | bathrooms | bedrooms | latitude | listing_id | longitude | price |
|---|---|---|---|---|---|---|---|
| **count** | 49352.000000 | 49352.00000 | 49352.000000 | 49352.000000 | 4.935200e+04 | 49352.000000 | 4.935200e+04 |
| **mean** | 62063.319724 | 1.21218 | 1.541640 | 40.741545 | 7.024055e+06 | -73.955716 | 3.830174e+03 |
| **std** | 35784.341886 | 0.50142 | 1.115018 | 0.638535 | 1.262746e+05 | 1.177912 | 2.206687e+04 |
| **min** | 4.000000 | 0.00000 | 0.000000 | 0.000000 | 6.811957e+06 | -118.271000 | 4.300000e+01 |
| **25%** | 31105.750000 | 1.00000 | 1.000000 | 40.728300 | 6.915888e+06 | -73.991700 | 2.500000e+03 |
| **50%** | 62033.500000 | 1.00000 | 1.000000 | 40.751800 | 7.021070e+06 | -73.977900 | 3.150000e+03 |
| **75%** | 93011.750000 | 1.00000 | 2.000000 | 40.774300 | 7.128733e+06 | -73.954800 | 4.100000e+03 |
| **max** | 124009.000000 | 10.00000 | 8.000000 | 44.883500 | 7.753784e+06 | 0.000000 | 4.490000e+06 |

Data Source URL → https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries/data

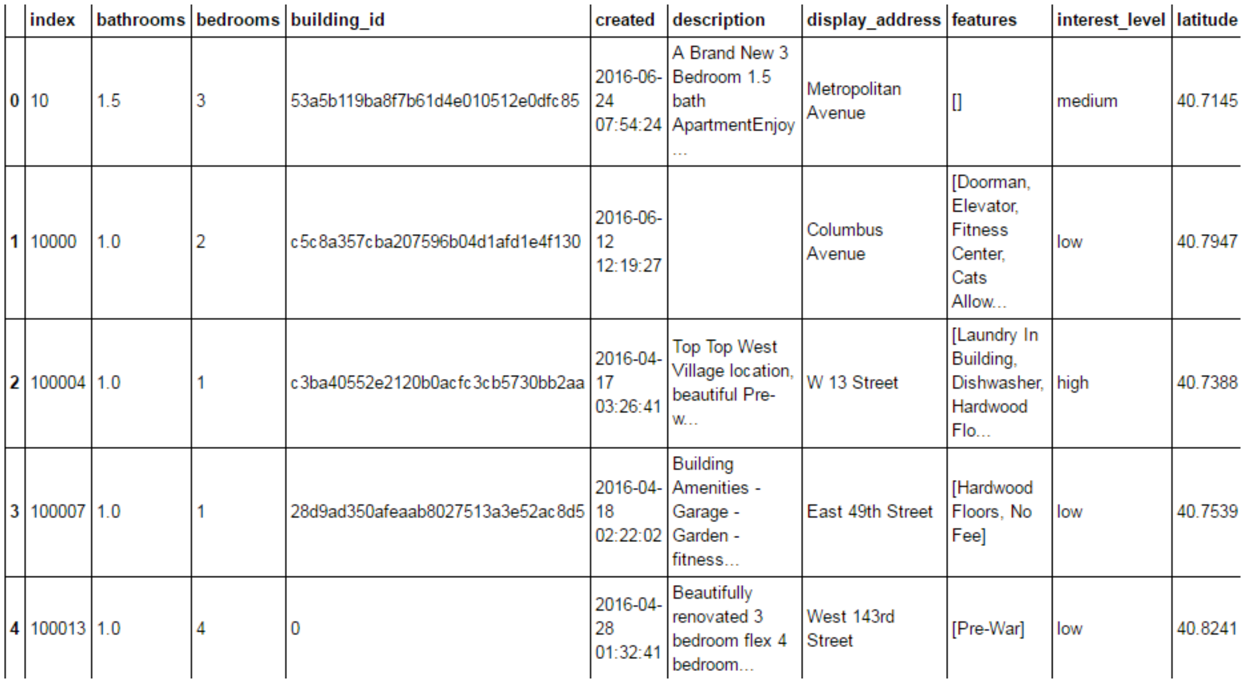
### ***Data Fields lists and their description***

- **bathrooms: number of bathrooms**
- **bedrooms: number of bathrooms**
- **building_id**
- **created : ad reation date**
- **description**
- **display_address**
- **features: a list of features about this apartment**
- **latitude**
- **listing_id**
- **longitude**
- **manager_id : id of the real estate manager in the renthop portal**
- **Photos: a list of photo links. You are welcome to download the pictures yourselves from renthop's site, but they are the same as imgs.zip.**
- **price: in USD**
- **street_address : house's street address in New York City**
- **interest_level: this is the target variable. It has 3 categories: 'high', 'medium', 'low'**

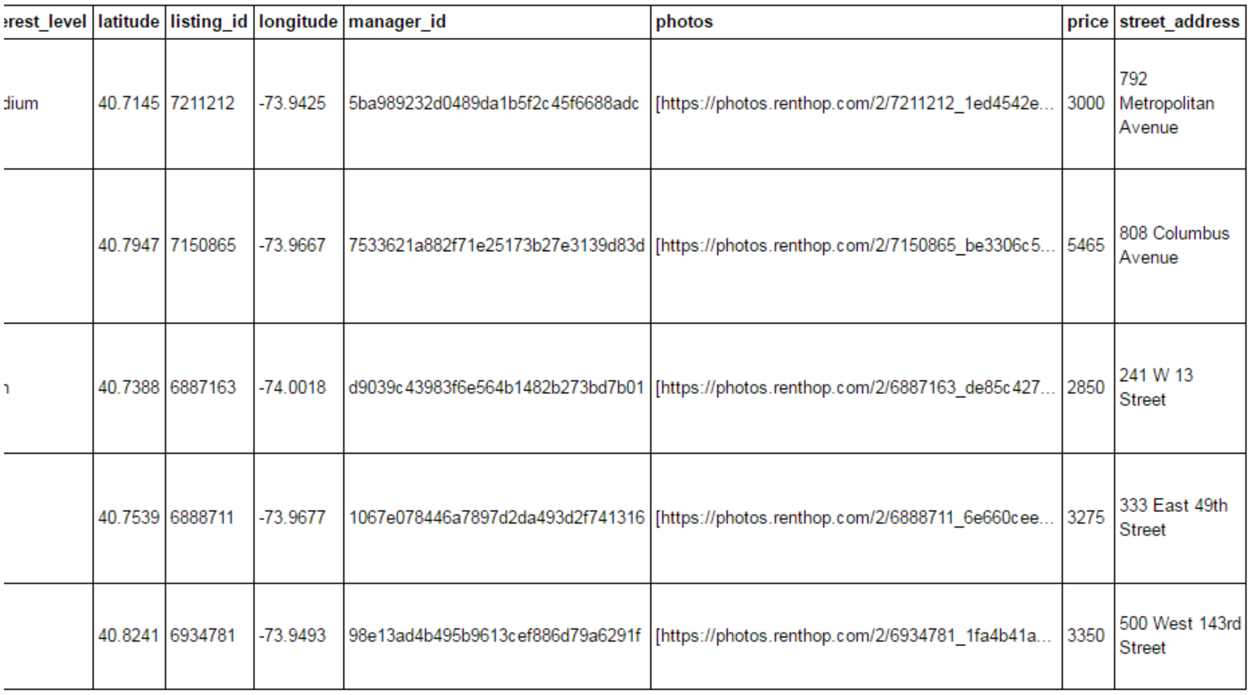
**The possible output class labels are High, Medium and Low for the Interest Level .The training data is huge so I did not applied K-Fold for train test split, I used only regular Train –Test Split. Apart from this I haven't used any other special methods considering the volume of data**

***Note***: test set doesn't contain target label, we need to actually split training set into train-test and get a model and apply this on our actual test data set. The results of actual test set are

Sample data set, it contains bathrooms, bedrooms, creation date, creation date, description, dispay_address, features, interest level, latitude, longitude, manager, photos, price and street address and out of these features we are going to use bathrooms, bedrooms, description, features, latitude, longitude and price to classify the interest level



| | index | bathrooms | bedrooms | building_id | created | description | display_address | features | interest_level | latitude |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 10 | 1.5 | 3 | 53a5b119ba8f7b61d4e010512e0dfc85 | 2016-06-24 07:54:24 | A Brand New 3 Bedroom 1.5 bath ApartmentEnjoy ... | Metropolitan Avenue | [] | medium | 40.7145 |
| 1 | 10000 | 1.0 | 2 | c5c8a357cba207596b04d1afd1e4f130 | 2016-06-12 12:19:27 | | Columbus Avenue | [Doorman, Elevator, Fitness Center, Cats Allow... | low | 40.7947 |
| 2 | 100004 | 1.0 | 1 | c3ba40552e2120b0acfc3cb5730bb2aa | 2016-04-17 03:26:41 | Top Top West Village location, beautiful Pre-w... | W 13 Street | [Laundry In Building, Dishwasher, Hardwood Flo... | high | 40.7388 |
| 3 | 100007 | 1.0 | 1 | 28d9ad350afeaab8027513a3e52ac8d5 | 2016-04-18 02:22:02 | Building Amenities - Garage - Garden - fitness... | East 49th Street | [Hardwood Floors, No Fee] | low | 40.7539 |
| 4 | 100013 | 1.0 | 4 | 0 | 2016-04-28 01:32:41 | Beautifully renovated 3 bedroom flex 4 bedroom... | West 143rd Street | [Pre-War] | low | 40.8241 |

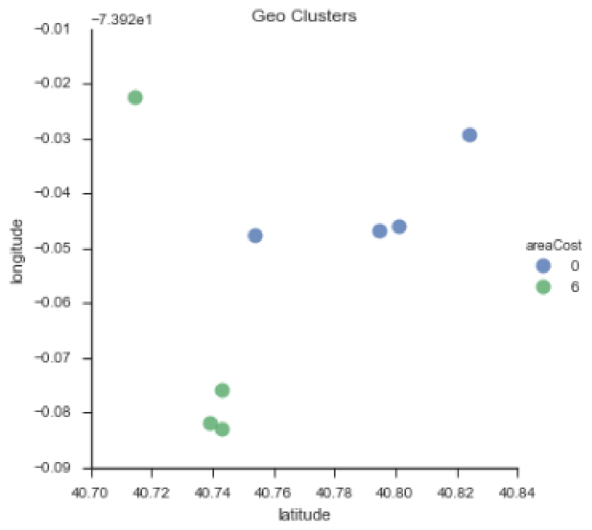


| erest_level | latitude | listing_id | longitude | manager_id | photos | price | street_address |
|---|---|---|---|---|---|---|---|
| dium | 40.7145 | 7211212 | -73.9425 | 5ba989232d0489da1b5f2c45f6688adc | [https://photos.renthop.com/2/7211212_1ed4542e... | 3000 | 792 Metropolitan Avenue |
| | 40.7947 | 7150865 | -73.9667 | 7533621a882f71e25173b27e3139d83d | [https://photos.renthop.com/2/7150865_be3306c5... | 5465 | 808 Columbus Avenue |
| | 40.7388 | 6887163 | -74.0018 | d9039c43983f6e564b1482b273bd7b01 | [https://photos.renthop.com/2/6887163_de85c427... | 2850 | 241 W 13 Street |
| | 40.7539 | 6888711 | -73.9677 | 1067e078446a7897d2da493d2f741316 | [https://photos.renthop.com/2/6888711_6e660cee... | 3275 | 333 East 49th Street |
| | 40.8241 | 6934781 | -73.9493 | 98e13ad4b495b9613cef886d79a6291f | [https://photos.renthop.com/2/6934781_1fa4b41a... | 3350 | 500 West 143rd Street |

Data is missing sometimes in description column, but as we are converting description into features columns using NLP, if the feature is not described in description then we will declare 0 and if present we declare it as 1.
The prices have some outliers in price column, so I normalized the price column using sklearn library functions.
Apart from these we don't have any missing data.

***Exploratory Visualization:***

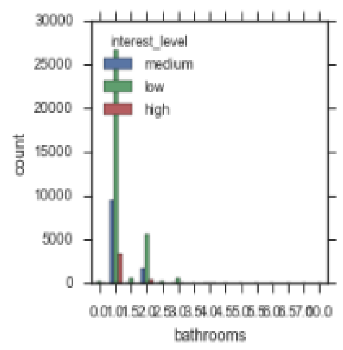
I performed clustering on longitude and latitude to form geo clusters. Here is the representation of clusters for first 8 columns of longitude and latitude as geo clusters

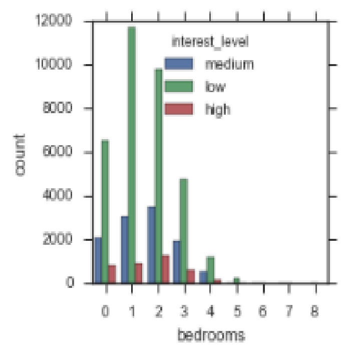


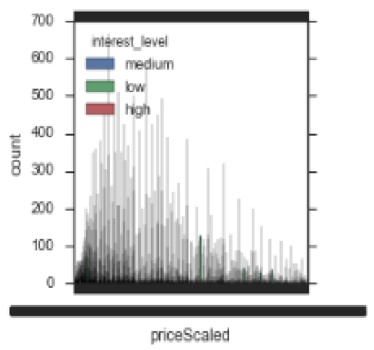
The first 8 longitudes and latitude columns fall into two clusters only 2 clusters (cluster 0 and cluster 7) but when we plot all the columns of latitudes and longitudes then we can see several other clusters.

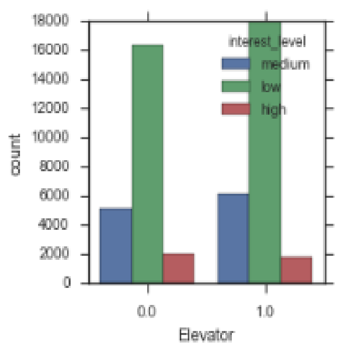
I have not displayed all columns of latitude and longitudes on visualizations as it makes the picture clumsy.
I used matplotlib and seaborn libraries for this visualization

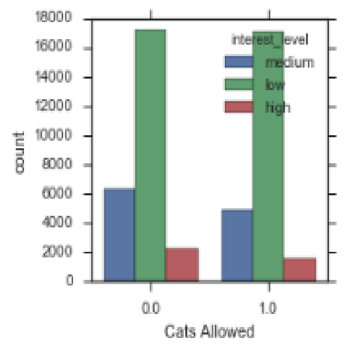
***<u>Visualization of columns count with respect to Interest level (“High”, “Medium” and “Low”)</u>***

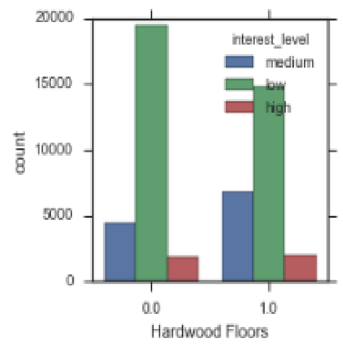


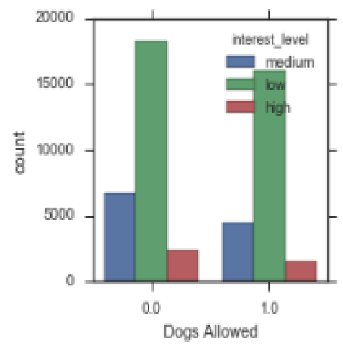


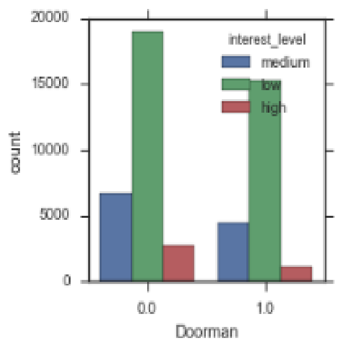


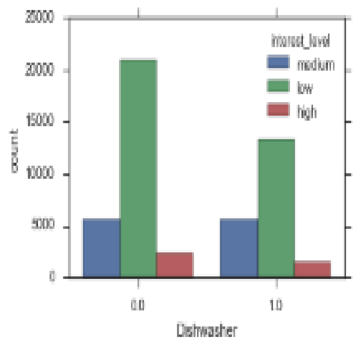


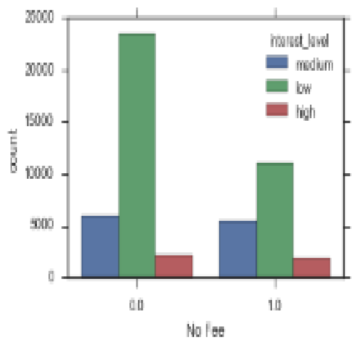


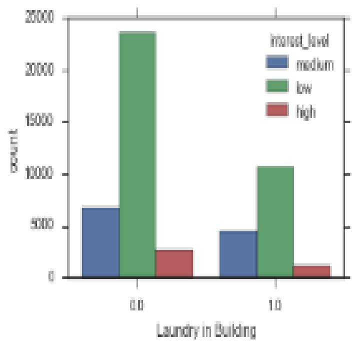


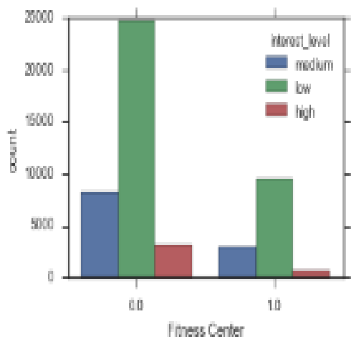


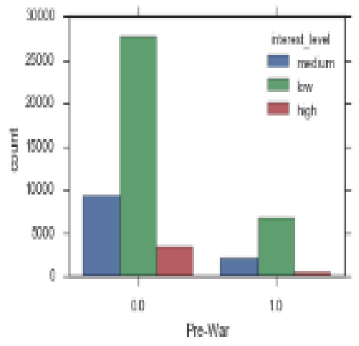


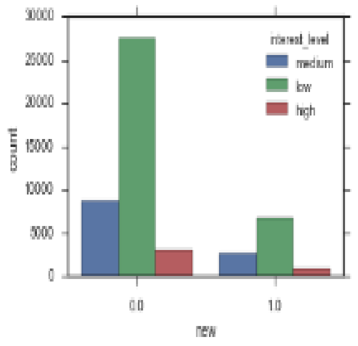


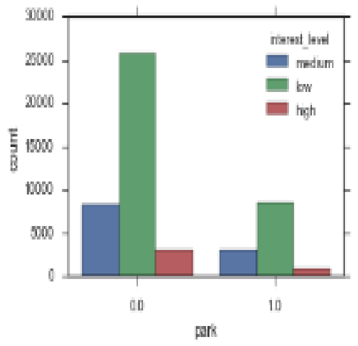


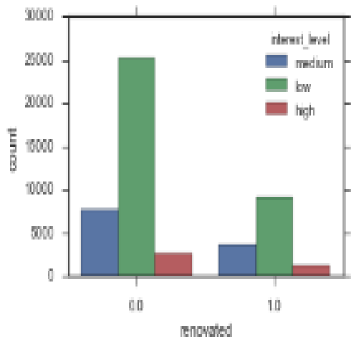


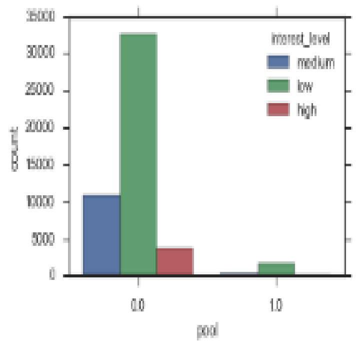


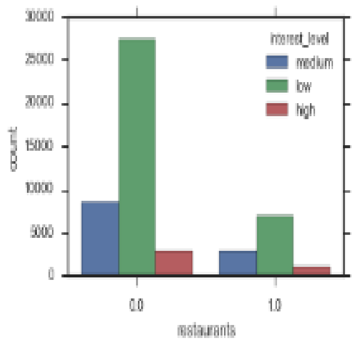


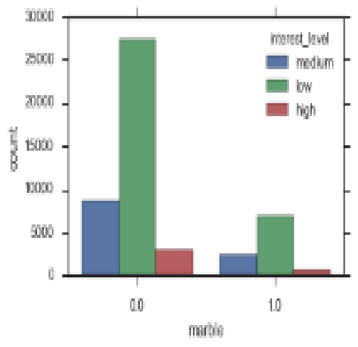


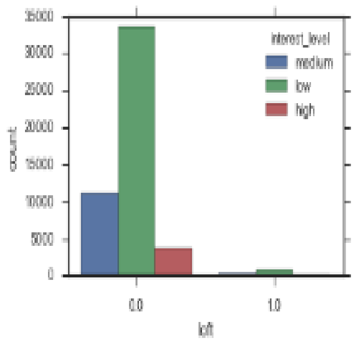












With the above visualizations, we can visualize how a feature impacts the interest level. We have 0 in several chart's x-axis and this means that feature is absent. We can ignore this case 0 for the features such as hardwood floors, park, restaurants etc.

From the above chart we get some valuable information in predicting the user's interest level.
The above chart answers the question, how a feature impacts the interest level.
For example bedrooms feature, when the number is 1 bedroom or 2 bedrooms then there are more users with high interest level.

Data , axis , titles are clearly defined for the above grid of charts.
Y-axis contains the count of users, X-axis represent the features quantity.

## ***Algorithms and Techniques***

I applied Natural Language processing to extract some valuable features from the input description label. Description label is in text format and our target is to extract valuable data using NLP. I converted al the rows of description label into as single string and applied NLTK on this text. I removed stop words ,puntutaion and words less than 3 from this text using NLTK library. Now on the processed text I wrote a snippet of the code to get the most common words. From the list of common words using my domain knowledge I have taken some words such as renovated , pool etc as features

The output is a multi-class output.so we need to used an multi class classifier algorithm. In my project initially I applied randomtree classifier and later feed it to OneVsRestClassifier.

(n_estimators=50,random_state=1,max_depth=80, min_samples_split=30, min_samples_leaf=1) are the parameters and values for randomtree forests

I choose randomtree classifier because it yield better results and I also tried adaboost, Support vector machines, but out of all these algorithms randomtree Classifier yields better results.
Later, I fed this algorithm to OneVsRestClassifier , because our output label is multi class, not bi-class.

Random forests are an improvement over bagged decision trees. Decision trees are greedy, they choose variable to split on using a greedy algorithm that minimizes error. In decision trees, when selecting a split point the learning algorithm considers all variables and all variables values to select the optimal split point. Random forests limited no.of features are searched .and several no.of the algorithms with limited no.of features are ensemble to give the best results.

Adaboost is a successful boosting algorithm developed for classification. Boosting is a ensemble method that creates a strong classifier from a number of week classifiers. First create a model from training data, then create another model that attempts to correct the errors from the first model. And this pattern continues i.e we keep on adding the models until we get best optimized performance. Adaboost is used to boost any machine learning model. By default decision tree classifier and in our case also we are implementing it on decision tree classification algorithm. Decision trees with one level are most used models because they are short and contain one decision for classification.

SVM is a popular machine learning algorithm for classification, where we separate the input data using a hyper plane .in case of 2D data the hyperplane is a line that splits the input variable space. This hyperplane is optimized so that the inputs belong to any of the given classes. In our case we feed our preprocessed data to SVM and later apply OneVsRestclassifier algorithm on SVM model for multi class classification.

I applied these 3 algorithms ( Adaboost, Random forest, Support vector machines) but I'm getting good score all times with randomtrees.

So I focused more on randomtrees algorithms as it the base of our target model.

Input data is huge so I am not using K-Fold for training the inputs. Several input labels are labeled data (text data) so I converted them to numerical data as classifier gives better results with numerical data.

***Benchmark***

The ideal benchmark model will be a classification algorithm and it should classify the output as multi classes. So, the benchmark model is a Random Tree forests as random tree forests is a multi-class classification algorithm. The output label is interest level and it has 3 classes High, Low and Medium so the benchmark model will yield the F1 score of 0.33. So my developed model should score 0.33 on F1 evaluation metric

## III.Methodology

***Data Preprocessing***

The classifier algorithm takes numerical data for analysis but our input data has categorical data. We convert this categorical data to numerical data using one hot encoding.

Not all features are selected and fed to the algorithm. I excluded some using my domain knowledge such as creation date, building id etc

Features is  a column in input data and contains a list of values. This is a very important column as this has impact on interest level of user. For ex: if features have a laundry, Dishwasher, Elevator then this listing has high interest level.

So to feed this label to our classification algorithm we need to convert this into numerical data. We cannot apply one hot encoding because features column is not categorical data, it has lists within it.
So, I extracted top 10 most occurred features and made them as columns and if these are present in actual list I assigned them 1, if not I assigned 0.

Description column is also an important column as it helps us to classify listings according to interest level.
For ex: brand new apartment with pool and nearby restaurant and park. This description for a listing definitely interest users. so I extracted some valuable data from description using Natural Language Processing and converted this data into columns  such as park, pool and restaurants.

In input data there are 2 columns longitude and latitudes. I clustered them into regions, because interest depends upon the regions also. For clustering the longitudes and latitudes, I used K-Means clustering algorithm.

The price is a very important input column and makes huge impact on the interest level. The input price column has some outliers' .so I normalized the price and removed outliers

For test train split, I haven't used K-Fold because my input data is huge and it takes so much time for training.
So I used the regular Test-Train split.

In test-train split, I have taken test size as 0.25 of input data and 0.75 of input data as train data

***Implementation***

I have implemented NLP on a input feature and later preprocessing of data , I have applied randomforest tree classifier on the train data which we got from the pre-processed data. The output is a multi- class label, so I fed my randomforest tree to OneVsRestClassifier which is a multi-class classifier algorithm.

I applied Natural Language processing to extract some valuable features from the input description label. Description label is in text format and our target is to extract valuable data using NLP. I converted al the rows of description label into as single string and applied NLTK on this text. I removed stop words ,puntutaion and words less than 3 from this text using NLTK library. Now on the processed text I wrote a snippet of the code to get the most common words. From the list of common words using my domain knowledge I have taken some words such as renovated , pool etc as features

For example : if the input for description is " brand new loft apartment with pool and near by restaurents"
After applying NLP, we extract loft ,pool, restaurant features from the text and in the loft column ,restaurant column and pool column I add 1 each where as for other features I add 0

I used F1 score as the performance metric. F1 score is the evaluation metric to measure accuracy. F1 score measures accuracy using the statistics precision p and recall r. Precision is the ratio of true positives (tp) to all predicted positives (tp + fp). Recall is the ratio of true positives to all actual positives (tp + fn). F1 is a very good evaluation metric for the classification as F1 weights recall and precision equally.

There is no such complexity in code to be documented; everything is explained clearly in jupyter notebooks

***Refinement***

I actually applied several algorithms. In the process of improvement.

I actually started with support vector machines and later tried with Adaboost classifier and finally got a better value with random forest tree  Classifier. (n_estimators=50,random_state=1,max_depth=80, min_samples_split=30, min_samples_leaf=1)  are the parameters and values for our model. n_estimator,random_state,max_depth,min_samples_split have less impact . min_samples_leaf  parameter has lot of impact on the solution. Changing min_samples_leaf varies the f1 score. I fed this Random forest trees to OneVsRestClassifier as our target label is a multi-class classifier and in OneVsRestClassifier, for n_jobs parameter I used -1.n_jobs is number of jobs to use for the computation. If -1 all cpus are used. I used -1 value so that all cores are used.

I tried to change adaboost parameters learning_rate and n_estimator but this adaboost doesn't out work the random forest classifier. But I haven't changed the any parameters of SVM. My focus is more on Random Forest classifier and AdaBoost classifier

## IV. Results

***<u>Model Evaluation and Validation:</u>***

The final model in our solution is Random forest tree model an ensemble algorithm fed to OneVsRestClassifier which is a multi-class classifier. Model is aligning with my solution expectations.  My models give an F1 score of 0.71 where as my bench model F1 score was 0.33. So my initial expectation is to get a score more than 0.33. So my final model results are beyond my expectations. The final parameters are appropriate, In our case Randomforest tree classifier, parameters are n_estimators, random_state, max_depth, min_samples_split

Base estimator is decision tree classifier. I tried logistic regression also as parameter but it yields F1 score of 0.61.

The final parameters are appropriate and these are yielding the better results.

The final model is robust for the problem. small perturbations in training data or the inputs don't have much effect on the results. Although we change some inputs , that is although I shuffled the inputs and outputs I got similar F1 score

The results found the model can be trusted. I manually tested some random results and I found true in most of the cases.

With different train test splits, we get similar results

In jupyter notebook's final section you can find the results although we can

The evaluation metric is F1 score which helps us to believe the model's robustness. When the F1 score is more we get more robustness.

***Justification.***

The final results are stonger than the benchmark model reported earlier. The benchmark model yields only 0.33 F1 score where as my model yield f1 score of 0.71.the final solution was a multi class classifier where we feed the processed input.in our project processing input was so valuable as it is the core of the project.the input was processed using one-hot encoding , NLP,K-Means etc and this processed input was fed to RandomTree classifier  which later fed to OneVsRest classifer which gives us the class to which the input data belongs . the final solution was thoroughly discussed and analyzed. the final solution has solved our problem of classifying the listings according to their interest level significantly.

## V. Conclusion

***Free-Form Visualization***

The most important visualizations are provided previously.

Here I'm mentioning another visualization, which helps us to understand the features.

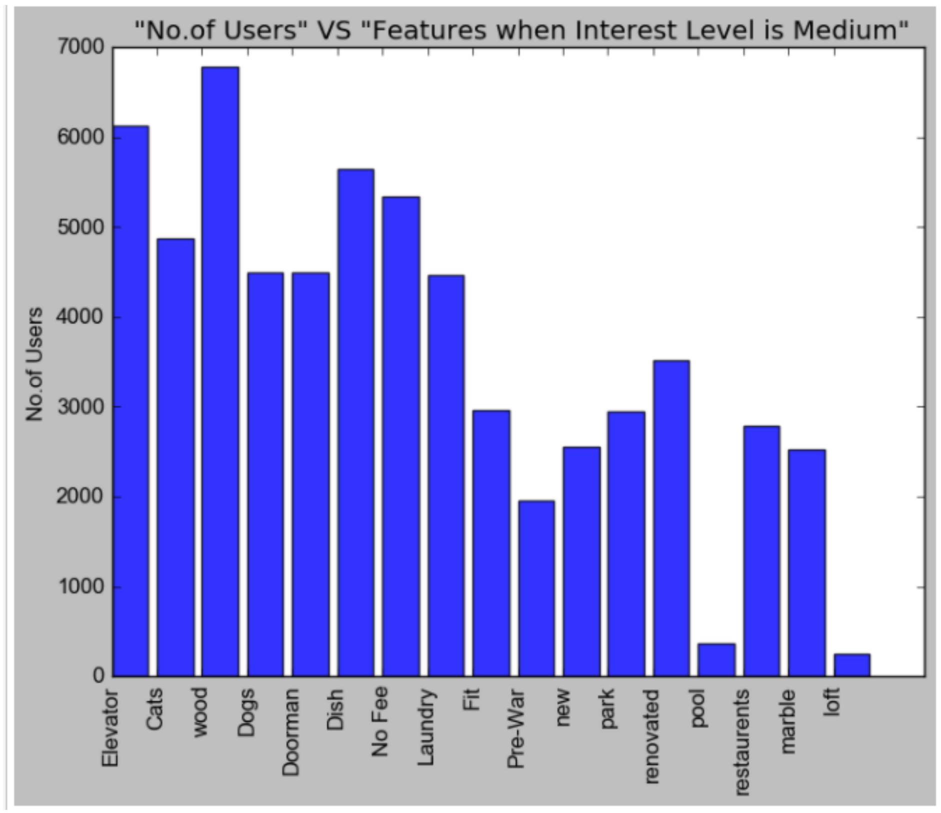
This visualization tells us how a user is interested when he is using a specific user.

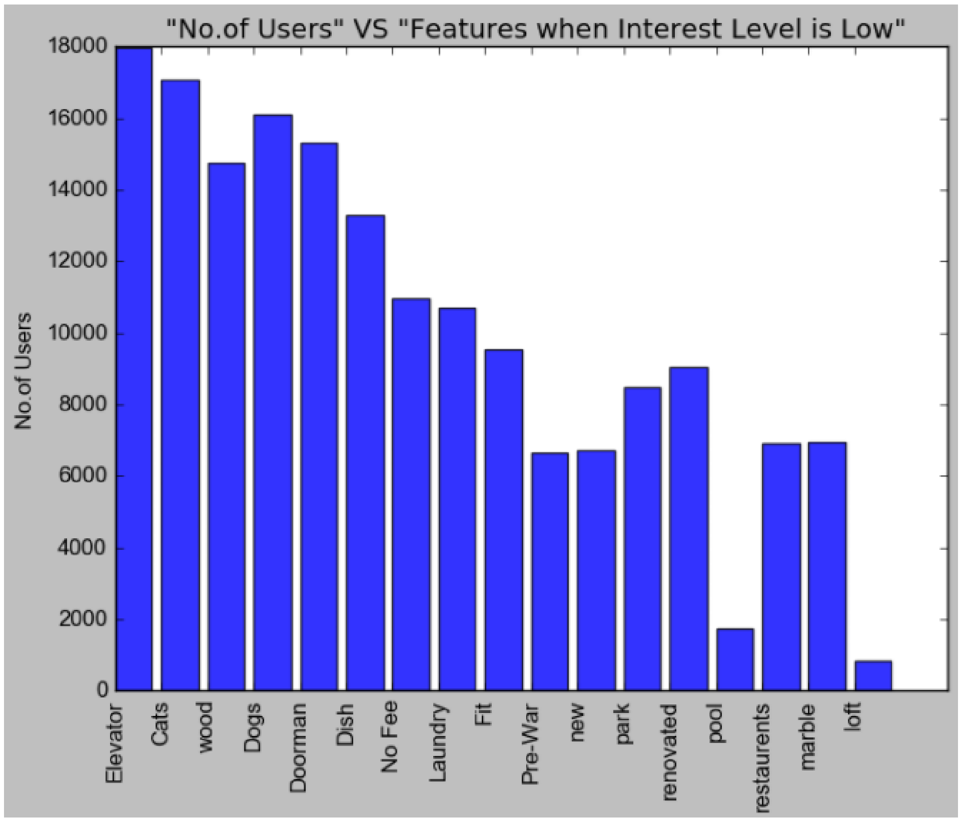
Provided 3 visualizations.

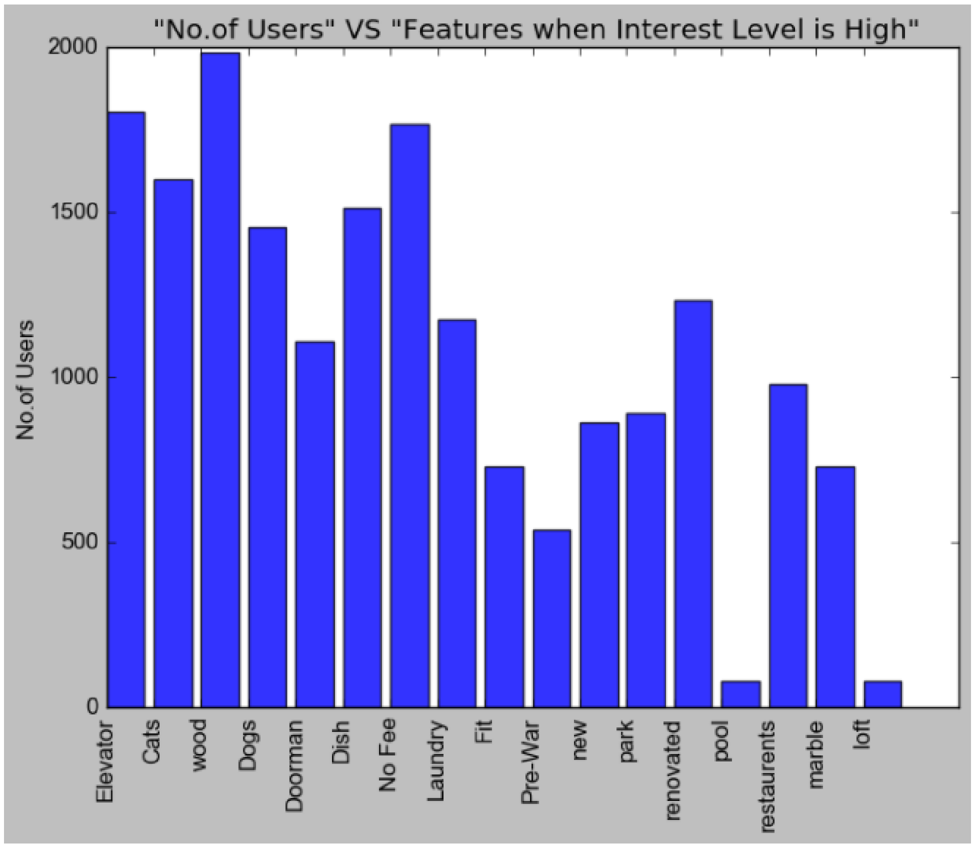
- No.of Users VS features when interest level of listing is Medium
- No.of Users VS features when interest level of listing is Low

- No.of Users VS features when interest level of listing is High

x-axis contains features.
y-axis contains No.of Users




"No.of Users" VS "Features when Interest Level is Low"
No.of Users
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
Elevator
Cats
wood
Dogs
Doorman
Dish
No Fee
Laundry
Fit
Pre-War
new
park
renovated
pool
restaurents
marble
loft


"No.of Users" VS "Features when Interest Level is High"
No.of Users
0
500
1000
1500
2000
Elevator
Cats
wood
Dogs
Doorman
Dish
No Fee
Laundry
Fit
Pre-War
new
park
renovated
pool
restaurents
marble
loft

## ***Reflection:***

The project's goal is to develop a model that predict the end user's interest level of the rental listing, so that customers don't spend much time in searching for the highly interested advertisements and also it reduces fraudulent activity on website. We have the sample data as input which has several features of listing and the interest level given by the rental hop's real estate agent manually. Now to predict the interest level of the next incoming listings, we need to apply machine learning techniques and develop a model which predicts the incoming listing's interest level. The benchmark model for our project has the F1 score of 0.33. So our target is to develop a model which has a better F1 score than 0.33 benchmark model. The inputs are given as text data, numerical data, array of text data and sentences also, out of this data using our domain knowledge we pick some data and exclude some data from out for processing. For example, I excluded building ID as it doesn't add anything to the interest level of user. The most difficult task in the project is data munging that is data processing. The difficult thing for me in the project is how to get the information out of the features column which is an array of features and description column which is text data. I applied NLP on description column and get the most useful text only and later converted this text into multiple features and applied one hot encoding on this to convert them into numerical data. All the text data in the project is converted into the numerical data using one hot encoding. There is one more complex part in the project, how to deal with longitude and latitude, geographical locations of the rental listings. I converted the longitude and latitudes into geographical clusters and these clusters using k-means clustering and used them in developing the model.to me NLP and K-Means are difficult and interesting aspects of the project. The final model which I developed has f1 score of 0.71 which is far beyond the bench mark model and this solution fit my expectations. This model is generic and this model really helps to solve the problem of identifying the interest levels of rental listings.

## ***Improvement***

Lot of improvement can be made to model. The algorithm can't be improved but the data processing can be further improved. The geographical longitudes and latitudes can be further processed and if we can get distance from the home to the nearest subway train stop , it really helps a lot. People in New York always look for easy commute and they prefer a home near to subway although it is small and has fewer amenities. This is really a huge task where I need to search for an api which takes input latitude and longitude and give distance as output. If such **api** doesn't exists then I need to develop this complex algorithm. Using longitudes and latitudes we can find the crime rate in the locality, crime rate also has the impact on the rental listing interest level, this is also a complex algorithm as we need to map the input longitude and latitude on crime rate database and get the crime rate of locality as output. If some other features are available then we can improve the algorithm more, for example if buildings age is available it can help us to find the interest level.