Johnson Xie, 301293703
Victor Chen, 301290167
Brendon Ho,301274189

CMPT 459 Final Project

Exploratory Data Analysis

Looking at the columns of the original dataset. A quick inspection can see “manager_id” and “building_id” do not look useful in training a model. This is because there is no order in the values as they aren’t categorical/continuous and it seems like records have unique “manage_id” and “building_id”. “Street_address” and “display_address” are very similar and are probably redundant. A lot of defaulted values are zero (e.g. there are lots of records where bedrooms or bathrooms are zero).

The first statistical analysis we did was compute the Pearson correlation coefficient matrix of interested features. As seen in table 1, there aren’t any significant correlations between these features. We do see the number of bathrooms and bedrooms increase as price increases which is expected.

Table 1. *Correlation coefficients of features.*

	price	bathrooms	bedrooms	interest	num_photos	num_features	nearest_subway
price	1.000000	0.667199	0.394994	-0.211442	0.148258	0.275553	-0.010857
bathrooms	0.667199	1.000000	0.454758	-0.072171	0.144160	0.213304	0.006491
bedrooms	0.394994	0.454758	1.000000	0.073145	0.090945	0.116845	-0.008345
interest	-0.211442	-0.072171	0.073145	1.000000	0.033093	0.043943	-0.009202
num_photos	0.148258	0.144160	0.090945	0.033093	1.000000	0.166775	0.017015
num_features	0.275553	0.213304	0.116845	0.043943	0.166775	1.000000	-0.007059
nearest_subway	-0.010857	0.006491	-0.008345	-0.009202	0.017015	-0.007059	1.000000

Histograms for Price, Latitude, Longitude

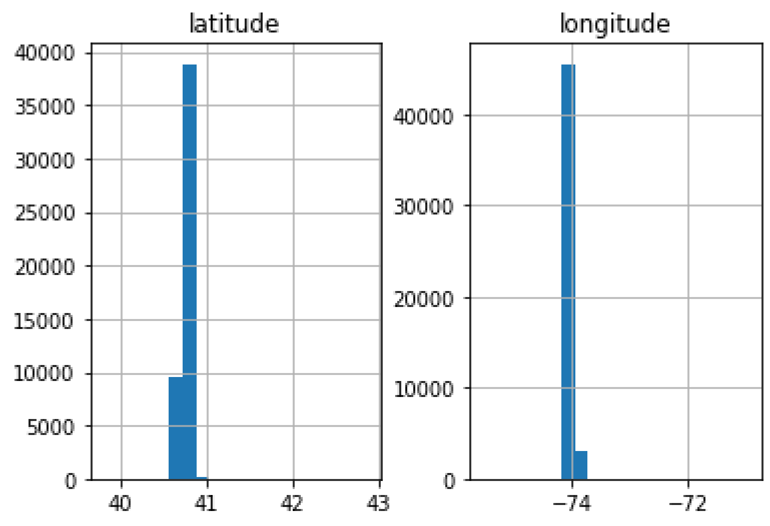


Figure 1. Histogram of latitude and longitude.

Johnson Xie, 301293703
Victor Chen, 301290167
Brendon Ho,301274189

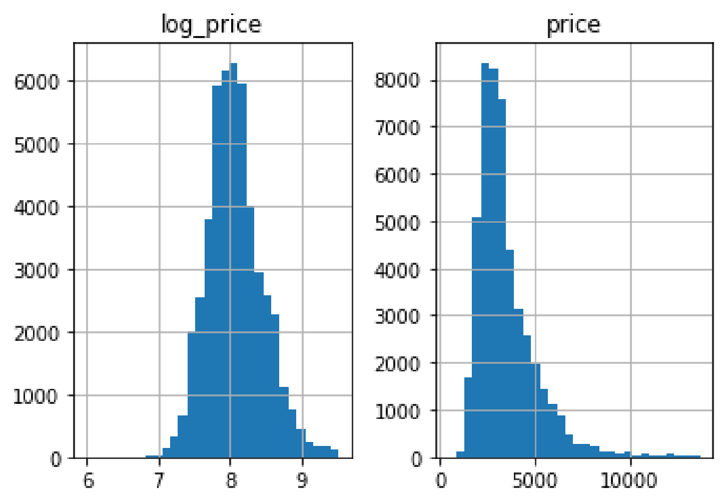


Figure 2. Histogram of logged price and price.

For the three features (price, latitude, longitude), we can see most records have the same values. This tells us the rental listings are all from similar locations and the number of rooms are same.

Histogram of bed/bathrooms

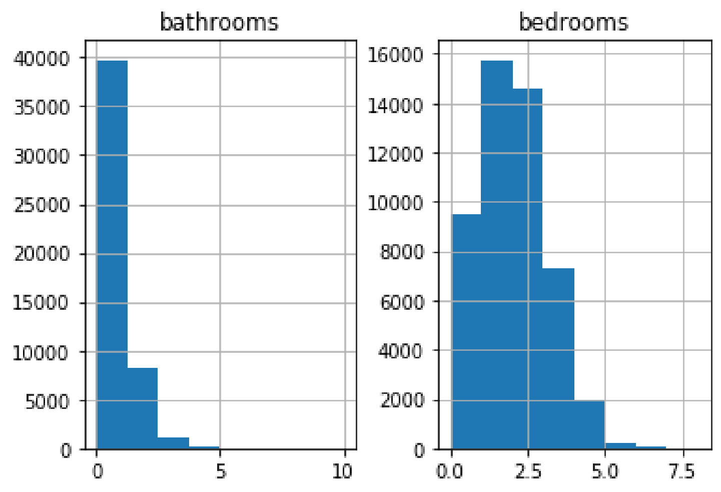


Figure 3. Histogram of bathrooms and bedrooms.

A quick inspection shows that any apartment with more than 1 bathroom is out of the norm. Most apartments have 1 or 2 bedrooms.

Johnson Xie, 301293703
Victor Chen, 301290167
Brendon Ho,301274189

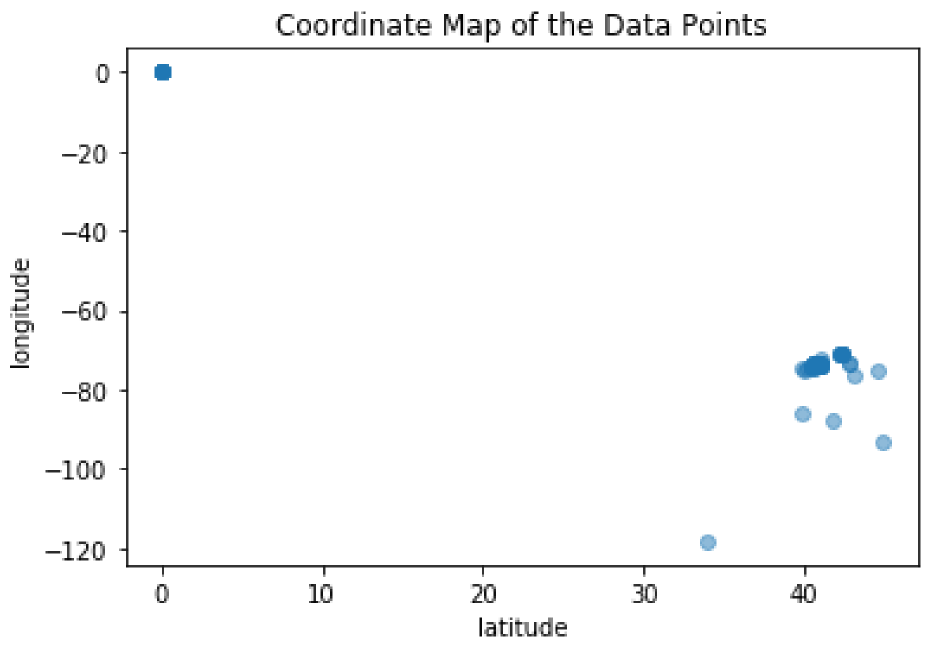


Figure 4. Coordinate mapping of rental posts

As expected from New York's rental postings, the data is mostly from the same data. We do notice there are some outliers and some data with default values at latitude/longitude = 0.

Hour-wise data

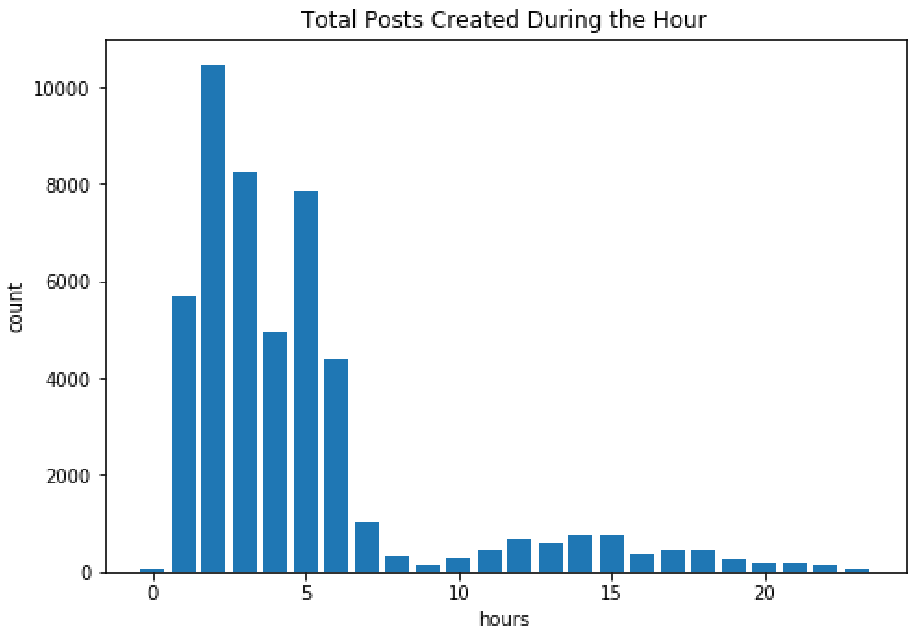
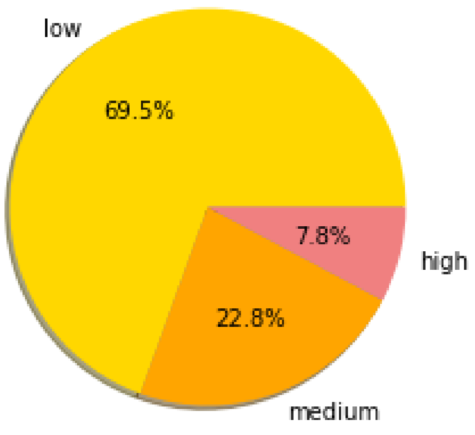


Figure 5. Posts created per hour
Top 5 busiest hours of posting: 2nd hour , 3rd hour , 5th hour, 1st hour, 4th hour
The distribution of the posts created per hour is right-skewed. This may be due to inconsistent use of 24hr standard time and AM/PM time. Another reason for the non-uniform distribution is because most posts are created during conventional work times of 9am-5pm.

Data Visualization Regarding the Target Variable

The first thing we did was count the total numbers of low/medium/high interest postings.



From figure 6, we see only 7.8% of the data has high interest. This may lead to overfitting when training a model and trying to detect which postings will be of high interest. Given the distribution of our target class, a classifier that can focus on a minority class may give the best results.

Figure 6. Distribution of interest level

Afterwards, we plotted some features to visually compare values of differing interest levels.

This figure combines bathrooms and bedrooms into rooms. Then the figure shows how common the number of rooms are for each interest level. We see the number of rooms does not necessarily influence the interest level.

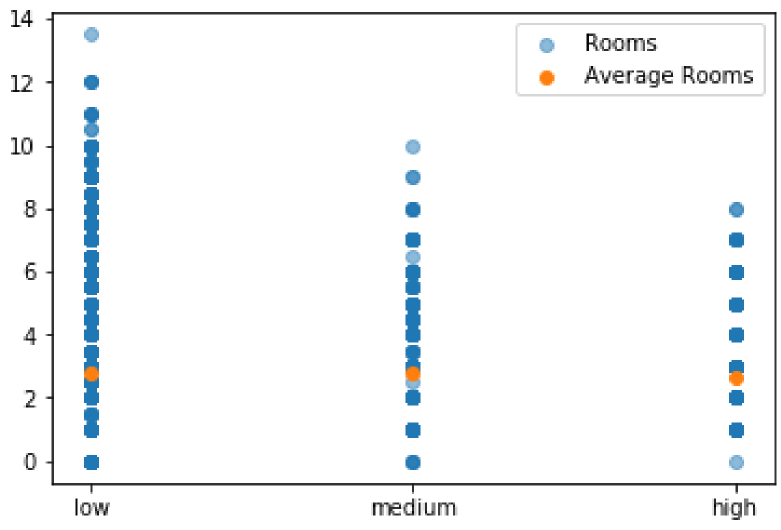


Figure 7. Distribution of rooms with respect to interest levels.

One last visualization we did with interest level, is finding out the ten most popular neighbourhoods and then calculating the average interest level.

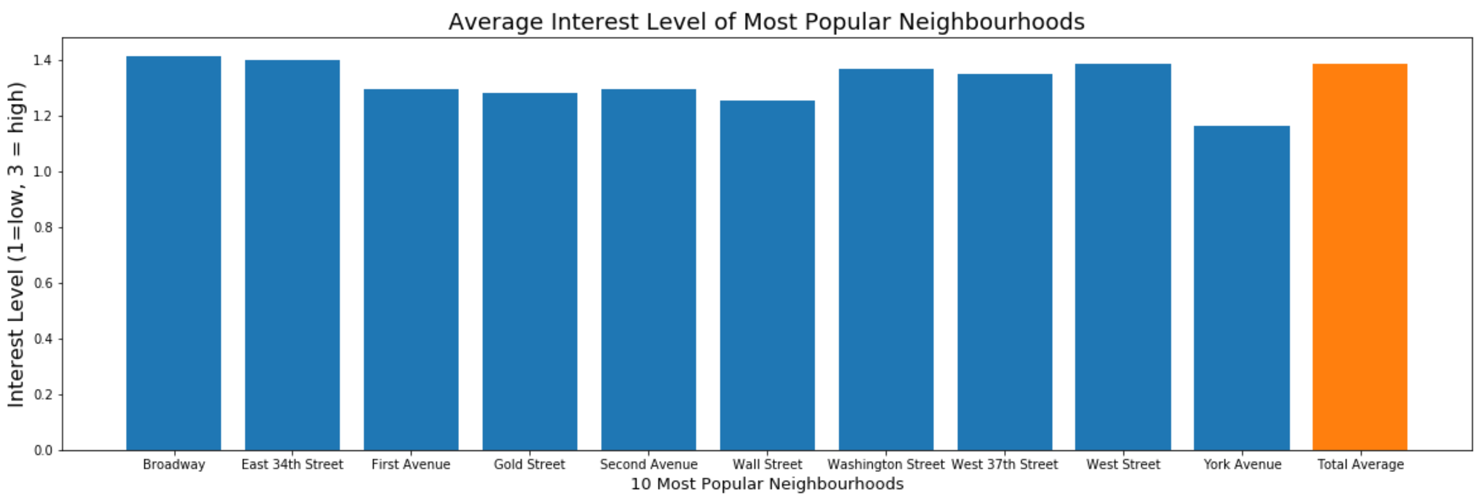


Figure 8. Average Interest Level of ten most popular neighbourhoods

From the figure above, we can see that popular neighbourhoods don't yield higher interest levels. There are some disparities between interest levels among different neighbourhoods.

Data Preprocessing

Missing Values in Dataset

For this dataset, values are assumed to be missing if it's empty or has the default value of zero. We assume no apartments can have zero bedrooms or zero bathrooms. For latitude and longitude, zero is a valid value but anything not in New York is an outlier.

Table 2. *Missing values of every feature*

Feature	**Missing Value Count**
bathrooms	9475
bedrooms	313
building_id	8286
date created	0
description	1446
display_address	135
features	3218
latitude	12
listing_id	0
longitude	12

manager_id	0
photos	3615
price	0
street_address	10
interest_level	0

There are multiple features with varying amounts of missing values. Some features will be near impossible to impute (e.g. "photos" and "building_id"). For bathrooms and bedrooms, We will impute those values through linear regression. For the other features, We can either remove the row with a missing value or remove the column itself if it will probably not affect training the classifier.

Outlier Detection

Outlier detection was done on features that were either numerical or categorical. Features that consist of strings did not have outlier detection done as it was too difficult to know the exact restrictions to consider points as norm or outlier. Outlier detection on the feature "photos" was also not done as there is no simple technique to go through each photo and decide if it's an outlier or not.

We start outlier detection on the more simpler features: "date created" and "interest level".
For "date created", records are outliers if any dates deviate too much from each other. We planned to do more complex algorithms to check for outliers, but we checked the earliest and latest date created and the time range was relatively small. We decided that there were no outliers for date creation.
Next, records were outliers if their interest level was not one of the three values (low/medium/high).
No records were returned as outliers.

With latitude and longitude. We used density-based scan to cluster the data points together and anything not part of the node clusters are deemed as outliers. The biggest challenge was adjusting the parameters to get the desired result. The parameters were tuned so that everything outside New York were outliers.

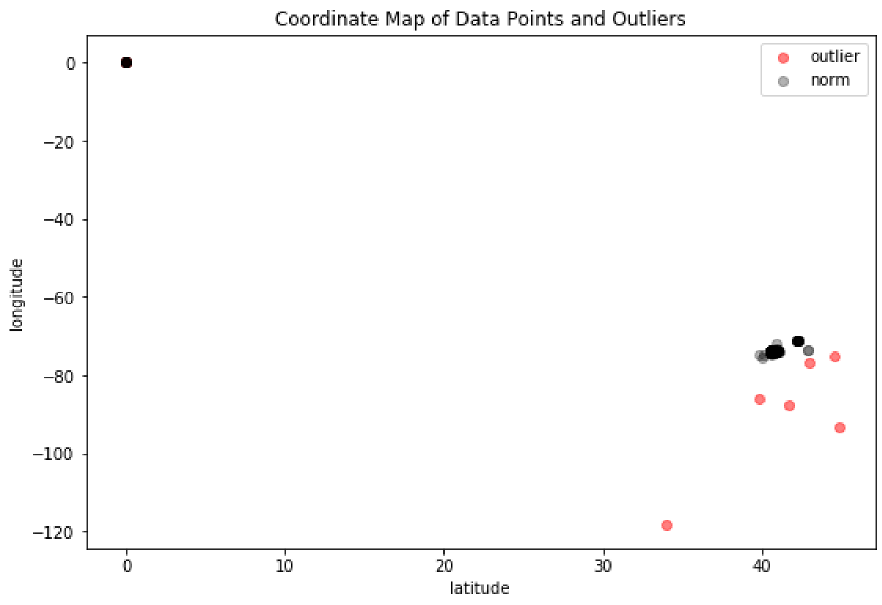


Figure 9. Outlier detection of rental posting locations

The data that comes up as a red dot on this figure will be removed. We got 10 outliers here if we don't include values with (0,0) latitude/longitude as outliers.

Outlier Detection for Price

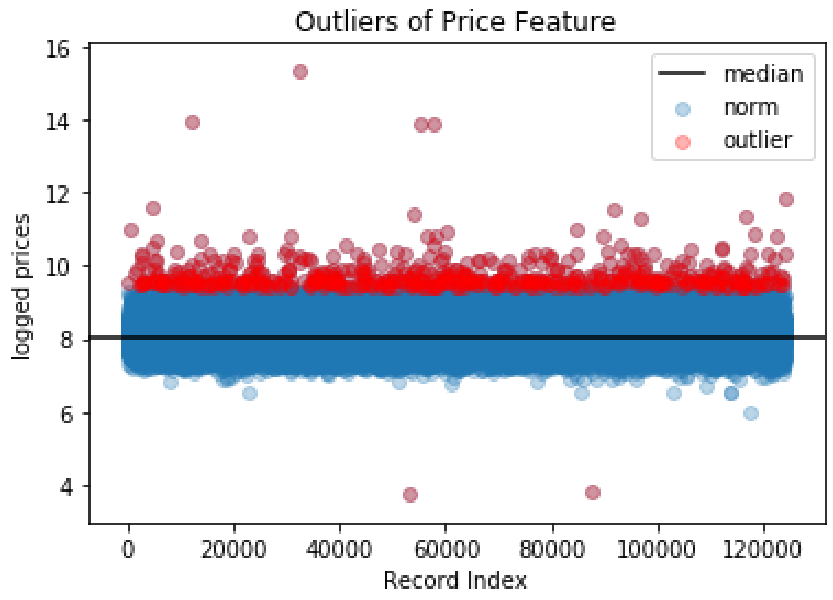


Figure 10. IQR outlier detection for price.

Regarding the distribution of price data, the extreme max of the data is much larger than the average price. We used an interquartile range (IQR) method for detecting outliers because using quartiles will be more effective when the prices can be magnitudes larger than the mean. The convention is to use 1.5*IQR, but this value returns over 5% of the data as anomalies. We decided to go with 5*IQR which returns about 585(1.2%) as outliers.

For the features "bathrooms" and "bedrooms", I used z score method to detect outliers. This method requires the data to be roughly normal distribution and when the mean of the data is an effective point. The max bathroom and max bedroom value isn't magnitudes larger than the mean value so this method was chosen. The convention is that if the data is normal distribution, 99.7% of the data is within 3 standard deviations of the mean.

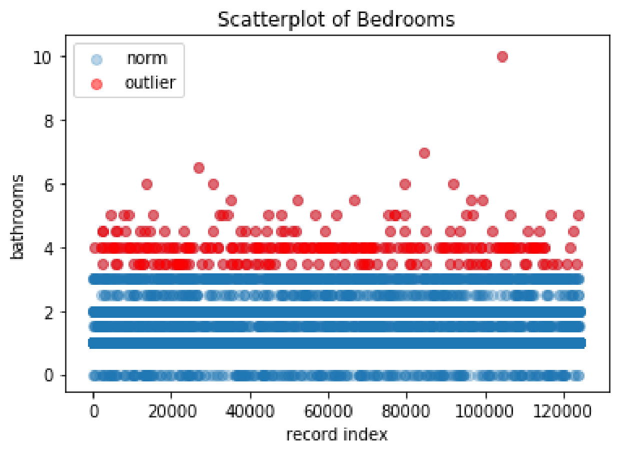


Figure 11. Outlier detection on bedrooms.

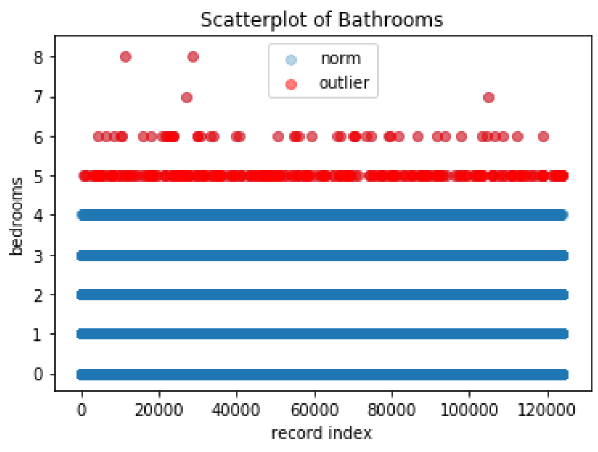


Figure 12. Outlier detection on bathrooms

For bedrooms, instead of three standard deviations from the mean, we chose four so that any rental listing with more than three bathrooms is an outlier. This returned 290 records as outliers.

For bathrooms, we just used the recommended three standard deviations away from mean as outlier benchmark. This means any rental posting with five or more bathrooms as an outlier; this returned 297 records as outliers.

Summarizing the features I did outlier detection on and the number of outliers in table 3.

Table 3. *Outlier count of interested features.*

Feature	**Outlier Count**
bathrooms	297
bedrooms	290
date created	0
latitude	10
longitude	10
price	585
interest_level	0

Imputation/Correction

Imputation was done for three feature's (display_address, bedrooms, bathrooms) missing values. For display_address, we merely copied the street_address onto display_address since the two values are similar. If both addresses are missing for that record, we remove it from the dataset (there were 10 of these records). For bathrooms and bedrooms, I interpolated the missing values by using linear interpolation with respect to the price. This will change rental listings with 0 bedrooms and 0 bathrooms to another value depending on its price.

Feature Extraction (Image) - Brendon Ho

For feature extraction for the images, we used the images inside the images_sample.zip to extract and transform our data which was then put into a Pandas DataFrame object. We first used the zipfile library to access each photo in their respective folders. For each photo, we used the process of binarization to transform a photo into grayscale to make it so we can use data mining algorithms in the future. An example is shown below.

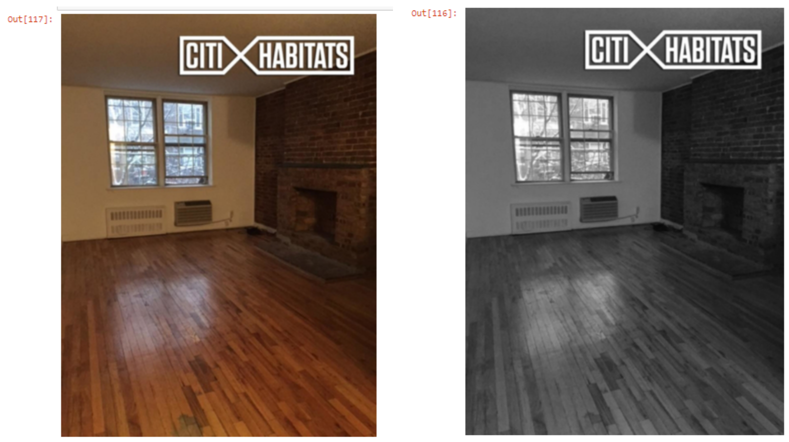


Figure 13. Example of greyscaling the pictures

As there are some listings with more than one picture and some with no pictures, we stored the mean of the array values for each listing_id and for those that have no pictures we filled it in with a NaN value.

Feature Extraction (Text) - Johnson Xie

We focused on extracting features from the description and the features column. We decided that those two were the two text-heavy categories that were necessary towards predicting the interest levels of apartments. We needed to transform the data into workable data. We first transformed the data into a list of strings for each row.

By calculating the TFIDF of features and descriptions, it would give the classifiers a better understanding of what words were important towards the interest level of the apartments. It will give each of the rows a weight on each of the words as shown below. It is normalized from 0 to 1. We also got rid of “stop words”, common words that add no meaning to the actual result. We then summed up the TFIDF values for each building listing so that it could be used in classifications. We needed to make it a numeric value for the classifiers.

Classifiers - Johnson Xie, Victor Chen, Brendon Ho

For all the classifiers we used, we achieved the best scores with the same set of features of price, bedroom, bathroom, latitude, longitude, day of week rental created, description tf idf, feature tf idf, num of features, and num of photos. We attempted removing features, but it just made our scores worse. In the end, giving the classifiers most of the features that we had gave the best scores.

Author: Brendon
Brendon: Decision Trees, Stacking Classifier
Milestone 2:
```
from sklearn.tree import DecisionTreeClassifier
```
Milestone 3:
```
from sklearn.ensemble import StackingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn,linear_model import LogisticRegression
```

Johnson:Logistic Regression, Bagging Classifier

Milestone 2:
from sklearn.linear_model import LogisticRegression
Milestone 3:
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier

Victor:SVM, Gradient Boosting Classifier
Milestone 2:
from sklearn.svm import LinearSVC
Milestone 3:
from sklearn.ensemble import GradientBoostingClassifier

Optimization of Classifiers:

Decision Trees: Changed min_samples_leaf to 5 (default 1), increased max_depth to 5 (default 1), and changed the criterion to be entropy (information gain) instead of gini index.
Accuracy (cross-validation):0.71
Kaggle:1.3

SVM: Set dual boolean parameter to false (n_samples > n_features), excluded ‘description’ feature, added more features (closest subway entrance). Had to use linear SVM due to long computation times of other SVM kernels.
Accuracy (cross-validation):0.69
Kaggle:0.875

Logistic Regression: Removed features like manager_id, changed solver to liblinear method, change weight_class = ‘balanced’ (takes into account the frequency of the features and attempts to balance it with infrequent feature)
Accuracy (cross-validation):0.68
Kaggle:1.1

Stacking: n_estimators from 10->100, min_samples split from 2 ->3, and max depth 10->15. For logistic regression parameters set random_state to 0, liblinear method, multi_class to auto and max_iter to 2000
Accuracy (cross-validation):0.71
Kaggle:0.98

Bagging: Removed low importance values (day, weeks and bathrooms), added features (number of photos, number of features), n_estimators -> 200, min_sample_leaf ->20
Accuracy (cross-validation):0.72
Kaggle:0.957

Gradient Boosting: Changed learning rate from 0.1 (helps find the more optimal weight), set warm_start parameter to true, subsample 1->0.8, removed tf_idf of features and increased k-fold validation from 5-fold to 10-fold
Accuracy (cross-validation):0.70
Kaggle:0.795

Table 4. *Results for various classifiers*

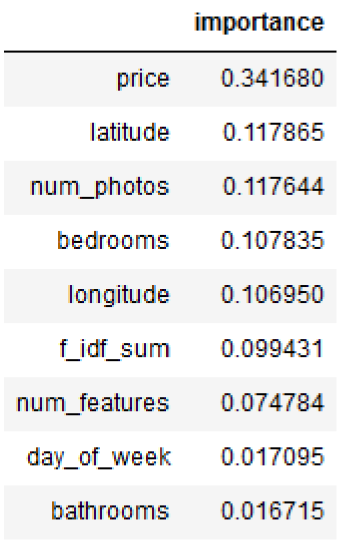
	Decision Tree	SVM	Logistic Regression	Stacking	Bagging	Gradient Boosting
Validation Score	0.71	0.69	0.68	0.71	**0.72**	0.70
Kaggle log loss	1.301	0.875	1.103	0.980	0.957	**0.795**

In terms of accuracy, Bagging yielded the highest result with an validation score of 72% and as for the lowest log loss (Kaggle score), Gradient Boosting had the lowest score of 0.795

Lessons Learned - Johnson, Victor

The top 3 most relevant features was the price, latitude, and number of photos. I derived the importance based on the feature importance function in the random forest classifier. The feature importance function in the random forest classifier is based on the Gini index of each feature. The importance value increases according to how much the feature decreases the impurity. So, price has the lowest gini index and decreases the impurity of the decision more than the other features.

Table 5. *Feature importance of features used for classification.*



	importance
price	0.341680
latitude	0.117865
num_photos	0.117644
bedrooms	0.107835
longitude	0.106950
f_idf_sum	0.099431
num_features	0.074784
day_of_week	0.017095
bathrooms	0.016715

We created one additional feature using NYC Transit Subway Entrance And Exit Data. We calculated each rental postings' nearest subway entrance using the haversine formula. It was very expensive to compute the nearest subway to each posting as it required a lot of memory and computations. After creating this feature, we compared the classifiers' accuracies with and without this additional feature; the results were mostly the same and in some cases it even lowered our accuracy.

Our gradient boosting classifier did significantly better than our other classifiers. Gradient boosting iteratively improved upon each decision tree and created the best predictions from the result. Our other classifiers did not iteratively improve upon each other like gradient boosting did. Our random forest classifier only reduces bias and overfitting by independently creating decision trees on random sets of the training data and picking the best decision tree. Our stacking classifier, we used a random forest classifier with logistic regression. Logistic regression with random forest was a bad combination of classifiers and logistic regression didn't improve the scores for the random forest classifier. We achieved a Kaggle log likelihood of 0.95 on a single random forest classifier and 0.98 on stacking random forest with logistic regression. We believe that logistic regression actually worsened our scores. The other classifiers of decision trees, logistic regression, and SVM, were too simple and were very susceptible to overfitting (decision tree), or forcing linear patterns (log regression, SVM).

The simple classifiers usually did better in terms of performance. Normal SVM did poorly in terms of speed because it had to compare every pair of points. However, logistic regression and decision trees did well in terms of speed. Logisticregression did the best because it's simple and finds a best fit line for the data provided. It did the best in terms of speed because it assumes a linear relationship. The more complex classifiers we used, such as random forest, did poorly on performance because it had to run numerous decision trees.

Overfitting was definitely a problem with the simple classifiers. Decision trees, logistic regression, and SVM were very prone to overfitting and underfitting. We attempted to solve overfitting on these simple classifiers by using K-folds cross validation to ensure that we don't commit too hard on patterns that don't exist. The ensemble classifiers experienced less overfitting, especially the ones that reduce variance. For example, gradient boosting had many parameters to prevent overfitting like subsampling. We also compared the test scores and the training scores to ensure that they were not too far apart as that is a sign of overfitting.

Recommendations for Rental Owners - Johnson, Victor

In the table below, we found the average values of low interest and high interest data.

Table 6. *Average values of low and high interest features*

features	**Low Interest**	**High interest**
price	3700.971	2667.609
bathrooms	1.210	1.112
bedrooms	1.824	1.955
num_photos	5.493	5.73
nearest_subway	0.484 km	0.391 km

An owner looking to increase the interest level of their rental listing should focus on minimizing the price which seems to have the strongest impact on interest levels. Price has the strongest negative correlation looking at table 1 and higher interest listings tend to have an average lower price of $1000. Looking at the correlation matrix from table 1, we can see price has the strongest relative significance with interest at -0.21142. Even though this value shows low correlation, it is relatively high compared to the other features. From the table above, we can see the average price of high interest listings is almost $1000 less than low interest. Also, with the feature importance table shown in Table 5, we can see that the importance of the price feature is three times higher than any of the other features we had.With the difference in average price in interests and the negative correlation with interest, we can conclude that a lower price will increase the interest level of the rental listing. We also see the distance to the nearest subway is 20% closer for high interest, but 93m is not a significant distance.

New rental property owners should have rentals with two bedrooms while minimizing the price. Interest level and bedrooms seem to have a positive correlation according to the matrix on Figure 1, and the feature importance value is quite high, (relatively similar to the second highest feature importance). As shown on the high interest, the average bedroom count is 1.955. Therefore, we can make an assumption that high interest rental listings are commonly 2 bedroom listings. The number of bathrooms is not important to the interest level as long as there is at least one.. The most popular neighbourhoods (street address, display address) had generally the same interest levels as the rest of the dataset, so it didn't really matter what neighbourhood the listing was in. Therefore, we believe minimizing price while having two bedrooms will maximize the interest level of the rental property.

Existing property owners should definitely highlight the price as it has the most influence to interest levels. As shown in the correlation matrix in Figure 1, it has the largest (negative) correlation value and it has the highest feature importance value (lowest gini index). The number of bedrooms can also be highlighted to supplement the increase in interest for the rental listing. However, the bedrooms are fixed and are hard to adjust after the property is built, so the price should be the main attraction highlighted to an existing rental listing.

Reference List

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.22.2 documentation. (2020). Retrieved 20 March 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

3.2.4.3.5. sklearn.ensemble.GradientBoostingClassifier — scikit-learn 0.22.2 documentation. (2020). Retrieved 20 March 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier

(2020). Retrieved 4 March 2020, from https://data.ny.gov/Transportation/NYC-Transit-Subway-Entrance-And-Exit-Data/i9wp-a4ja

derricw. (2020). Fast Haversine Approximation (Python/Pandas). Retrieved 4 March 2020, from http://stackoverflow.com/a/29546836/2901002

Explaining Feature Importance by example of a Random Forest. (2020). Retrieved 21 March 2020, from https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random-forest-d9166011959e

sklearn.ensemble.BaggingClassifier — scikit-learn 0.22.2 documentation. (2020). Retrieved 22 March 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

sklearn.ensemble.StackingClassifier — scikit-learn 0.22.2 documentation. (2020). Retrieved 23 March 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html

sklearn.linear_model.LogisticRegression — scikit-learn 0.22.2 documentation. (2020). Retrieved 21 March 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

sklearn.svm.LinearSVC — scikit-learn 0.22.2 documentation. (2020). Retrieved 2 March 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

sklearn.tree.DecisionTreeClassifier — scikit-learn 0.22.2 documentation. (2020). Retrieved 1 March 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html