Two Sigma Connect: Rental Listing Inquiries

https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries

Cathy Qian, 2017

Outline

1. Problem description
2. Train Data analysis
3. Machine Learning Outline
4. Feature engineering
5. Machine learning Results Summary
6. Take-home message

1. Problem Description

- Predict the popularity of an apartment rental listing.
- Figure out key features responsible for the popularity of apartment rental listings.

training dataset:
9352 entries, 15 columns (with interest_level as target)
testing dataset:
74659 entries, 14 columns (without interest_level).

15 features

1. bathrooms: number of bathrooms, float
2. bedrooms: number of bedrooms, int
3. building_id: the id of the building, string
4. created: date and time when the post is created, string
5. description: description of the apartment, string
6. display_address: display address of the apartment in the posting, string
7. features: a list of features about this apartment, string
8. latitude: latitude of the apartment, float
9. listing_id: listing id of the apartment, int
10. longitude: longitude of the apartment, float
11. manager_id: id of the manager of the apartment, string
12. photos: a list of photo links. string
13. price: in USD, int
14. street_address: street address of the apartment, string
15. interest_level: this is the target variable. It has 3 categories: 'high', 'medium', 'low'

2. Train Data Analysis

Numerical features:

bathrooms: [0, 10], mean = 1.2

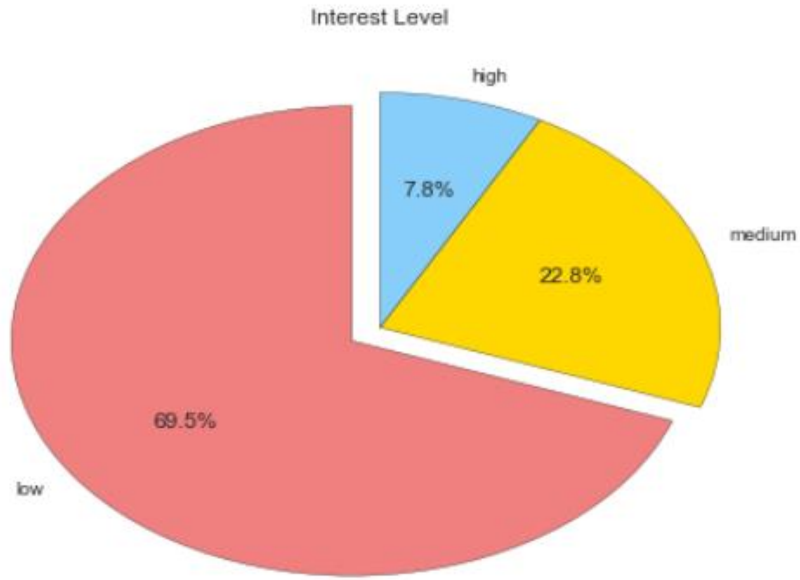
bedrooms:[0, 8.0], mean = 1.5

latitude:[0.000000, 44.883500]

longitude:[-118.271000, 0.00000]

listing_id:[6811957, 7753784], unique for each listing

price:[43, 4490000]



Texts:

created → extract day, hour, week of day

photos → extract number of photos

features → extract length of features or key words

description → extract length or key words

building_id: there're multiple listings with the same building_id

manager_id: there're multiple listings with the same manager_id

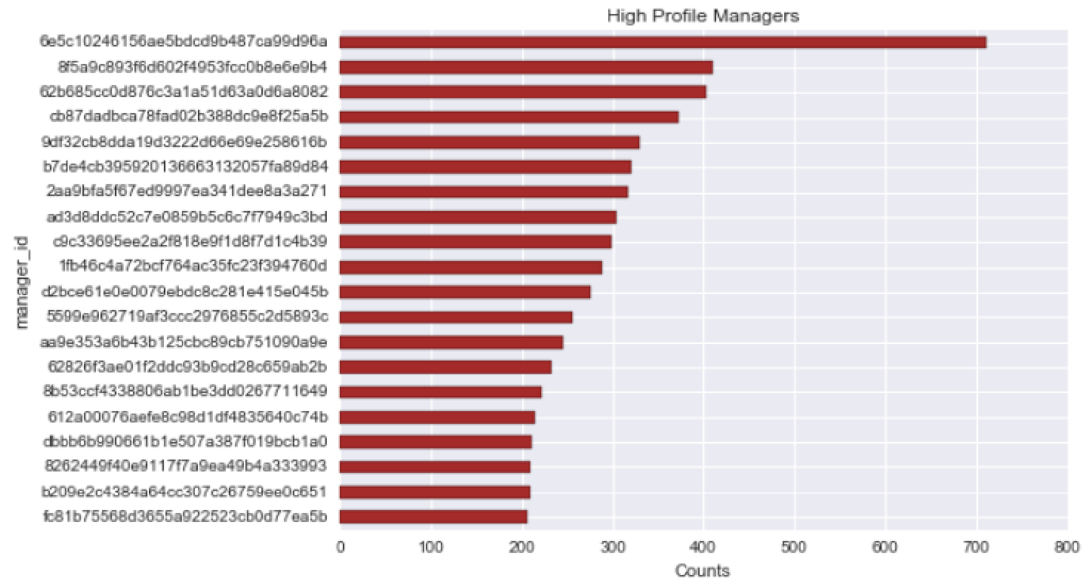
display_address: there're multiple listings with the same display_address

street_address: there're multiple listings with the same street_address

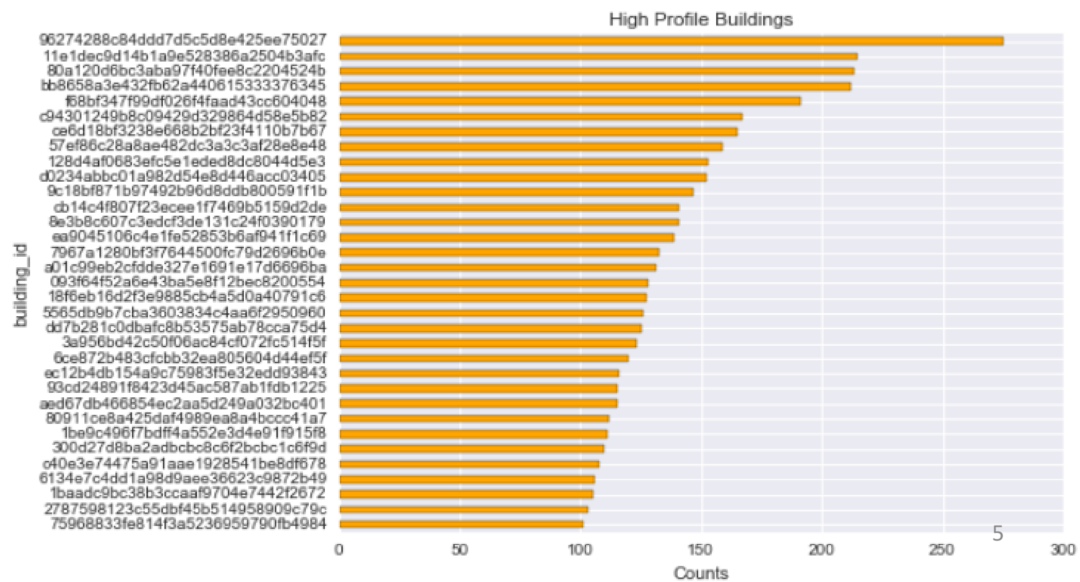
interest_level: distribution shown above

2. Train Data Analysis

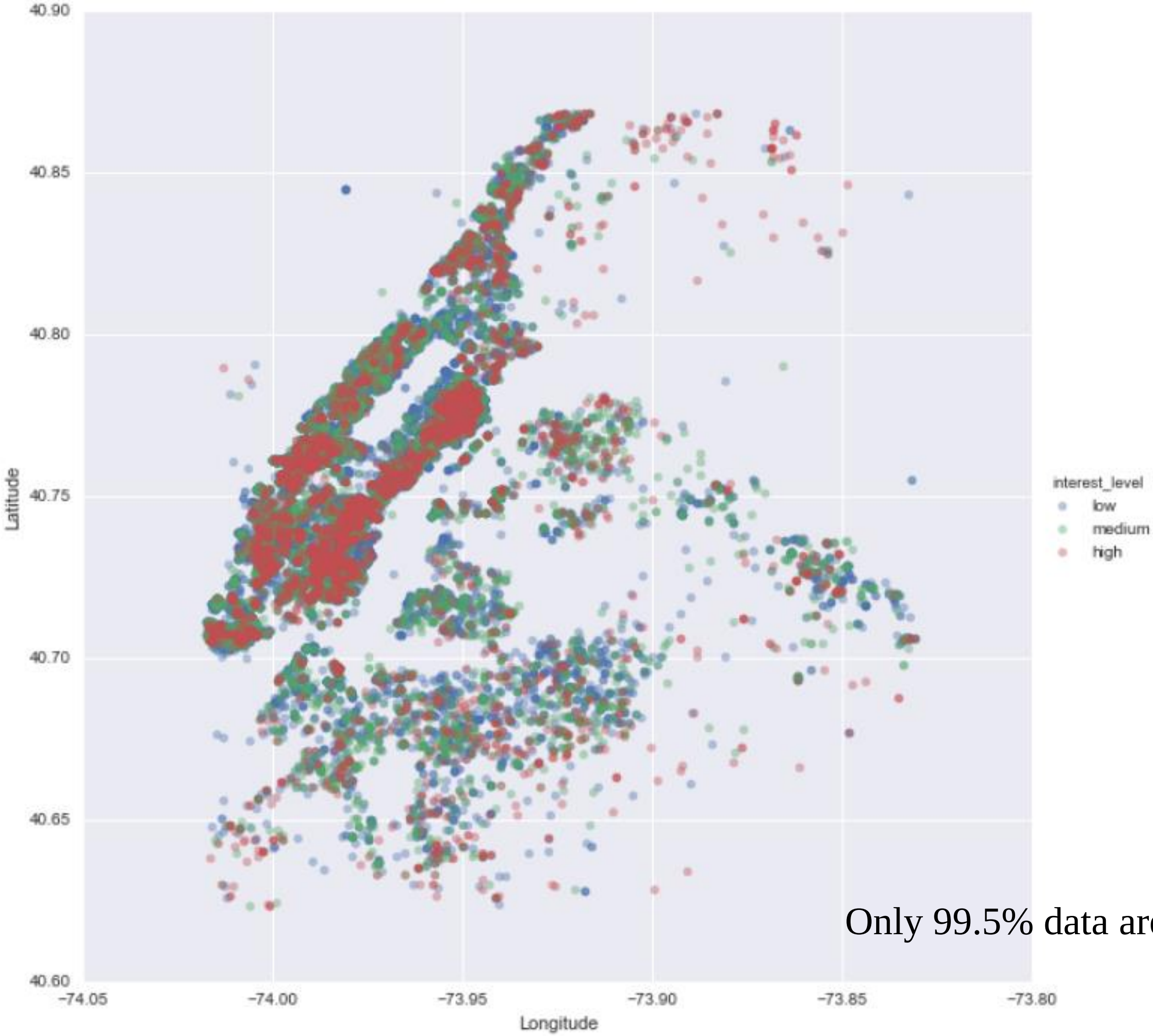
manager_id with counts over 200



building_id with counts over 100

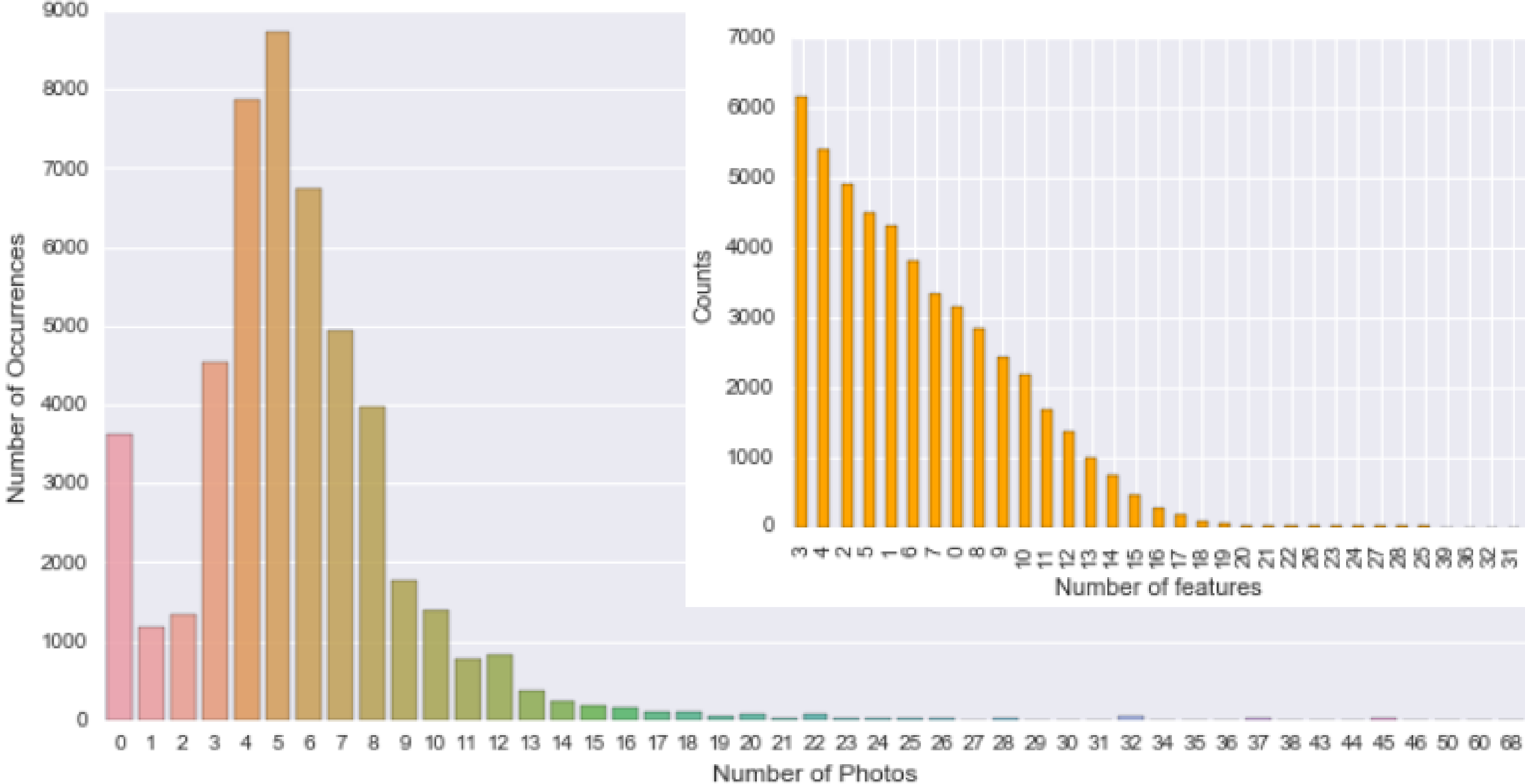


2. Train Data Analysis



Only 99.5% data are shown.

2. Train Data Analysis



Skewed distribution

3. Machine Learning Outline

Data Cleaning

- Remove outliers(ML_0)
- Do nothing *(similar results as ML_0)*

Feature Engineering

- Naïve feature engineering (FE_0)
- get_statistics (FE_1, FE_4, FE_5)
- CV_statistics (FE_2)
- Clustering (FE_3)
- Factorization (FE_6)

Machine Learning

- Logistic regression
- Random Forests
- XGBoost
- LightGBM

4. Feature Engineering

- Naïve feature engineering
  - total number of rooms
  - average price per room
  - number of photos
  - length of features
  - number of words in description
  - created day, month, hour

- get_statistics

  group the dataframe by group column (manager_id, building_id), then calculate the count, mean, std, median, max, min of the target column (bathrooms, bedrooms, latitude, longitude, price etc) feature

- cv_statistics

  calculate building_level = {manager_id: low_count, medium_count, high_count} Then update three new features: low_count%, medium_count% and high_count% for both train_df and test_df. If this manager_id only shows up in train_df but not test_df, nan is added.

  encode categorical values into numerical values between 0 and n_classes - 1

- Clustering

  categorize 'features' by the top ten features

  separate Friday from the rest of the days and clustering the time of day into four categories

- Factorization

  Encode input values as an enumerated type or categorical variable (i.e., 0, 1, 2,…..) and return the unique values

5. Machine learning Results Summary

Ref on logloss: http://www.exegetic.biz/blog/2015/12/making-sense-logarithmic-loss/

Log loss on X_validation/lb	Logistic Regression	Random Forests	XGBoost (without NAN)	XGBoost (withNAN)	LightGBM (withnan)	Notes
FE_0	0.70304 /0.72024	0.62085 /0.63383	0.631709 /0.62627	0.628339 /0.62627	0.6068221 /0.60777	Too simple FE
FE_1	0.667379 /1.00415	0.6153 /1.4447	0.572166 /1.15843	0.582721 /1.07354	0.550506 /1.16804	Overfitting?? Or data leaking
FE_2	0.459439 /test data contain NAN	0.4482 /test data contain NAN	0.408885 /0.87942	0.407328 /0.87324	0.40023587 /0.86359	Data leaking??
FE_3	0.795639/0 .82395	0.8112 /0.82395	0.796590 /0.79797	0.796626 /0.796626	0.79183 /0.79195	Bad performance
FE_4	0.70040 /0.94479	0.61523 /1.35541	0.603335 /1.02034	0.600084 /0.94132	0.5619336 /1.17163	Overfitting?? Or data leaking
FE_5	N.A.	N.A.	N.A.	0.592260 /1.05636	0.534319 /1.27489	Overfitting?? Or data leaking
FE_6 (with listing_id)	0.646260 /0.64718	0.57552 /0.58216		0.550675 /0.55800	0.538097 /0.54073 0.53814(replace NAN with -1)/0.54114	**Best result**

6. Take-home message

Lessons learned from this competition:

1, Naïve Bayes and SVM performs really bad. Maybe because naïve assumption doesn't hold while SVM is good for "linear" separation which may not be the case here.

2, Merge test and train data before doing feature engineering may give better results than doing feature engineering solely on train data because this gives more data.

3, Tree based models like xgboost and lightGBM are not sensitive to features scales, so feature scaling is not needed.

4, LightGBM performs better and faster than xgboost in all investigated cases.

5, listing_id is important for getting small logloss value.

6, Among all tried ML algorithms, only XGBoost and lightGBM can handle NAN value.

7, More feature engineering, single algorithm training and ensembing/stacking are needed to decrease the logloss and increase the prediction accuracy given enough time. There is leaking in the dataset, which is subjected to further investigation.