JUNE, 2020 • CAPSTONE PROJECT

Airbnb Price Prediction

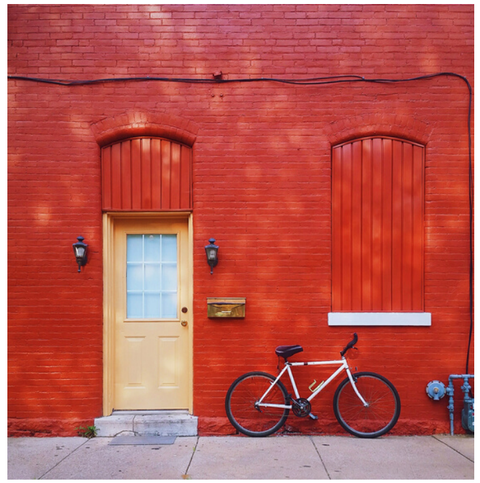
Aisha Awan
Daniela Collaguazo
Darien Thayne
Jessica Saenz
Mulyono Kertajaya

Georgetown Data Science Certificate

GEORGETOWN UNIVERSITY
School of Continuing Studies

airbnb

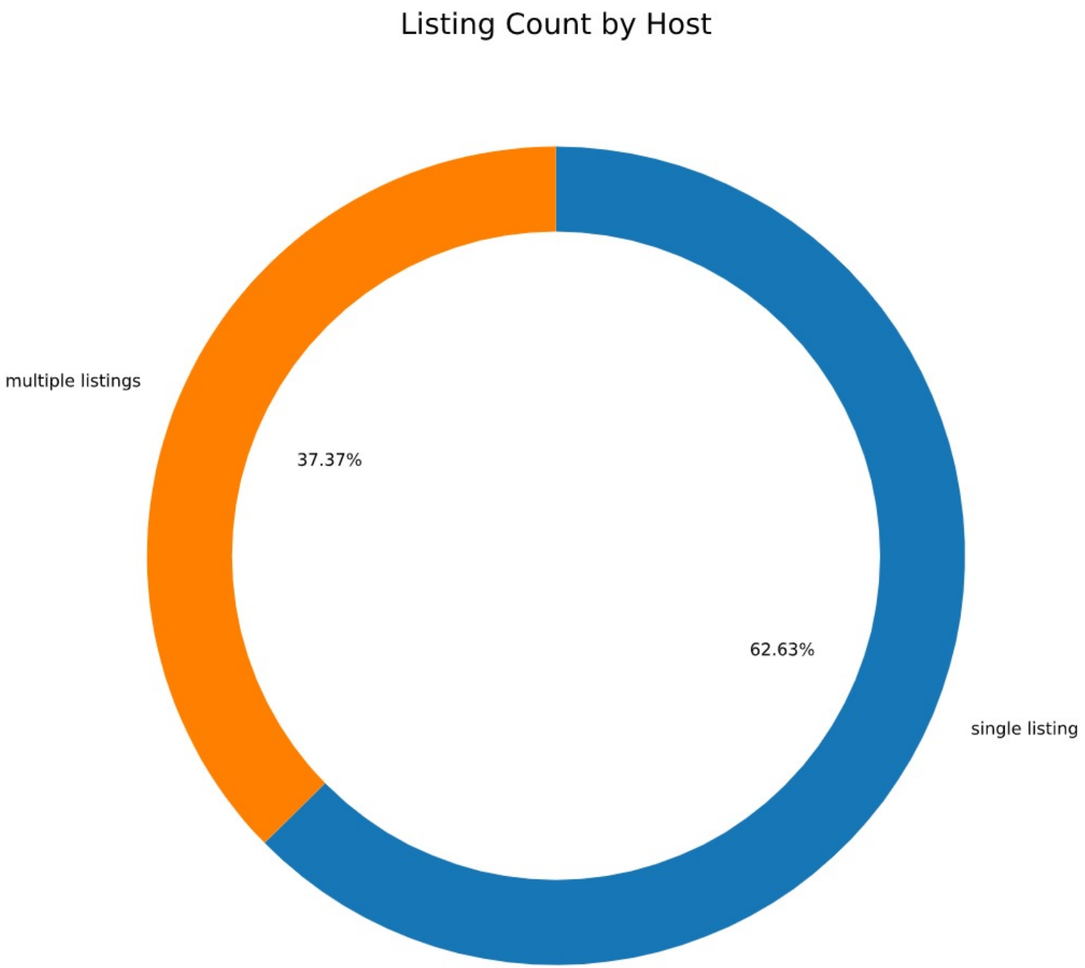
Jun, 2020 • Airbnb Prediciton Price



Airbnb began in 2008, it is a U.S based online marketplace that allows people to rent their properties or spare rooms to guests around the world.

The company’s business model is to act as a broker that receives commission from each booking from both hosts and guests.

Airbnb listing is now filled with professional hosts



We are trying to help new individual listers

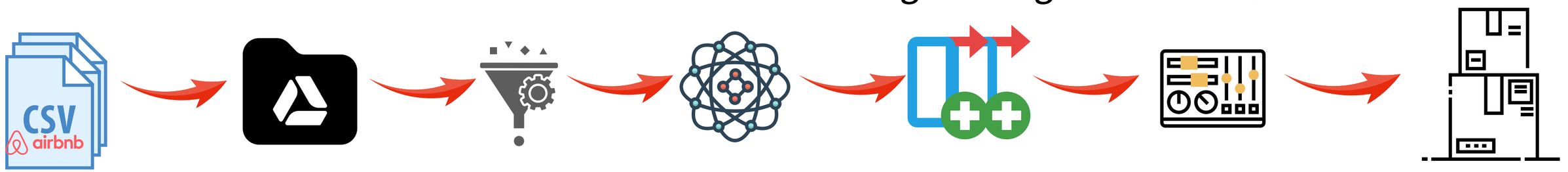
Hypothesis

By understanding the type of property, location, booking availability per year, together with an analysis of the accommodations, we will be able to predict a price per night that will optimize revenue for a new Airbnb host.

The Architecture

BUILD PHASE

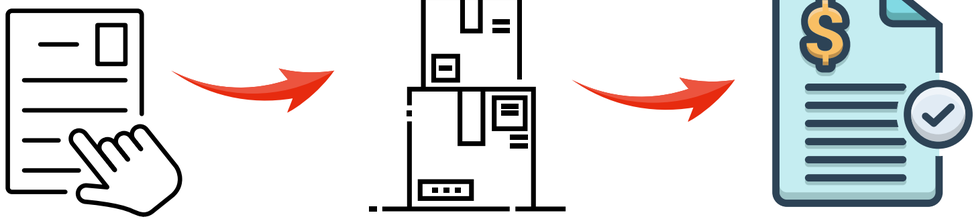
Data Ingestion → Storage → Cleaning → Modeling → Feature Engineering → Hyperparameter Tuning → New York Model



OPERATIONAL PHASE (Option 1)

Flask

User Input → New York Model → Price Prediction



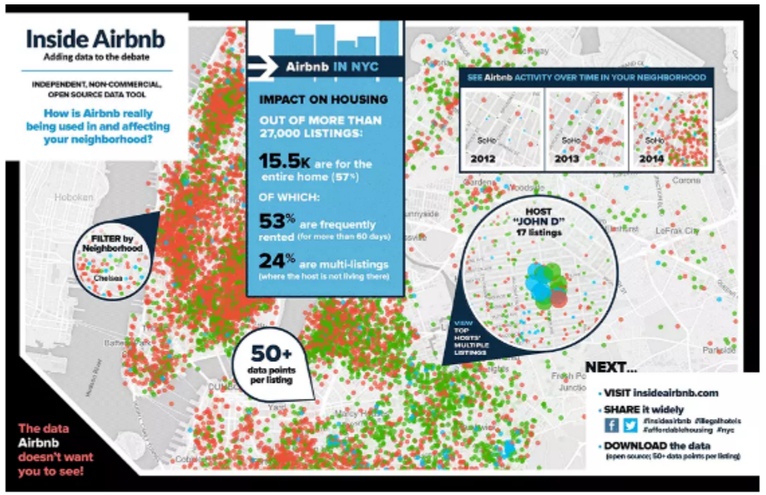
OPERATIONAL PHASE (Option 2)

Pipeline

New Data Sets → Machine Learning Pipeline → New Predictive Models

Data ingestion & Data Acquisition from AIRBNB OPEN DATA

- Listings -- this csv has 105 columns and 50k records for NY
- Reviews -- this csv has reviews by listing
- Calendars -- this has prices up to 1 year in the future



New York City. Adding data to the debate.

Inside Airbnb is a set of independent tools and open data that allows you to explore how Airbnb is REALLY being used in cities around the world.

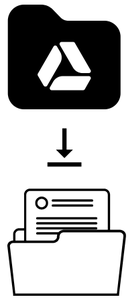
Inside Airbnb

JUNE, 2020 • CAPSTONE PROJECT

Storage

DATA STORAGE

- Originals pulled from insideairbnb and stored on Google Drive
- Data pulled via code from Google Drive and stored locally
- Cleaned Data stored within separate cleaning stages during initial build stage.
- Simplified model is stored on flask hosting site.



Data ingestion

REVIEWS

- Only used for data exploration and trying out text analysis but didn't work out.

CALENDAR

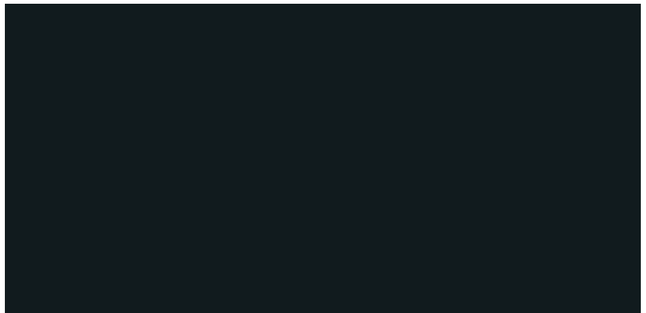
We were planning to use this to see if the prices go up for different seasons or holidays.

But then we realize the dates are for future dates. Most users probably don't setup their prices that advance.

LISTINGS

- All listings that have reviews in last 1 year.
- Exclude records that are type of hotel, boutique hotel, resort, etc.
- Limit the price to between $10 and $800 to deal with outliers.

This listing.csv is the main data set



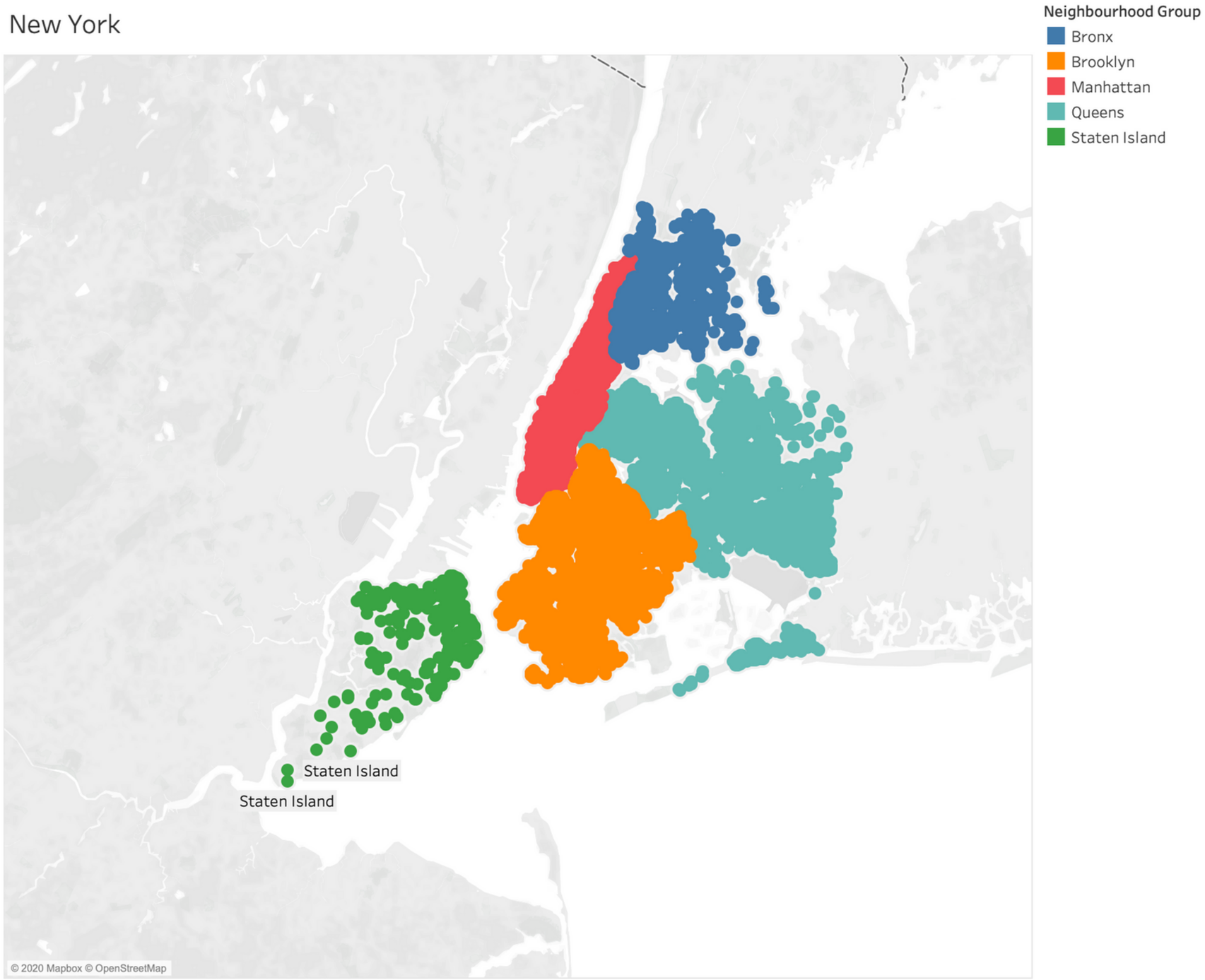
Listings data

50796 rows × 105 columns

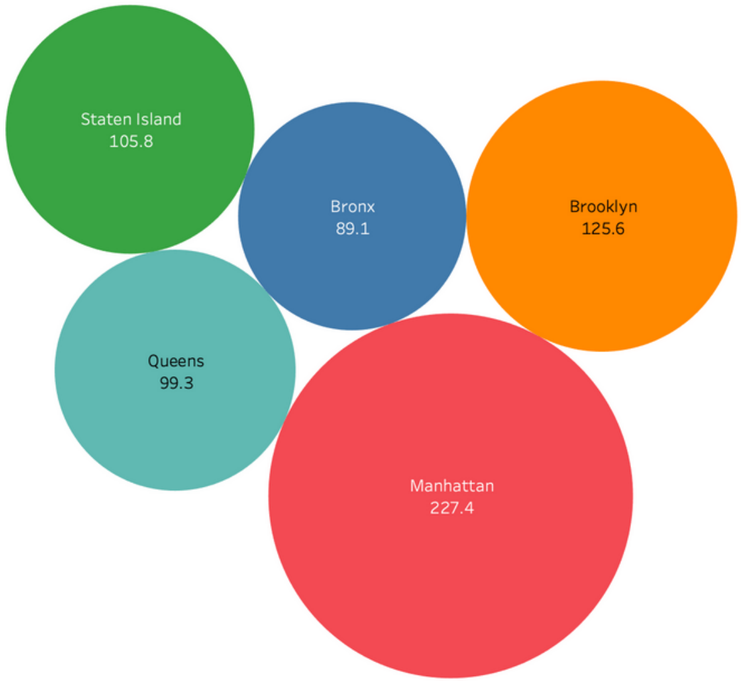
- Columns with same values
- Duplicates values
- Nan values
- Pictures and thumbail columns

'id', 'listing_url', 'name', 'host_id', 'host_url',
'host_name', 'host_since', 'host_location', 'host_response_time',
'host_response_rate', 'host_acceptance_rate', 'host_is_superhost',
'host_neighbourhood', 'host_listings_count',
'host_total_listings_count', 'host_verifications',
'host_identity_verified', 'street', 'neighbourhood',
'neighbourhood_cleansed', 'neighbourhood_group_cleansed', 'city',
'state', 'zipcode', 'market', 'smart_location', 'latitude', 'longitude',
'is_location_exact', 'property_type', 'room_type', 'accommodates',
'bathrooms', 'bedrooms', 'beds', 'bed_type', 'amenities', 'price',
'security_deposit', 'cleaning_fee', 'guests_included', 'extra_people',
'minimum_nights', 'maximum_nights', 'minimum_minimum_nights',
'maximum_minimum_nights', 'minimum_maximum_nights',
'maximum_maximum_nights', 'minimum_nights_avg_ntm',
'maximum_nights_avg_ntm', 'calendar_updated', 'availability_30',
'availability_60', 'availability_90', 'availability_365',
'number_of_reviews', 'number_of_reviews_ltm', 'first_review',
'last_review', 'review_scores_rating', 'review_scores_accuracy',
'review_scores_cleanliness', 'review_scores_checkin',
'review_scores_communication', 'review_scores_location',
'review_scores_value', 'instant_bookable', 'cancellation_policy',
'calculated_host_listings_count',
'calculated_host_listings_count_entire_homes',
'calculated_host_listings_count_private_rooms',
'calculated_host_listings_count_shared_rooms', 'reviews_per_month'

Listings by Neighborhood



Average Price



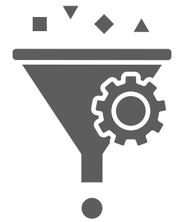
Room Type

Neighbourh..	Room Type	
Bronx	Entire home/apt	433
	Private room	761
	Shared room	55
Brooklyn	Entire home/apt	9,852
	Hotel room	29
	Private room	10,141
	Shared room	433
Manhattan	Entire home/apt	13,531
	Hotel room	336
	Private room	8,053
	Shared room	528
Queens	Entire home/apt	2,271
	Hotel room	37
	Private room	3,754
	Shared room	204
Staten Island	Entire home/apt	187
	Private room	186
	Shared room	5

0K 1K 2K 3K 4K 5K 6K 7K 8K 9K 10K 11K 12K 13K 14K

Count of Room Type

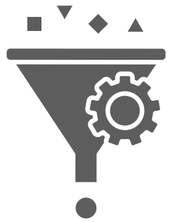
Ingestion & Cleaning Data



- Columns Removed
  - < 50% Populated
  - 95% + rows with one value
  - Picture / Thumbnail URLs
  - Remove object fields
  - Remove other parts we didn't plan on using

- Rows Removed
  - No reviews within a year (non-active)
  - High price outliers (Hotels primarily)
- Features Columns added from Amenities text.
- Basic conversions (Boolean, Date, Category)

Ingestion & Cleaning Data



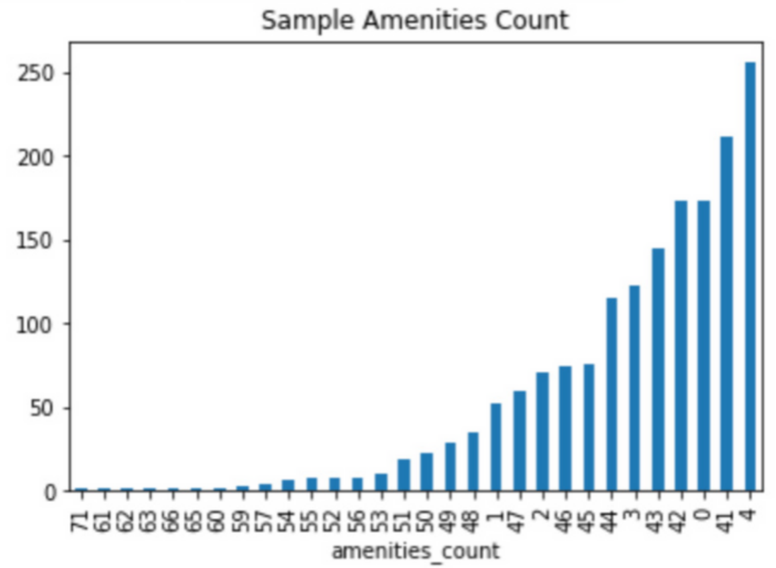
CLEANED DATA

- 27926 rows × 45 columns
- 68 columns removed
- 22870 rows removed
- Feature Columns Engineered

Feature engineering

TROUBLESHOOTING

- First model used all the possible features (more than 120).
- Two features in particular, host verification and amenities have a lot of unique values where hot encoding expanded our dimensions dramatically.



- We also found out that the service fee and cleaning fee caused the model to behave differently that we would expect.
  - As we increased Cleaning Fees, the listing price also increased.
  - These were likely highly correlated with higher priced listing, but not predictive.

Feature engineering

OUR RESOLUTION

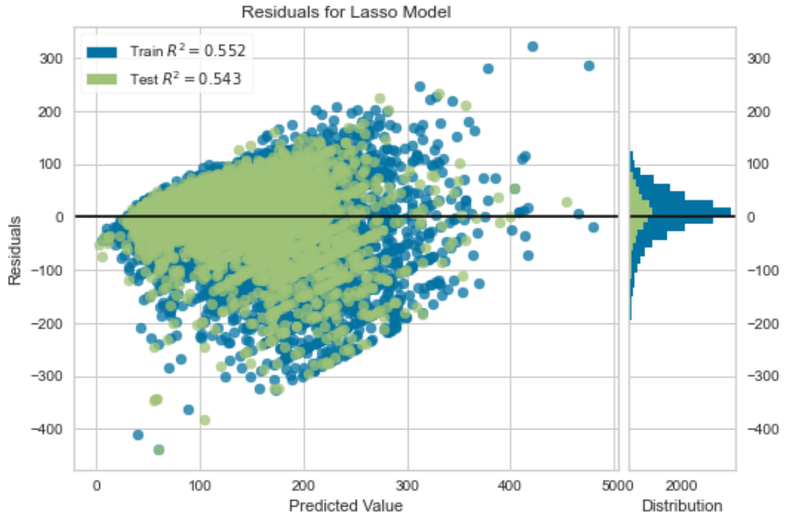
So we engineer some features to make the model simpler. Here are some of them:

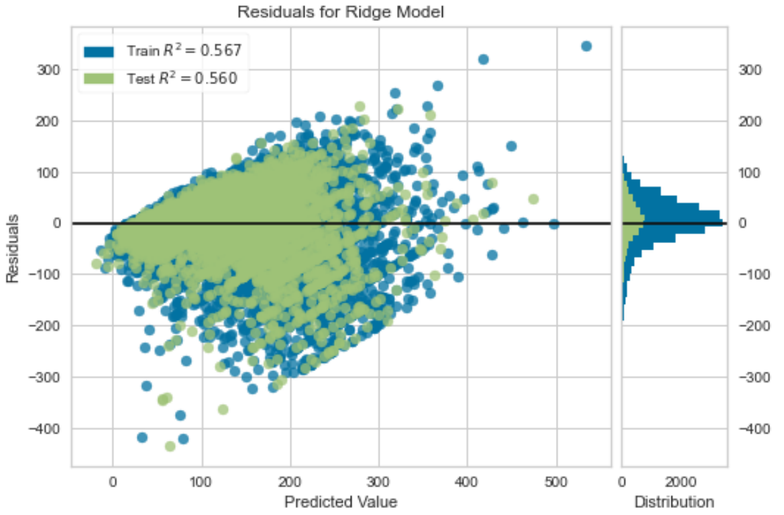
- amenities , instead of using one hot encoding, we are counting them
- using yes or no for cleaning fee, instead of dollar amount
- host response type, property type, acceptance rate calculation, real bed, cancellation policy, listing counts

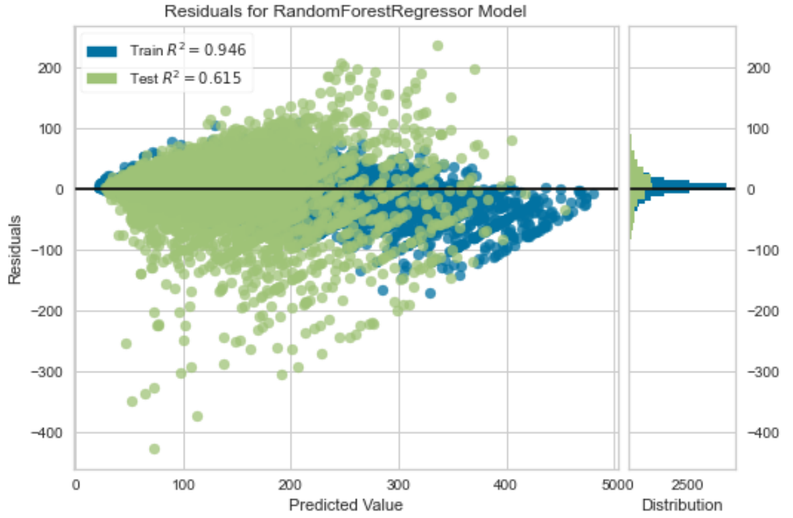
Modeling: Base Models

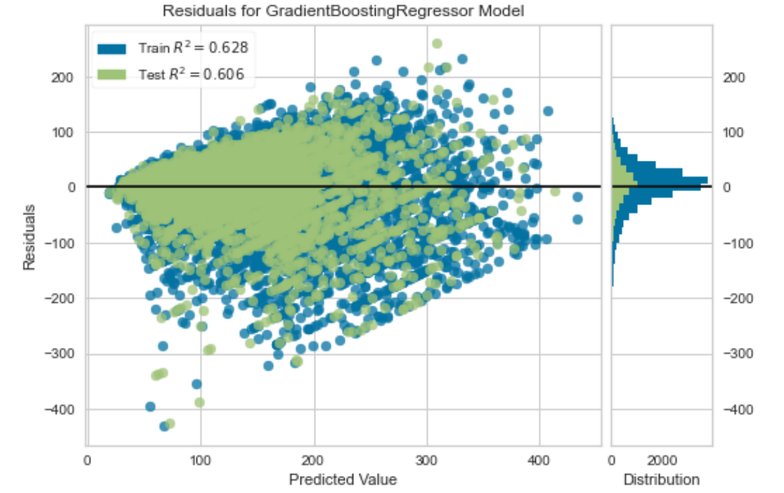
June, 2020 • Capstone Project

Base Modeling







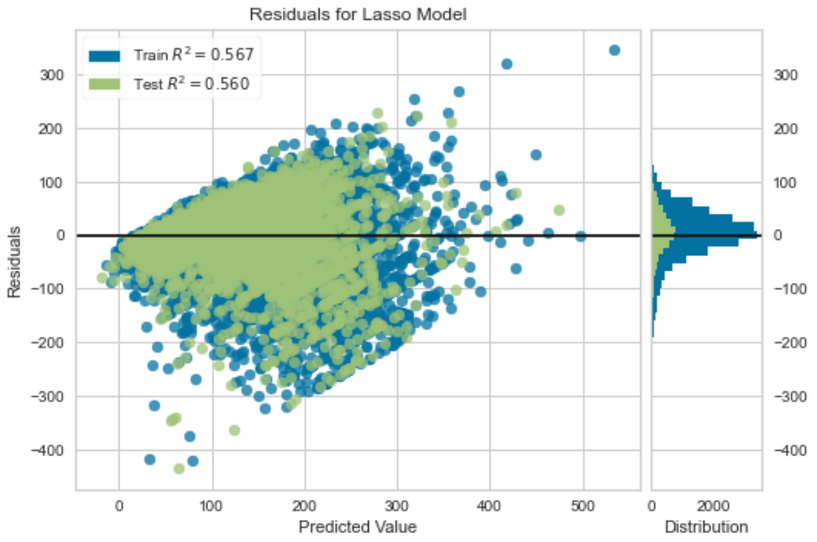


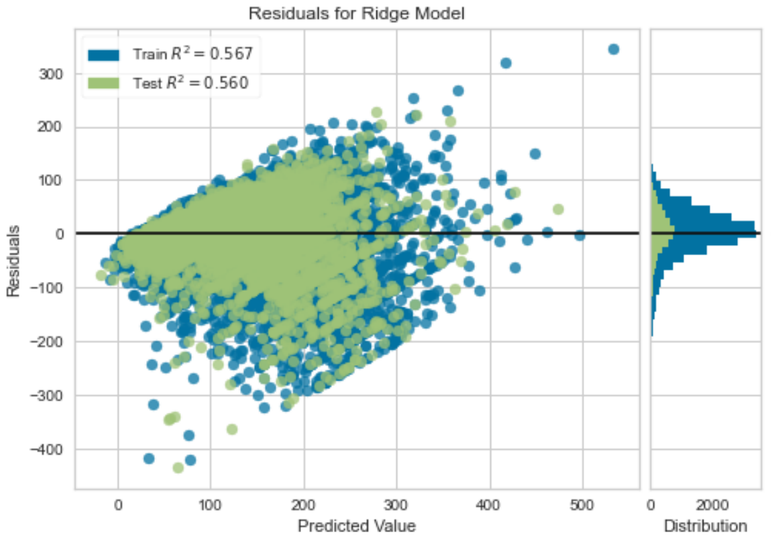
Modeling: Hyper Parameter Tuning

June, 2020 • Capstone Project

Lasso & Ridge

- For Lasso & Ridge Regression models, we used Grid Search to find the optimal alpha to choose.
- For Lasso, Optimal alpha was .001
  - Applying this alpha improved R^2
    - On Train Data, from .552 -> .567
    - On Test Data, from .543 -> .560
- For Ridge, Optimal alpha was 10
  - Applying this alpha had no impact on R^2
- As you can see below, after optimization, the models performed identically.



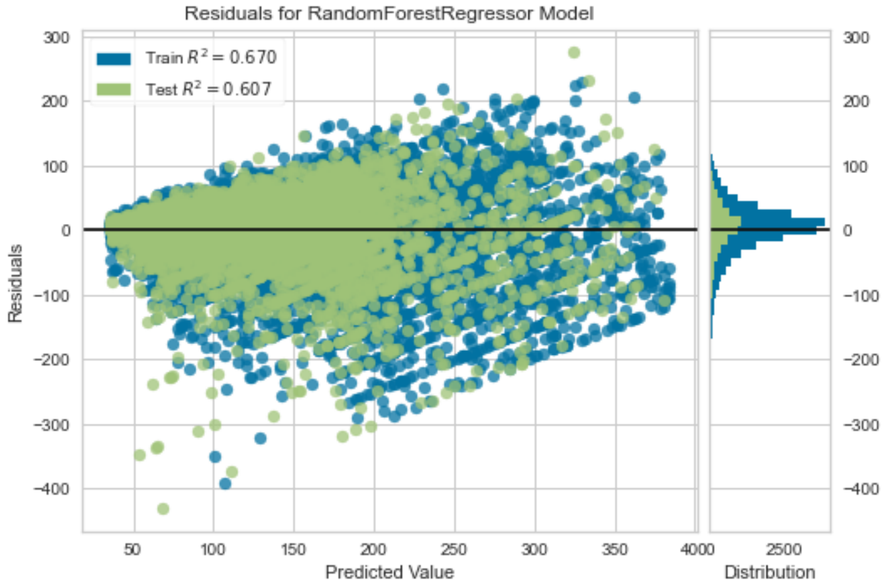


Modeling: Hyper Parameter Tuning

June, 2020 • Capstone Project

Random Forest

- For Random Forest we were searching for the optimal number of estimators, which random state to use, and minimum sample splits.
- Base model showed high Train R^2 but much lower Test R^2. This likely means the model is over-fit with the base model. This is the main issue we need to resolve if we want to use this model.
- Optimal parameters based on grid search we found were:
  - n_estimators = 100
  - random_state = none
  - min_samples_split_100 = 100
- This still may be slightly over-fit still, but overall is much less than before while maintaining a relatively good test R^2.



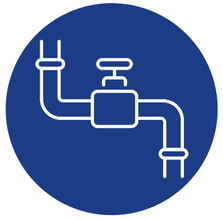
Modeling: Hyper Parameter Tuning

June, 2020 • Capstone Project

Gradient Boost

- For the last model we tried, Gradient boost, the parameter we were looking to optimize was the max depth.
- Base model showed better R^2 scores compared to the Lasso and Ridge, and was similar to the random forest regression scores on the test R^2.
- Optimal max depth from the grid search gave 6.
- Using this, we saw an increase in both R^2 scores, but it may have been slightly over-fit.

Data Pipelines



June, 2020 • Capstone Project

Testing Models on other cities

Now that we had done each stage of the data pipeline, we were then able to stitch together a pipeline procedure so we could expand our testing to other cities.

Because the Lasso and Ridge performed identically after optimization, we decided to just use Lasso with the optimized alpha alongside both the Random forest and Gradient Boost models.

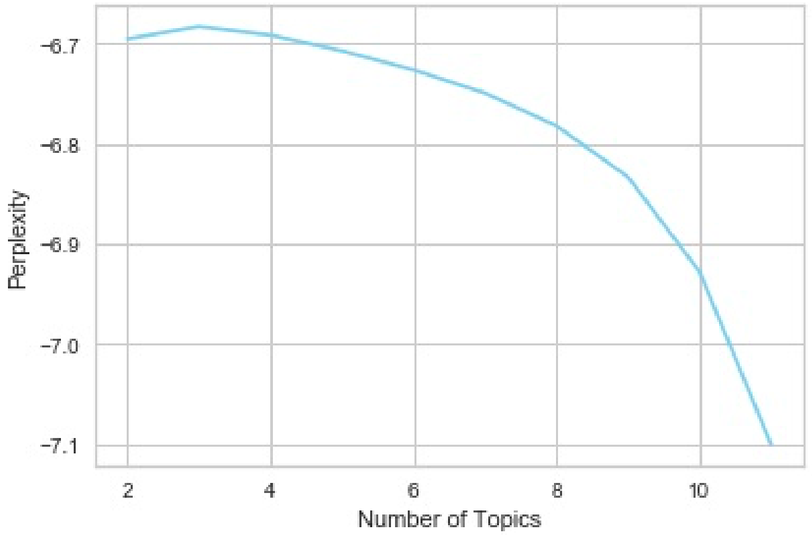
Additional Exploration: Topic Modeling

Missing values in Airbnb dataset (TEXT)

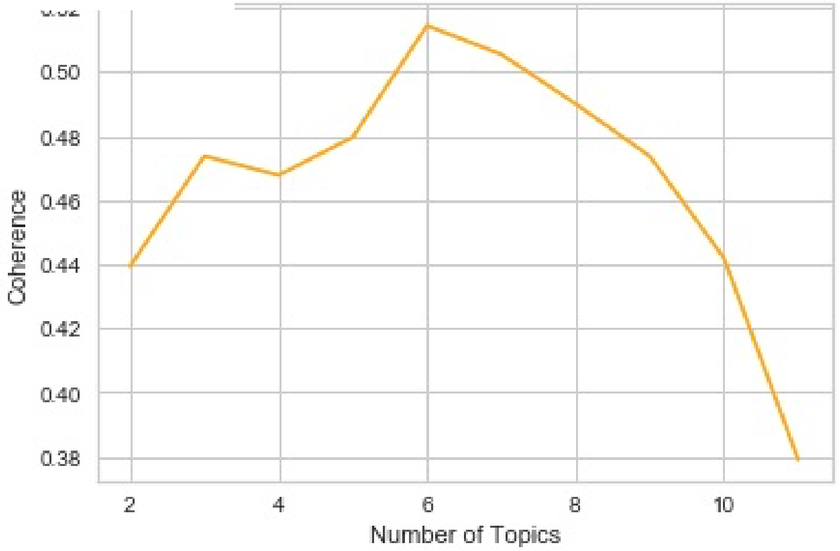
- 4.8% missing in summaries
- 28.0% missing of space description
- 2.4% description
- 34.4% missing overviews on their neighborhood
- 59.6% missing notes from the host
- 34.4% missing information on transit
- 47.2% missing information on accessibility
- 40.2% missinginformation on interaction
- 39.1% missing on house_rules
- 0% mising in name

Additional Exploration: Topic Modeling

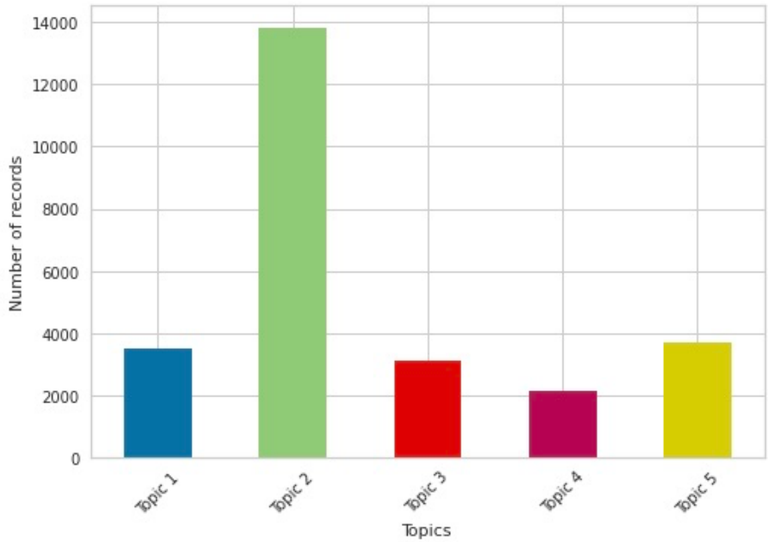
LDA Text Analysis



Visualization

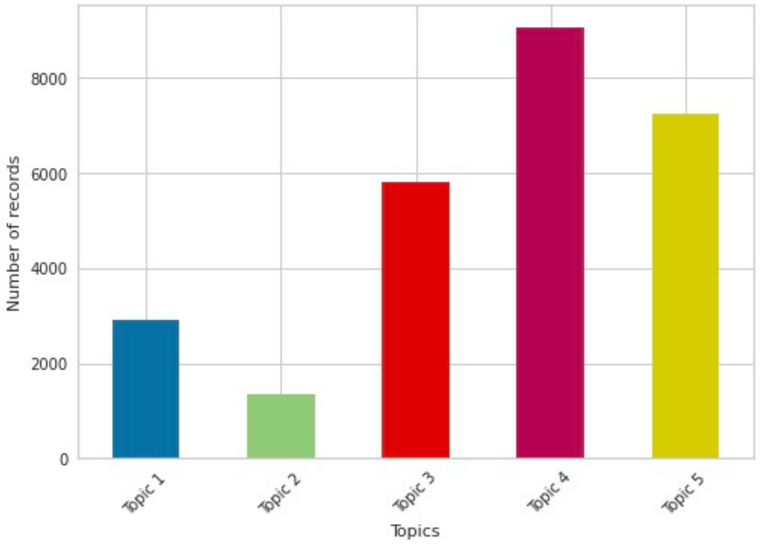


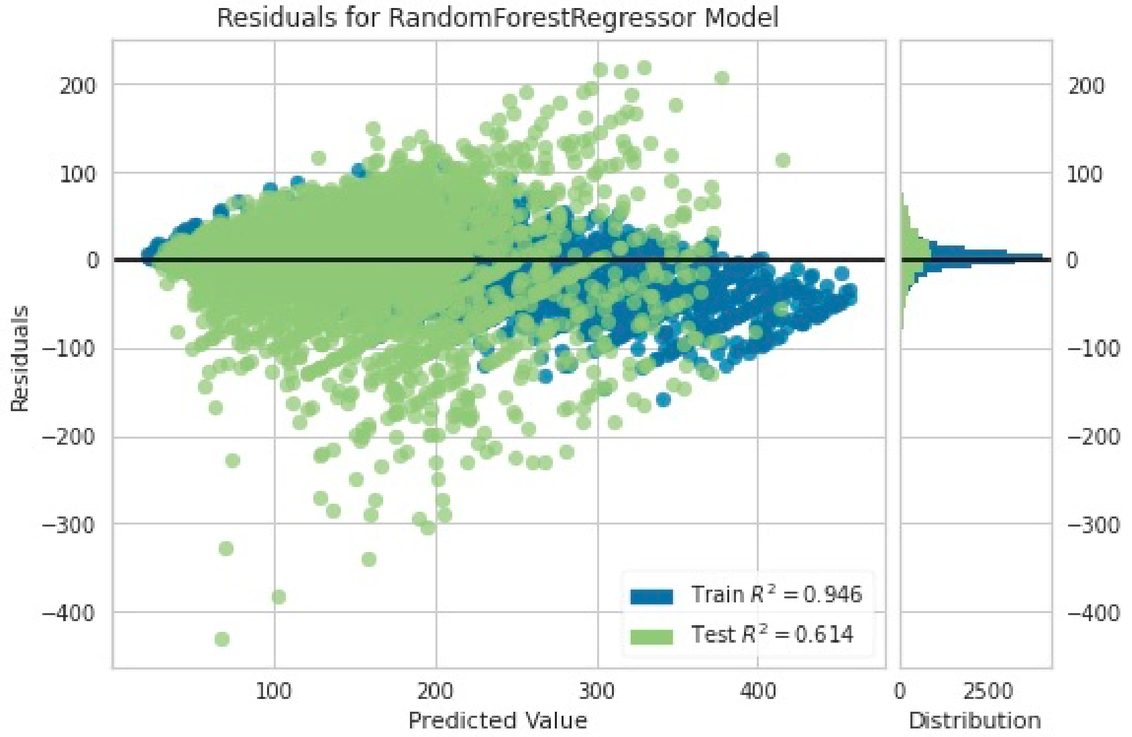
Additional Exploration: Topic Modeling

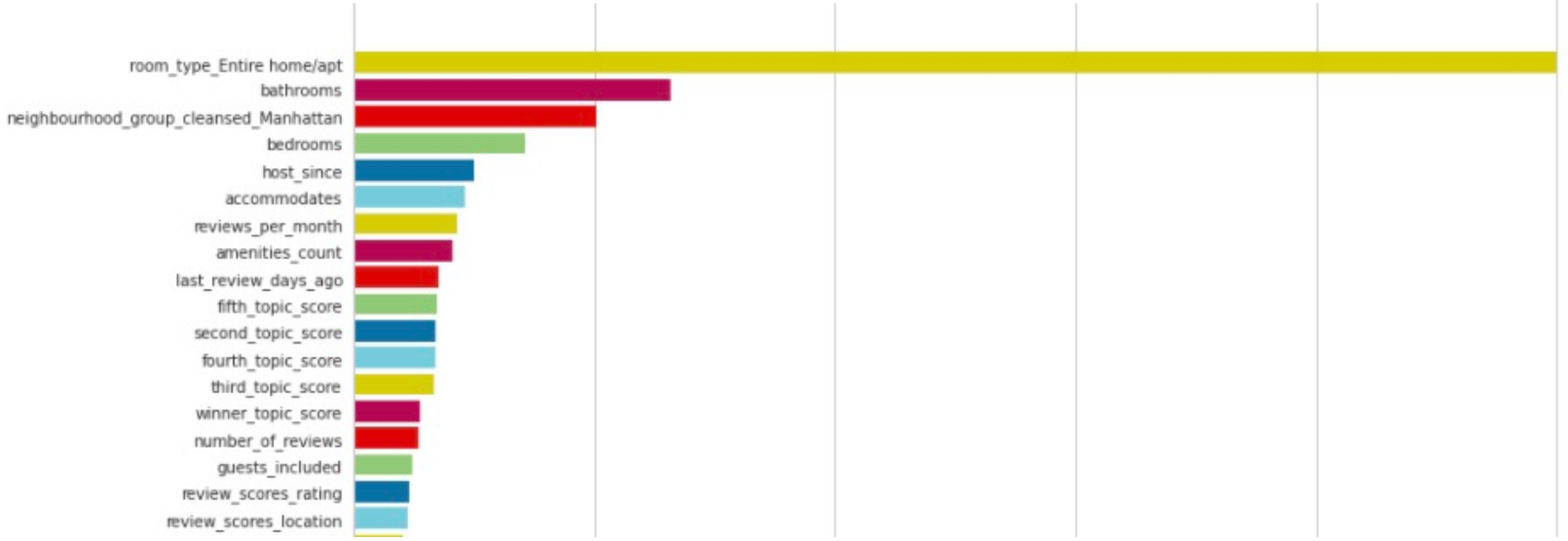


Topic # 2

Modeling with TM scores







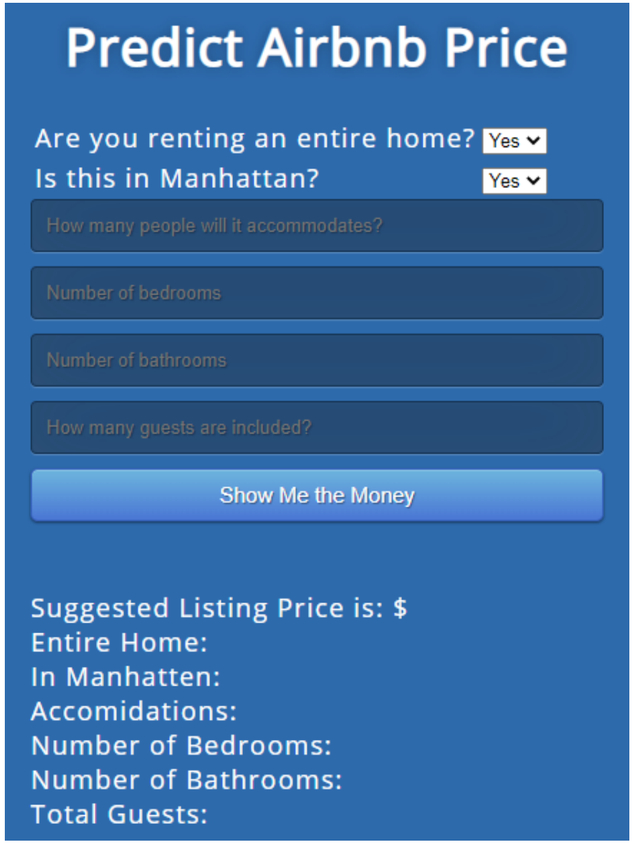
Application End Product

June, 2020 • Capstone Project

FLASK MODEL

- To simplify the usage of the model in a live system, we decided to use a simplified model (6 features).
- Though this app is interesting, and works as expected, without further testing in the field, we won't be able to verify its accuracy in the real world.

Link to Flask App



Further Research

Address heteroscedasticity of our model and R2.

Better understand of why the topic scores are important for prediction

Do a bigram or trigram analysis instead of single tokens because text is all about context

Use other algorithms to calculate frequency like TF-IDF to have a more detailed understanding of the text

Picture analysis