Two-Sigma/Renthop Kaggle Machine Learning Project

Team: [] (NULL)

Ray (Xu) Gao,

Tommy (Yaxiong) Huang,

Scott Edenbaum,

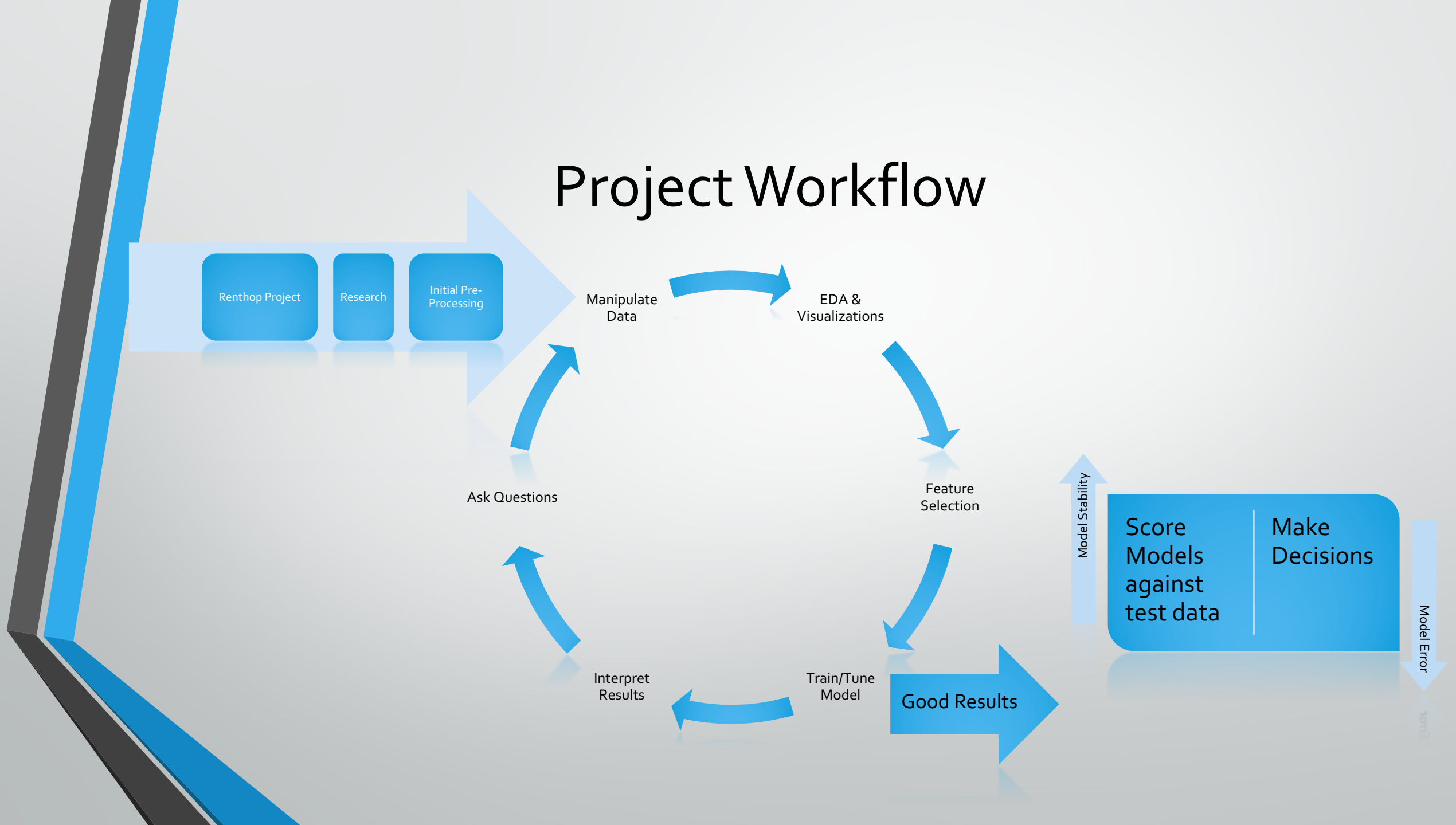
Dodge Coates

Table Of Contents

Overview

- Research
- Formulate approach to project
- Execution
  - Feature pre-process
  - EDA
  - Feature selection
  - Model development
- Model structure
- Results

Project Workflow



Research

- Kaggle Kernels
- Inspect sample data set
- Define problem
  - Supervised machine learning
  - Inspect website – add screenshot of Renthop listing page
  - What is interest level? What does it mean?
  - Base assumptions? Expectations for high/medium/low interest property

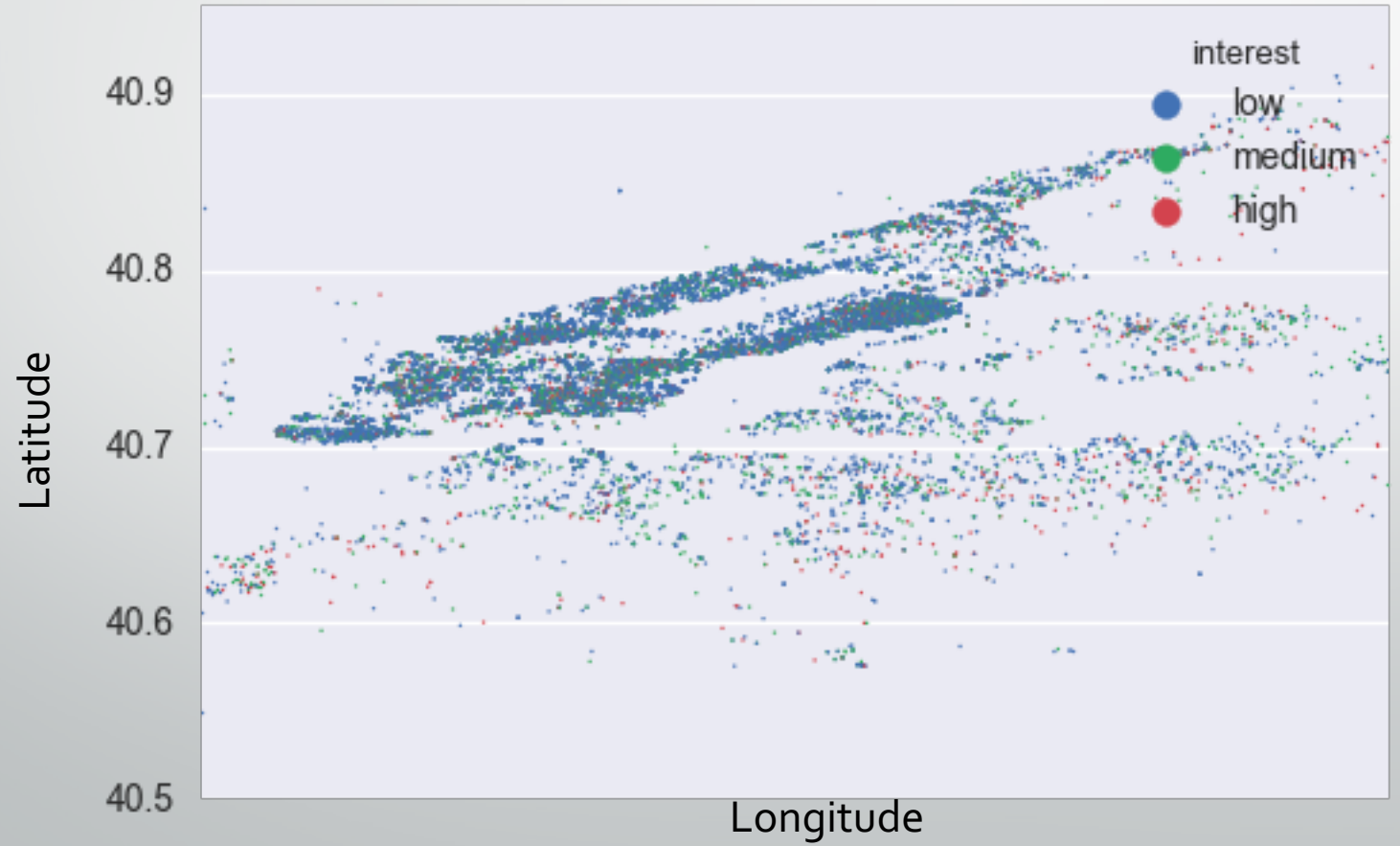
Renthop Listing

- Examining Renthop's listing page
  - 18 Distinct Features, and an input box

Formulate Approach

- Work in python
  - Why?
    - Personal preference, portability, large Machin learning development community
    - JSON data
    - Parse images

Property Map Swarmplot



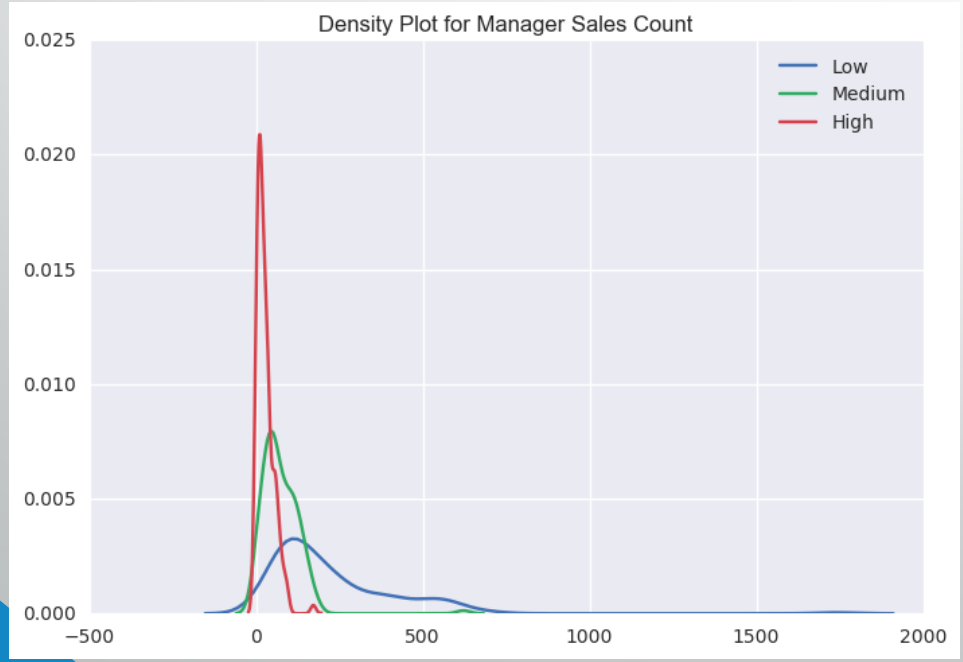
Execution

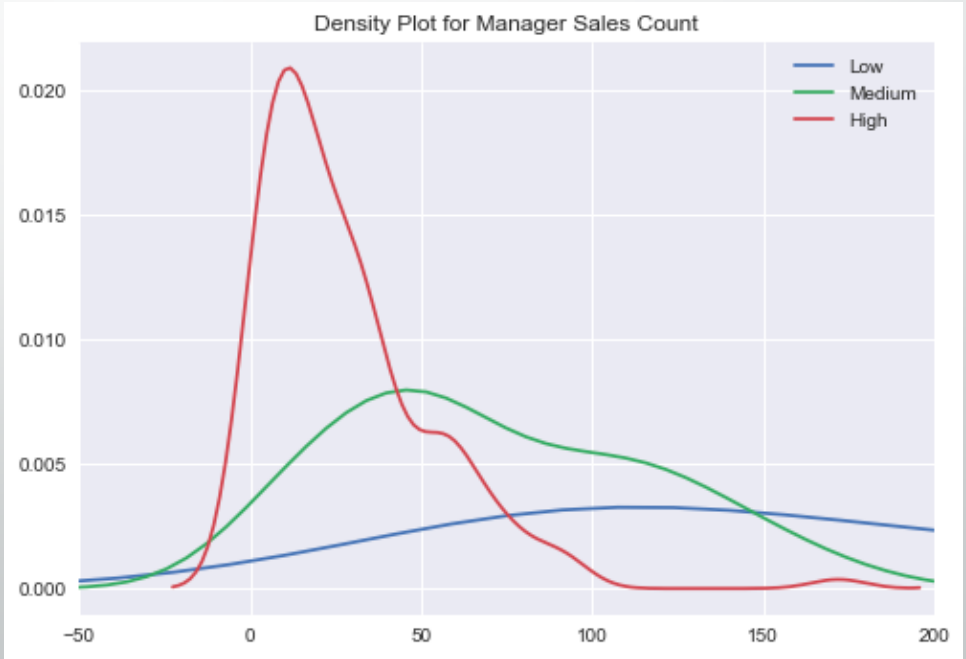
- Feature Pre-Process
  - Assume least important features:
    - Building_id, manager_id, listing_id
    - Display_address, street_address
    - Images?
    - created
  - Downloading and parsing ~83gB image data
  - Use EXIF.py, PILLOW, OS packages to extract info
  - Graph latitude/longitude

Derivative Features

- Manager_count
- Building_count
- Word_count
- Word_diversity
- Dummify
  - Number_distinct
- Sentiment
- Image_count, Avg_brightness, Avg_luminance, Avg_filesize…
- Etc.

Manager Count

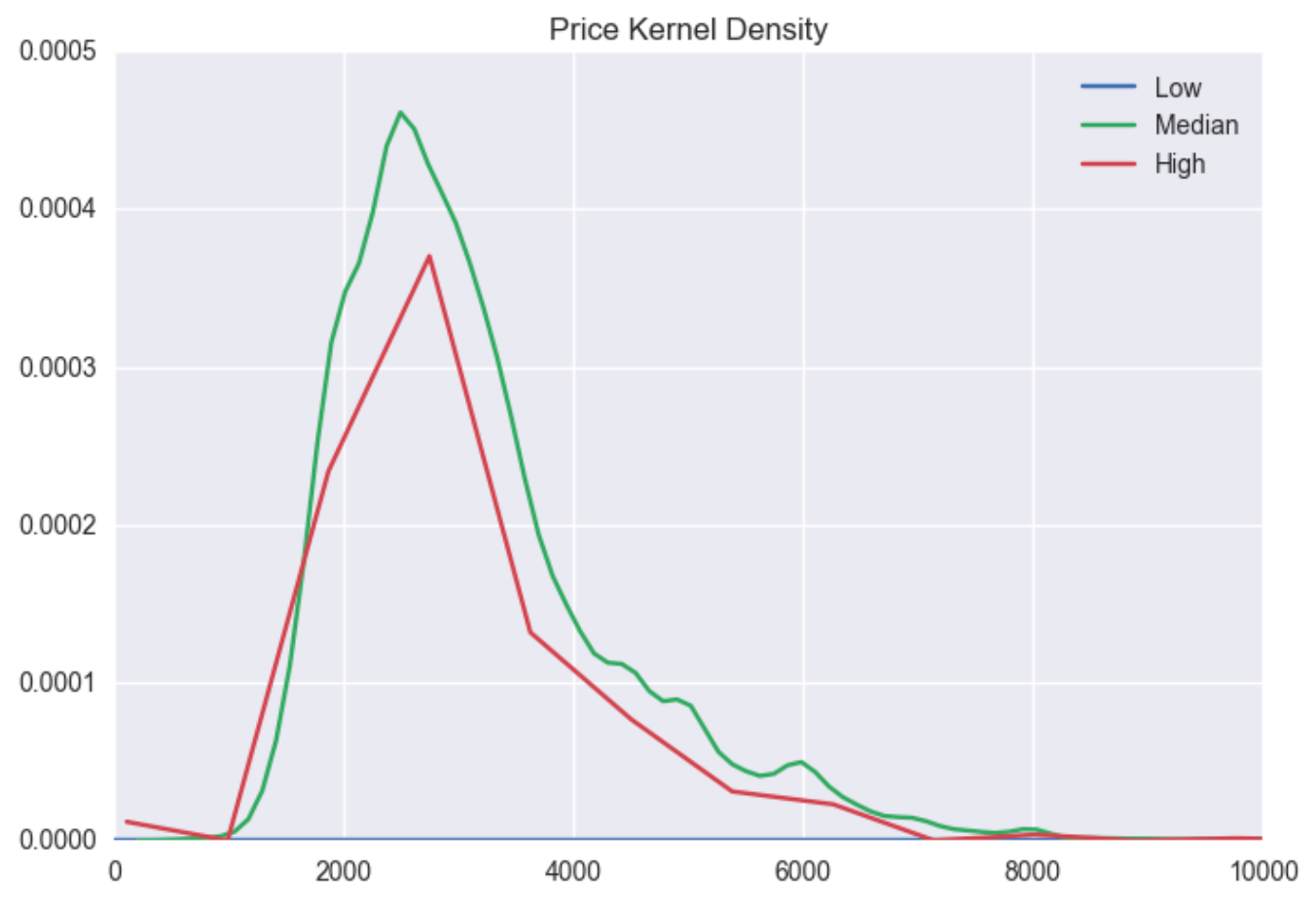




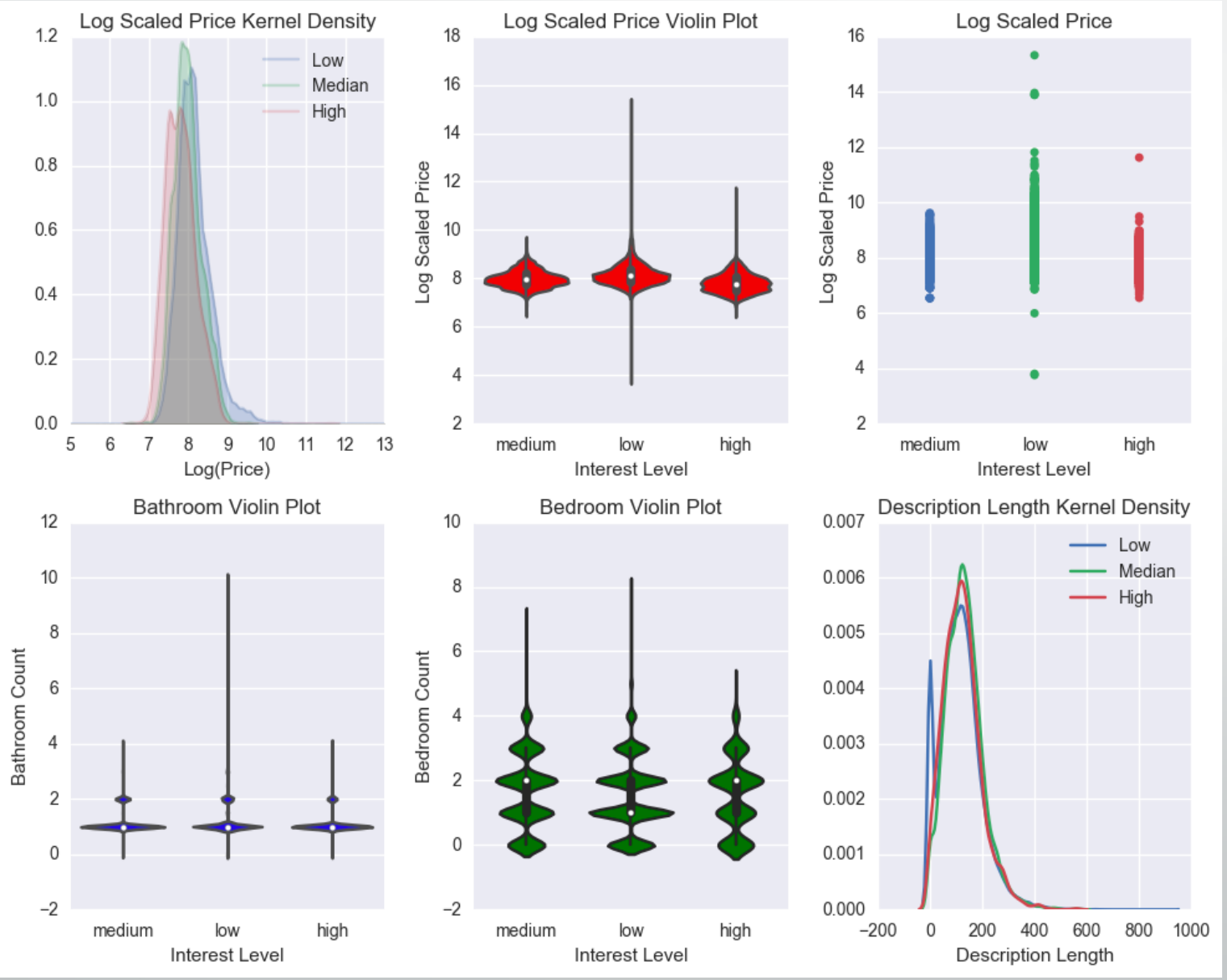
EDA

- Price
  - log Transformation
- Bathrooms
- Bedrooms
- Word count

Price – Kernel Density



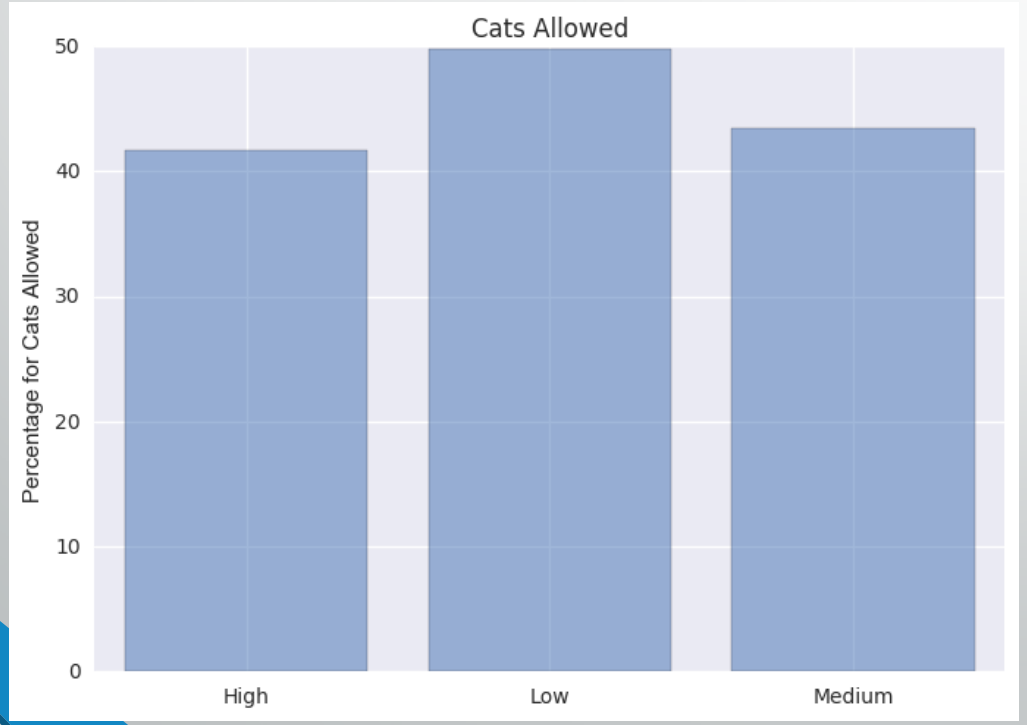
Base Factors Visualizations

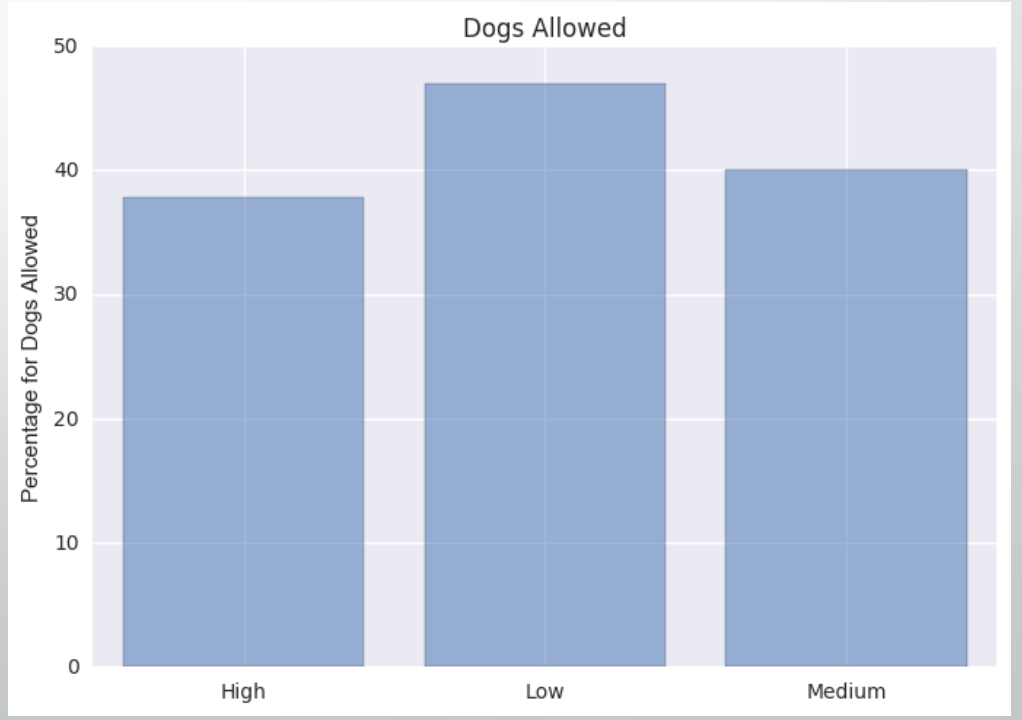


EDA cont.

- Distinct “Features” – distribution
  - Dogs & Cats – Low importance
    - Interest vs Value
- Unique “Features”
  - Transformed to count
  - Limited impact from sentiment analysis

Cats & Dogs

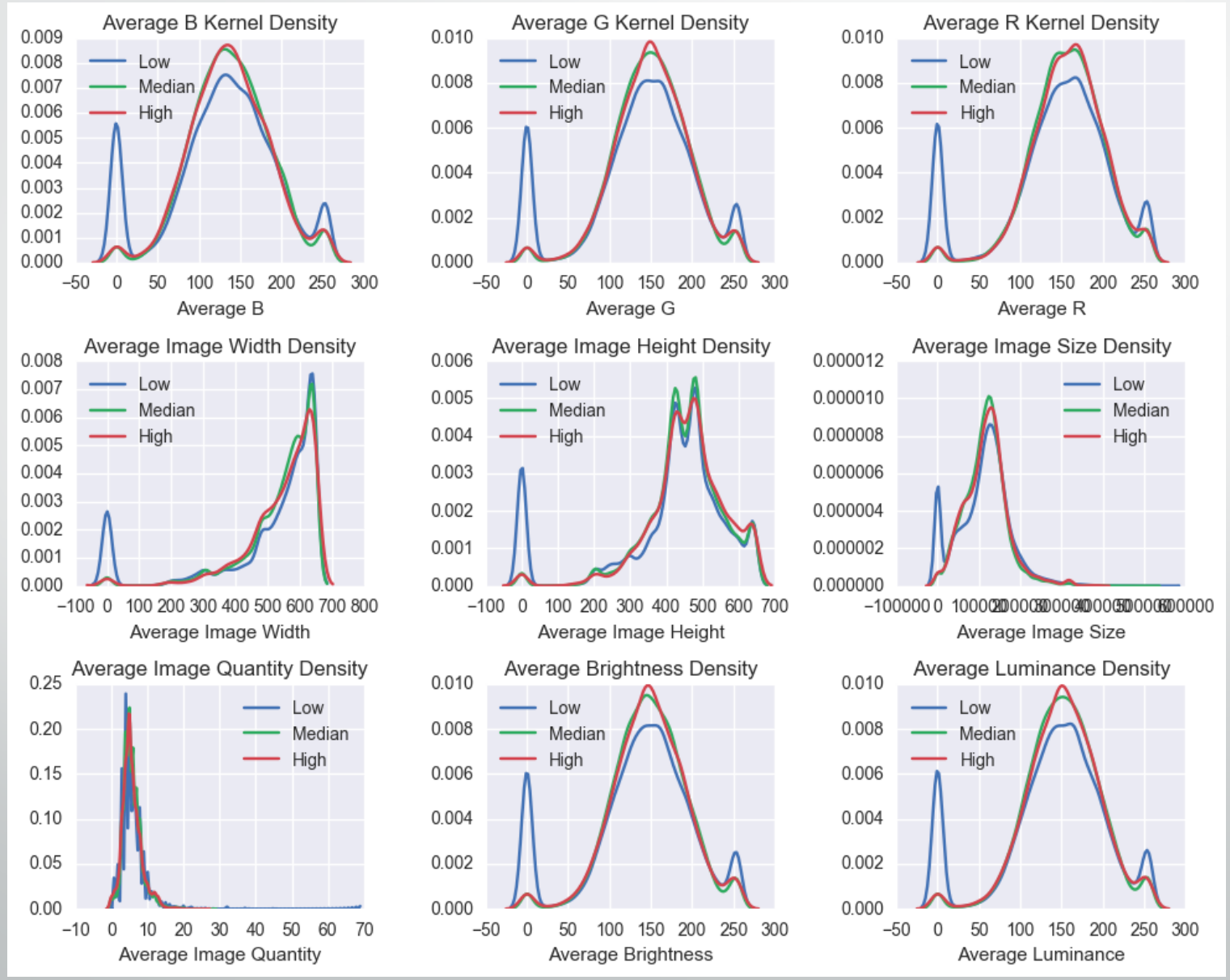

Cats Allowed
Percentage for Cats Allowed
50
40
30
20
10
0
High
Low
Medium


Dogs Allowed
Percentage for Dogs Allowed
50
40
30
20
10
0
High
Low
Medium

EDA cont.

- Images
  - RGB
  - Average image size
  - Brightness, Luminance
  - Average image count
  - Metadata ratio
  - Pixel Height, Pixel Width

Image EDA Visualizations



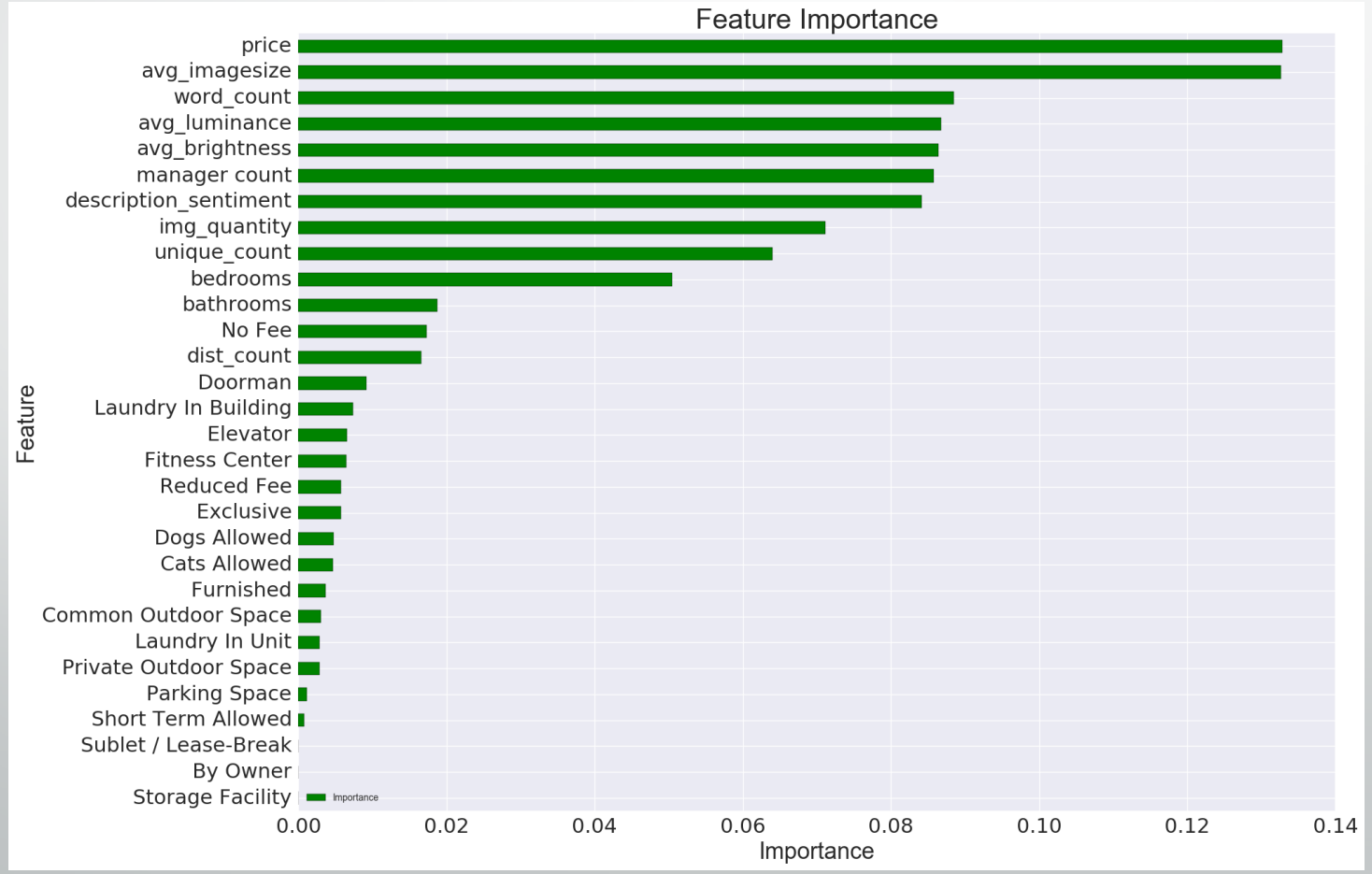
EDA cont.

- Manager count
  - Log transform
- Building count
  - Log transform

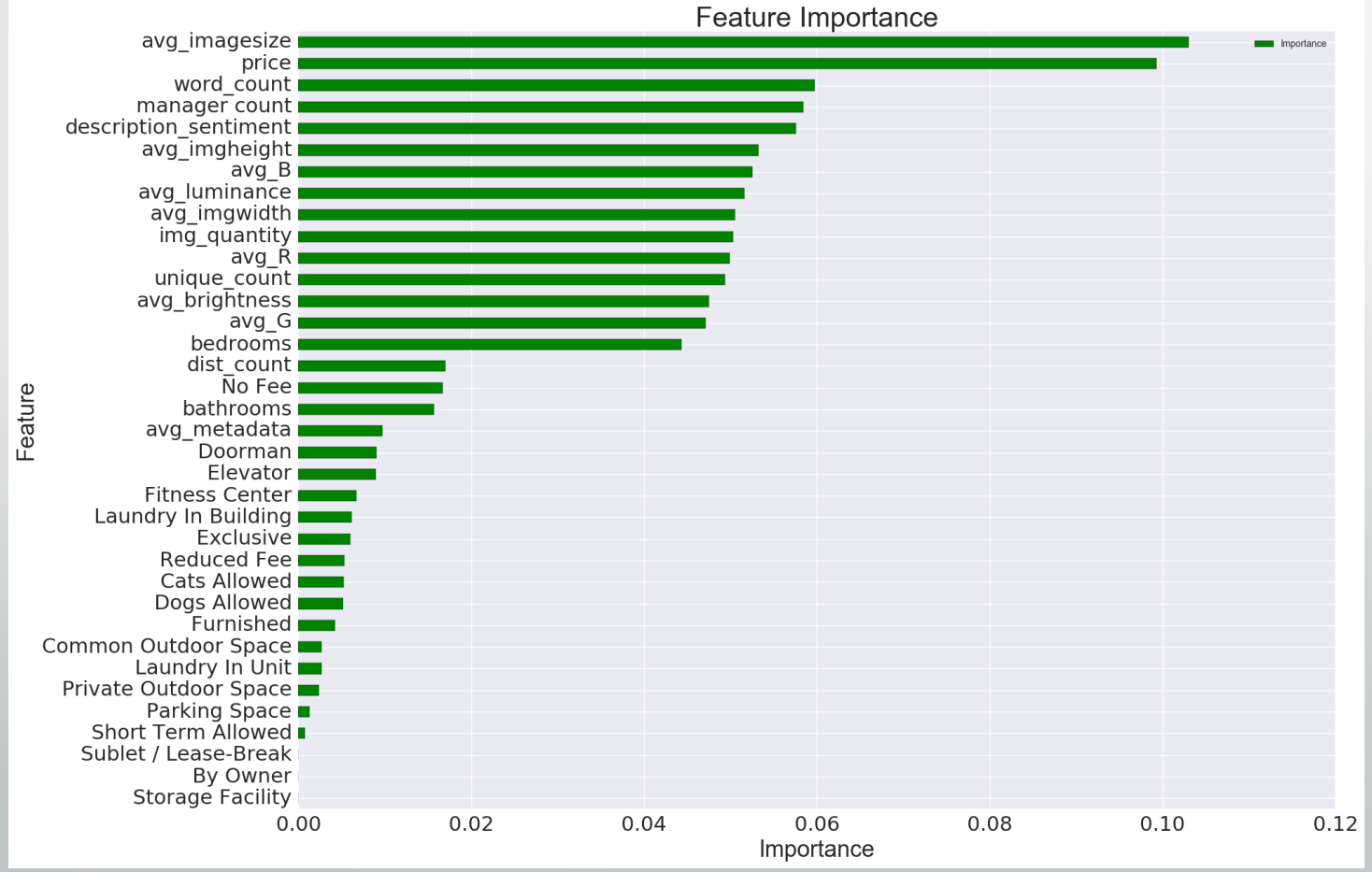
(Initial) Feature Selection

- "Trial" Feature Selection
  - Initially select 5 features:
    - Price, Bathrooms, Bedrooms, Latitude, Longitude
      - Form benchmark
    - Compare against Price, Bathrooms, Bedrooms
- Extra Tree classifier for Feature Selection
  - Orders Feature Importance

Feature Importance



Feature Importance:
Revised with Additional Derivative Features



Model Development

- Stacking Model
  - Not optimized
- Voting Classifier
  - Fast but unstable
- Artificial Neural Network
  - Very Long Processing time

Model Structure

- Stacking Model – each classifier's result -> next classifier's input
  - Logistic Regression, KNN, Gradient Boosting, Random Forest, Ada Boost
    - Results passed to XG Boost
  - Copied underlying model code
    - Not optimized, slow
    - Discovered stacking model library 2 days ago

Stacking in 5 Steps

1. Partition train set into 5 test sets (aka Folds)
2. Create empty data frames *train_meta* and *test_meta*
   - row Ids & fold Ids from training dataset
3. For each test fold, combine remaining 4 folds as training fold
   - Predict results for given fold and save inside *train_meta*
4. Fit each base model to full training dataset
   - Predict results on test dataset, save inside *test_meta*
5. Fit a new Model, S (Logistic Regression, XGBoosting, etc.) to *train_meta* using M1, M2, … as features
   - Optionally, include other features from the original training dataset or engineered features

Model Cont.

- Voting Classifier
  - Easy way to combine models
    - Softweight – weighted average amongst models
  - Logistic Regression, Random Forest, Naïve Bayes, Decision Tree, Gradient Boost, Ada Boost
  - Fast! But unstable
  - Stepwise process generates optimized linear combination of underlying models

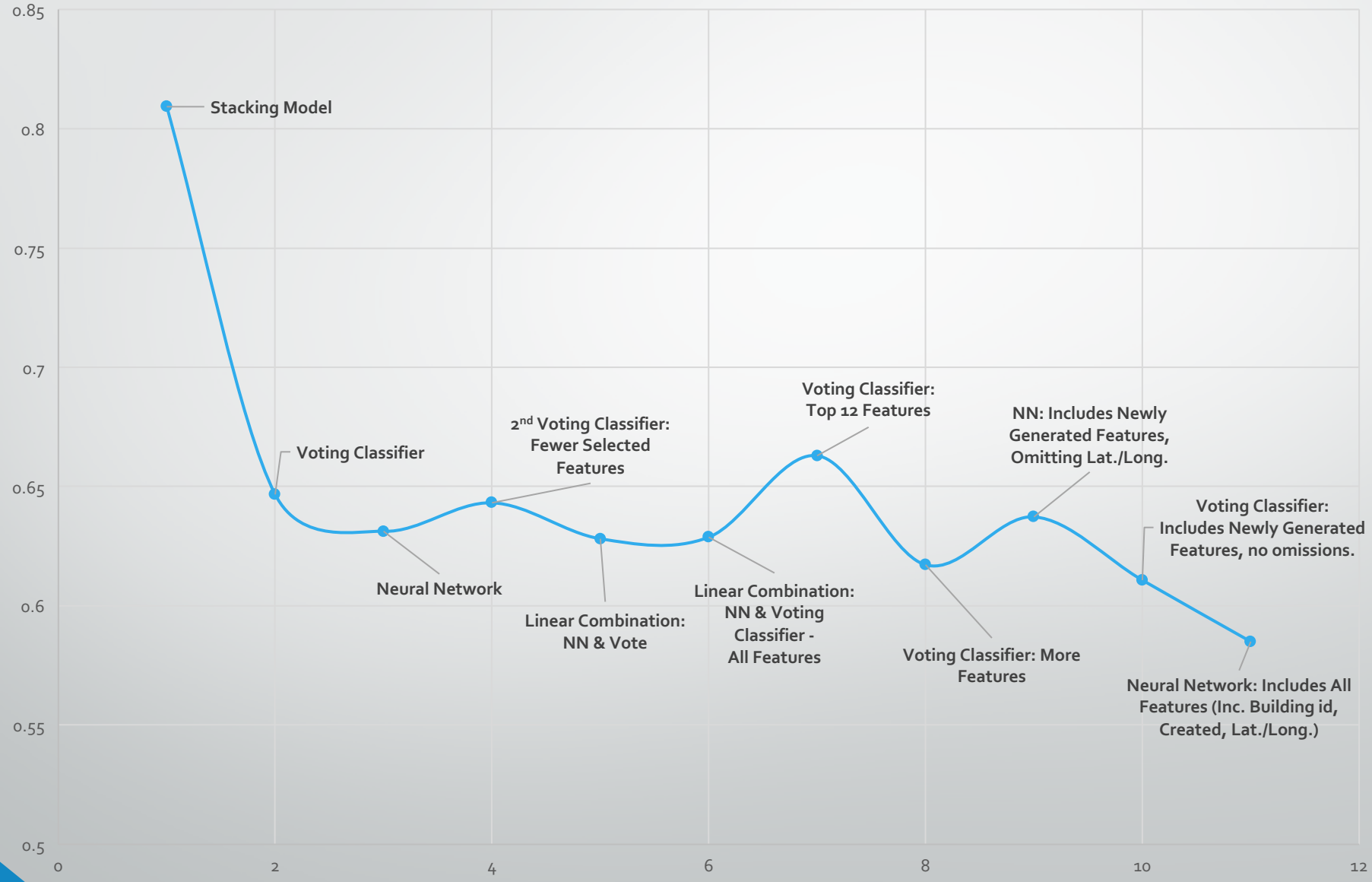
Models

- Artificial Neural Network
  - Tensorflow
  - Theano
  - Structure
    - Input layer
      - 300 nodes, Sigmoid activation function
    - 2 Hidden layers
      - 50 nodes, Sigmoid activation function
    - Output layer
      - 3 nodes, softmax activation function
        - Multiple mutually exclusive classes
    - Downside - Process time

Results

- Include all "Important" features
  - Omitting seemingly useless and highly correlated features
    - Decrease dimensionality
    - Decreases overall model performance
      - Why?
    - Initial process we ignored building id, manager id, description, Latitude, Longitude, Created
      - Manager id, description id added
        - Model improved
      - Ultimately added Created (dates), Latitude, Longitude

Kaggle Score Submissions



Conclusions and Insights

- Group Cohesion
  - Organized github repo
  - Discuss strengths & weakness
- Concurrent Development
  - Assign parallel tasks during development
  - Simultaneous development
    - feature engineering
    - EDA
    - Model Deployment
    - Parameter Tuning
- Development Environment
  - Ipython notebook
  - Python library is very powerful
  - Running models in terminal
- Python is incredibly powerful for machine learning

Additional Tools

- Subplot – display multiple histograms in same image
- EXIF.py – read image metadata
- PILLOW (aka PIL) – basic image processing
- JSON – convenient file format
  - Very well supported within Pandas
- SSH, Screen – run models and test various code remotely
- Make – consistent data processing
- OS – file processing
- Git – version control
- Emacs, Sublime, iPython Notebook