Predicting the Level of Interest of a Rental Listing

Dane Noland

DS-SEA-06

The Company

RentHop

- Web and Mobile-based search engine that allows users to search for apartments in major cities

The Problem

Finding the right rental is difficult in a city

- Sometimes there are thousands of options to choose from
- Sometimes are there's little to no options
- Can you find the right setup (# bathrooms, # bedrooms)?

Listing a rental can be difficult

- Sometimes there are thousands of rentals to compete with
- Loss of income with empty units
- How can you make your listing "**POP**"

The Question

Can we predict the level of interest a rental posting will have?

- If we can predict how popular a rental will be, we can look at key items/attributes that draw renters in

The Data

Kaggle

- Data was obtained from a Kaggle Competition online and downloaded from their website

- Data was in JSON format, there were two (2) files

    1. Train.json - model building data

    2. Test.json - model testing data

The Data - Test.JSON

- bathrooms
- bedrooms
- building_id
- created
- description
- display_address
- features
- latitude
- listing_id
- longitude
- manager_id
- photos
- price
- interest_level

The Data - Test.JSON

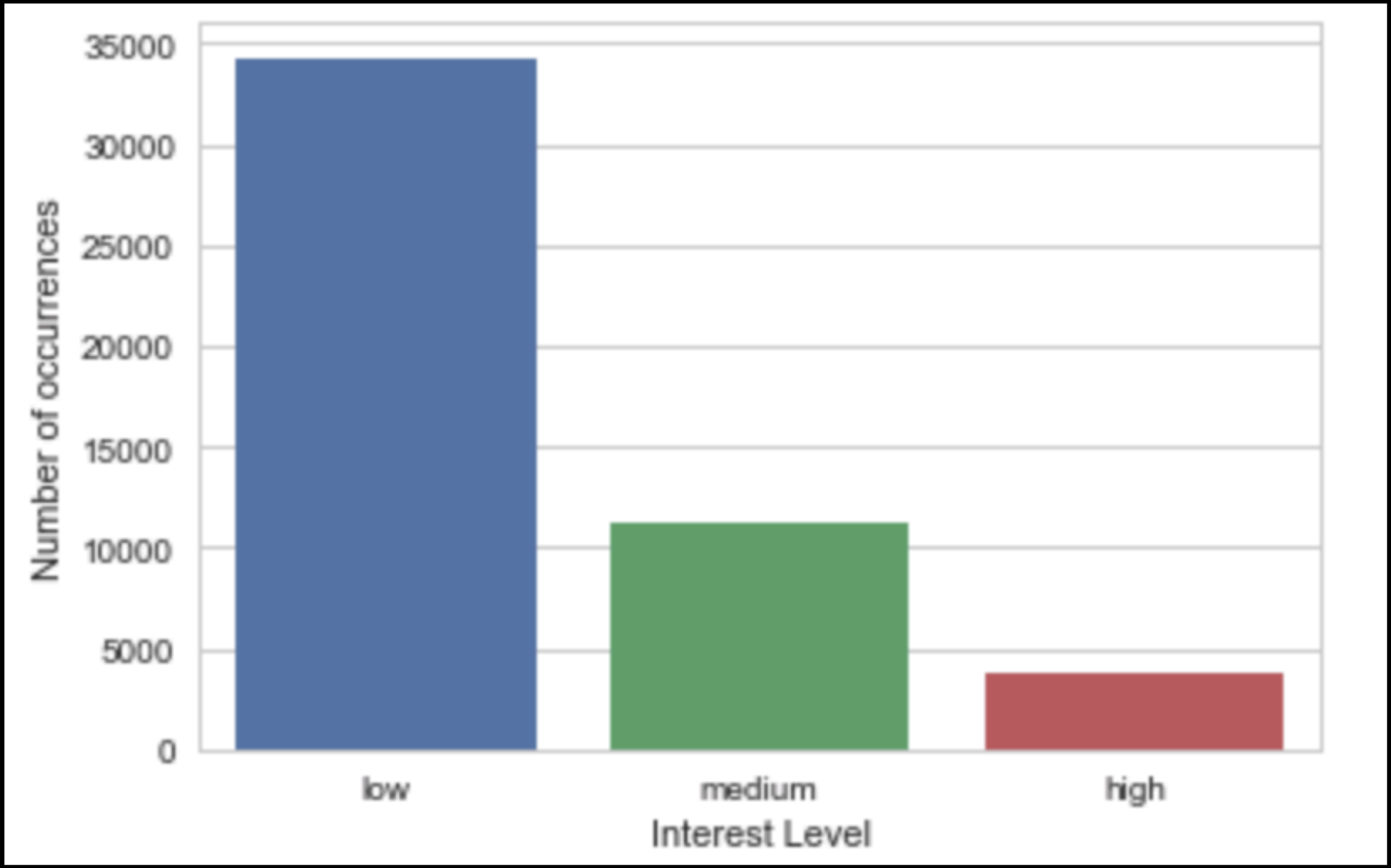
Interest_level….that's the ticket!

- **Defined**: # of inquiries a listing has during the life of the post
- **Categories**: Low, Medium, High

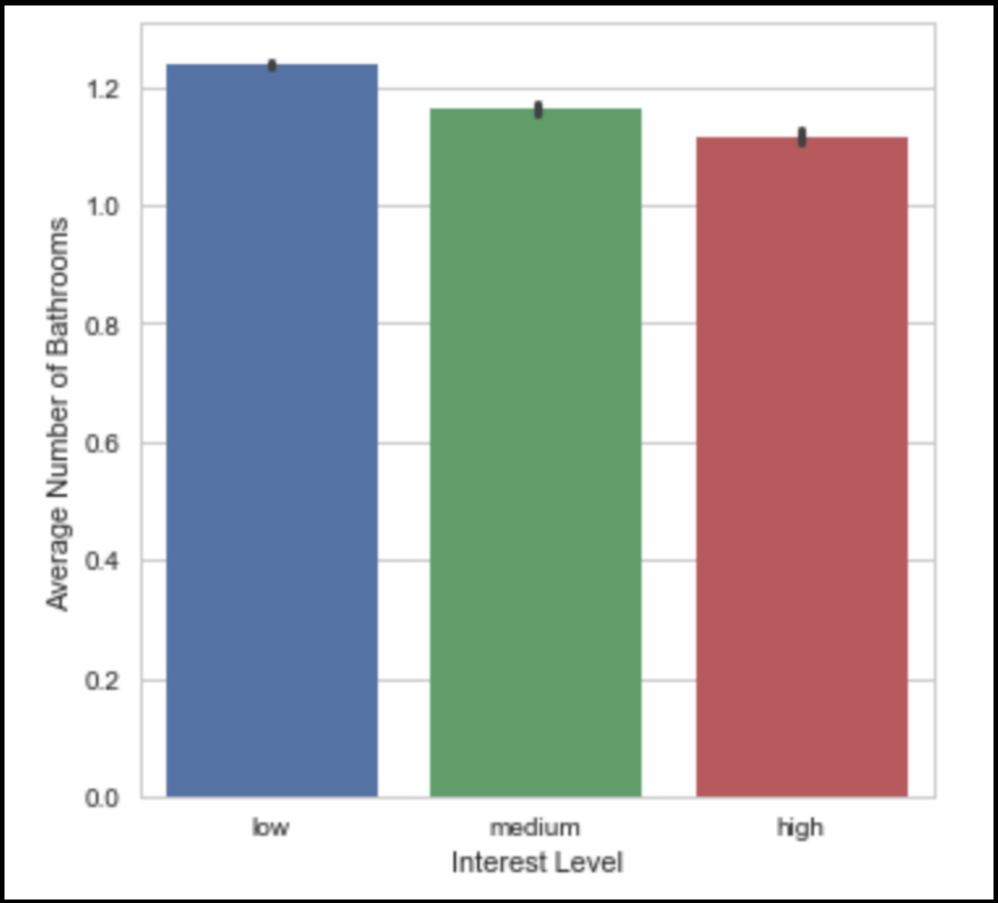
How can we predict interest level?

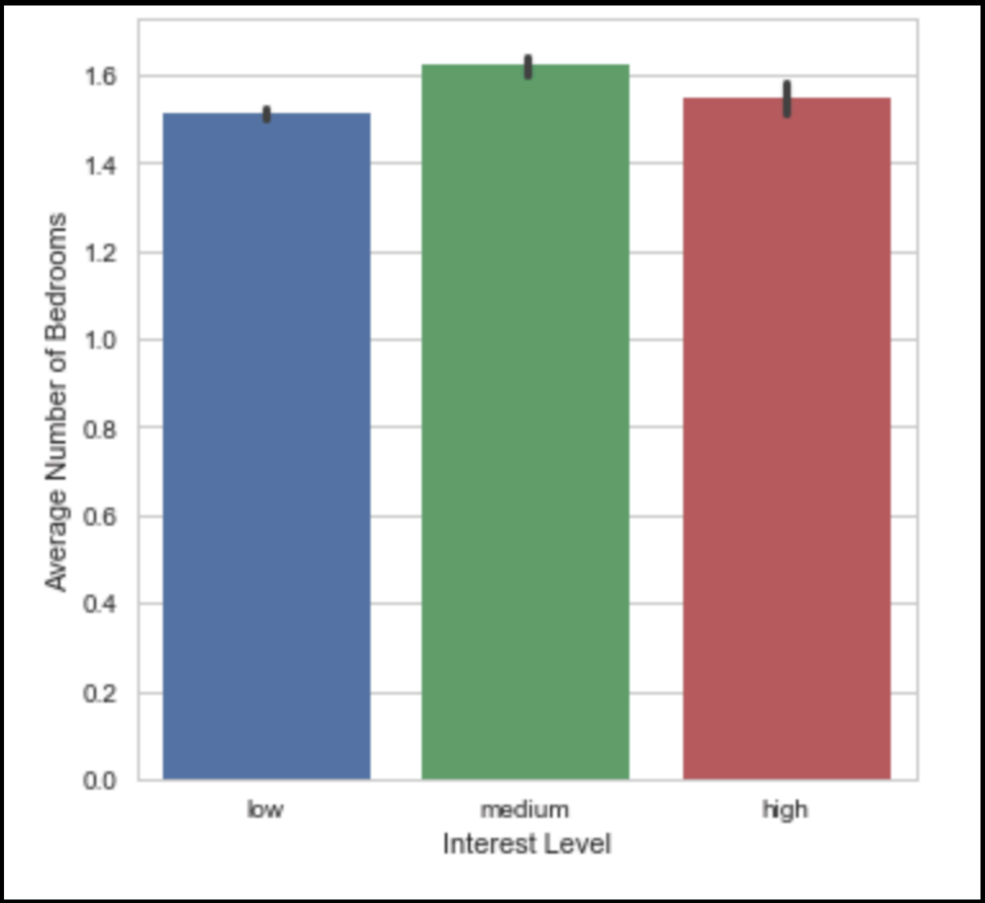
- This data set has 49,352 entries!

The Data - Digging In

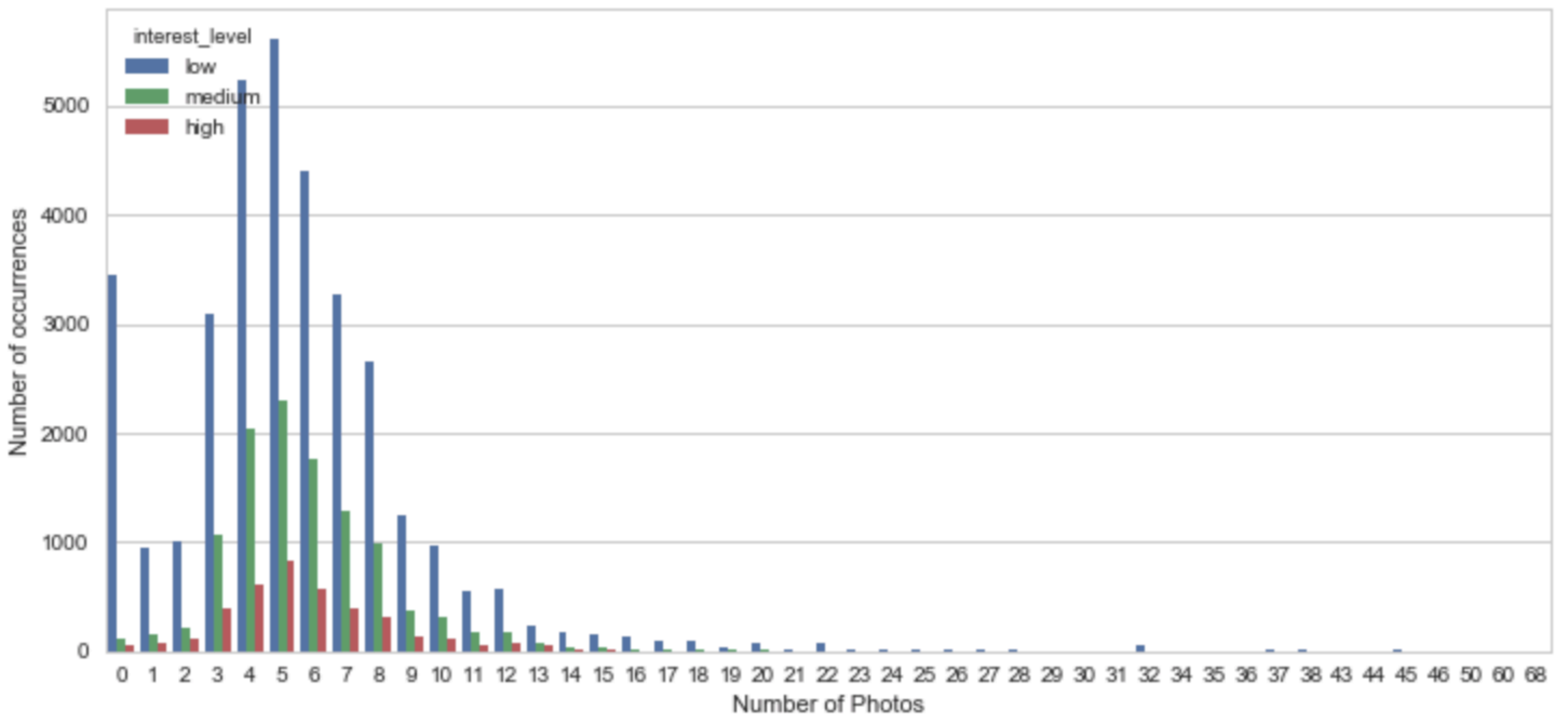


The Data - Digging In

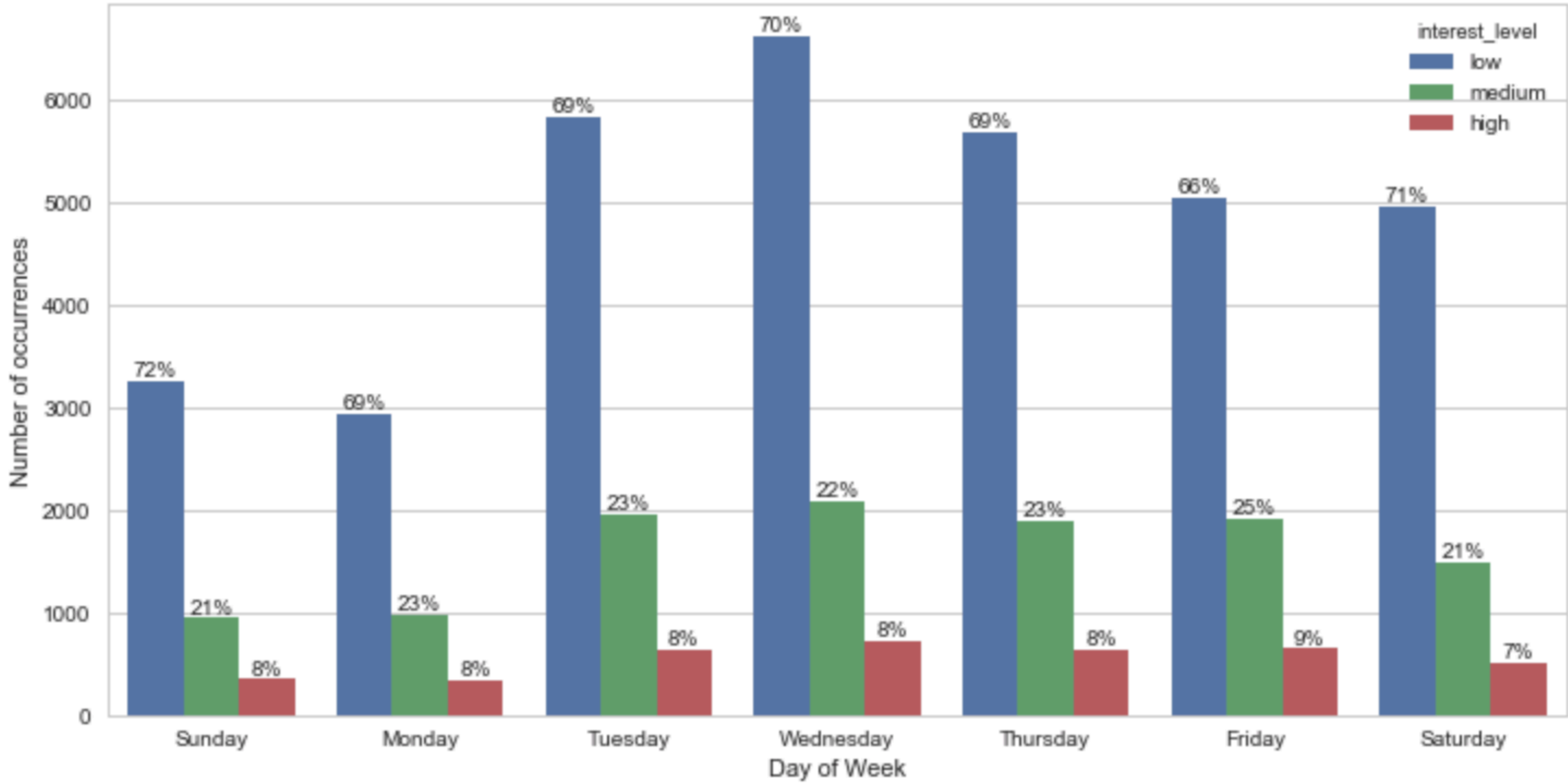




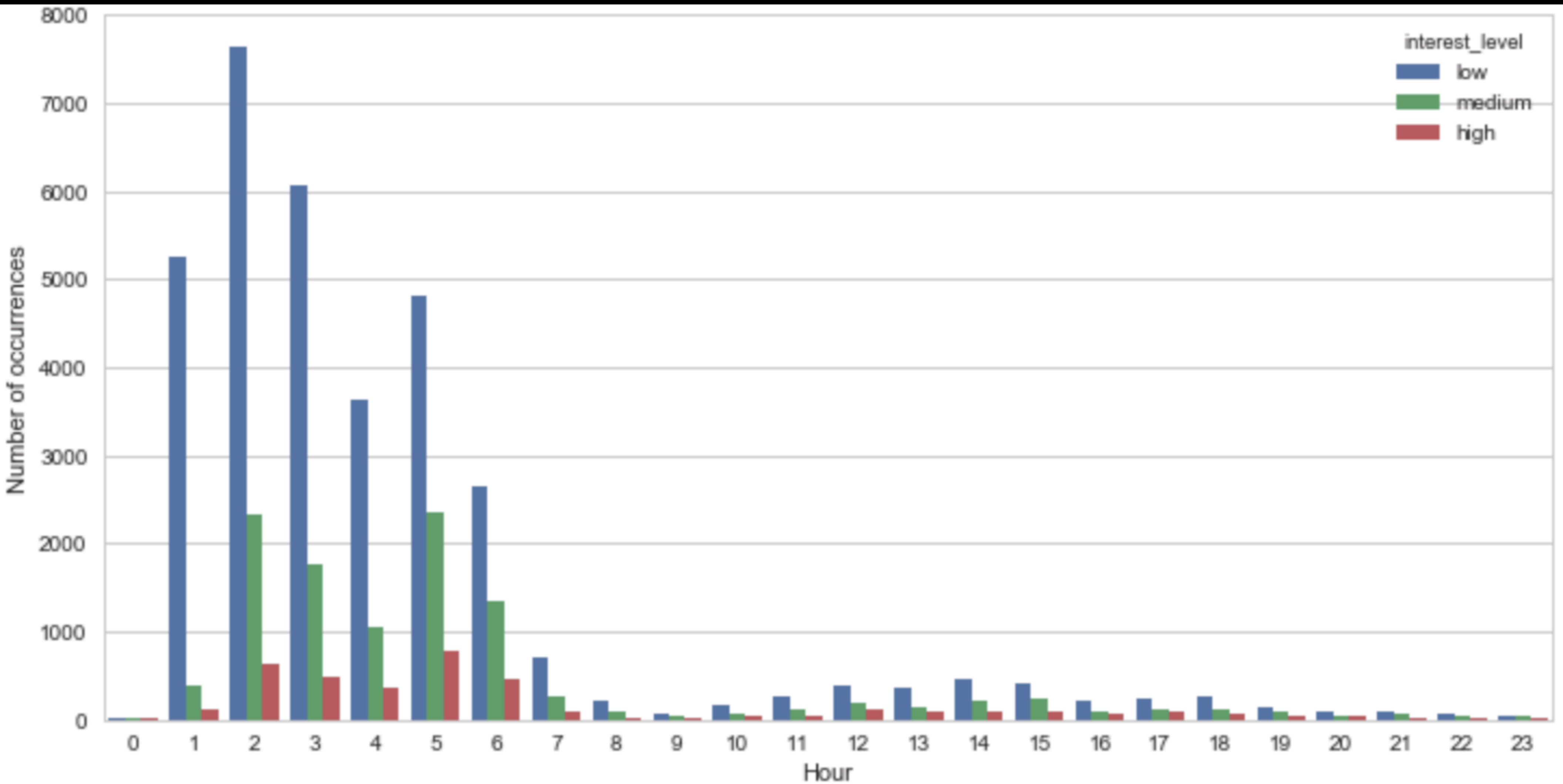
The Data - Digging In



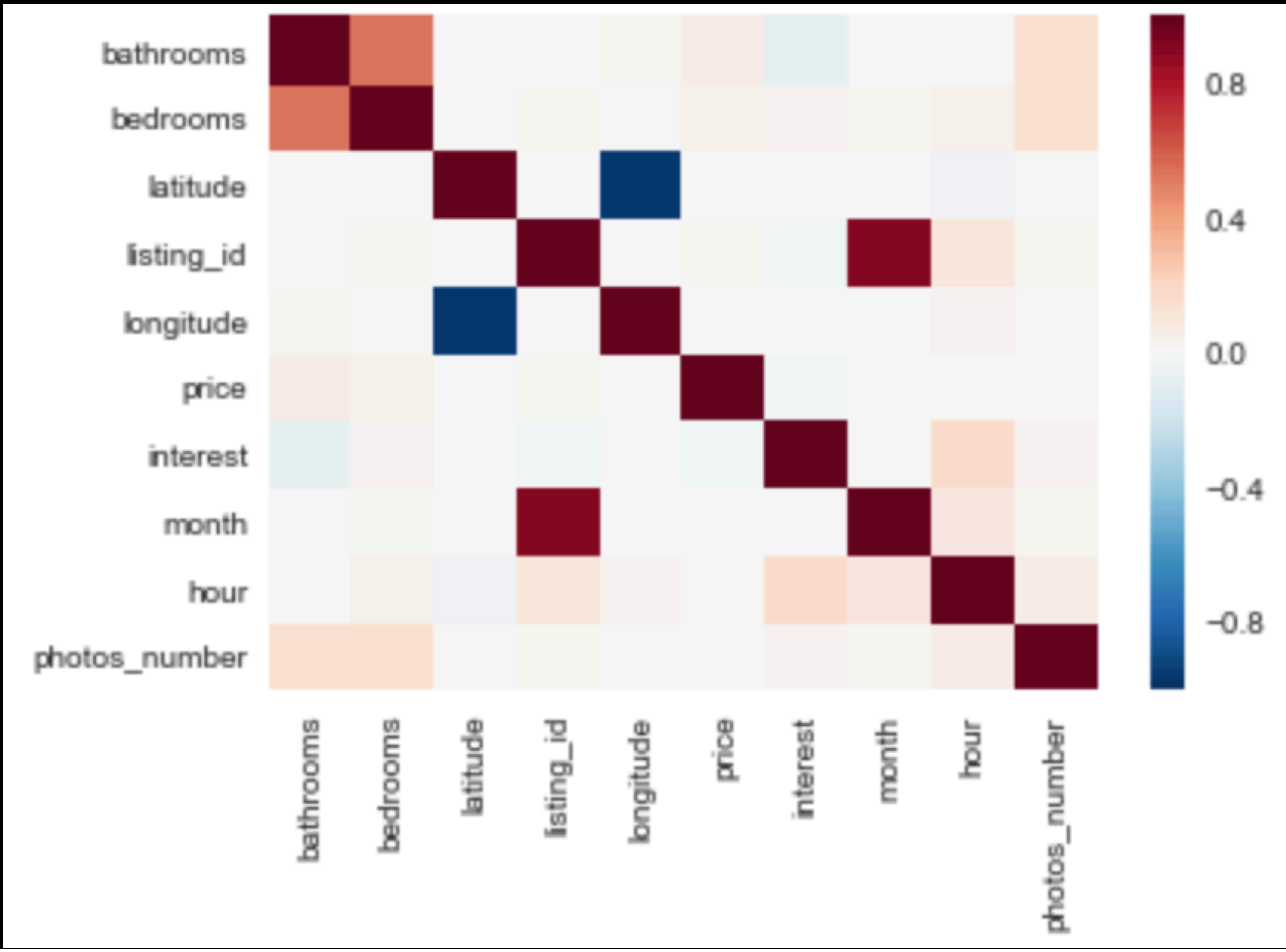
The Data - Digging In



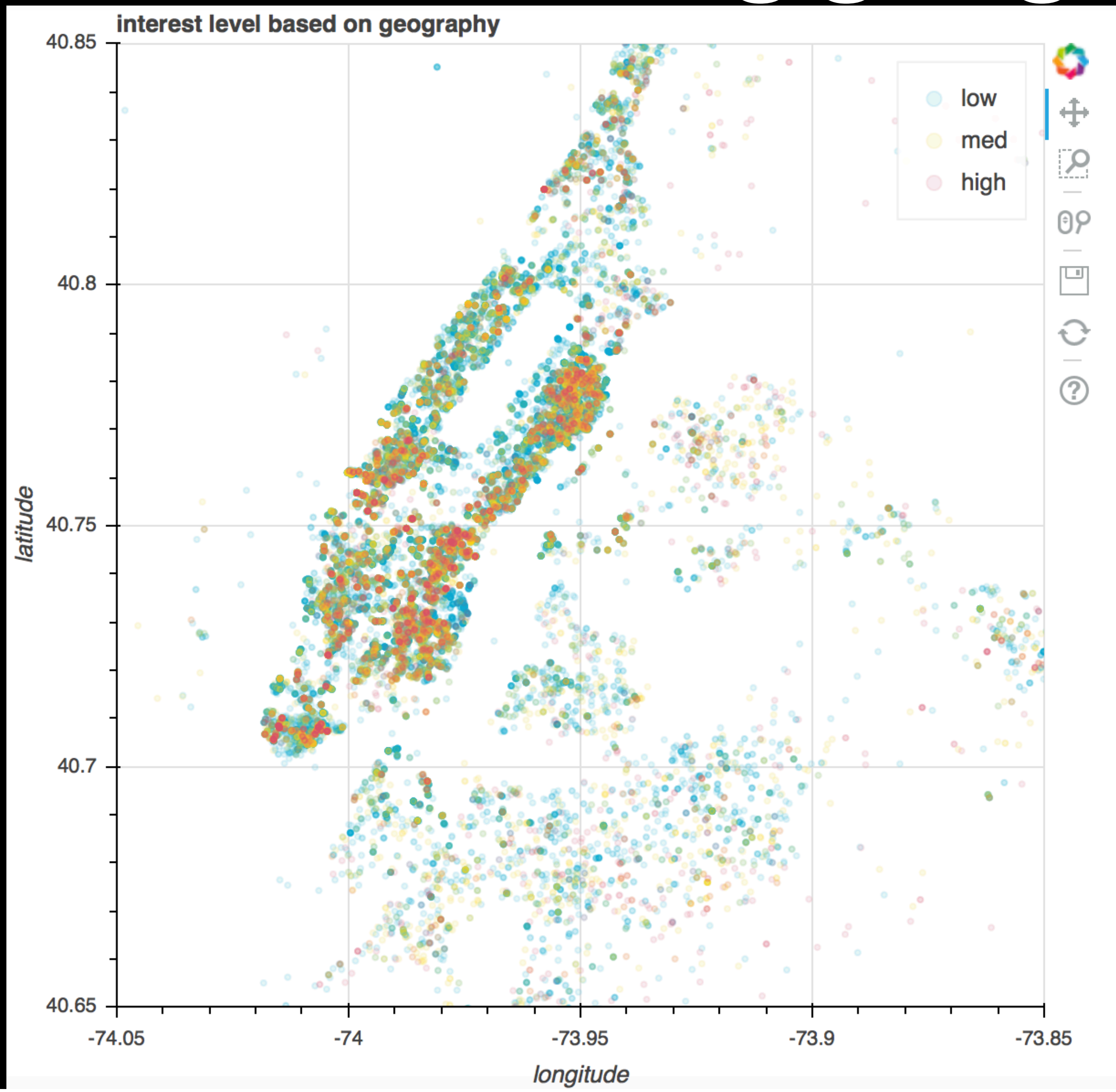
The Data - Digging In



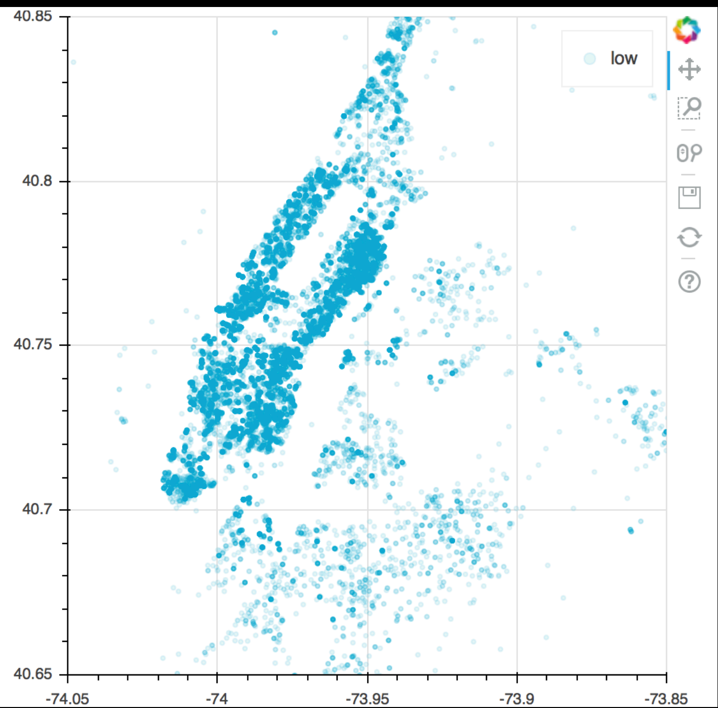
The Data - Digging In

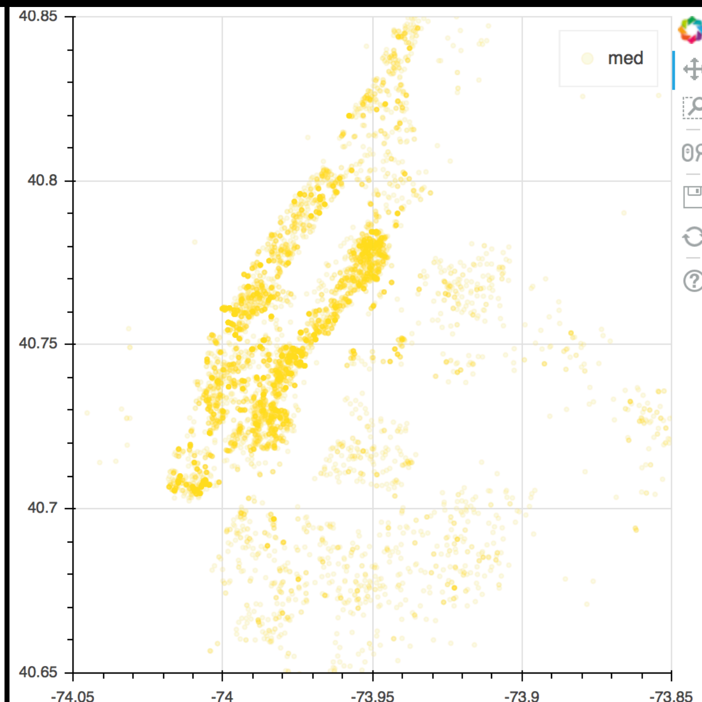


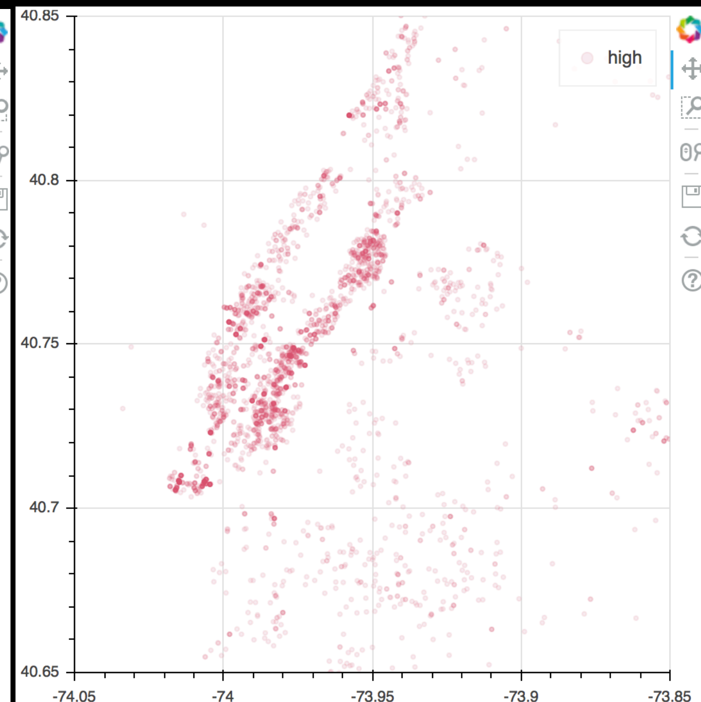
The Data - Digging In



The Data - Digging In







Key Findings

- **Neighborhood/Location is important**
  - Longitude and Latitude must be analyzed together
- **Hour of Posting is important**
  - Postings between 0200 - 0600 have greatest potential for interest
- **Number of photos**
  - Posting with 3 - 6 photos had most interest

Building a Model - KNN

KNN

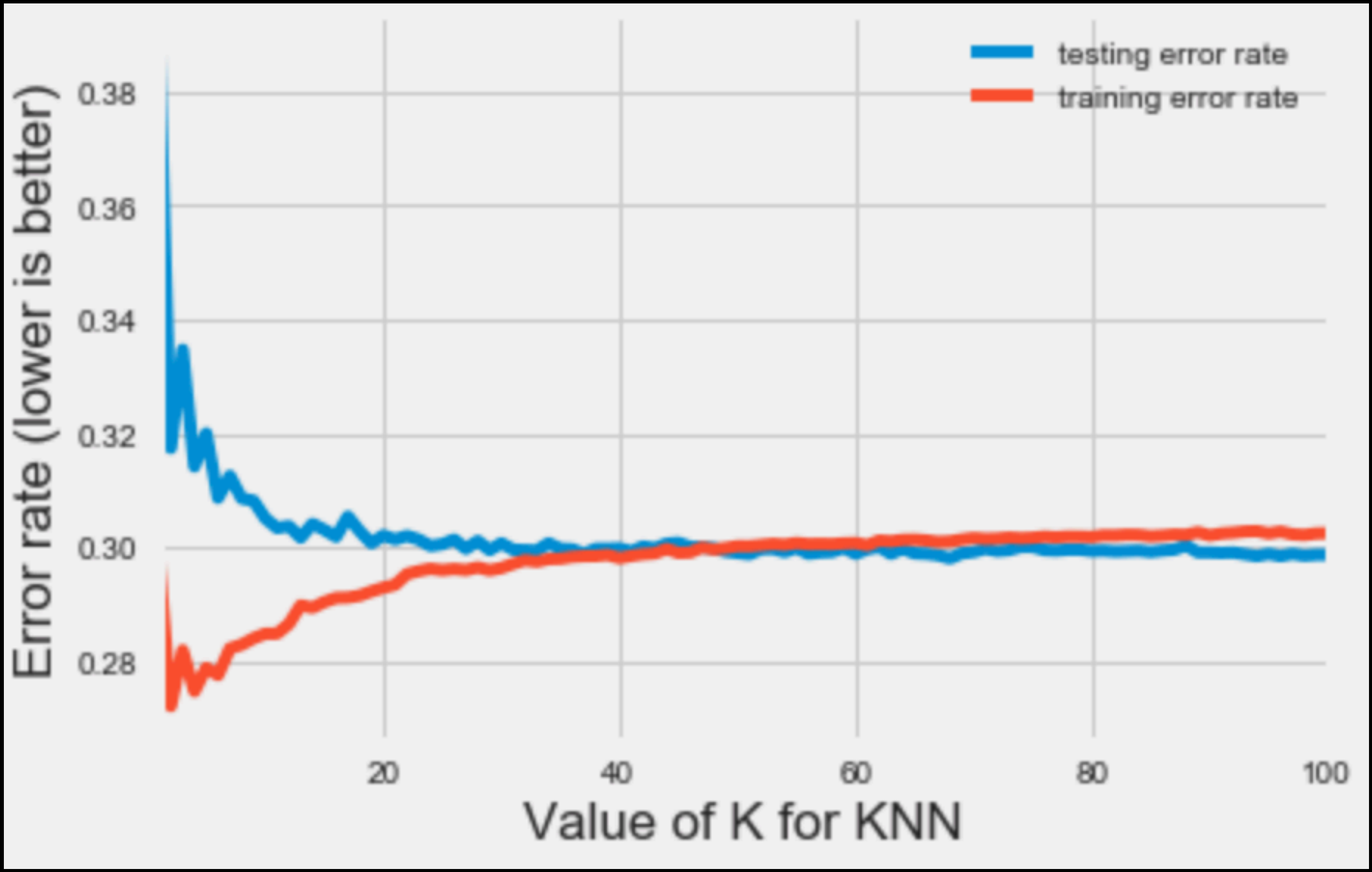
- Can Use Latitude and Longitude together to predict interest level
- Define interest level as a numerical classification
  - Low = 0
  - Med = 1
  - High = 2

Building a Model - KNN

```python
#STEP 1a: define X and y
X=pd.concat([train_data['latitude'],train_data['longitude']],axis=1)
y=train_data['interest']

# STEP 1b: split X and y into training and testing sets (using random_state for reproducibility)
from sklearn.model_selection import train_test_split
import scipy as sp
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=99)

#STEP 2: Choose and import the estimator
from sklearn.neighbors import KNeighborsClassifier

#STEP 3: Instantiate into a variable
# instantiate the estimator with K=68
knn = KNeighborsClassifier(n_neighbors=68)

#STEP 4: Fit model
knn.fit(X_train, y_train)

# STEP 5: test the model on the testing set
y_pred_class = knn.predict(X_test)

#STEP 6: Calculate the accuracy using the testing set predictions
print 'Testing Accuracy =', metrics.accuracy_score(y_test, y_pred_class)
# compute null accuracy
print 'Null Accuracy =' , y_test.value_counts().head(1) / len(y_test)
```

```
Testing Accuracy = 0.701977630086
Null Accuracy = 0    0.699384
Name: interest, dtype: float64
```

Building a Model - KNN




```
min(zip(testing_error_rate, k_range))
```

```
(0.29802236991408659, 68)
```

Building a Model - Logistic Regression

LogReg

- Can use hour, photos, location, price, to predict interest level
- Define interest level as a numerical classification
  - Low = 0
  - Med = 1
  - High = 2

Building a Model - Logistic Regression

```
#STEP 1a: define X and y
feature_cols = ['hour', 'price', 'photos_number','latitude', 'longitude']
X = train_data[feature_cols]
y = train_data.interest

# STEP 1b: split X and y into training and testing sets
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=99)

#STEP 2: Choose and import the estimator
from sklearn.linear_model import LogisticRegression

#STEP 3: Instantiate into a variable
logreg = LogisticRegression(C=1e9)

#STEP 4: Fit model
logreg.fit(X_train, y_train)

# STEP 5: test the model on the testing set
y_pred_class = logreg.predict(X_test)

#STEP 6: Calculate the accuracy using the testing set predictions
# compute testing accuracy
print 'Testing Accuracy =', metrics.accuracy_score(y_test, y_pred_class)
# compute null accuracy
print 'Null Accuracy =' , y_test.value_counts().head(1) / len(y_test)
```

```
Testing Accuracy = 0.699465067272
Null Accuracy = 0    0.699384
Name: interest, dtype: float64
```

Building a Model - Ensembling

```
from sklearn.ensemble import VotingClassifier
from sklearn import model_selection
# create the sub models
estimators = []
model1 = logreg
estimators.append(('logistic', model1))
model2 = knn
estimators.append(('KNN', model2))

# create the ensemble model
ensemble = VotingClassifier(estimators)
results = model_selection.cross_val_score(ensemble, train_data[feature_cols], y, cv=99)
print 'Testing Accuracy =', metrics.accuracy_score(y_test, y_pred_class)
# compute null accuracy
print 'Null Accuracy =' , y_test.value_counts().head(1) / len(y_test)
```

```
Testing Accuracy = 0.701977630086
Null Accuracy = 0    0.699384
Name: interest, dtype: float64
```

Building a Model - Comparison

- **Accuracy**
    - Overall, how often the classifier is correct
- **Null Accuracy**
    - How often you would be correct if you predicted the majority class (low interest)

Conclusions

- **KNN was the best predictor of interest**
- **Logistic Regression had no difference between accuracy and null accuracy**
- **Ensembling results were the same as KNN results.**
  - Indicates accuracy is dependent on KNN
  - Much slower process, KNN is best estimator

Conclusions

- **As a potential landlord, how can you increase interest levels (location not included)**
  - **Post in the morning**
    - Posting early makes your show up during searches
    - Sweet Spot: 0200 - 0600
  - **Post in the middle of the week**
    - Most people search during work or during the workweek
    - Sweet Spot: Tuesday - Thursday
  - **Have photos, but not too much**
    - Sweet Spot: 4 to 6 photos

QUESTIONS?