Tiffany Wilson

General Assembly - Santa Monica
Data Science Immersive
Summer 2016

Real Estate Property Description Analyzer

August 2016

Problem Statement

It is a Realtor's job to market a property listing to the best of their ability. Any tool available to help them sell a home quickly and for the highest price possible is of interest to Realtors - especially if it's easy and inexpensive to do!

Every listing should have a property description. We want to use property descriptions from closed sales to find keywords/phrases that may be highly correlated with closed MLS (Multiple Listng System) property listings (ie. sold) that have a high "Realtor Quality Score". Then using this information, we should be able to evaluate the property description for an existing "for sale" listing (ie active) and predict its "Realtor Quality Score".

Approach

1. **Geographic Areas:** Certain geographic areas have different attributes, from a real estate perspective. We identified 3 different areas to focus our efforts on, and these areas were chosen because of characteristics unique to the area (ie. home values, community features, etc.). The areas we chose are "The Hill" (Palos Verdes, Rolling Hills, etc.), South Bay beach cities (El Segundo, Manhattan Beach, Hermosa Beach, and

Redondo Beach), and the rest of the Greater South Bay (ie. Torrance, Hawthorne, Carson, etc.)

2. **Realtor Quality Score:** No score like this currently exists, but using my domain knowledge, I was able to identify the features of a closed property listing, which could be used to define and calculate a Realtor Quality Score.

3. **Naive Bayes Classifier:** A commonly used model for classifying freeform-text data is Naive Bayes classifier, and we use Scikit Learn's MultinomialNB class for our experiment.

Data Exploration & Preparation

Let's take a look at our data:

Property Descriptions

A typical MLS property description can be up to 1275 characters, but on average it is quite a bit shorter:

3

```
select geo_listings.geo_code as geo_code,
count(listings.property_description) as num_recs,
max(char_length(listings.property_description)) as max_pd_length,
round(avg(char_length(listings.property_description))) as avg_pd_length
from geo_listings join listings
on geo_listings.mls_id = listings.mls_id
group by geo_listings.geo_code;
```

Output pane

Data Output | Explain | Messages | History

	geo_code text	num_recs bigint	max_pd_length integer	avg_pd_length numeric
1	GSB	3413	1275	680
2	SBBeach	3199	1275	820
3	TheHill	1389	1275	869

There is a high likelihood that a property description will be provided (the rules of the MLS require a property description, and a Realtor can receive a reprimand for not having an appropriate property description), but we will check to see if any nulls exist.

```
In [13]: dfData.count()

Out[13]: geo_code                8054
         mls_id                  8054
         property_description    8001
         dtype: int64
```

```
In [14]: # Remove any rows with null values in the property_description
         dfCleaningData = dfData.dropna()
         dfCleaningData.count()

Out[14]: geo_code                8001
         mls_id                  8001
         property_description    8001
         dtype: int64
```

There were just over 50 property listings that didn't have a property description, so those records were removed.

Realtor Quality Score

We will need to calculate a "Realtor Quality Score", based on certain features of a closed listing: cumulative days on market (cdom), price margin, escrow length, sold terms and type of financing. Let's check this data:

```
dfData.count()
```

```
property_description      8001
cdom                      8001
list_price                8001
sold_price                8001
price_margin              8001
purchase_contract_date    8001
closed_sale_date          8001
escrow_length             8001
sold_terms                7598
financing                 7745
dtype: int64
```

We are missing values from sold_terms and financing, so we need to handle this. The approach we'll take is to fill the missing values with the most common value.

The most common sold_term is 'Standard Sale', making up 91% of the values in this column. We will use this value to fill in the missing sold_terms columns.

The most common financing value is 'Conventional', making up 56% of the values in this column. We will use this value to fill in the missing financing columns.

```
select count(listings.*) as total_listings,
(select count(*)
from listings
where listings.sold_terms = 'Standard Sale') as standard_sale_count
from listings;
```

Output pane

Data Output | Explain | Messages | History

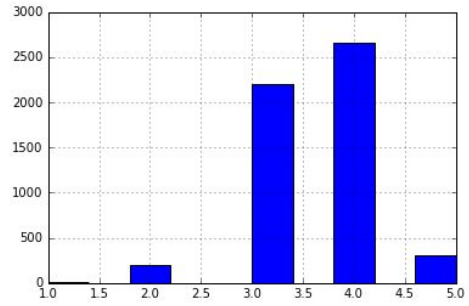
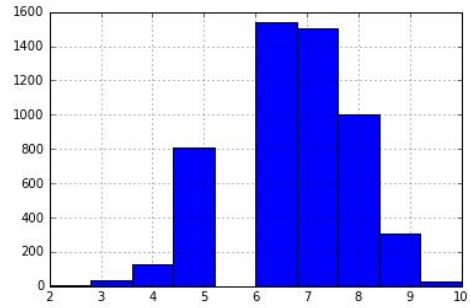
	total_listings bigint	standard_sale_count bigint
1	8054	7361

Now we will calculate a 5 and a 10 “Realtor Quality Score” for each record:

_length	sold_terms	financing	rqual_score5	rqual_score10
	Standard Sale	Cash	3	5
	Standard Sale	Conventional	4	8
	Standard Sale	Cash to New Loan	4	7
	Standard Sale	Cash	5	9
	Standard Sale	Cash	4	8

Let’s explore these values further:

5 Score	10 Score

`pd.Series(data=y5_train).hist()`
`<matplotlib.axes._subplots.AxesSubplot at 0x1311e96d0>`
 
	`pd.Series(data=y10_train).hist()`
	`<matplotlib.axes._subplots.AxesSubplot at 0x1344cfe50>`
	 

```
pd.Series(data=y5_train).value_counts()
```

```
4    2656
3    2204
5     304
2     195
1       1
dtype: int64
```

```
pd.Series(data=y10_train).value_counts()
```

```
6     1539
7     1502
8     1003
5      809
9      310
4      130
3       37
10      29
2        1
dtype: int64
```

Run Model & Evaluate Results

```
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
print 'X_train_counts.shape:' + str(X_train_counts.shape)

tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
print 'X_train_tfidf.shape: ' + str(X_train_tfidf.shape)

# Train our Naive Bayes model
clf5 = MultinomialNB().fit(X_train_tfidf, y5_train)
clf10 = MultinomialNB().fit(X_train_tfidf, y10_train)

# Test our NB model
X_test_counts = count_vect.transform(X_test)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)

predicted5 = clf5.predict(X_test_tfidf)
predicted10 = clf10.predict(X_test_tfidf)
```

```
X_train_counts.shape:(5360, 11164)
X_train_tfidf.shape: (5360, 11164)
```

We used Scikit Learn's MultinomialNB class to perform a Naive Bayes classifier, which is well suited for our text data. Before plugging it into the model though, we transformed the data to TF-IDF values.

Results

Using the pandas_confusion library, we produced a confusion matrix, and all of the associated metrics. Let's explore:

5 Score

Confusion Matrix:

Predicted	1	2	3	4	5	__all__
Actual						
1	0	0	0	1	0	1
2	0	0	13	79	0	92
3	0	0	105	947	0	1052
4	0	0	50	1288	0	1338
5	0	0	4	154	0	158
__all__	0	0	172	2469	0	2641

10 Score

Confusion Matrix:

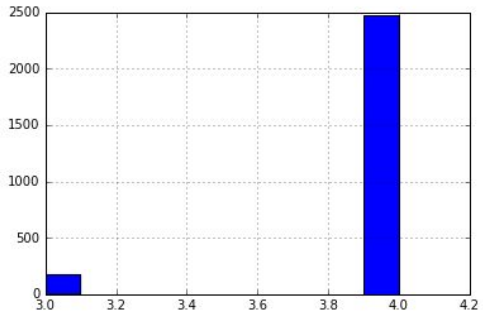
Predicted	2	3	4	5	6	7	8	9	10	__all__
Actual										
2	0	0	0	0	0	1	0	0	0	1
3	0	0	0	0	20	7	0	0	0	27
4	0	0	0	0	40	21	0	0	0	61
5	0	0	0	0	269	133	0	0	0	402
6	0	0	0	0	445	262	2	0	0	709
7	0	0	0	0	394	354	0	0	0	748
8	0	0	0	0	271	243	3	0	0	517
9	0	0	0	0	75	88	1	0	0	164
10	0	0	0	0	4	8	0	0	0	12
__all__	0	0	0	0	1518	1117	6	0	0	2641

```
pd.Series(data=predicted5).value_counts()
```

```
4    2469
3     172
dtype: int64
```

```
pd.Series(data=predicted5).hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x12fe7fd50>
```

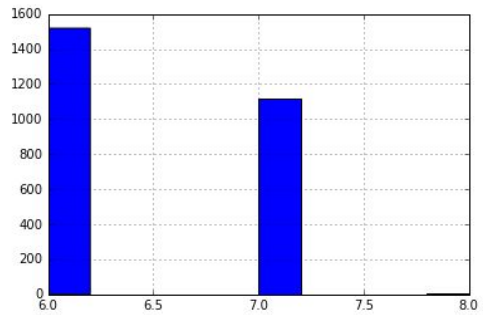



```
pd.Series(data=predicted10).value_counts()
```

```
6    1518
7    1117
8       6
dtype: int64
```

```
pd.Series(data=predicted10).hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x12ef0da90>
```



Hmmmm... the results are not looking like they support the hypothesis.

Let's check some of the metrics:

5 Score	10 Score
Overall Statistics: Accuracy: 0.527451722832 95% CI: (0.50820035491806381, 0.54664216487046713) No Information Rate: ToDo P-Value [Acc > NIR]: 1.0 Kappa: 0.0557088483748 Mcnemar's Test P-Value: ToDo	Overall Statistics: Accuracy: 0.303672851193 95% CI: (0.28617135291477369, 0.32161055490754187) No Information Rate: ToDo P-Value [Acc > NIR]: 1.0 Kappa: 0.0401584900873 Mcnemar's Test P-Value: ToDo

Ack! These values are certainly concurring with the results we see in the confusion matrix. The Accuracies for the 5 and 10 score models are 53% and 30% respectively - quite poor.

Also note that the P-Value is 1.0, which indicates to us that we should accept the null hypothesis, which is that the property descriptions can not be used to predict the Realtor Quality Score.

	1	2	3	4	5
Sensitivity Rate	0	0	0.09	0.96	0
Specificity	1	1	0.96	0.09	1
False Omission Rate	0.00038	0.035	0.38	0.29	0.06
False Discovery Rate	nAn	nAn	0.39	0.48	nAn

These values are also quite poor. The only good results are with the Sensitivity Rate and Specificity, for Realtor Quality Scores of 4 and 3 (respectively). But unfortunately, this is most certainly not good enough.

The results of the 10 score model concur with the results of the 5 score model, so we won't examine those further.

Conclusion

Based on the analysis of our result metrics, it is clear that the property description for a property listing is not a good predictor of the Realtor Quality Score.

But Wait...There is More!!

I wanted to explore NLP further, so I "played" with the following NLP concepts to further explore the property description:

Topic Modeling

- I used LDA (Latent Dirichlet Allocation) and LSA (Latent Semantic Allocation) to perform some topic modeling on the dataset.
  - Scikit Learn - excellent documentation.
  - TrucatedSVD for LSA and LatentDirichletAllocation for LDA.
- LDAvis to visualize LDA results.
- Could not find a suitable tool to visualize LSA results.
- LDA using TF (term frequency) produced "better" results than using TFIDF (Term Frequency-Inverse Document Frequency).

Word2Vec

- Load the model using tokenized sentences.
- Built a model using unigrams and bigrams.
- Perform "natural language queries" against the Word2Vec model.
- FUTURE: experiment with tokenizing formatted data (ie. data in database) and load into Word2Vec to see if it can be used to produce property listing results based on more natural language queries:
  - "I want a house in Torrance with 2 bedrooms 3 bathrooms near North High school"