Hotel Review Sentiment Analysis

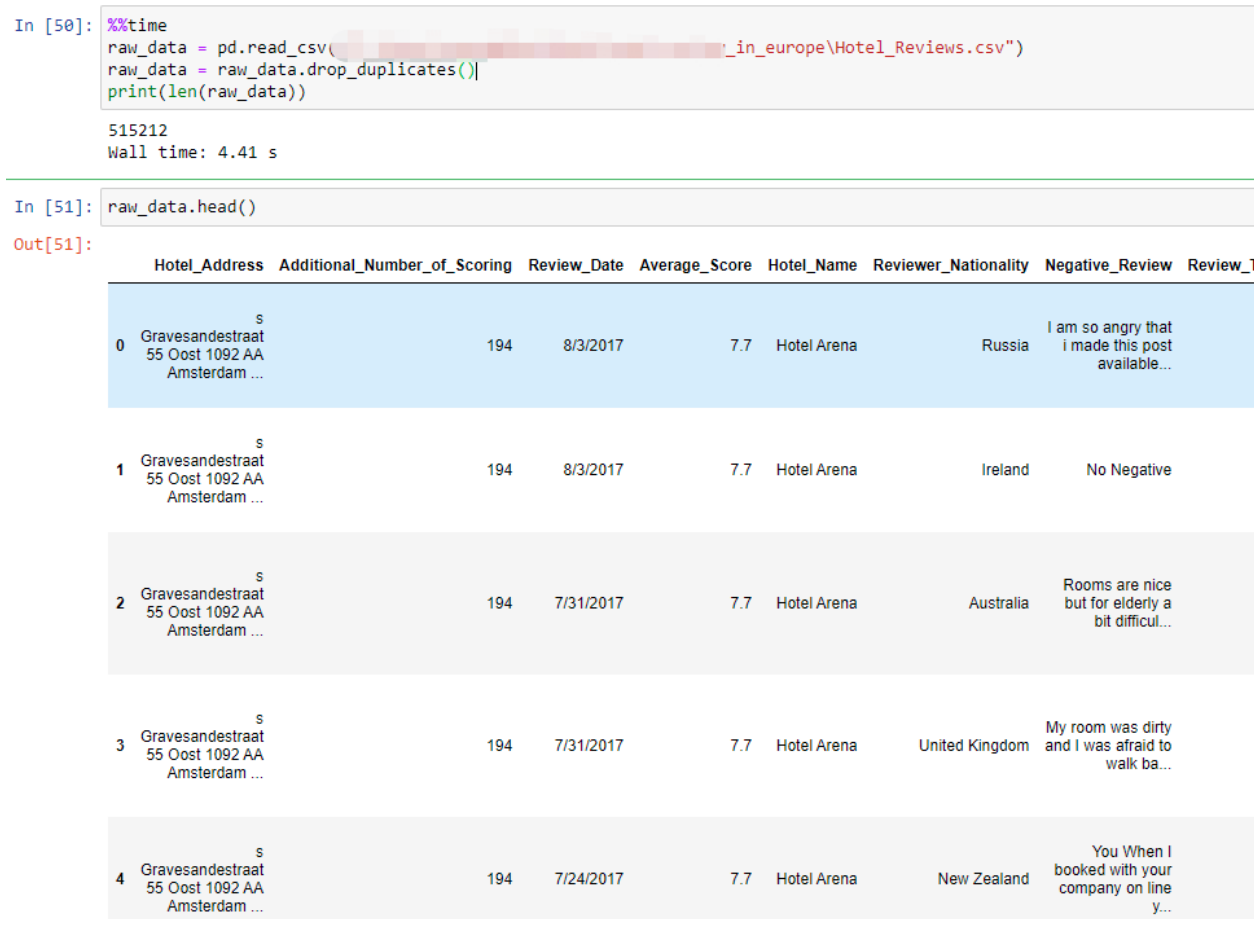
Team Mr. Goose
Runyu Cao (runyuc2@illinois.edu) (Captain)
Weijie Wang (weijiew2@illinois.edu)
Qijing Zhu (qijingz2@illinois.edu)

Project Progress Report

Current Progress

Dataset Preparation

We use the dataset from Hotel Reviews Data in Europe. The data size is 515k. For the purpose of trying things out, we only took the first 10k rows.



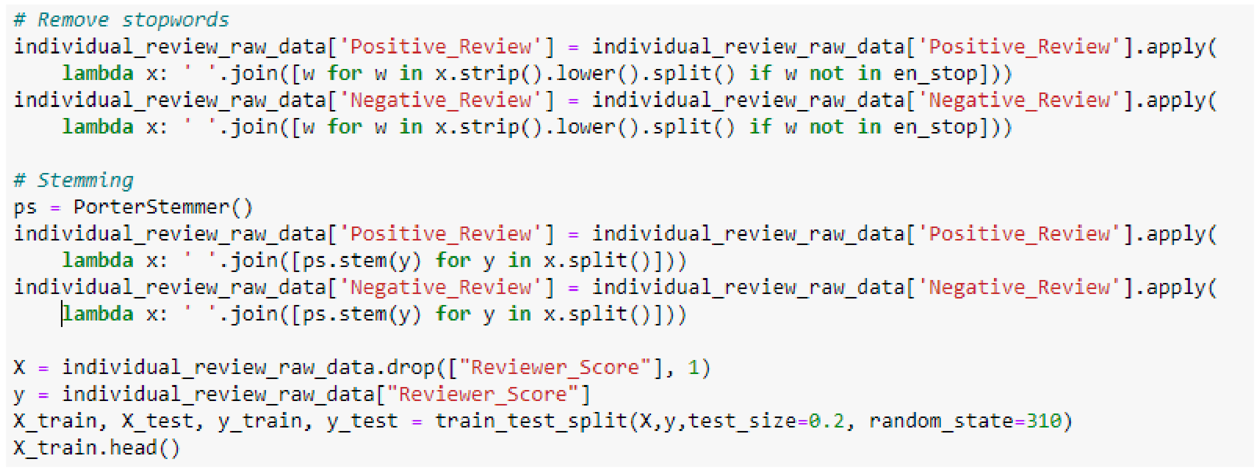
```
In [50]: %%time
         raw_data = pd.read_csv([redacted]_in_europe\Hotel_Reviews.csv")
         raw_data = raw_data.drop_duplicates()
         print(len(raw_data))

         515212
         Wall time: 4.41 s
```

```
In [51]: raw_data.head()
Out[51]:
```

	Hotel_Address	Additional_Number_of_Scoring	Review_Date	Average_Score	Hotel_Name	Reviewer_Nationality	Negative_Review	Review_T
0	s Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	8/3/2017	7.7	Hotel Arena	Russia	I am so angry that i made this post available...	
1	s Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	8/3/2017	7.7	Hotel Arena	Ireland	No Negative	
2	s Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	7/31/2017	7.7	Hotel Arena	Australia	Rooms are nice but for elderly a bit difficul...	
3	s Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	7/31/2017	7.7	Hotel Arena	United Kingdom	My room was dirty and I was afraid to walk ba...	
4	s Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	7/24/2017	7.7	Hotel Arena	New Zealand	You When I booked with your company on line y...	

Data Cleaning

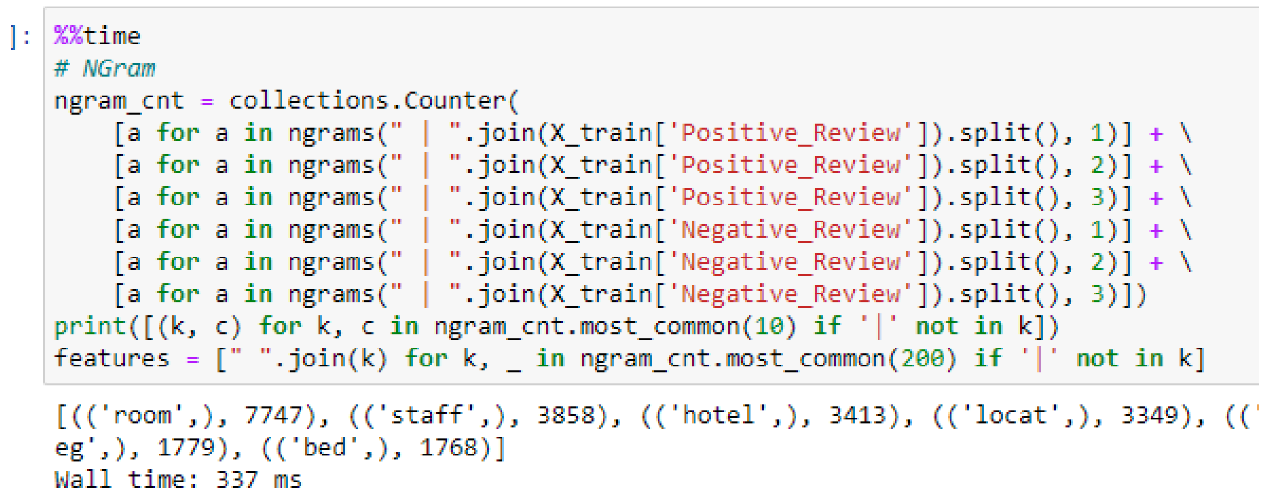
We eliminated irrelevant columns, made everything into lower case, split them into words, removed English stopwords, and stemmed the words. The data was later split into training and testing dataset with rate 80%-20%.



```
# Remove stopwords
individual_review_raw_data['Positive_Review'] = individual_review_raw_data['Positive_Review'].apply(
    lambda x: ' '.join([w for w in x.strip().lower().split() if w not in en_stop]))
individual_review_raw_data['Negative_Review'] = individual_review_raw_data['Negative_Review'].apply(
    lambda x: ' '.join([w for w in x.strip().lower().split() if w not in en_stop]))

# Stemming
ps = PorterStemmer()
individual_review_raw_data['Positive_Review'] = individual_review_raw_data['Positive_Review'].apply(
    lambda x: ' '.join([ps.stem(y) for y in x.split()]))
individual_review_raw_data['Negative_Review'] = individual_review_raw_data['Negative_Review'].apply(
    lambda x: ' '.join([ps.stem(y) for y in x.split()]))

X = individual_review_raw_data.drop(["Reviewer_Score"], 1)
y = individual_review_raw_data["Reviewer_Score"]
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=310)
X_train.head()
```

Feature Engineering (v1)

We then performed 1-3 gram(s) on the positive and negative reviews and picked 200 of the most common grams as sentimental analysis features.

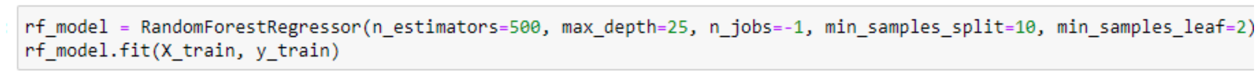


```
]: %%time
   # NGram
   ngram_cnt = collections.Counter(
       [a for a in ngrams(" | ".join(X_train['Positive_Review']).split(), 1)] + \
       [a for a in ngrams(" | ".join(X_train['Positive_Review']).split(), 2)] + \
       [a for a in ngrams(" | ".join(X_train['Positive_Review']).split(), 3)] + \
       [a for a in ngrams(" | ".join(X_train['Negative_Review']).split(), 1)] + \
       [a for a in ngrams(" | ".join(X_train['Negative_Review']).split(), 2)] + \
       [a for a in ngrams(" | ".join(X_train['Negative_Review']).split(), 3)])
   print([(k, c) for k, c in ngram_cnt.most_common(10) if '|' not in k])
   features = [" ".join(k) for k, _ in ngram_cnt.most_common(200) if '|' not in k]
```

```
[(('room',), 7747), (('staff',), 3858), (('hotel',), 3413), (('locat',), 3349), (('
eg',), 1779), (('bed',), 1768)]
Wall time: 337 ms
```

Model Training (v1)

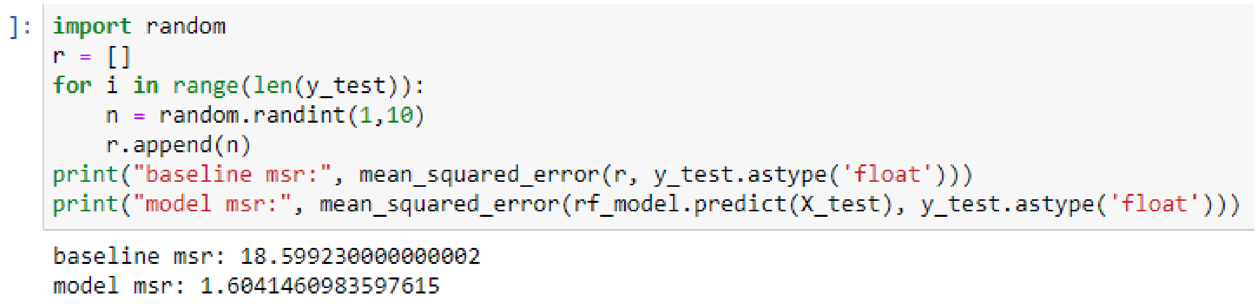
We used the random forest regressor model as our initial model for it's robustness and ability to play a role in further feature selection.



```
rf_model = RandomForestRegressor(n_estimators=500, max_depth=25, n_jobs=-1, min_samples_split=10, min_samples_leaf=2)
rf_model.fit(X_train, y_train)
```

Model Evaluation (v1)

We compared the model msr with baseline, and based on the data our model shows it is learning (indicating signs of learning). We also tried some of our own inputs and the results looked good.



```
]: import random
   r = []
   for i in range(len(y_test)):
       n = random.randint(1,10)
       r.append(n)
   print("baseline msr:", mean_squared_error(r, y_test.astype('float')))
   print("model msr:", mean_squared_error(rf_model.predict(X_test), y_test.astype('float')))

   baseline msr: 18.599230000000002
   model msr: 1.6041460983597615
```

Things we learned so far

We got a valuable chance to get our hands dirty on a real dataset and learned how to use Python data processing libraries like Pandas and Scikit-learn.

Pending Tasks

The pending tasks include further feature engineering, training other classifiers, fine-tuning classifier parameters, and comparing the performance of each classifier. A stretch goal is to build the web tooling interface for users to get the label for new comments produced by the trained model.

Challenges

The main challenge we faced was the large size of the dataset, 238 MB, 515k rows, which leads to exceptionally long time in computing n-grams. We can solve this challenge by computing n-grams on only a random subset of the whole dataset and use that as the initial features.