Steps

1. Data Preprocessing

   1.1 Data Cleaning

   - Process emojis: :(and): to Frowny // :), :D, (: to Smiley // ;) to Winky
   - Replace "&" to "and"
   - Remove extra punctuations

   Some sentences end with several punctuations, like !!!(exclamation), ……(period)

   - Remove contraction

   'm to am // won't to will not // He's to He is

   - Remove extra spaces
   - Remove repeated characters whose length >2

   Some words are mis-spelled, like 'Amaaaaaazing'. I convert it to 'Amazing'.

   1.2 Create new var 'sentiment': positive 1 // negative 0

   If star rating is highly positive (4/5), sentiment=1

   If star rating is highly negative (1/2), sentiment = 0

2. Extract Opinion Segments and Sentiment Analysis – Simple Sentiment Classification

   2.1 Extract Opinion Segments using a Regex pattern parser

   grammar = r"""

   Nominal:{<NN\w?\w?>?<NN\w?\w?>}

   o1: {<DT>?<Nominal><CC>?<DT>?<Nominal>?<VB\w?>+<RB\w?>*<JJ\w?>+<CC>?<JJ\w?>*}

   RBJJN: {<RB\w?>*<JJ\w?>+<Nominal>}

   o2: {<RBJJN>+(<,>?<CC>?<RBJJN>)*}

   """

   O1 sample: The chocolate cake and the apple pie are extremely delicious, super good-looking and cheap.

   O2 sample: very clean bed sheet, clean windows and nice showers.

   Example from dataset:

   pattern 1: ('The', 'DT'), ('chicken', 'NN'), ('and', 'CC'), ('vegetables', 'NNS'), ('were', 'VBD'), ('nice', 'JJ'), ('and', 'CC'), ('fresh', 'JJ')

   pattern 2: ('great', 'JJ'), ('service', 'NN'), ('and', 'CC'), ('tasty', 'JJ'), ('fod', 'NN')

2.2 Compute Polarity Score for matched segments

As we know, positive polarity score means positive sentiment, and negative polarity score means negative. And the original dataset has given us the real sentiment from authors of reviews. If the computed polarity score for extracted segments is positive, and the sentiment given by user is also positive, there is a match. A match means the opinion mining can show the real attitude from user. Although it is an NLP method, it is also a way to conduct sentiment classification.

```
------------------------------------------------
The percentage of matched positive reviews = 30%
------------------------------------------------
The percentage of matched negtive reviews = 21%
```

3. Sentiment Classification with Machine Learning Methodologies

3.1 Counting Word Frequency

Count the frequency of each word after lemmatizing the word.

3.2 Features Engineering

Transform raw data into features.

	aaron	aback	abandon	abe	abide	ability	abita	able	abodova	abound	...	ziped
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0

5 rows × 10716 columns

3.3 Apply Supervised Learning Classifiers

Before applying classifiers, I split the dataset into training set and test set by 70% and 30%.

For each classifier, I perform parameter configuration to select the best parameter. It can help with the problem of over-fitting.

Classifier	Accuracy on Training Set	Accuracy on Test Set
Logistic Regression	0.998	0.852
Decision Tree	0.883	0.764
Random Forest	0.772	0.764
KNN	0.852	0.72

The classification report for Logistic Regression Classifier:

```
Logistic Regression-Average accuracry score on the train dataset is: 0.998
Logistic Regression-Average accuracry score on the test dataset is: 0.852
---------------------------------------------------------------------------
             precision    recall  f1-score   support

          0       0.86      0.48      0.62        62
          1       0.85      0.97      0.91       188

avg / total       0.85      0.85      0.84       250
```

4. Topic Modeling

Latent Dirichlet allocation (LDA) is a topic model that generates topics based on word frequency from a set of documents. LDA is particularly useful for finding reasonably accurate mixtures of topics within a given document set. And I get help from the website https://rstudio-pubs-static.s3.amazonaws.com/79360_850b2a69980c4488b1db95987a24867a.html

4.1 Cleaning documents

Tokenizing: converting a document to its atomic elements.

Stopping: removing meaningless words.

Stemming: merging words that are equivalent in meaning.

4.2 Constructing a document-term matrix

4.3 Applying the LDA model

The dataset contains 1000 lines of review, so the number of topics is assigned 30.

4.4 Examining the results

Each line is a topic with individual topic terms and weights

```
[(13,
  '0.021*"place" + 0.017*"fod" + 0.012*"restaurant" + 0.011*"time" + 0.007*"peop
 (26,
  '0.017*"fod" + 0.009*"chicken" + 0.008*"place" + 0.007*"thing" + 0.007*"god" +
 (2,
  '0.015*"place" + 0.013*"time" + 0.007*"burger" + 0.007*"location" + 0.006*"ser
 (15,
  '0.026*"place" + 0.015*"time" + 0.012*"fod" + 0.008*"god" + 0.008*"service" +
 (28,
  '0.016*"place" + 0.009*"time" + 0.007*"fod" + 0.007*"sauce" + 0.006*"night" +
```

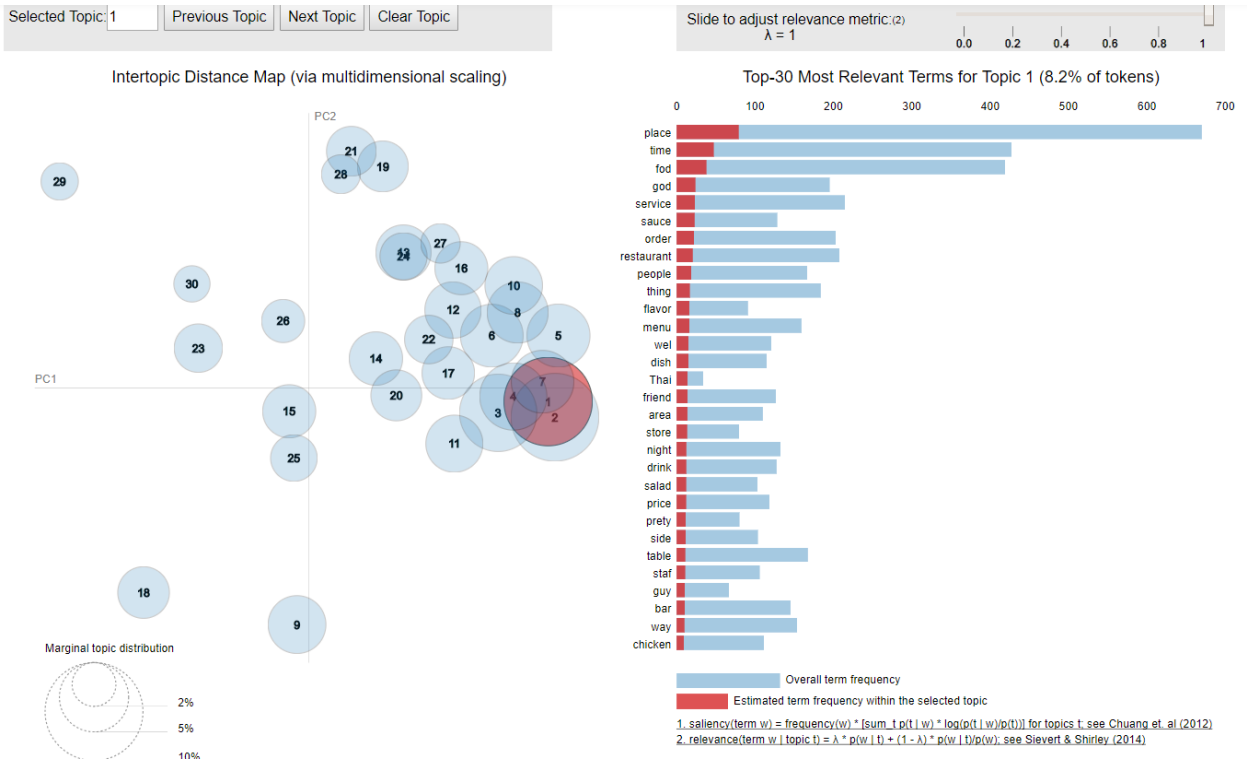
Let's look at the top 10 terms of this topic. The words 'service', 'window', 'car', 'vehicle' create a topic related to 'car' or 'body shop'.

```
model.show_topic(1, topn=10)
```

```
[('time', 0.014544184),
 ('place', 0.013783222),
 ('service', 0.0068973554),
 ('order', 0.006851297),
 ('window', 0.006315692),
 ('year', 0.006091858),
 ('car', 0.0060835225),
 ('vehicle', 0.0055893688),
 ('phone', 0.00551071),
 ('thing', 0.005392492)]
```

4.5 Dynamic Visualization of Topic Words using pyLDAvis

In order to visualize words in different topics, I generate a dynamic visualization interface, which shows the most relevant terms of different topics.



The output of topic modeling is not good enough. And the reason will be discussed in the error analysis part.

Error Analysis

1. The accuracy of opinion mining and sentiment analysis part is 30% and 21% separately for positive reviews and negative ones.

It has something to do with the extracted segments. I extract the segments with Regex pattern parser, but negative reviews can also be extracted as positive one, like [('prety', 'JJ'), ('smoky', 'JJ'), ('inside', 'NN')]. This segment express a negative attitude, but it is extracted from positive review set.

Besides, due to the large amount of mis-spelling words, the accuracy of extracted positive sentiments can be classified as neutral or negative, like the difference between 'god' and 'good', from [('Al', 'NNP'), ('were', 'VBD'), ('amazingly', 'RB'), ('god', 'JJ')]. The user would say that 'All were amazingly good.' 'Good' is positive word, but 'god' is a neutral word.

2. The output of topic modeling is not good enough. And the reason for it can also be the mis-spelling words. 'Fod' and 'food'. And 'Prety' is a adv, but it is classified as a noun in the topic modeling part.

Future Improvement Ideas

The above two issues are both coming from the mis-spelling words to some extent. Therefore, in the future work, spelling correction will be the most important part.

Words plays an important part in the analysis. But I am not able to conduct spelling correction so far.

- Autocorrect lib is accessible, but it does not work well. For example, 'fod' cannot be detected.

```
print(spell('fod'))
```

```
fod
```

- Enchant is a great lib for spelling correction, but it can support 32-digit system only.

In future, I will look for other useful spelling correction methods to update my research.