Sentiment Analysis Model

By: Cindy Suyitno

Layout

- Introduction
- Database Information
- Data Cleaning
- Data Preprocessing
- Data Modelling
- Deployment
- Conclusion and Suggestion

1. Introduction

Why sentiment analysis?

➔ **Definition**

Sentiment analysis is an automatic process that uses artificial intelligence to assign a value from +1 (extremely positive) to -1 (extremely negative) to bits of text.

➔ **Function**

Through sentiment analysis, business owners can capture the emotion behind a star rating and begin to understand what attributes contributed to or detracted from a positive experience at the business. For example, a review can still be rated five stars and mention that the burgers were burnt.

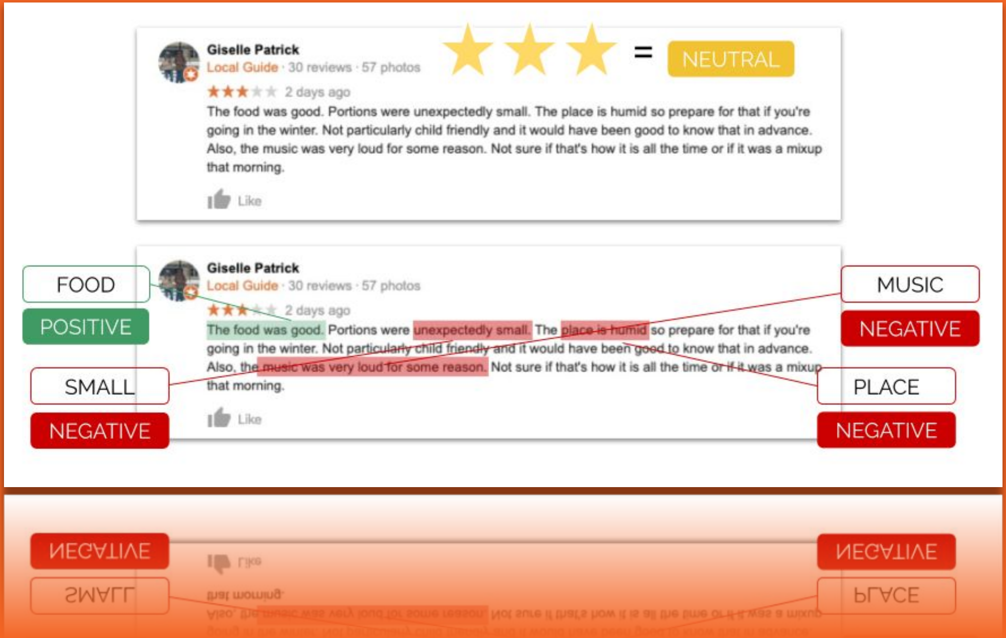


Image source: https://www.vendasta.com/blog/sentiment-analysis-turns-reviews-into-insights/

Customers
leave tons of advice, reviews, complaints, and appreciation in a business portal.

Do we need to read them one by one?

In this project we will use amazon product reviews to get sentiment analysis model.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 459436 entries, 0 to 459435
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   overall         459436 non-null  float64
 1   verified        459436 non-null  bool   
 2   reviewTime      459436 non-null  object 
 3   reviewerID      459436 non-null  object 
 4   asin            459436 non-null  object 
 5   style           234401 non-null  object 
 6   reviewerName    459412 non-null  object 
 7   reviewText      459370 non-null  object 
 8   summary         459380 non-null  object 
 9   unixReviewTime  459436 non-null  int64  
 10  vote            127853 non-null  object 
 11  image           1508 non-null    object 
dtypes: bool(1), float64(1), int64(1), object(9)
memory usage: 42.5+ MB
None
```

2. Database Info

The columns that are used:

- ➔ **reviewTime:** time when the reviews are added
- ➔ **reviewText:** customer reviews text
- ➔ **overall:** overall ratings by customers

Data source: http://deepyeti.ucsd.edu/jianmo/amazon/index.html

3. Data Cleaning

Change data type → Drop null values
Drop duplicated values

reviewTime
03 11, 2014
02 23, 2014
02 17, 2014
02 17, 2014
10 14, 2013

date
2014-03-11
2014-02-23
2014-02-17
2014-02-17
2013-10-14

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 459436 entries, 0 to 459435
Data columns (total 12 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   overall        459436 non-null  float64
 1   verified       459436 non-null  bool
 2   reviewTime     459436 non-null  object
 3   reviewerID     459436 non-null  object
 4   asin           459436 non-null  object
 5   style          234401 non-null  object
 6   reviewerName   459412 non-null  object
 7   reviewText     459370 non-null  object
 8   summary        459380 non-null  object
 9   unixReviewTime 459436 non-null  int64
 10  vote           127853 non-null  object
 11  image          1508 non-null    object
dtypes: bool(1), float64(1), int64(1), object(9)
memory usage: 42.5+ MB
None
```

4. Data Preprocessing Timeline

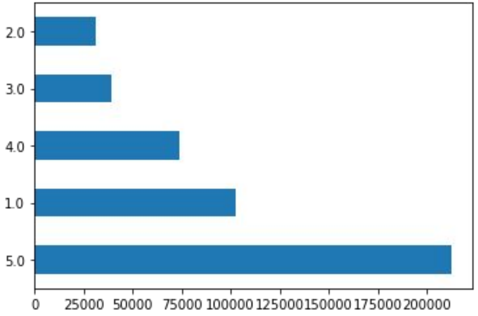
Checking Imbalance

The data is imbalance
(higher 5 ratings)

```
df_['label'].value_counts()

2    272625
0    130249
1     38178
Name: label, dtype: int64
```

df['rating'].value_counts().plot(kind='barh')



def get_rating_cat(rating)

Labelling Data

The ratings will be divided
into 3 sentiment category:
Negative: rating 1-2
Neutral: rating 3
Positive: rating 4-5

undersampling.RandomSampler(), train_test_split()

Splitting Data

1. Split the data into X,y
2. Undersampling X,y
3. Split into half since the data is so big
4. Split into train-test data (ratio: 80-20)

TF-IDF (Term Frequency-Inverse Document Frequency)

Feature Extraction

Vectorize the review text
into a usable vector

Data Visualization

Using wordclouds for text analysis (the bigger the text, the more times it appears in the reviews) we can see that lots of reviews are in neutral terms.

4. Modelling

- ➔ Decision Tree Classifier
- ➔ Random Forest Classifier
- ➔ Logistic Regression (multiclass='multinomial', solver='lbfgs')

```
DecisionTreeClassifier()
        Training time: 52.932s
        Prediction time: 0.959s
        Accuracy: 0.547339030374820
```

```
RandomForestClassifier()
        Training time: 219.038s
        Prediction time: 0.577s
        Accuracy: 0.547339030374820
```

```
LogisticRegression(multi_class='multinomial')
        Training time: 6.370s
        Prediction time: 0.005s
        Accuracy: 0.692698197817074
```

Hypertuning parameter:

```
solvers = ['lbfgs','sag', 'saga','newton-cg']
penalty = ['none','l2']
c_values = [100, 10, 1.0, 0.1, 0.01]
```

```
Best: 0.684566 using {'C': 1.0, 'penalty': 'l2', 'solver': 'lbfgs'}
0.644553 (0.010609) with: {'C': 100, 'penalty': 'none', 'solver': 'lbfgs'}
0.647328 (0.010329) with: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
0.644553 (0.010609) with: {'C': 10, 'penalty': 'none', 'solver': 'lbfgs'}
0.662983 (0.009766) with: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
0.644553 (0.010609) with: {'C': 1.0, 'penalty': 'none', 'solver': 'lbfgs'}
0.684566 (0.010189) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'lbfgs'}
0.644553 (0.010609) with: {'C': 0.1, 'penalty': 'none', 'solver': 'lbfgs'}
0.675515 (0.009708) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
0.644553 (0.010609) with: {'C': 0.01, 'penalty': 'none', 'solver': 'lbfgs'}
0.642955 (0.010577) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
```

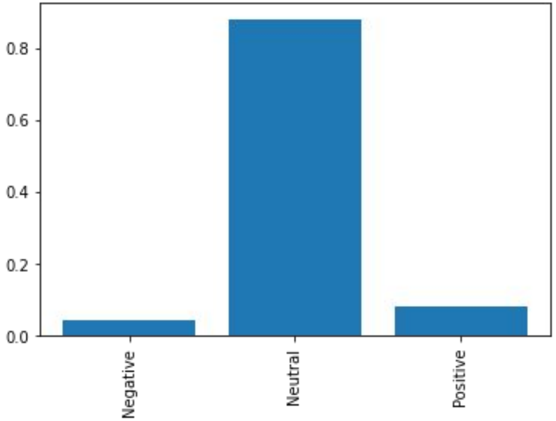
Train all dataset:

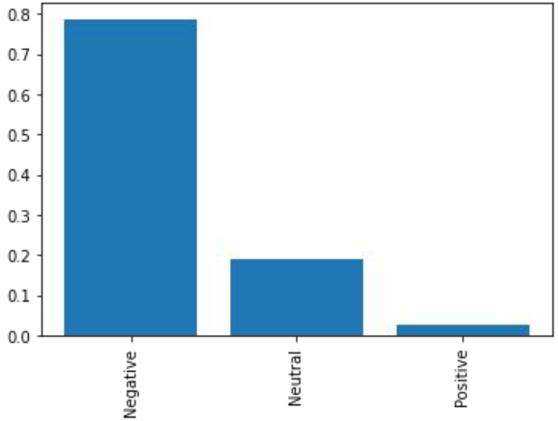
```
model_final =
LogisticRegression(multi_class='multinomial',
solver='lbfgs', penalty='l2', C=1)
model_final.fit(tfid_train, y_train)
y_pred_final = model_final.predict(tfid_test)
```

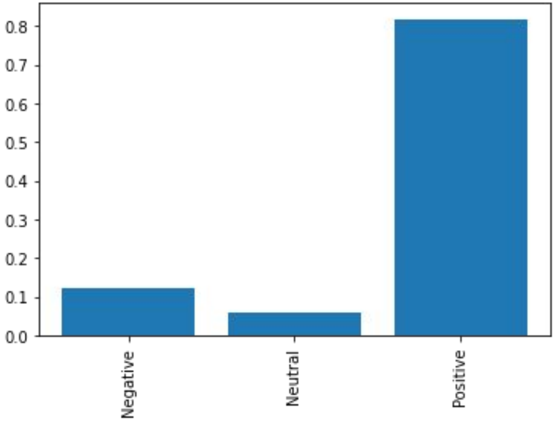
```
LogisticRegression(C=1, multi_class='multinomial')
Accuracy: 0.6968312391589457
```

Testing Model

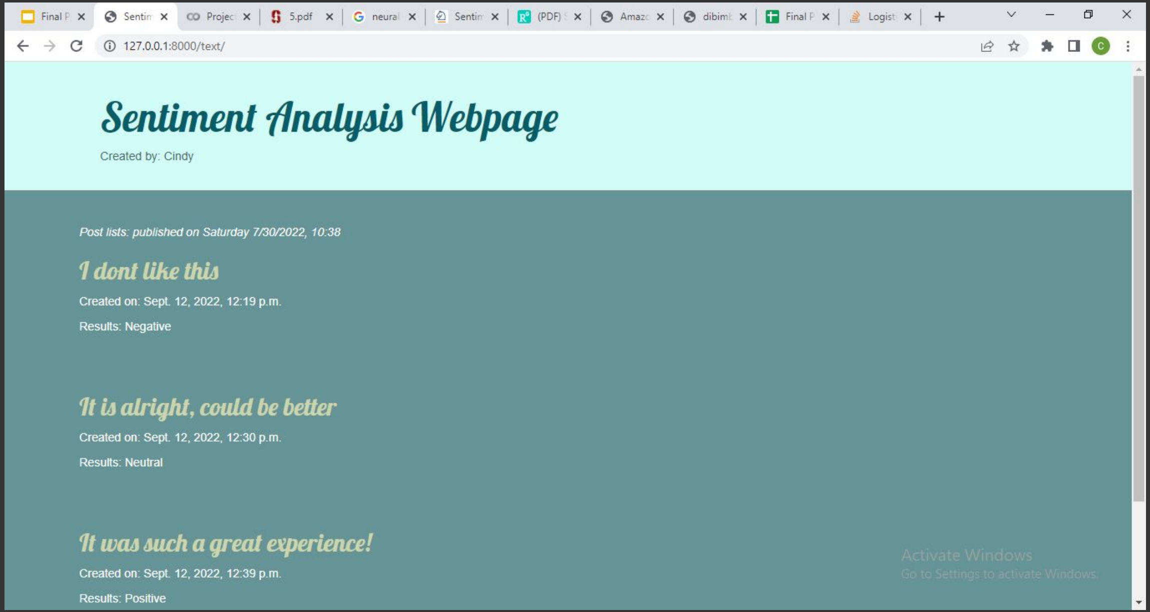
```
test_text = ['It is alright','I hate it','Oh it was a wonderful experience!']
```







5. Deployment



Link Here (only local computer)

5. Conclusion and Suggestion

The best sentiment analysis model for this database is Logistic Regression

It is better to combine the database with other company's customer reviews, so that we can get more general train data.

If there is more time and RAM, other models can also be explored (like NaiveBayes, neural networks, etc).

Thank you!

<u>Google colab link</u>