YASH RANDIVE

Restaurant Rating Prediction Using Sentiment Analysis and Machine Learning

Part II

QUICK RECAP

YASH
RANDIVE

Loading the Data

The files loaded are:

- #### review_meta_train.csv (rating label in this file)
- #### review_text_train.csv: original review text.
- #### review_meta_test.csv
- #### review_text_test.csv: original review text.

In [2]:

```
review_text_train = pd.read_csv('../input/review_text_train.csv', index_col = False, delimiter
= ',', header=0)
review_text_test = pd.read_csv('../input/review_text_test.csv', index_col = False, delimiter =
',', header=0)
review_meta_train = pd.read_csv('../input/review_meta_train.csv', index_col = False, delimiter
= ',', header=0)
review_meta_test = pd.read_csv('../input/review_meta_test.csv', index_col = False, delimiter =
',', header=0)
```

QUICK RECAP

YASH
RANDIVE

Examining Shapes of all Datasets

Here we examine the number of rows and columns that we'll be dealing with

In [3]:

```
print('review_text_train :\t', str(review_text_train.shape))
print('review_text_test :\t', str(review_text_test.shape))
print('review_meta_train :\t', str(review_meta_train.shape))
print('review_meta_test :\t', str(review_meta_test.shape))
```

```
review_text_train :     (28068, 1)
review_text_test :      (7018, 1)
review_meta_train :     (28068, 8)
review_meta_test :      (7018, 7)
```

Check for Null Values

In [6]:

```
print('review_text_train :\n', str(review_text_train.isna().sum()), "\n**********\n")
print('review_text_test :\n', str(review_text_test.isna().sum()), "\n**********\n")
print('review_meta_train :\n', str(review_meta_train.isna().sum()), "\n**********\n")
print('review_meta_test :\n', str(review_meta_test.isna().sum()), "\n**********\n")
```

QUICK RECAP

Creating Train and Test DataFrames

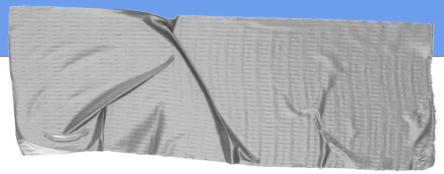
```
df_train['vote_funny'] = review_meta_train.vote_funny
df_train['vote_cool'] = review_meta_train.vote_cool
df_train['vote_useful'] = review_meta_train.vote_useful
df_train['rating'] = review_meta_train.rating
df_train
```

Test DF

```
df_test = review_text_test.copy()
df_test['vote_funny'] = review_meta_test.vote_funny
df_test['vote_cool'] = review_meta_test.vote_cool
df_test['vote_useful'] = review_meta_test.vote_useful
df_test
```

YASH
RANDIVE

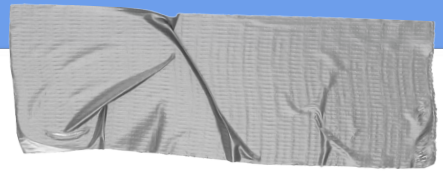
YASH
RANDIVE



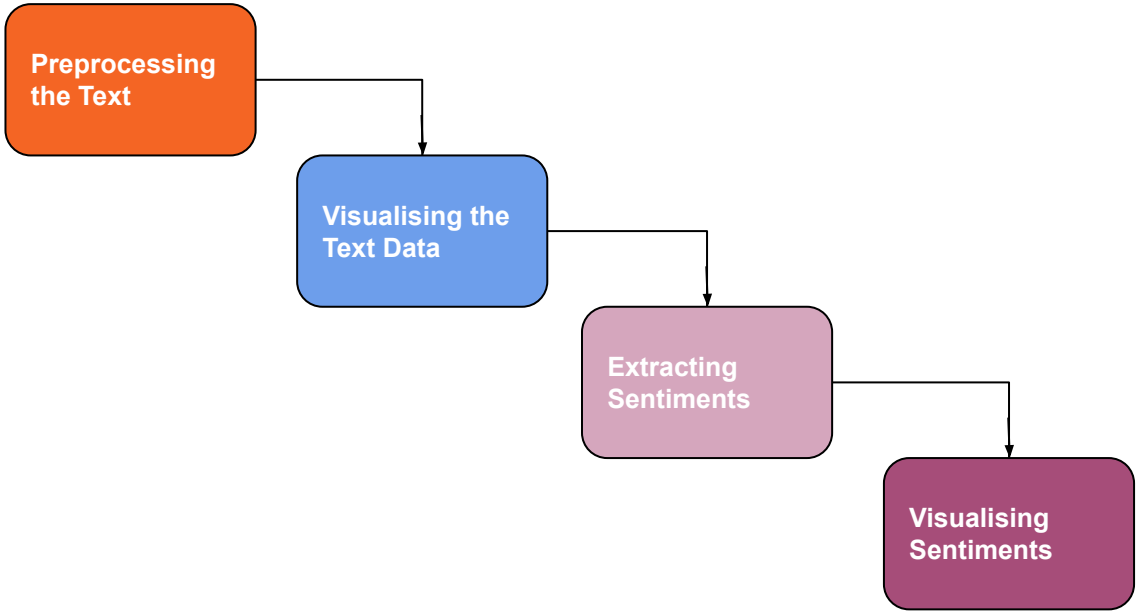
Cleaning the Text Data

- ➔ Remove Punctuations, Alphanumeric Characters, Numbers, symbols from the Review Text (‘,”._!@#$%^&*()_+=, etc.)
- ➔ Creating a Document Term Matrix(DTM)
- ➔ Extracting Sentiments from the Cleaned Reviews Text (Polarity and Subjectivity)

YASH
RANDIVE



Cleaning the Text Data



YASH
RANDIVE

Cleaning the Reviews Text is important as it helps python effectively Sentiments like Polarity and Subjectivity from the Text

YASH
RANDIVE

Preprocessing the Data

1. Removing Punctuation, Numbers and Alphanumeric Characters
2. Converting all characters to lowercase

In [12]:
```
# The re module raises the exception re. error if an error occurs while compiling or using a regu
lar expression.
import re
# This module contains a number of functions to process standard Python strings.
import string
```

In [13]:
```
def clean_text_round_1(text):
    text = text.lower()
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text) # Removes Punctuation
    text = re.sub('\w*\d\w*','',text) # Removes Alphanumeric Characters
    return text
round_1 = lambda x: clean_text_round_1(x)
```

In [14]:
```
data_review_cleaned = pd.DataFrame(df_train.review.apply(round_1))
```

In [15]:
```
data_review_cleaned
```

YASH
RANDIVE

Preprocessing the Data

Creating a string of ALL the reviews combined to check whether there still exist any special characters/symbols in it, if yes, remove them with Round 2 of Cleaning

In [16]:
```
rev_string = " "
for i in range(len(data_review_cleaned)):
    rev_string +=data_review_cleaned['review'][i]+" "
#rev_string
```

- ### It is observed that the "Review Text" requires to be pre-processed further as the text still contains the '\n' characrter(s) that are undesirabe and do not add any significant value to our data
- ### Hence, we define another function for the 2nd round of cleaning data

In [17]:
```
def clean_text_round_2(text):
    text = re.sub('\n', ' ', text)
    text = re.sub('['""...]', '', text)
    return text
round2 = lambda x: clean_text_round_2(x)
```

In [18]:
```
data_review_cleaned = pd.DataFrame(data_review_cleaned.review.apply(round2))
```

Preprocessing the Data

1. **Creating a Document term Matrix(DTM)**
2. **Transposing the Document Term Matrix**

YASH
RANDIVE

Creating a Document Term Matrix (DTM)

- ### A Document-Term Matrix(DTM) or Term-Document Matrix(TDM) is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms.

In [19]:

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(stop_words = 'english')
data_cv = cv.fit_transform(data_review_cleaned.review)
data_dtm = pd.DataFrame(data_cv.toarray(), columns = cv.get_feature_names())
data_dtm.index = data_review_cleaned.index
data_dtm
```

Out[19]:

	aa	aaa	aaaaaaaaaaaaa	aaaaaaaaaaaaaand	aaaaaaaaaaah	aaaaaaaaaggggh	aaaaaaahh	aaaaaawesome	aaaa
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0

In [20]:

```
data_dtm_transposed = data_dtm.T
```

In [21]:

```
data_dtm_transposed
```

Similarly,
Preprocessing the Testing Data

YASH
RANDIVE

Preprocessing the Testing Data

Cleaning the Testing Data

In [22]:
```
data_review_cleaned_test = pd.DataFrame(df_test.review.apply(round_1))
data_review_cleaned_test = pd.DataFrame(data_review_cleaned_test.review.apply(round2))
```

Extracting Polarity and Subjectivity from the Data

In [23]:
```
from textblob import TextBlob

# Polarity is how positive or Negative the expressioin is
# Subjectivity is how Factual or Opinionated the expression is
pol_test = lambda x: TextBlob(x).sentiment.polarity
sub_test = lambda x: TextBlob(x).sentiment.subjectivity

df_test['polarity'] = df_test['review'].apply(pol_test)
df_test['subjectivity'] = df_test['review'].apply(sub_test)
```

In [24]:
```
df_test
```

Exploratory Data Analysis(EDA)

YASH
RANDIVE

Extracting the "Top Words" used by reviewers in their reviews

In [25]:
```
top_words = {}
for o in data_dtm_transposed.columns:
    top = data_dtm_transposed[o].sort_values(ascending = False).head(50)
    top_words[o] = list(zip(top.index, top.values))
#top_words
```

In [26]:
```
data = data_dtm_transposed
```

Exploratory Data Analysis(EDA)

Calculating the “Most Common Words” in the Corpus and Adding them to a DataFrame

YASH
RANDIVE

In [27]:
```
from collections import Counter

words = []
for user_id in data.columns:
    top = [word for (word,count) in top_words[user_id]]
    for t in top:
        words.append(t)
```

In [28]:
```
common_words_count  = Counter(words).most_common()
```

Creating a dataframe of the most commonly used words in the reviews

In [29]:
```
df_common_words = pd.DataFrame(common_words_count, columns = ['word', 'count'])
df_common_words
```

Exploratory Data Analysis(EDA)

Visualising the “Top 30” most common words using a Seaborn Barplot

YASH
RANDIVE

Visualising the Top 30 Words in the Corpus

```
# plot
fig, ax = plt.subplots()
# the size of A4 paper
fig.set_size_inches(15, 12)
ax = sns.barplot(x='count', y='word', data=df_common_words[:30])
ax.set_title('Top 30 Words in the Corpus', size = 24)
ax.set_xlabel('Count', size = 20)
ax.set_ylabel("Words", size = 20)

fig.savefig('top_30_words.png')
```

YASH
RANDIVE

Creating A Word Cloud

A word cloud is a novelty visual representation of text data, typically used to depict keyword metadata on websites, or to visualize free form text. In Word Clouds the importance of each tag is shown with font size or color.

```
from wordcloud import WordCloud
```

```
from sklearn.feature_extraction import text
from sklearn.feature_extraction.text import CountVectorizer
```

```
#Adding the Stop Words
stop_words = text.ENGLISH_STOP_WORDS
```

```
wc = WordCloud(width = 800, height = 400, stopwords = stop_words, background_color = 'white', c
olormap = 'Dark2',
               max_font_size = 170, random_state = 45)
```

```
data_for_wc = pd.DataFrame()
data_for_wc['review'] = data_review_cleaned['review']
data_for_wc = data_for_wc.reset_index(drop = True)
```

```
#data_for_wc
```

YASH
RANDIVE

```
#Extracting text of all reviews into one string for WordCloud
text_wc = ' '
for i in range(len(data_for_wc)):
    text_wc += data_for_wc['review'][i]
```

```
cloud = wc.generate(text_wc)
plt.figure(figsize=(30,15))
plt.title('Reviews Text WordCloud', fontsize = 30)
plt.imshow(wc, interpolation = 'bilinear')
plt.axis("off")
```

```
(-0.5, 799.5, 399.5, -0.5)
```

```
cloud.to_file('word_cloud.png')
```

```
<wordcloud.wordcloud.WordCloud at 0x7f4b262b8710>
```

Sentiment Analysis

YASH
RANDIVE

Polarity is how Positive or Negative the expression is

- polarity >= 0.5 signifies that the statement is positive
- polarity < 0.5 signifies that the statement is negative

Subjectivity is how Factual or Opinionated the expression is

- subjectivity >=0.5 signifies that the statement is 'opinionated'
- subjectivity <0.5 signifies that the statement is 'factual'

Sentiment Analysis

YASH
RANDIVE

Updating Train DataFrame with cleaned "reviews" text

In [39]:
```
df_train['review'] = data_review_cleaned['review']
df_train
```

Extracting Sentiments

```
data = df_train
```

```
from textblob import TextBlob
```

Sentiment Analysis

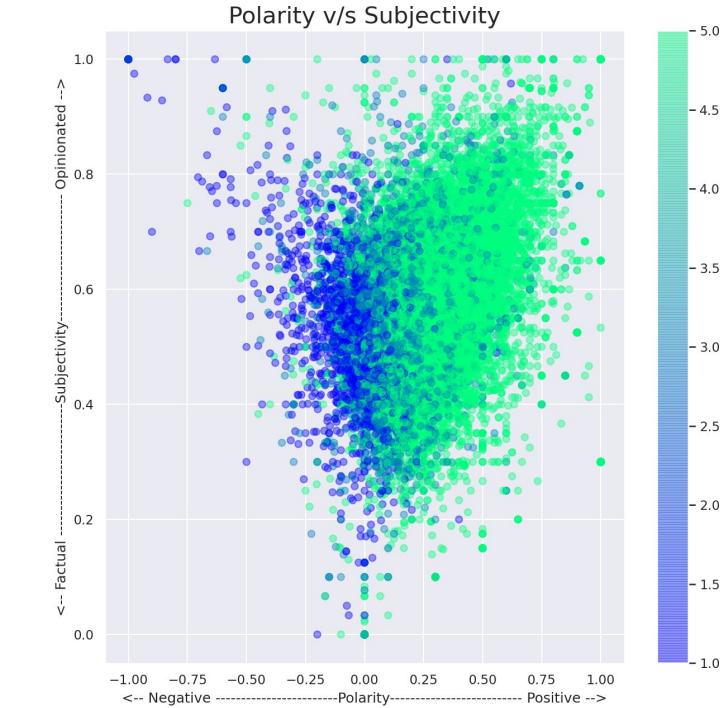
YASH
RANDIVE

```
pol = lambda x: TextBlob(x).sentiment.polarity
sub = lambda x: TextBlob(x).sentiment.subjectivity

data['polarity'] = data['review'].apply(pol)
data['subjectivity'] = data['review'].apply(sub)
```

```
plt.figure(figsize = (10,10))
scatter = plt.scatter(data['polarity'], data['subjectivity'], c= data['rating'], cmap = 'winte
r', alpha = 0.4)
plt.xlabel('<-- Negative ----------------------Polarity-------------------------- Positive --
>')
plt.ylabel('<-- Factual -------------------------Subjectivity----------------------- Opinion
ated -->')
plt.title('Polarity v/s Subjectivity', size  = 20)
plt.colorbar()
plt.show

plt.savefig('polarity_vs_subjectivity_sentiment', dpi = 150)
```

Sentiment Analysis

YASH
RANDIVE



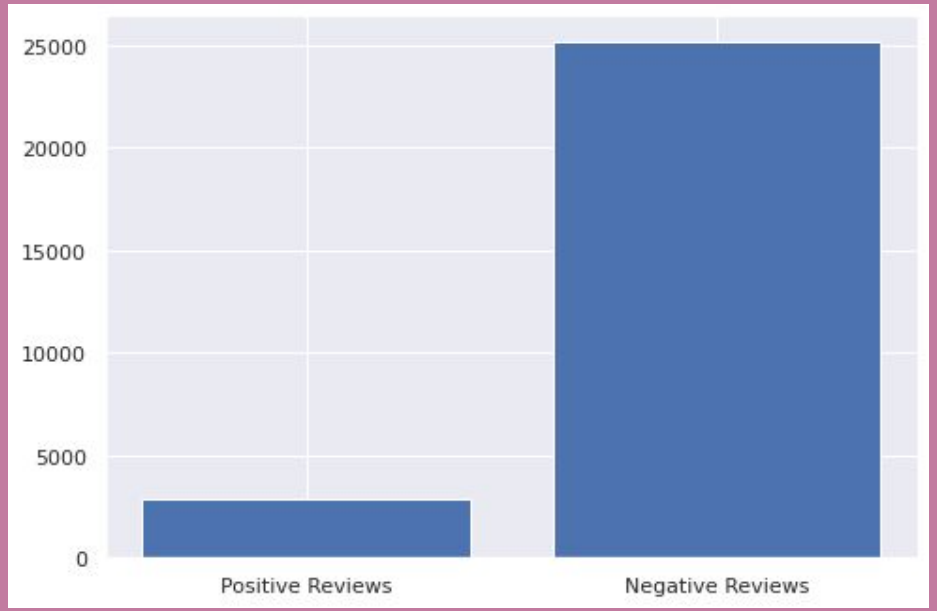
Sentiment Analysis

YASH
RANDIVE

Visualising Positive v/s Negative Reviews

```
positive_values = 0
negative_values = 0

for i in range(len(data['polarity'])):
    if data['polarity'][i]<0.5:
        negative_values +=1
    else:
        positive_values +=1
```

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
polarity_prop = ['Positive Reviews', 'Negative Reviews']
polarity_vals = [positive_values, negative_values]
ax.bar(polarity_prop, polarity_vals)
plt.show()

plt.savefig('pos_vs_neg_reviews.png', dpi = 150)
```



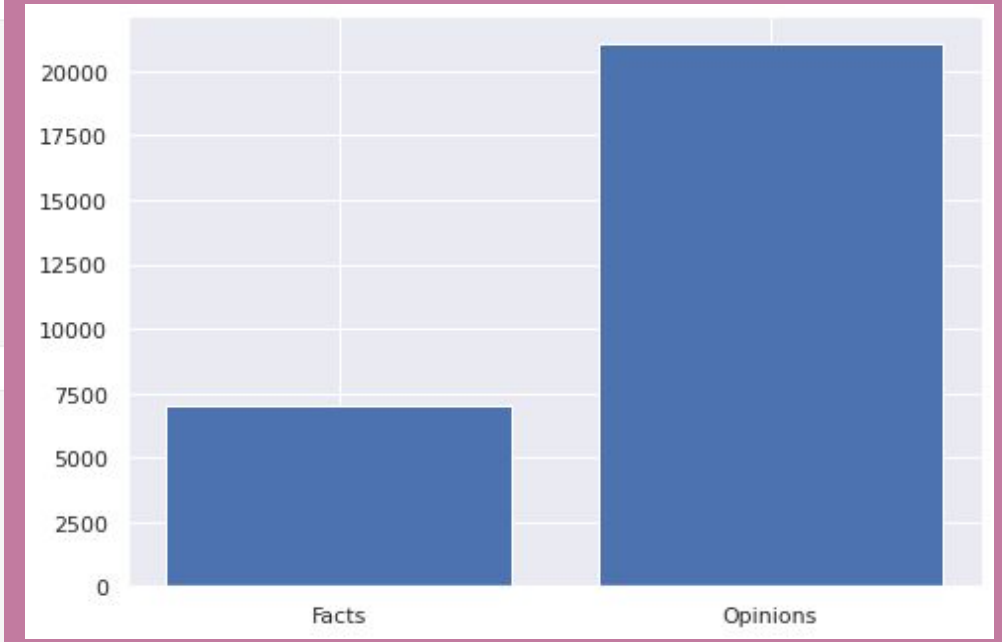
Sentiment Analysis

YASH
RANDIVE

Visualising Facts v/s Opinions in the Reviews

```
factual_values = 0
opinionated_values = 0

for i in range(len(data['subjectivity'])):
    if data['subjectivity'][i]<0.5:
        factual_values +=1
    else:
        opinionated_values +=1
```

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
subjectivity_prop = ['Facts', 'Opinions']
subjectivity_vals = [factual_values, opinionated_values]
ax.bar(subjectivity_prop, subjectivity_vals)
plt.show()

plt.savefig('facts_vs_opinions.png', dpi = 150)
```



Sentiment Analysis

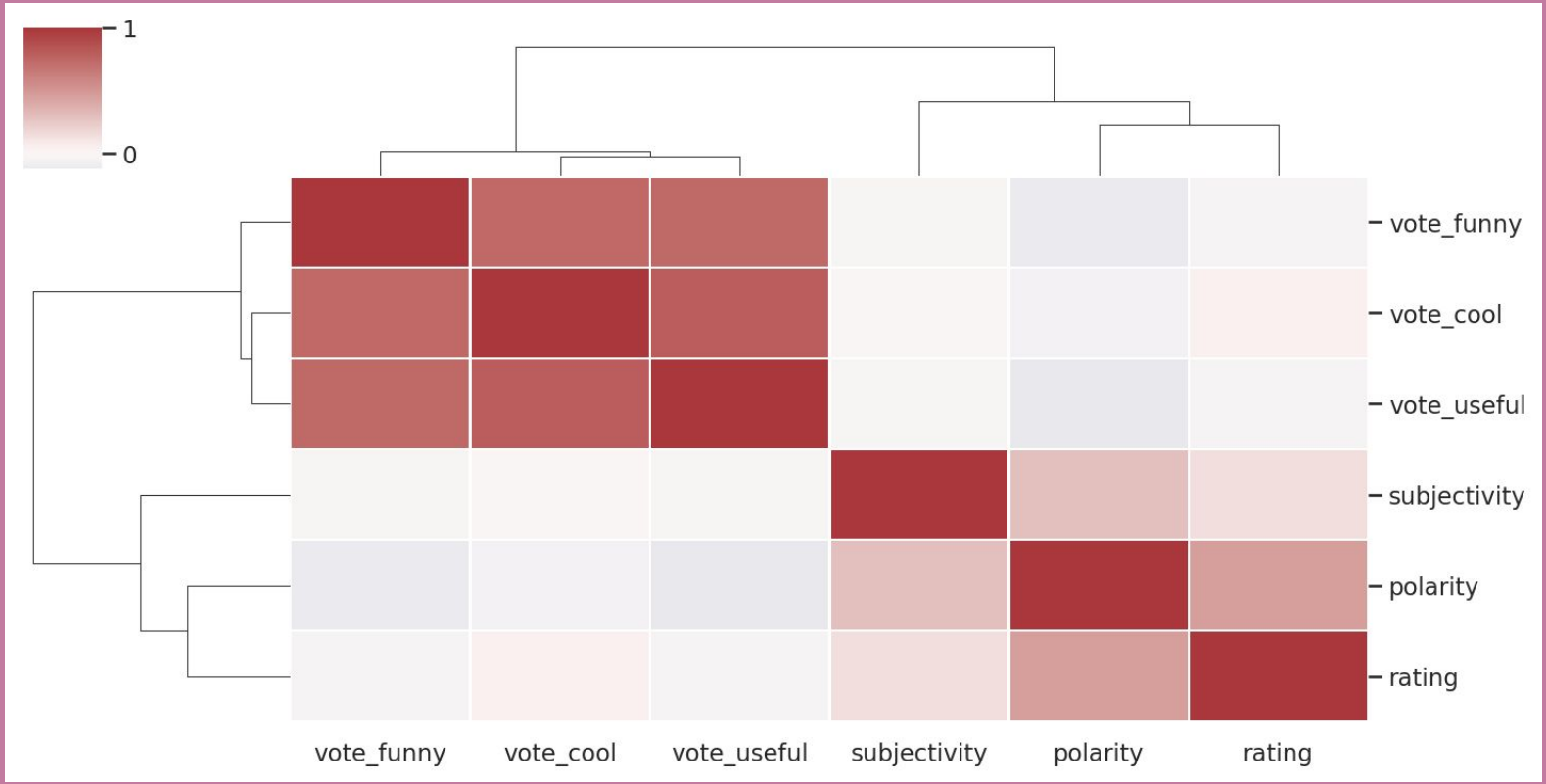
YASH
RANDIVE

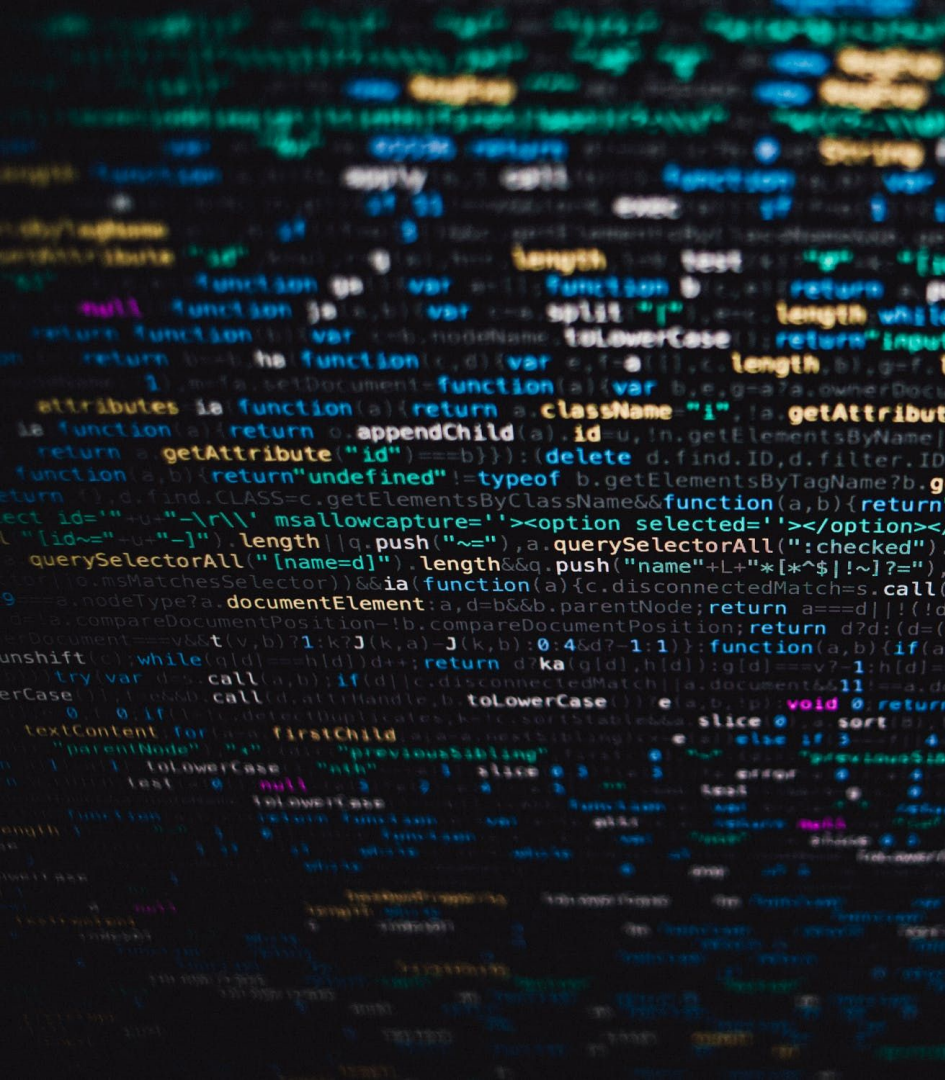
Plotting Correlation between Features

```
cor = df_train.loc[:,["vote_funny","vote_cool",'vote_useful',"polarity", "subjectivity","ratin
g"]]
correlation_map = sns.clustermap(cor.corr(), center=0, cmap="vlag",
               linewidths=.75, figsize=(10, 5))

correlation_map.savefig('feature_correlation.png', dpi = 150)
```

Sentiment Analysis

YASH
RANDIVE





YASH
RANDIVE

Next Up

Predictive Analysis

1. Logistic Regression
2. KNN Classifier
3. Naive Bayes Classification
    a. Gaussian NB
    b. Bernoulli NB
    c. Multidimensional NB
4. Random Forest Classifier
5. Support Vector Machine(SVM) Classifier
6. Deep Neural Network (Tensorflow 2.0)

YASH
RANDIVE

Congratulations!

You have reached the end of this part of the project and are all set to move ahead!

For queries or questions in the code reach me at :

yashrandive.datascience@gmail.com

All the Best!