Sentiment Analyzer

Luis A.
Hicksville, NY
0 friends
15 reviews
16 photos

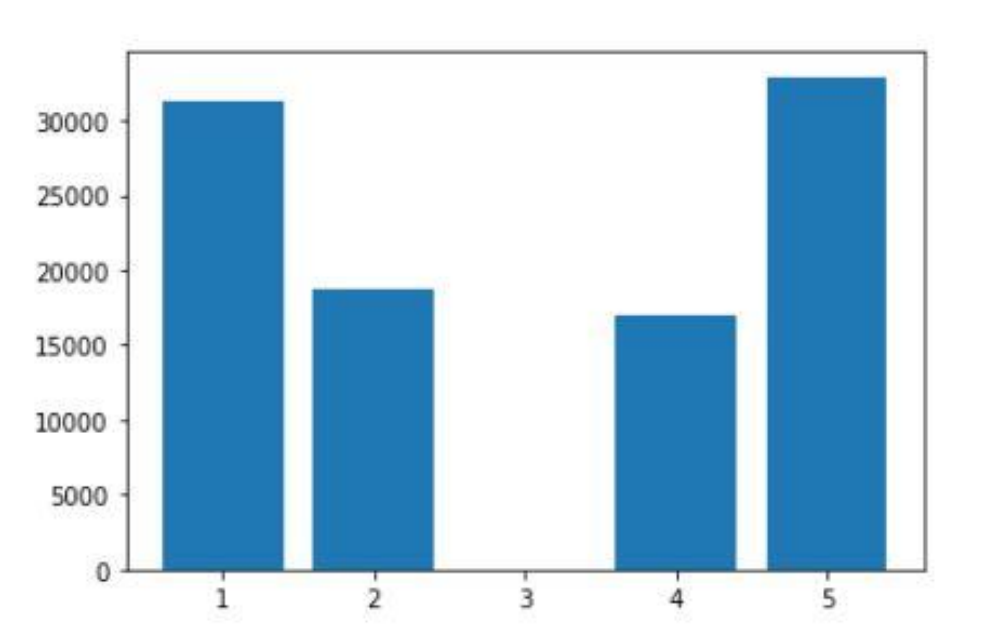
Share review
Embed review

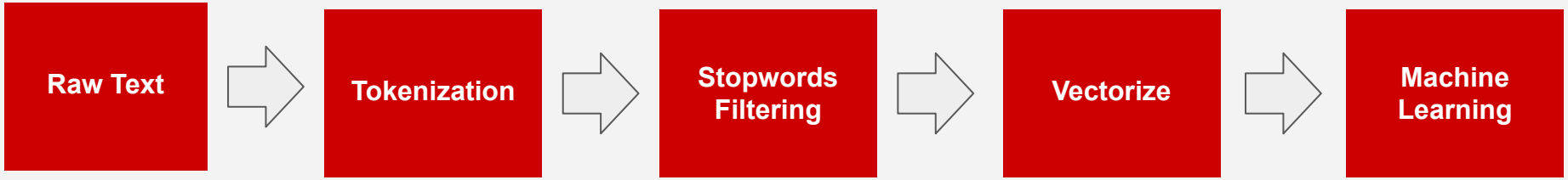
1 star rating 3/31/2019

I stopped by here to pick up coffee and a sandwich for my wife. I asked for a small coffee with three splendas and three creams. When I returned to Brooklyn college campus, because we were proctoring an exam their, and gave her the coffee she realized the coffee was black and had no Splenda.... unbelievable... Brooklyn, you're killing me here....

Big Data (Yelp original dataset over 5M restaurant reviews)

- Negative Review: Randomly selected 50K reviews of 1 & 2 Star restaurants
- Positive Review: Randomly selected 50K reviews of 4 & 5 Star restaurants




Raw Text
Tokenization
Stopwords Filtering
Vectorize
Machine Learning

Cleaning Raw Text

Removing mentions
Removing hashtags
Removing numbers

```
import re #regular expression
def function_clean(text):
    text = re.sub(r"http\S+", "", text) #removing the URL Http
    # Removal of mentions
    text = re.sub("@[^\s]*", "", text)
    # Removal of hashtags
    text = re.sub("#[^\s]*", "", text)
    # Removal of numbers
    text = re.sub('[0-9]*[+-:]*[0-9]+', '', text)
    text = re.sub("'s", "", text)
    return text
```

```
data_final['text'] = data_final['text'].apply(lambda text: function_clean(text))
```

Tokenizing & Stopwords

Lemmatization
Punctuation

```
import nltk

import string
#nltk.download('stopwords')

from nltk.corpus import stopwords
ENGLISH_STOP_WORDS = stopwords.words('english')

def my_tokenizer(sentence):

    listofwords = sentence.strip().split()
    listof_words = []
    for word in listofwords:
        if not word in ENGLISH_STOP_WORDS:
            lemm_word = WordNetLemmatizer().lemmatize(word)
            # remove the stop words
            for punctuation_mark in string.punctuation:
                word = word.replace(punctuation_mark, '').lower()
            if len(word)>0:
                listof_words.append(word)
    return(listof_words)
```

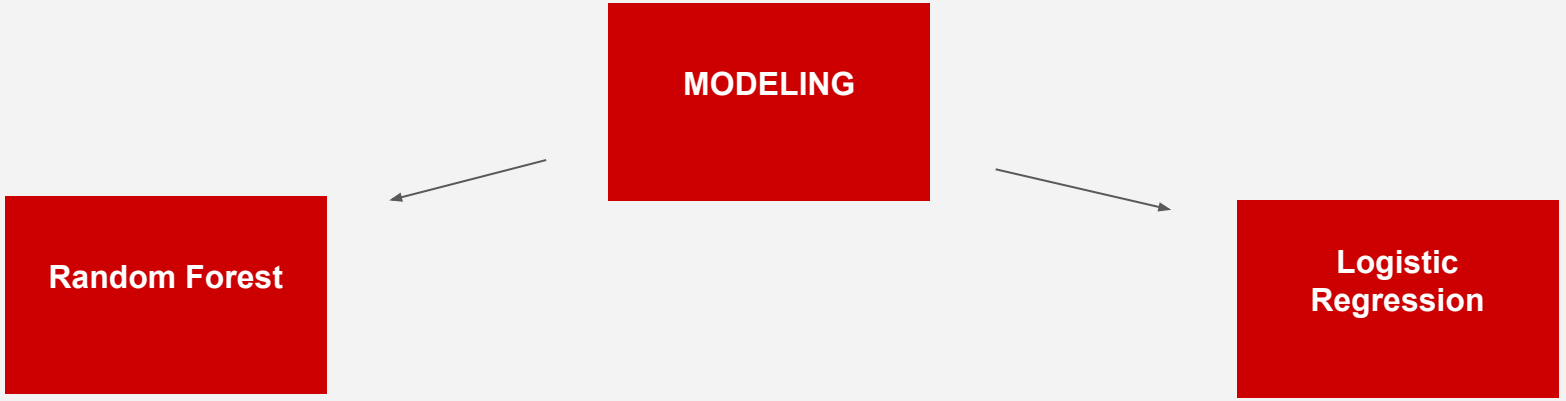
Vectorizer

Ngram = Range 1, 3,
Unigrams, Bigrams, Trigrams


```
#term frequency-inverse document frequency
from sklearn.feature_extraction.text import TfidfVectorizer
nltk.download('wordnet')
vect_1 = TfidfVectorizer(min_df=100,tokenizer=my_tokenizer, stop_words={'english'}, ngram_range=(1,3)).fit(X_train)
X_train1 = vect_1.transform(X_train)
X_test1 = vect_1.transform(X_test)
```

	word	count
1966	food	2310.008738
4054	place	2058.430759
2412	great	1929.061689
2318	good	1910.801663
4809	service	1819.912651
5542	time	1509.335826
3113	like	1472.305078
2181	get	1456.556882
355	back	1431.844515
3760	one	1420.229538
6213	would	1331.273918
2272	go	1311.082828
4395	really	1158.696885
5840	us	1140.180184
2379	got	1059.759123
1648	even	1043.986561
2860	it	997.693854

Counting The Most Repetitive Words




```
GridSearchCV(cv=5,
estimator=RandomForestClassifier(random_state=42),
             param_grid={'criterion': ['gini', 'entropy'],
                         'n_estimators': [5, 50]})
```

Best Parm = 'criterion': 'gini', 'n_estimators': 50

```
params = {'classifier__C' : [10**j for j in
range(-4,4)]}
grid_search = GridSearchCV(pipe,
param_grid=params,cv=5)
```

Pipeline(steps=[('classifier', LogisticRegression(C=10, random_state=1))])

Random Forest scores:

Training: 99%
Test: 91%

The Confusion Matrix for Random Forest model:

	Predicted 0	Predicted 1
True 0	9062	872
True 1	1001	9065

Logistic Regression scores:

Training: 95%
Test: 94%

The Confusion Matrix for Logistic regression model:

The Confusion Matrix

	Predicted 0	Predicted 1
True 0	9373	561
True 1	582	9484

The Classification report

	precision	recall	f1-score	support
0	0.941537	0.943527	0.942531	9934.00000
1	0.944151	0.942182	0.943165	10066.00000
accuracy	0.942850	0.942850	0.942850	0.94285
macro avg	0.942844	0.942854	0.942848	20000.00000
weighted avg	0.942853	0.942850	0.942850	20000.00000

Tools used:

Python libraries/modules:

Numpy
Pandas
Natural Language Toolkit (stopwords)
Matplotlib
Seaborn
Regular expression operations
Sklearn (GridSearch, train_test_split, StandardScaler, TfidfVectorizer, LogisticRegression, confusion_matrix, classification_report, RandomForestClassifier)

Pickle

For the application:

Flask
HTML
CSS


Exported models
and
TfidfVectorizer
Flask
HTML
CSS
APP