Title: Sentiment analysis of twitter data

Name: Neha Damele

Contents

Abstract 1
Introduction 1
Purpose 1
Task 2
Training data 2
Test data 2
Techniques 2
Programming tools used 2
Libraries 2
Steps involved in Sentiment Analysis 2
Cleaning the given data set 2
Data Pre-processing steps 4
Cleaning the text 4
Stemmization 5
Vectorization 5
Experiment results 5
Conclusion 7

Abstract

Sentiment analysis is contextual mining of text that analyses texts for polarity, from positive to negative. In this project the tweets from Obama and Romney ware analysed to predict the sentiment of the tweets. In this project I am using multiple machine learning models to predict the polarity of the tweet. Different measures were considered to define which model performed the best. Final model was selected based on the model accuracy and used on the test data to perform the prediction.

Introduction

Purpose

The purpose of the Data mining research project is to perform Sentiment analysis on the tweets downloaded from Twitter during a presidential debate between Obama and Romney. In order to classify the sentiments, build the best classification model which gives the highest accuracy and F1 score.

Task

To classify the sentiment or opinion expressed in tweets into one of the three classes: positive (1), negative (-1), or neutral (0) (which means no opinion).

Training data

- Tweets about Obama
  This dataset consists of 7198 records, 6 columns
- Tweets about Romney
  This dataset consists of 7200 records, 6 columns

The training data has 4 classes: positive (1), negative (-1), neutral (0), and mixed (2). The mixed class (2) means that a tweet expresses both positive and negative opinions (or sentiments). We are not using class label 2 for the classification purpose.

Test data

The test data contains only positive, negative and neutral tweets about the two presidential candidates in two separate datasets without class labels.

Techniques

Programming tools used

Language – Python

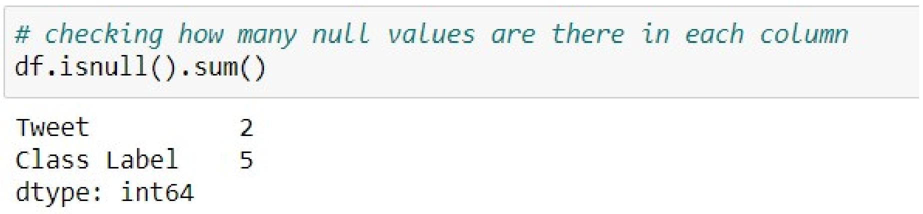
Libraries

- pandas
- nltk
- re
- Model
- String
- Groupby
- PreProcessing
- SnowballStemmer
- linear_model
- svm, neighbors, tree, neural_network, feature_extraction, pipeline, metrics, model_selection

Steps involved in Sentiment Analysis

Cleaning the given data set

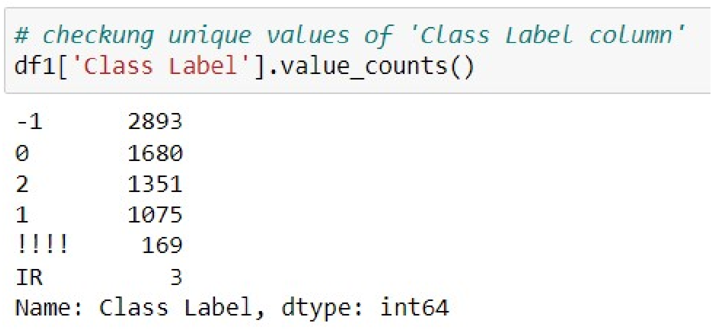
- The training data consisted of few null values. Dropped the records with the null values



```
# checking how many null values are there in each column
df.isnull().sum()
```

```
Tweet          2
Class Label    5
dtype: int64
```

- Since we are only interested in 0, -1, 1 classes, therefore it was needed to drop those rows which have values other values.

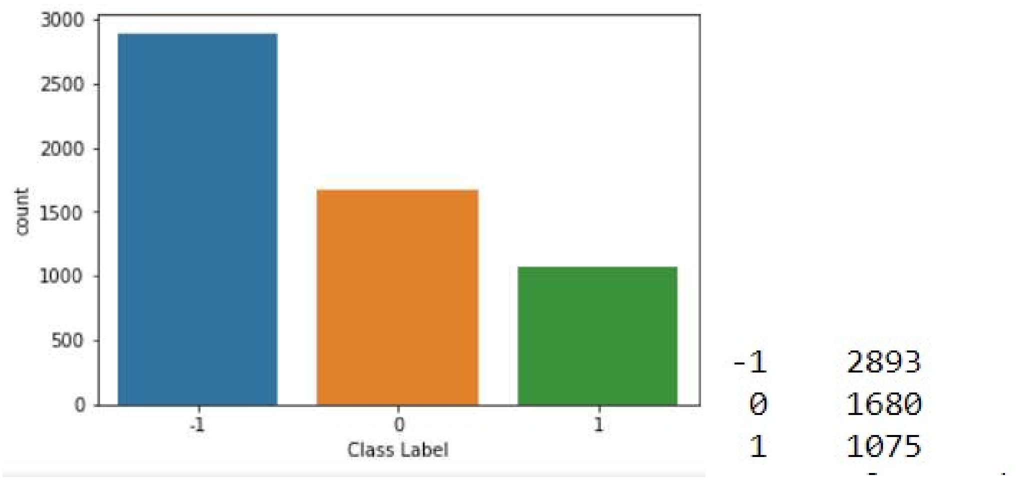


```
# checkung unique values of 'Class Label column'
df1['Class Label'].value_counts()
```

```
-1      2893
0       1680
2       1351
1       1075
!!!!     169
IR         3
Name: Class Label, dtype: int64
```

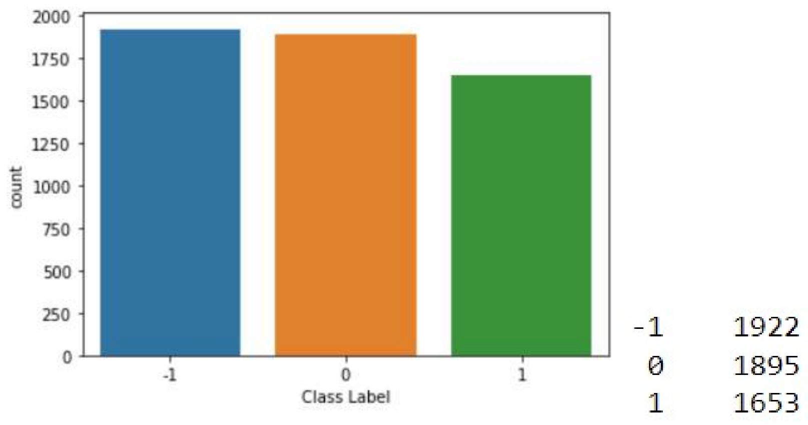
Above image shows unique class label values like -1, 0, 1, 2, !!!!, IR in Romney dataset

- Dealing with class imbalance problem:



```
-1     2893
 0     1680
 1     1075
```

Figure 1: Class distribution in Romney data



```
-1     1922
 0     1895
 1     1653
```

Figure 2: Class distribution in Obama data

From the above figures, we can observe that there is class imbalance in the dataset, especially Romney dataset.

Class weight attribute of model was used in every model to deal with this class imbalance problem.

Data Pre-processing steps

1. Cleaning the text
2. Stemmization
3. Vectorization
4. Classification

Cleaning the text

Cleaning the text with unwanted words, numbers, punctuations, special characters, emojis, hashtags, URLS, username mentions which do not contribute to the sentiment of the tweet. For this I am using the regular expression operations library of Python - 'Regex'. Similarly, removal of stop words from text. Stop words in English are words that carry very little useful information. Using NKTL library which has a list of English language stop words.

Removal of comma:

`Replacing the ','` → `' '`

Removing hashtags:

e.g. "#debate"→ " "

Expanding acronyms:

```
e.g.    "w/o" → "without "
        "w/"→ "with "
```

Removing html tags:

```
e.g.    "<e>" → " "
        "<a>"→ " "
```

Removing links:

```
e.g.    "http://t.co/oid9jiTK" → " "
```

Removing username mentions:

```
e.g.    "@WardBrenda" → " "
```

Removing numbers in the text:

```
e.g.    "Romneys 12 million jobs scam" → "Romneys million jobs scam"
```

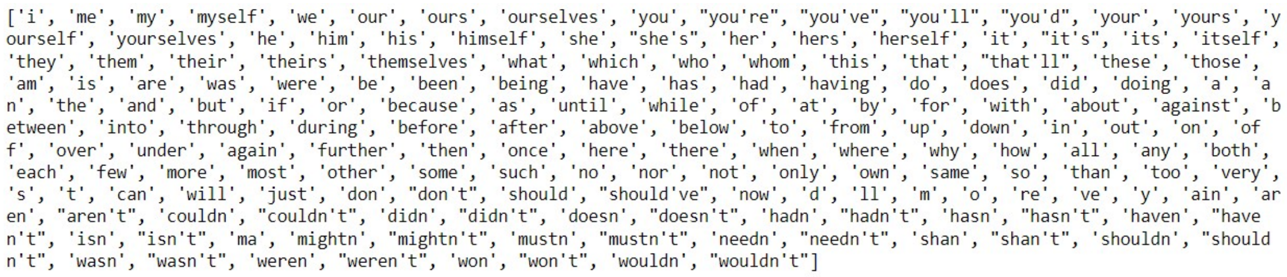
Removing punctuations:

e.g. "@WardBrenda" → " "

Converted all characters to lowercase

e.g. "Obama's Business Experience" → " obamas business experience "

Removable of stop words:



```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Figure 3: Common English stop words in NKTL Library

Stemmization

Stemming is a process that reduces the words to their root forms. The input to the stemmer is the tokenized words.

Used two types of Stemmerization process:

1. Porter Stemmer
2. Snowball Stemmer

Stemmerization gives some meaningless root words as it simply chops off some characters in the end.

e.g.

"people " → "peopl"

" anything " → " anyth"

Vectorization

The next step in our workflow was Feature Extraction. Used Count Vectorization to transform the text into a numerical feature vector on the basis of count of each word that appear in the entire document. Considering ngram-range=(1, 2): both unigrams and bigrams.

Further transformed the transformed vector into normalized term-frequency times inverse document-frequency representation using Tfidfvectorizer.

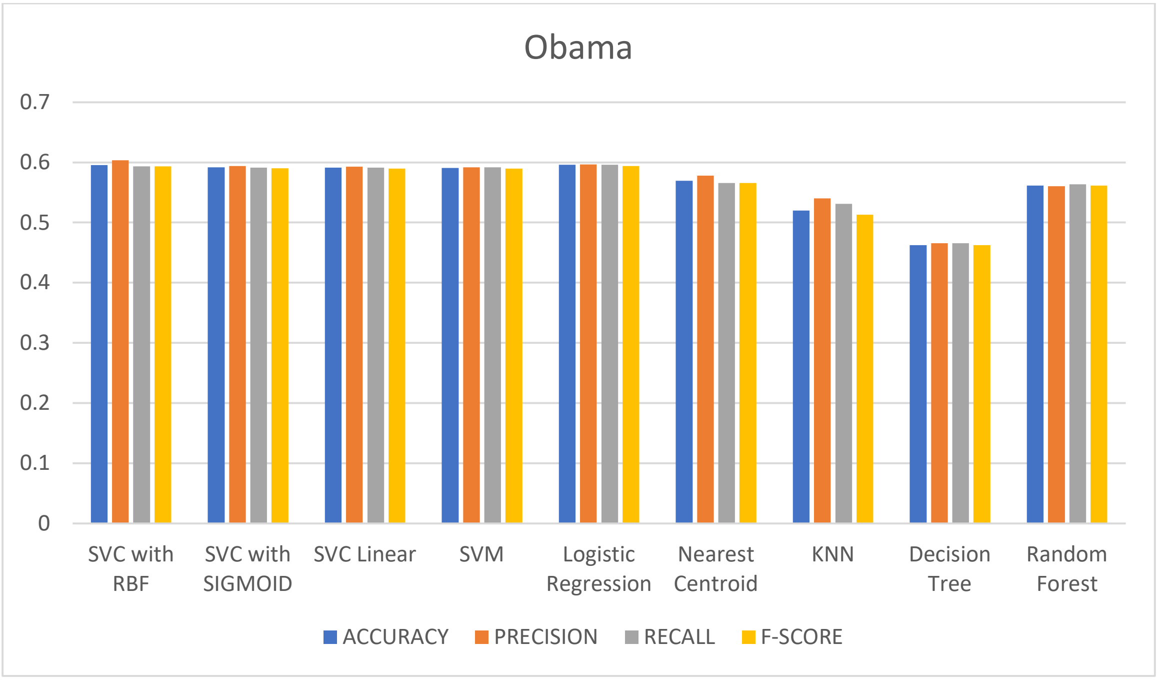
Experiment results

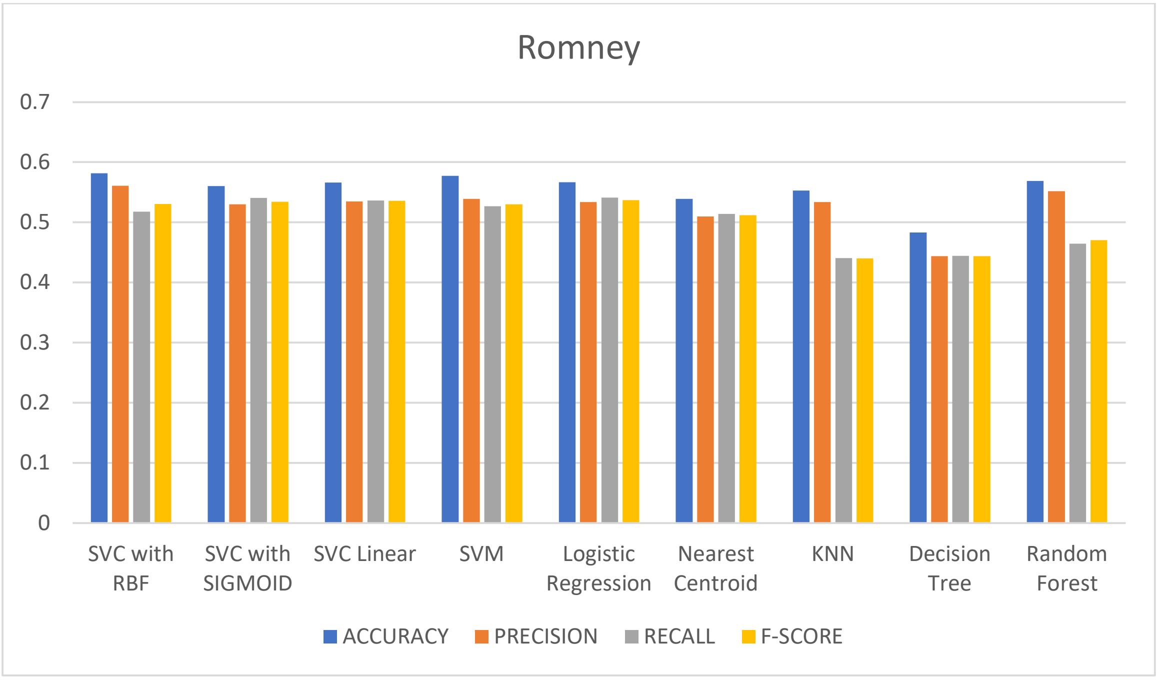
I experimented with a number of Machine learning models (listed in the below table) using the numerical representation of text data to analyse which ML Model will give the highest accuracy. I have used 10 K-Fold Cross-Validation to train and test on different folds of data.

SVC with RBC gave the highest accuracy of 59.5% on Obama data and 58.1 % on Romney data.

ALGORITHM		ACCURACY	PRECISION	RECALL	F-SCORE
SVC with RBF	Obama	0.5954	0.6033	0.5932	0.5932
	Romney	0.5816	0.5607	0.5178	0.5304
SVC with SIGMOID	Obama	0.5915	0.5938	0.5914	0.5901
	Romney	0.5601	0.5300	0.5406	0.5344
SVC Linear	Obama	0.5912	0.5929	0.5910	0.5896
	Romney	0.5660	0.5348	0.5365	0.5356
SVM	Obama	0.5908	0.5915	0.5915	0.5897

	Romney	0.5773	0.5392	0.5267	0.5300
Logistic Regression	Obama	0.5959	0.5968	0.5962	0.5941
	Romney	0.5665	0.5339	0.5410	0.5370
Nearest Centroid	Obama	0.5692	0.5778	0.5657	0.5655
	Romney	0.5389	0.5095	0.5142	0.5116
KNN	Obama	0.5201	0.5399	0.5312	0.5131
	Romney	0.5529	0.5339	0.4407	0.4398
Decision Tree	Obama	0.4627	0.4656	0.4657	0.4627
	Romney	0.4831	0.4437	0.4441	0.4437
Random Forest	Obama	0.5612	0.5603	0.5635	0.5612
	Romney	0.5688	0.5519	0.4645	0.4704





Conclusion

I have compared multiple supervised learning algorithms, of all the classifiers, Support Vector Machine produced the best results. The pipeline used first cleaned the text followed by running the Stemmization, Vectorization and model building. The lemmatization was also considered instead of Stemmization, but Stemmization gave the best results.