CS5542 Big Data Apps and Analytics

In Class Programming –3 Report
(Jongkook Son)

Project Overview:

Use the same data (that we obtained by in source code `Data = pd.read_csv('https://raw.githubusercontent.com/dD2405/Twitter_Sentiment_Analysis/master/train.csv')` **) and perform the sentiment analysis task on this data using one of the scikit learn classifier for text.**

Requirements/Task(s):

1) Data cleaning and preprocessing (at minimum have the following: Removing unnecessary columns or data, Removing Twitter Handles(@user), Removing punctuation, numbers, special characters, Removing stop words, Tokenization, and Stemming, TFIDF vectors, POS tagging, checking for missing values , train/test split of data). (70 points)
2) Data Visualization and analysis for critical steps (WordCloud, Bar plots, etc) (10 points)
3) Model building and successfully executing the model to make prediction. (10 points)
4) Code quality, Pdf Report quality, video explanation (10 points)

What I learned in ICP:

I could have learned the basics of nltk. First of all, before doing a data analysis job, Data cleaning and preprocessing are very, very important. If this is not done enough, then garbage in garbage out of the situation can happen. So If You want to get a good insight from data analysis, clean the data first. Also, I could have learned about Regex in python. Whenever I try to use it, it is always confusing, but by doing this ICP, it becomes easier for me. Finally, using the module in nltk I learned how to tokenize and stem for each word and pos tagging job, which was very meaningful to me.

ICP description what was the task you were performing and Screen shots that shows the successful execution of each required step of your code

1. Data cleaning and preprocessing

Remove unnecessary column in this case id column is not necessary

Removing unnecessary columns or data

```
[4] #Remove id column
    Data = Data.drop("id", axis=1)
```

```
[5] Data
```

	label	tweet
0	0	@user when a father is dysfunctional and is s...
1	0	@user @user thanks for #lyft credit i can't us...
2	0	bihday your majesty
3	0	#model i love u take with u all the time in ...
4	0	factsguide: society now #motivation
...	...	...
31957	0	ate @user isz that youuu?ð□□□□ð□□□□ð□□□□ð□□□□ð□□□□ð...
31958	0	to see nina turner on the airwaves trying to...
31959	0	listening to sad songs on a monday morning otw...
31960	1	@user #sikh #temple vandalised in in #calgary,...
31961	0	thank you @user for you follow

31962 rows × 2 columns

Remove Twitter handler by using re module

```
#library for data cleaning
import re
import numpy as np


def remove_handle(text, pattern):
  # finds the pattern @ and put it in a list
  words = re.findall(pattern, text)

  for word in words:
    #remove @ and replace it with blank
    text = re.sub(word,"",text)

  return text


#make a new column named cleaned tweet
Data["Cleaned_Tweets"] = np.vectorize(remove_handle)(Data['tweet'], "@[\w]*")

#Remove tweet column
Data = Data.drop("tweet", axis=1)
```

	label	Cleaned_Tweets
0	0	when a father is dysfunctional and is so sel...
1	0	thanks for #lyft credit i can't use cause th...
2	0	bihday your majesty
3	0	#model i love u take with u all the time in ...
4	0	factsguide: society now #motivation
...	...	...
31957	0	ate isz that youuu?ð□□□□ð□□□□ð□□□□ð□□□□ð□□□□ð□□□□ð□...
31958	0	to see nina turner on the airwaves trying to...
31959	0	listening to sad songs on a monday morning otw...
31960	1	#sikh #temple vandalised in in #calgary, #wso...
31961	0	thank you for you follow

31962 rows × 2 columns

Remove punctuation, numbers, special characters simply using Series.str

Removing punctuation, numbers, special characters

```
[8] #replace everything except characters. Series.str can be used to access the values of the series
    Data["Cleaned_Tweets"] = Data["Cleaned_Tweets"].str.replace("[^a-zA-Z]", " ")
```

```
Data
```

	label	Cleaned_Tweets
0	0	when a father is dysfunctional and is so sel...
1	0	thanks for lyft credit i can t use cause th...
2	0	bihday your majesty
3	0	model i love u take with u all the time in ...
4	0	factsguide society now motivation
...	...	...
31957	0	ate isz that youuu ...
31958	0	to see nina turner on the airwaves trying to...
31959	0	listening to sad songs on a monday morning otw...
31960	1	sikh temple vandalised in in calgary wso...
31961	0	thank you for you follow

31962 rows × 2 columns

Remove stopwords which are loaded from nltk

```
[11] #import stopword from nltk
     from nltk.corpus import stopwords
```

```
stopwords = stopwords.words("english")
print(stopwords)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'your
```

```
[13] #Series.apply Invoke function on values of Series and remove words in stopword
     Data["Cleaned_Tweets"] = Data["Cleaned_Tweets"].apply(lambda x: " ".join([word for word in x.split() if word not in stopwords]))
```

```
[14] Data
```

	label	Cleaned_Tweets
0	0	father dysfunctional selfish drags kids dysfun...
1	0	thanks lyft credit use cause offer wheelchair ...
2	0	bihday majesty
3	0	model love u take u time ur
4	0	factsguide society motivation
...	...	...
31957	0	ate isz youuu
31958	0	see nina turner airwaves trying wrap mantle ge...
31959	0	listening sad songs monday morning otw work sad
31960	1	sikh temple vandalised calgary wso condemns act
31961	0	thank follow

31962 rows × 2 columns

Tokenization, and Stemming

Tokenization, and Stemming

[15]
```
#Tokenize the text by word
Data["Cleaned_Tweets"] = Data["Cleaned_Tweets"].apply(lambda x: word_tokenize(x))
```

[16]
```
Data
```

	label	Cleaned_Tweets
0	0	[father, dysfunctional, selfish, drags, kids, ...
1	0	[thanks, lyft, credit, use, cause, offer, whee...
2	0	[bihday, majesty]
3	0	[model, love, u, take, u, time, ur]
4	0	[factsguide, society, motivation]
...	...	...
31957	0	[ate, isz, youuu]
31958	0	[see, nina, turner, airwaves, trying, wrap, ma...
31959	0	[listening, sad, songs, monday, morning, otw, ...
31960	1	[sikh, temple, vandalised, calgary, wso, conde...
31961	0	[thank, follow]

31962 rows × 2 columns

```
#Import stemming library
from nltk.stem import PorterStemmer
porter = PorterStemmer()
#Stemming for each Series values
Data["Cleaned_Tweets"] = Data["Cleaned_Tweets"].apply(lambda x: [porter.stem(word) for word in x])
```

```
Data
```

	label	Cleaned_Tweets
0	0	[father, dysfunct, selfish, drag, kid, dysfunc...
1	0	[thank, lyft, credit, use, caus, offer, wheelc...
2	0	[bihday, majesti]
3	0	[model, love, u, take, u, time, ur]
4	0	[factsguid, societi, motiv]
...	...	...
31957	0	[ate, isz, youuu]
31958	0	[see, nina, turner, airwav, tri, wrap, mantl, ...
31959	0	[listen, sad, song, monday, morn, otw, work, sad]
31960	1	[sikh, templ, vandalis, calgari, wso, condemn,...
31961	0	[thank, follow]

31962 rows × 2 columns

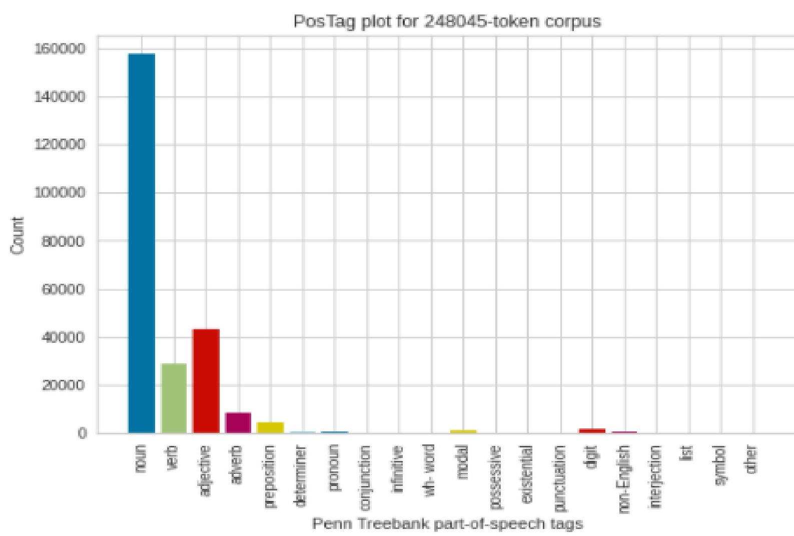
POS tagging

Visualization of postag(Using yellowbrick)

```
[19] # list for contain postagged words
     tagged_words = []

     #Pos tagging eac
     for words in Data["Cleaned_Tweets"]:
       tagged_words.append(nltk.pos_tag(words))

     #Change the form to use in yellowbrick
     tagged_words = [tagged_words]
```

```
#library to visualizae postag
from yellowbrick.text import PosTagVisualizer
```

```
# Create the visualizer, fit, score, and show it
viz = PosTagVisualizer()
viz.fit(tagged_words)
viz.show()
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7effe85c4ac8>
```

Data Visualization(FreqDist, Barplot, wordcloud)

2) Data Visualization and analysis for critical steps

```
[23]
     from nltk.probability import FreqDist
     import matplotlib.pyplot as plt
```

```
#Empty lists to store positive and negative words
positive_words  =[]
negative_words = []
```

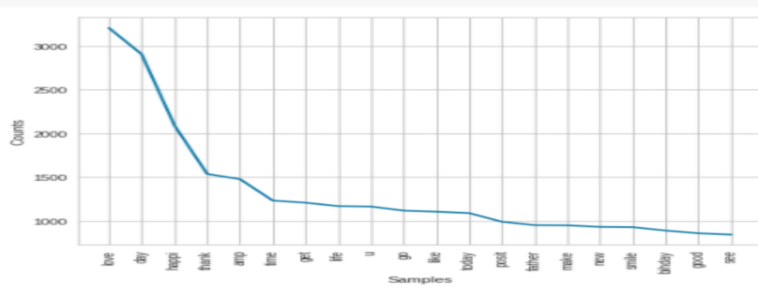
```
[25] #Classfy positive words from cleaned tweets
     for word in Data["Cleaned_Tweets"][Data["label"]==0]:
       for w in word:
         positive_words.append(w)
```

```
[26] #Classfy negative words from cleaned tweets
     for word in Data["Cleaned_Tweets"][Data["label"]==1]:
       for w in word:
         negative_words.append(w)
```

Visualization using FreqDist

```
[27] fdist = FreqDist(positive_words)
     positive_frequent = fdist.most_common(20)
```

```
#Visualize the most 20 frequennt word in positive tweet
fdist.plot(20)
```



Visualization using barplot

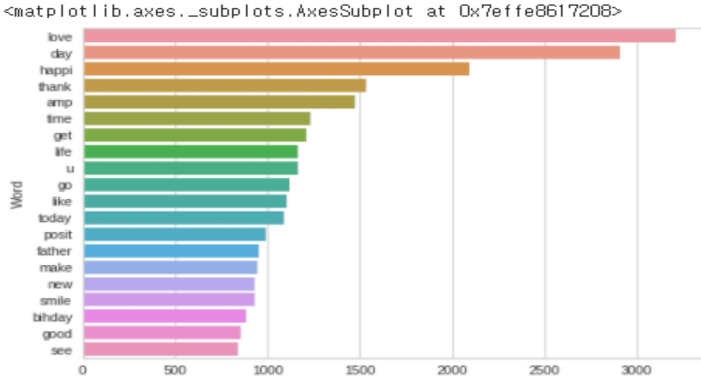
```
[31] #make dataframe for barplot
     df_positive = pd.DataFrame(positive_frequent, columns=["Word", "Count"])
     df_negative = pd.DataFrame(negative_frequent, columns=["Word", "Count"])
```

```
#library for barplot
import seaborn as sns
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
  import pandas.util.testing as tm
```

```
[33] #Positve words barplot
     sns.barplot(data=df_positive, y="Word", x="Count")
```

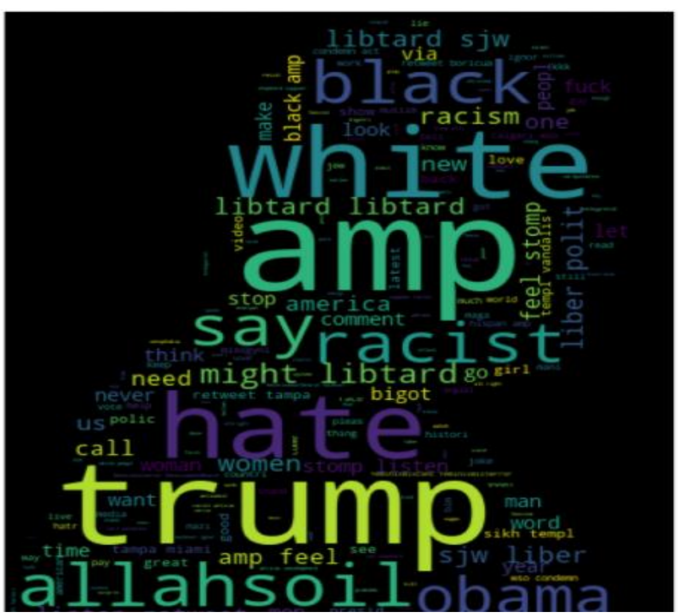
```
<matplotlib.axes._subplots.AxesSubplot at 0x7effe8617208>
```



Word cloud(Positive words and negative words frequency)

```
Mask = np.array(Image.open(requests.get('https://res.cloudinary.com/practicaldev/image/fetch/s--sKgAIMeY--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https://dev-to-uploads.s3.amazonaws.com/i/9fzqwxykz0ha9emyrjin.jpg', stream=True).raw))

# We use the ImageColorGenerator library from Wordcloud
# Here we take the color of the image and impose it over our wordcloud
image_colors = ImageColorGenerator(Mask)

#Generating the wordcloud :
wordcloud = WordCloud(background_color='black', height=1500, width=4000,mask=Mask).generate(" ".join(positive_words))

#Plot the wordcloud :
plt.figure(figsize = (12, 12))
plt.imshow(wordcloud)

#To remove the axis value :
plt.axis("off")
plt.show()
```


```
#Change the mask
Mask = np.array(Image.open(requests.get('https://miro.medium.com/max/878/1*ALByHE3fv8xNfD1eTx12XQ.png', stream=True).raw))
image_colors = ImageColorGenerator(Mask)

#Generating the wordcloud with negative words
wordcloud = WordCloud(background_color='black', height=1500, width=4000,mask=Mask).generate(" ".join(negative_words))

#Plot the wordcloud
plt.figure(figsize = (12, 12))
plt.imshow(wordcloud)

#To remove the axis value
plt.axis("off")
plt.show()
```



TFIDF vectors

(Feature extraction for making training data it is useful you need to reduce the number of resources needed for processing without losing important or relevant information.)

TFIDF vectors

```
[38] #Joining listed text to use in tfidf vector
     Data["Cleaned_Tweets"] = Data["Cleaned_Tweets"].apply(lambda x: " ".join(x))
```

```
     #import required library
     from sklearn.feature_extraction.text import TfidfVectorizer
     #create an object
     vectorizer = TfidfVectorizer(norm = None)
```

```
[40] #Generating output for TF_IDF
     X = vectorizer.fit_transform(Data["Cleaned_Tweets"])
```

```
[41] X.todense()

     matrix([[0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             ...,
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.],
             [0., 0., 0., ..., 0., 0., 0.]])
```

```
[41]
```

train/test split of data(assign target data and split train/test for 7:3 ratio)

train/test split of data

```
[42] from sklearn.model_selection import train_test_split
     #target data for model
     y = Data["label"]
```

```
[43] # Split dataset into training set and test set
     x_train, x_test, y_train, y_test = train_test_split(X,y,train_size=0.7, test_size=0.3,random_state=15)#70% training and 30% test
```

Model building and successfully executing the model to make prediction

LogisticRegression

```
[44] #Import Log_Reg model
     from sklearn.linear_model import LogisticRegression
```

```
[45] #Create a Log_Reg
     Log_Reg = LogisticRegression()
```

```
[46] #Train the model using the training sets
     model1 = Log_Reg.fit(x_train,y_train)
```

```
[47] #Predict the response for test dataset
     pred_y = model1.predict(x_test)
```

```
[47]
```

```
[48] #Import scikit-learn metrics module for accuracy calculation
     from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score
```

```
[49] #Evaluation of model
     print("For LogisticRegression")
     print("Accuracy {}".format(accuracy_score(y_test, pred_y)))
     print("Recall {}".format(recall_score(y_test, pred_y)))
     print("f1_Score {}".format(f1_score(y_test, pred_y)))
     print("Precision {}".format(precision_score(y_test, pred_y)))
```

```
For LogisticRegression
Accuracy 0.9551569506726457
Recall 0.5739130434782609
f1_Score 0.6481178396072013
Precision 0.7443609022556391
```

Complement Naive Bayes

```
[50] #Import CNB model
     from sklearn.naive_bayes import ComplementNB
```

```
[51] gnb = ComplementNB()
```

```
[52] model2 = gnb.fit(x_train ,y_train)
```

```
[53] pred_y = model2.predict(x_test)
```

```
[54] #Evaluation of model
     print("For CNB model")
     print("Accuracy {}".format(accuracy_score(y_test, pred_y)))
     print("Recall {}".format(recall_score(y_test, pred_y)))
     print("f1_Score {}".format(f1_score(y_test, pred_y)))
     print("Precision {}".format(precision_score(y_test, pred_y)))
```

```
For CNB model
Accuracy 0.8093648972781312
Recall 0.8666666666666667
f1_Score 0.39550264550264547
Precision 0.25621251071122536
```

DecisionTreeClassifier

```
[ ] from sklearn import tree
```

```
[56] clf = tree.DecisionTreeClassifier()
     model3 = clf.fit(x_train, y_train)
```

```
[57] pred_y = model3.predict(x_test)
```

```
[58] #Evaluation of model
     print("For DecisionTreeClassifier")
     print("Accuracy {}".format(accuracy_score(y_test, pred_y)))
     print("Recall {}".format(recall_score(y_test, pred_y)))
     print("f1_Score {}".format(f1_score(y_test, pred_y)))
     print("Precision {}".format(precision_score(y_test, pred_y)))
```

```
For DecisionTreeClassifier
Accuracy 0.9529669412868912
Recall 0.5797101449275363
f1_Score 0.6394884092725819
Precision 0.7130124777183601
```

Challenges that I faced:

The most difficult challenge that I faced was that I was not used to building model for maschine learning. But I overcame this problem by exploring many materials that explain model building. Scikit learn documentation was escpescially helpful for me.

Video link

https://youtu.be/QFMRd58rLhY

Any inside about the data or the ICP in general:

I was confused about predict, predict_proba which are method for prediction in model. And I found out that one is predicting value based on x while other one is predict the probability of each class will be. For example, if labels are 0 ,1, **Predict** will give either 0 or 1 as output **Predict_proba** will give the only probability of 1.