Amritpal Singh

Project - Sarcasm Detection

Introduction: In this project, I predicted sarcasm in the news headlines from the dataset given with the help of Machine Learning algorithms. I choose two algorithms to detect sarcasm: Logistic Regression and Random Forest. I made this project on Jupyter Notebook. Below are the steps of this project**:**

Technologies Used: Python, Machine Learning, Logistic Regression, Random Forest, MongoDB, Word Cloud, Matplotlib, Text Mining, NLTK, json, Joblib, etc.

Instructions: There are two files:

- Sarcasm Detection.ipynb (main file)
- userPrediction.ipynb (for inputting new single headline)

After saving the folder as it is, just need to run the files and all the code and it's output/result can be seen in the main file once running. It will take some time to upload the TermDocumentFrequency data file as it is very large and required high RAM and SSD. But except it, everything will work fast and efficiently. Also, the "userPrediction.ipynb" file can be run from the main file(in the last cell).

Code

Word Cloud for top 200 frequent words:

```
#splitting sarcastic and not sarcastic headlines for visualizing wordcloud

sarcastic = []

notSarcastic = []

for i in range (0,len(dataset)):

if dataset['is_sarcastic'][i] == 1:

        sarcastic.append(dataset['headline'][i])

else:

        notSarcastic.append(dataset['headline'][i])

#Visualizing sarcastic headlines in wordcloud

wordcloud_sarcastic=WordCloud(width=1000,height=500,stopwords=STOPWORDS
        ,background_color =’white’).generate(‘’.join(sarcastic)[0:200])
plt.figure(figsize = (15,8))
plt.imshow(wordcloud_sarcastic)
plt.axis('off')
plt.title("Sarcastic", fontsize = 50, fontweight='bold')
plt.show()
```

1. (Filename – Sarcasm Detection.ipynb)

Step 1: Importing Libraries

```
#Loading libraries
import pymongo as pymongo
from pymongo import MongoClient
import pprint
import json
import csv
from textblob import TextBlob
import os
import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from textblob.sentiments import NaiveBayesAnalyzer
from nltk.corpus import stopwords
import string
import nltk
import textmining
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from wordcloud import WordCloud, STOPWORDS
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

import seaborn as sns

from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score

from sklearn.ensemble import RandomForestClassifier

import joblib
```

Step 2: Creating database and fetching dataset from the database(MongoDB)

```
#Saving the dataset in the database, we can also read csv file dirctly or can fetch the dataset from database(MongoDB)

def Database():

    # Connect to the MongoDB, change the connection string per your MongoDB environment

    Channel = MongoClient(port=27017)

    # Set the db object to point to the business database

    Select_db = Channel.SarcasmDetection

    # Set collection name

    Select_collection = Select_db.SarcasmHeadlinesDataset

    # Extract data from mongodb

    # Create json file

    cursor = Select_collection.find()

    # Convert json into pandas

    result = pd.DataFrame(list(cursor))

    return result

dataset = Database()
```

Step 3: Deleting Unnecessary columns from the dataset, checking null values in the dataset if any

```
dataset = dataset.drop(['_id', 'article_link'], axis = 1)

#Checking is there is any null value

dataset.isnull().sum()

#Counting all the values.

dataset.count()
```

Step 4: Pre-processing

```
#stop words and punctuations

stop = set(stopwords.words("english"))

exclude = set(string.punctuation)

#removing stop words, punctuations, numbers from our dataset

def clean(doc):

        stop_free = ' '.join([i for i in doc.lower().split() if i not in stop])

        punc_free = ''.join([i for i in stop_free if i not in exclude])

        num_free = ''.join(i for i in punc_free if not i.isdigit())

        return (num_free)

post_corpus = [clean(dataset.iloc[i,1]) for i in range(0, dataset.shape[0])]
```

Step 5: Visualization(WordCloud)

```
#splitting sarcastic and not sarcastic headlines for visualizing wordcloud

sarcastic = []

notSarcastic = []

for i in range (0,len(dataset)):

        if dataset['is_sarcastic'][i] == 1:
```

```
                        sarcastic.append(dataset['headline'][i])
                else:
                        notSarcastic.append(dataset['headline'][i])



#Visualizing sarcastic headlines in wordcloud

wordcloud_sarcastic = WordCloud(width = 1000, height = 500, stopwords =
            STOPWORDS,  background_color = 'white').generate(''.join(sarcastic))
plt.figure(figsize = (15,8))
plt.imshow(wordcloud_sarcastic)
plt.axis('off')
plt.title("Sarcastic", fontsize = 50, fontweight='bold')
plt.show()

#Visualizing non-sarcastic headlines in wordcloud
wordcloud_notSarcastic = WordCloud(width = 1000, height = 500, stopwords =
            STOPWORDS, background_color = 'white').generate(''.join(notSarcastic))
plt.figure(figsize = (15,8))
plt.imshow(wordcloud_notSarcastic)
plt.axis('off')
plt.title("Non-Sarcastic", fontsize = 50, fontweight='bold')
plt.show()
```

Step 6: Creating Document Term Matrix

```
# Create document term matrix
tdm = textmining.TermDocumentMatrix()
for i in post_corpus:
    tdm.add_doc(i)

# Write tdm - saving matrix in csv format in system.

tdm.write_csv("TDM_DataFRame.csv", cutoff = 1)


# Load dataframe from system.

df = pd.read_csv("TDM_DataFRame.csv")
```

Step 7: Prediction

```
x = df

y = dataset['is_sarcastic']

#Splitting data

x_train, x_test, y_train, y_test = train_test_split(x,y, stratify = y, test_size = 0.3,

        random_state = 101)

#Normalizing the values

y_train.value_counts(normalize = True)

y_test.value_counts(normalize = True)
```

Step 7(a): Logistic Regression

```
def LR(dataset):

        #SAVING THE MODEL BECAUSE WE DO NOT NEED TO TRAIN THE
        MODEL AGAIN & AGAIN. SO, AFTER TRAINING, SAVE THE MODEL with
        joblib library.

        #TRAINING & PREDICTING

        #logmodel = LogisticRegression()

        #logmodel.fit(x_train, y_train)

        #joblib.dump(logmodel, 'lr.pkl')

        lr_model = joblib.load('lr.pkl')

        lr_predictions = lr_model.predict(x_test)

        return lr_predictions
```

```
lr_predictions = LR(dataset)
```

Accuracy:

```
#ACCURACY
def LR_accuracy():
        accuracy = accuracy_score(y_test, lr_predictions)
        print("Accuracy:",int(accuracy*100),"%\n")
        #CONFUSION MATRIX
        print("Confusion Matrix:\n", confusion_matrix(y_test, lr_predictions),"\n")
        print("Classification Report:\n",classification_report(y_test, lr_predictions))
        print("HeatMap:\n", sns.heatmap(confusion_matrix(y_test, lr_predictions)))
LR_accuracy()
```

Step 7(b): Random Forest

```
def RF(dataset):
        #rf_model = RandomForestClassifier(n_estimators = 20).fit(x_train, y_train)
        #rf_predictions = RF_model.predict(x_test)
        #SAVING MODEL
        #joblib.dump(rf_model, 'rf.pkl')
        rf_model = joblib.load('rf.pkl')
        rf_predictions = rf_model.predict(x_test)
        return rf_predictions
rf_predictions = RF(dataset)
```

Accuracy:

```
def RF_accuracy():
        #ACCURACY
        RF_accuracy = accuracy_score(y_test, rf_predictions)
        print("Accuracy:",int(RF_accuracy*100),"%\n")
        #CONFUSION MATRIX
        print("Confusion Matrix:\n", confusion_matrix(y_test, rf_predictions),"\n")
        print("Classification Report:\n",classification_report(y_test, rf_predictions))
        print("HeatMap:\n", sns.heatmap(confusion_matrix(y_test, rf_predictions)))
RF_accuracy()
```

After applying two algorithms: Logistic Regression & Random Forest, we got predictions from both of the algorithm and found that Logistic Regression is more accurate than Random Forest as:

Logistic Regression: 79%

Random Forest: 74%

Step 8: Running another file to detect sarcasm for single headline entered by the user

```
%run userPrediction.ipynb
```

2. (Filename – userPrediction.ipynb)

Step 1: Importing Libraries

```
#Loading libraries
import pymongo as pymongo
from pymongo import MongoClient
import pprint
import json
import csv
from textblob import TextBlob
import os
import pandas as pd

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

from textblob.sentiments import NaiveBayesAnalyzer

from nltk.corpus import stopwords

import string

import nltk

import textmining

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

import joblib
```

Step 2: Connecting Dataset to fetch the dataset stored in it(MongoDB)

```
#saving the dataset in the database, we can also read csv file dirctly or can fetch the dataset from database(MongoDB)

def Database():

    # Connect to the MongoDB, change the connection string per your MongoDB environment

    Channel = MongoClient(port=27017)

    # Set the db object to point to the business database

    Select_db = Channel.SarcasmDetection

    # Set collection name

    Select_collection = Select_db.SarcasmHeadlinesDataset

    # Extract data from mongodb

    # Create json file

    cursor = Select_collection.find()

    # Convert json into pandas

    result = pd.DataFrame(list(cursor))

    return result
```

```
dataset = Database()

dataset
```

Step 3: Removing unnecessary columns

```
dataset = dataset.drop(['_id', 'article_link'], axis = 1)

dataset
```

Step 4: Selecting subset of the data as data is very large to train and load again & again

```
dataset = dataset.iloc[:2000]

dataset
```

Step 5: Get input from the user to get sarcasm detected for it.

```
def giveInput():

    userInput = input("Text: ")

    return userInput

obj = giveInput()

obj
```

Step 6: PreProcessing – removinf stop words, punctuations & numbers)

```
#stop words and punctuations

stop = set(stopwords.words("english"))

exclude = set(string.punctuation)

#removing stop words, punctuations, numbers from our dataset

def clean(doc):

    stop_free = ' '.join([i for i in doc.lower().split() if i not in stop])

    punc_free = ''.join([i for i in stop_free if i not in exclude])
    num_free = ''.join(i for i in punc_free if not i.isdigit())
    return (num_free)
post_corpus = [clean(dataset.iloc[i,1]) for i in range(0, dataset.shape[0])]
```

Step 6: PreProcessing – for single sentence(input) entered by user. And, append it in the post_corpus

```
userInput = clean(obj)
post_corpus.append(userInput)
post_corpus
```

Step 7: Creating Document Term Matrix

```
# Create document term matrix
tdm = textmining.TermDocumentMatrix()
for i in post_corpus:
        tdm.add_doc(i)
# Write tdm - saving matrix in csv format in system.
tdm.write_csv("userPrediction\\userInput.csv", cutoff = 1)

# Load dataframe from system.
df = pd.read_csv("userPrediction\\userInput.csv")
userInput = df.iloc[len(df)-1]
df = df.drop(len(df)-1)
```

Step 8: Spliting Dataset (test, train)

```
x = df
y = dataset['is_sarcastic']
```

```
#Splitting data

x_train, x_test, y_train, y_test = train_test_split(x,y, stratify = y, test_size = 0.3, random_state
= 101)

#Normalizing the values

y_train.value_counts(normalize = True)

y_test.value_counts(normalize = True)
```

Step 9: Prediction – using Logistic Regression

```
def LR(dataset):

        #SAVING THE MODEL BECAUSE WE DO NOT NEED TO TRAIN THE MODEL
        AGAIN & AGAIN. SO, AFTER TRAINING SAVE THE MODEL.

          #TRAINING & PREDICTING

          logmodel = LogisticRegression()

          logmodel.fit(x_train, y_train)

          joblib.dump(logmodel, 'userPrediction\\lr.pkl')

          lr_model = joblib.load('userPrediction\\lr.pkl')

          lr_predictions = lr_model.predict(x_test)

          return lr_predictions

lr_predictions = LR(dataset)
```

Step 10: Getting output (Sarcasm or Not)

```
def Prediction(lr_predictions):

          if lr_predictions[-1] == 0:

                    prediction = "Sarcasm is Not Present"

                    print(prediction)

          elif lr_predictions[-1] == 1:

                    prediction = "Sarcasm is Present"
```

```
        print(prediction)
    else:
        print("There is some error!")

Prediction(lr_predictions)
```