Sentiment Twitter Analysis
Course Project
CS644 Introduction to Big Data
Michael Woo and Swapnil Basu

Installed Packages

We used the following packages in order to perform all operations and analyses on our acquired data:

1. Numpy - used to perform a wide variety of mathematical operations on arrays.
2. Pandas - used to perform mathematical operations on tables
3. Scikit-Learn - used to perform classification metrics
4. PySpark - used to initiate an use Spark Sessions within Python
5. Google_drive_downloader - library to access files within Google Drive
6. NLTK - library for statistical natural language processing
7. Pymongo - library for working with MongoDB instances and tables

Used IDE

We used the Python IDE Jupyter Notebook in order to write and test all code.

Data Processing Techniques

1. Bag-Of-Words
    a. The bag-of-words model is a simplifying representation used in machine learning and natural language processing. In this model, a text is represented as a bag of words with no regard to grammar but keeps multiplicity. It is used for methods of document classification where the frequency of occurrence of each word is used as a feature for training a classifier.
2. Removing stop words/common words
    a. We identify stop words or common words in our tweets to ignore in our analysis because we do not need these words taking up space in our database and ultimately skewing the results.

Applied Model

1. We used the sklearn.metrics module with a split of 70-30 with our acquired tweets from a dataset in order to train our model. After the model was trained we were able to distinguish reasonably well whether or not a tweet is positive or negative.
2. Then we retrained our model using Logistic Regression with the full dataset.

Streaming Twitter Data

1. We created a Twitter Developer Account and generated all the necessary access tokens which would allow us to use the Python library Tweepy to access the Twitter Streaming API.
2. We set up the proper authentication using our login details, created an instance of class Stream, filtered on English tweets and saved the data sample of 100 tweets from the Stream.

Storing the text and predictions into MongoDB

1. From the tweets acquired in the stream, we created a new dataframe and transformed the data to fit our original data pipeline. Next, we extracted the columns upon which we would run our model ("text" and "features") and performed our Logistic Regression model on it.
2. Finally, we imported the Pymongo library and established a connection to our "mongodb://localhost:27017/" and we used a for loop to insert the output of the Logistic Regression model into this MongoDB table. We can then query our results to MongoDB to show that our model worked and our data was stored.

Screenshots

Create spark session object (Data Processing)

```
In [4]: spark=SparkSession.builder.appName('classification_tweet').getOrCreate()
```

Load in data

```
In [5]: training_data = spark.read.csv("/Users/mwoo/Downloads/training.1600000.processed.noemoticon.csv",header=False)
```

SS.1 Loading in the Data

Exploratory

```
In [9]: training_data.describe()
Out[9]: DataFrame[summary: string, target: string, id: string, date
```

Selecting the target value and text

These are the values that we believe are important in predicting whether a twee

```
In [10]: df = training_data.select('text','target')

In [11]: df.show(5)
+--------------------+------+
|                text|target|
+--------------------+------+
|@switchfoot http:...|     0|
|is upset that he ...|     0|
|@Kenichan I dived...|     0|
|my whole body fee...|     0|
|@nationwideclass ...|     0|
+--------------------+------+
only showing top 5 rows
```

SS.2 Exploring the data

```
13]: df.groupBy("target").count().orderBy(col("count").desc()).show()
```

```
+------+------+
|target| count|
+------+------+
|     0|800000|
|     4|800000|
+------+------+
```

SS.3 We can see that it is an even split between positive and negative tweets from the data file

Stop Words Remover

Remove unnecessary words

```
n [16]: sp = set(string.punctuation)
stop_words = set(stopwords.words('english'))
extra_words = {"http","https","amp","rt","t","c","the"}
for i in extra_words:
    stop_words.add(i)
stop_words = list(stop_words)
```

```
n [17]: stopwordsRemover = StopWordsRemover(inputCol="words", outputCol="filtered").setStopWords(stop_words)
```

Bag of words count

This is a type of feature engineering

```
n [18]: countVectors = CountVectorizer(inputCol="filtered", outputCol="features", vocabSize=10000, minDF=5)
```

SS.4 Data processing and feature engineering

```
label_stringIdx = StringIndexer(inputCol = "target", outputCol = "label")
```

```
pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, countVectors, label_stringIdx])
pipelineFit = pipeline.fit(df)
dataset = pipelineFit.transform(df)
dataset.show(5)
```

```
+--------------------+------+--------------------+--------------------+--------------------+-----+
|                text|target|               words|            filtered|            features|label|
+--------------------+------+--------------------+--------------------+--------------------+-----+
|@switchfoot http:...|     0|[switchfoot, http...|[switchfoot, twit...|(10000,[1,10,16,6...|  0.0|
|is upset that he ...|     0|[is, upset, that,...|[upset, update, f...|(10000,[6,70,172,...|  0.0|
|@Kenichan I dived...|     0|[kenichan, i, div...|[kenichan, dived,...|(10000,[4,213,251...|  0.0|
|my whole body fee...|     0|[my, whole, body,...|[whole, body, fee...|(10000,[3,325,374...|  0.0|
|@nationwideclass ...|     0|[nationwideclass,...|[nationwideclass,...|(10000,[20,486],[...|  0.0|
+--------------------+------+--------------------+--------------------+--------------------+-----+
only showing top 5 rows
```

SS.5 *The training spark dataframe that was made through a pipeline*

```
(trainingData, testData) = dataset.randomSplit([0.7, 0.3], seed = 100)
print("Training Dataset Count: " + str(trainingData.count()))
print("Test Dataset Count: " + str(testData.count()))
```

```
Training Dataset Count: 1120280
Test Dataset Count: 479720
```

SS.6 Splitting the data into 2 sets (training and testing)

```
lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0).fit(trainingData)
predictions = lr.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
    .select("text","probability","label","prediction") \
    .orderBy("probability", ascending=False) \
    .show(n = 10, truncate = 30)
predictions = lr.transform(testData)
predictions.show(10)
```

```
+------------------------------+------------------------------+-----+----------+
|                          text|                   probability|label|prediction|
+------------------------------+------------------------------+-----+----------+
|@KoolioHoolio see i didnt e...|[0.9983509482201712,0.00164...|  1.0|       0.0|
|you suck you suck you suck ...|[0.9956474979112804,0.00435...|  0.0|       0.0|
|super pissed that another t...|[0.9952937171301914,0.00470...|  0.0|       0.0|
|Things I'm feeling now: ang...|[0.9942597194931533,0.00574...|  0.0|       0.0|
|so sad, me equal sad, no so...|[0.9926117815441087,0.00738...|  0.0|       0.0|
|is feeling sad and stressed...|[0.9921148433314205,0.00788...|  0.0|       0.0|
|today i kinda feel sick of ...|[0.9918217293827665,0.00817...|  0.0|       0.0|
|Been sick with sore throat ...|[0.9900972897854806,0.00990...|  0.0|       0.0|
|Throat is killing me, runny...|[0.9884381180833721,0.01156...|  0.0|       0.0|
|Ugh my nose is stuffy, my t...|[0.9883616611740987,0.01163...|  0.0|       0.0|
+------------------------------+------------------------------+-----+----------+
only showing top 10 rows
```

SS.7 Used a Logistic Regression Classification approach to classify tweets

```
Test Area Under ROC: 0.8472088126136625
```

	precision	recall	f1-score	support
0.0	0.79	0.75	0.77	239942
1.0	0.76	0.80	0.78	239778
accuracy			0.77	479720
macro avg	0.77	0.77	0.77	479720
weighted avg	0.77	0.77	0.77	479720

SS.8 Classification Metrics

```
tweet_list = list()
# Subclass Stream to print IDs of Tweets received
class IDPrinter(tweepy.Stream):

    def on_status(self, status):
        tweet_list.append(status.text)
        #print(tweet_list)
        #print(status.text)
        if len(tweet_list) == 100:
            Stream.disconnect(self)
# Initialize instance of the subclass
printer = IDPrinter(
  API_KEY, API_KEY_SECRET,
  ACCESS_TOKEN, ACCESS_TOKEN_SECRET
)

printer.sample(languages=['en'])
```

```
Stream connection closed by Twitter
```

SS.9 Twitter Stream Data with a limit of 100 tweets that are sampled in English

```
df_2 = pd.DataFrame(np.array(tweet_list))
df_2.columns = ['text']
df_2 = spark.createDataFrame(df_2)
df_2.show()
```

```
+--------------------+
|                text|
+--------------------+
|RT @NotOwenMeany:...|
|RT @CryptoTownEU:...|
|RT @MCU_Source: B...|
|@composerchris Th...|
|i actually want t...|
|RT @emailmanROCKS...|
```

SS.10 New Dataframe from stream

```
dataset_1 = pipelineFit.transform(df_2)
dataset_1.show()
```

```
+--------------------+--------------------+--------------------+--------------------+
|                text|               words|            filtered|            features|
+--------------------+--------------------+--------------------+--------------------+
|RT @NotOwenMeany:...|[rt, notowenmeany...|[notowenmeany, dc...|(10000,[910,1208,...|
|RT @CryptoTownEU:...|[rt, cryptotowneu...|[cryptotowneu, ai...|(10000,[40,263,34...|
|RT @MCU_Source: B...|[rt, mcu_source, ...|[mcu_source, brea...|(10000,[90,1122,1...|
|@composerchris Th...|[composerchris, t...|[composerchris, l...|(10000,[883,1612,...|
|i actually want t...|[i, actually, wan...|[actually, want, ...|(10000,[26,187,73...|
|RT @emailmanROCKS...|[rt, emailmanrock...|[emailmanrocks, r...|(10000,[38,145,59...|
|RT @secretyarrow:...|[rt, secretyarrow...|[secretyarrow, lo...|(10000,[996,1122,...|
```

SS.11 Process the stream data through a pipeline

```
model_predictions = lr.transform(dataset_test)
```

```
model_predictions.show()
```

```
+--------------------+--------------------+--------------------+--------------------+----------+
|                text|            features|       rawPrediction|         probability|prediction|
+--------------------+--------------------+--------------------+--------------------+----------+
|RT @NotOwenMeany:...|(10000,[910,1208,...|[-0.4205481364755...|[0.39638559333384...|       1.0|
|RT @CryptoTownEU:...|(10000,[40,263,34...|[-0.2426675444595...|[0.43962908141970...|       1.0|
|RT @MCU_Source: B...|(10000,[90,1122,1...|[-0.1111318162142...|[0.47224560461601...|       1.0|
|@composerchris Th...|(10000,[883,1612,...|[0.33328114913442...|[0.58255751614137...|       0.0|
|i actually want t...|(10000,[26,187,73...|[0.41763114414529...|[0.60291626474052...|       0.0|
|RT @emailmanROCKS...|(10000,[38,145,59...|[-1.3859129547787...|[0.20006103199723...|       1.0|
+--------------------+--------------------+--------------------+--------------------+----------+
```

SS.12 Predictions made on the stream data

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["my_database"]
mycol = mydb["predictions"]
```

SS.13 MongoDB to store the text and predictions of the streamed data

```
for index, row in mp_df.iterrows():
    #print(row['text'], row['prediction'])
    mydict = { "text": row['text'], "prediction": row['prediction'] }
    mycol.insert_one(mydict)
```

```
RT @NotOwenMeany: DC Christmas Light Hunters: The area north of American University (roughly Van Ness to Davenport an
d 43rd to 49th streets… 1.0
RT @CryptoTownEU: 🚀 Airdrop: BAoE Global
💰 Value: 10 $BAoE
👥 Referral: 3 $BAoE
📒 Partnership: Groo INTERNATIONAL, Meta Ultra Holdings, SEM… 1.0
RT @MCU_Source: BREAKING: #Deadpool's first appearance at the #MCU will be in #DoctorStrangeInTheMultiverseOfMadness!
https://t.co/KZYAVcAC… 1.0
```

SS.14 Inserting the rows into MongoDB

```
myquery = { "prediction": 1.0 }
mydoc = mycol.find(myquery)
for x in mydoc:
    print(x)
```

```
{'_id': ObjectId('61bfc4d9b8bc24cd819b369c'), 'text': 'RT @NotOwenMeany: DC Christmas Light Hunters: The area north o
f American University (roughly Van Ness to Davenport and 43rd to 49th streets…', 'prediction': 1.0}
{'_id': ObjectId('61bfc4d9b8bc24cd819b369d'), 'text': 'RT @CryptoTownEU: 🚀 Airdrop: BAoE Global\n💰 Value: 10 $BAoE
\n👥 Referral: 3 $BAoE\n📒 Partnership: Groo INTERNATIONAL, Meta Ultra Holdings, SEM…', 'prediction': 1.0}
{'_id': ObjectId('61bfc4d9b8bc24cd819b369e'), 'text': 'RT @MCU_Source: BREAKING: #Deadpool's first appearance at the
#MCU will be in #DoctorStrangeInTheMultiverseOfMadness! https://t.co/KZYAVcAC…', 'prediction': 1.0}
{'_id': ObjectId('61bfc4d9b8bc24cd819b36a1'), 'text': "RT @emailmanROCKS: The Rules of Creative Writing by children's
author Kurt Chambers https://t.co/lrQszZIoxL … This is a great guide to help…", 'prediction': 1.0}
```

SS.15 Retrieving tweets that were predicted to be positive

```
myquery = { "prediction": 0.0 }
mydoc = mycol.find(myquery)
for x in mydoc:
    print(x)
```

```
{'_id': ObjectId('61bfc4d9b8bc24cd819b369f'), 'text': '@composerchris They should lose their tax exempt status', 'pre
diction': 0.0}
{'_id': ObjectId('61bfc4d9b8bc24cd819b36a0'), 'text': 'i actually want to die.', 'prediction': 0.0}
{'_id': ObjectId('61bfc4d9b8bc24cd819b36a4'), 'text': "RT @djrothkopf: Manchin did not have to say anything today. Di
d not have to go on Fox. Did not have to issue a statement. He could've kept…", 'prediction': 0.0}
{'_id': ObjectId('61bfc4d9b8bc24cd819b36a5'), 'text': 'RT @itsfkntxna: you are allowed to criticize your own communit
y and i hate to tell you this but other people are allowed to criticize your…', 'prediction': 0.0}
{'_id': ObjectId('61bfc4d9b8bc24cd819b36a8'), 'text': 'RT @ohits_laurenn: i be asking for a sign then i be ignoring t
he sign 😂🤦🏾\u200d♀️ ', 'prediction': 0.0}
```

SS.16 Retrieving tweets that were predicted to be negative