GROUP 4

CISC 886-PROJECT

PART B

PREPARED BY

Ahmed Basha	20398547
Ahmed Mohamed	20398549
Amal Fawzy	20399126
Galena Wagdy Zareef	20399124
Areeg Mansour	20399122

Submitted to

Dr. Anwar Hossain

Table of content

1. Problem specification: **2**
2. Data Collection **2**
3. Setting Up the Environment **3**
1. Checking Java Version 3
2. Setting Java 8 Environment 3
3.Downloading Spark 4
.Extracting Spark Files 4
5.Install FindSpark 4
6.Install Pyspark 4
7.Setting Up Home Environment 4
8.Create Spark Session 5
4. Getting The Data and Import Libraries **5**
1.Importing The Required Libraries 5
2.Read and Explore the data 6
5. Data Preparation and Visualization **6**
1. Feature selection 6
2. Data cleaning 7
3. Data Visualization 8
4. Data Splitting 10
6. Data Preprocessing 11
A- Tokenization 11
B- Stopwords 12
C- Hashing 13
6. Model Building **14**
1. Model Training 14
2. Model Prediction 14
7. Model Evaluation **15**
1. Classification Report: 15
2. Confusion Matrix: 15
8. Participation **16**

1. Problem specification:

Sentiment Analysis model by using PySpark to analyze social media data for gaining insights into user sentiment. We analyze the collected data to find out which tweets are positive and negative.

2. Data Collection

This is the sentiment140 dataset from Kaggle. It contains 1,600,000 tweets extracted using the Twitter API. The tweets have been annotated (0=negative, 1=positive) and they can be used to detect sentiment.

It contains the following 6 fields:

1. target: the polarity of the tweet (*0* = negative, *2* = neutral, *4* = positive)
2. ids: The id of the tweet (*2087*)
3. date: the date of the tweet (*Sat May 16 23:58:44 UTC 2009*)
4. flag: The query (*lyx*). If there is no query, then this value is NO_QUERY.
5. user: the user that tweeted (*robotickilldozr*)
6. text: the text of the tweet (*Lyx is cool*)

3. Setting Up the Environment

1. Checking Java Version

```
! java -version
```

```
openjdk version "11.0.17" 2022-10-18
OpenJDK Runtime Environment (build 11.0.17+8-post-Ubuntu-1ubuntu220.04)
OpenJDK 64-Bit Server VM (build 11.0.17+8-post-Ubuntu-1ubuntu220.04, mixed mode, sharing)
```

```
[ ] !sudo apt update
```

```
Get:1 https://cloud.r-project.org/bin/linux/ubuntu focal-cran40/ InRelease [3,622 B]
Ign:2 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu2004/x86_64  InRelease
Get:3 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Hit:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  InRelease
Hit:5 https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu2004/x86_64  Release
Hit:6 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:7 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu focal InRelease
Get:8 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Hit:10 http://ppa.launchpad.net/cran/libgit2/ubuntu focal InRelease
Hit:11 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu focal InRelease
Get:12 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:13 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [1,000 kB]
Hit:14 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu focal InRelease
Hit:15 http://ppa.launchpad.net/ubuntugis/ppa/ubuntu focal InRelease
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [2,069 kB]
```

2. Setting Java 8 Environment

```
[ ] !apt-get install openjdk-8-jdk-headless -qq
```

```
Selecting previously unselected package openjdk-8-jre-headless:amd64.
(Reading database ... 128126 files and directories currently installed.)
Preparing to unpack .../openjdk-8-jre-headless_8u352-ga-1~20.04_amd64.deb ...
Unpacking openjdk-8-jre-headless:amd64 (8u352-ga-1~20.04) ...
Selecting previously unselected package openjdk-8-jdk-headless:amd64.
Preparing to unpack .../openjdk-8-jdk-headless_8u352-ga-1~20.04_amd64.deb ...
Unpacking openjdk-8-jdk-headless:amd64 (8u352-ga-1~20.04) ...
Setting up openjdk-8-jre-headless:amd64 (8u352-ga-1~20.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to provide /usr/bin/orbd (orbd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool to provide /usr/bin/servertool (servertool) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/tnameserv to provide /usr/bin/tnameserv (tnameserv) in auto mode
Setting up openjdk-8-jdk-headless:amd64 (8u352-ga-1~20.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/idlj to provide /usr/bin/idlj (idlj) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/wsimport to provide /usr/bin/wsimport (wsimport) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jsadebugd to provide /usr/bin/jsadebugd (jsadebugd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/native2ascii to provide /usr/bin/native2ascii (native2ascii) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/javah to provide /usr/bin/javah (javah) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/clhsdb to provide /usr/bin/clhsdb (clhsdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/xjc to provide /usr/bin/xjc (xjc) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/hsdb to provide /usr/bin/hsdb (hsdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/schemagen to provide /usr/bin/schemagen (schemagen) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/extcheck to provide /usr/bin/extcheck (extcheck) in auto mode
```

```
[ ] !sudo update-alternatives --config java # We choose Selection 2- java-8-openjdk-amd64
```

```
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority   Status
------------------------------------------------------------
* 0            /usr/lib/jvm/java-11-openjdk-amd64/bin/java      1111      auto mode
  1            /usr/lib/jvm/java-11-openjdk-amd64/bin/java      1111      manual mode
  2            /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java   1081      manual mode

Press <enter> to keep the current choice[*], or type selection number: 2
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java to provide /usr/bin/java (java) in manual mode
```

3. Downloading Spark

```
[ ]  !wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz
```

4. Extracting Spark Files

```
[ ]  !tar xf spark-3.0.0-bin-hadoop3.2.tgz
```

5. Install FindSpark

```
[ ]  !pip install -q findspark
```

6. Install Pyspark

```
[ ]  ! pip install --ignore-installed -q pyspark==2.4.4
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 215.7/215.7 MB 5.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 197.3/197.3 KB 16.9 MB/s eta 0:00:00
  Building wheel for pyspark (setup.py) ... done
```

7. Setting Up Home Environment

```
[ ]  import os
     os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
     os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"
```

8. Create Spark Session

```
import findspark
findspark.init()
findspark.find()
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").appName("Project 2-SocialMedia-Analytics").config('spark.sql.execution.arrow.pyspark.enabled', True).
```

4. Getting The Data and Import Libraries

```
[ ]  !pip install opendatasets --upgrade --quiet
```

1. Importing The Required Libraries

```
[ ]  from pyspark.sql.types import *
     from pyspark.sql.functions import *
     from pyspark.ml.feature import HashingTF, Tokenizer, StopWordsRemover
     import opendatasets as od
     import seaborn as sns
     import pandas as pd
     import numpy as np
     from wordcloud import WordCloud, STOPWORDS
     import matplotlib.pyplot as plt
     from pyspark.ml.classification import LogisticRegression
```

2. Read and Explore the data

```
[ ] df = spark.read.csv("/content/sentiment140/training.1600000.processed.noemoticon.csv", header = False, encoding = "ISO-8859-1")
```

```
[ ] df.show(n = 10)
```

```
+---+----------+--------------------+--------+--------------+--------------------+
|_c0|       _c1|                 _c2|     _c3|           _c4|                 _c5|
+---+----------+--------------------+--------+--------------+--------------------+
|  0|1467810369|Mon Apr 06 22:19:...|NO_QUERY|_TheSpecialOne_|@switchfoot http:...|
|  0|1467810672|Mon Apr 06 22:19:...|NO_QUERY|  scotthamilton|is upset that he ...|
|  0|1467810917|Mon Apr 06 22:19:...|NO_QUERY|       mattycus|@Kenichan I dived...|
|  0|1467811184|Mon Apr 06 22:19:...|NO_QUERY|        ElleCTF|my whole body fee...|
|  0|1467811193|Mon Apr 06 22:19:...|NO_QUERY|         Karoli|@nationwideclass ...|
|  0|1467811372|Mon Apr 06 22:20:...|NO_QUERY|       joy_wolf|@Kwesidei not the...|
|  0|1467811592|Mon Apr 06 22:20:...|NO_QUERY|        mybirch|         Need a hug |
|  0|1467811594|Mon Apr 06 22:20:...|NO_QUERY|          coZZ|@LOLTrish hey  lo...|
|  0|1467811795|Mon Apr 06 22:20:...|NO_QUERY|2Hood4Hollywood|@Tatiana_K nope t...|
|  0|1467812025|Mon Apr 06 22:20:...|NO_QUERY|        mimismo|@twittera que me ...|
+---+----------+--------------------+--------+--------------+--------------------+
only showing top 10 rows
```

5. Data Preparation and Visualization

Text Preprocessing is traditionally an important step for Machine Learning tasks. It transforms text into a more digestible form so that machine learning algorithms can perform better.

1. Feature selection

we Select the related Columns Using **PySpark.SQL,**

```
[ ] #Selecting the related Columns
    df2 = df.select(col('_c5').alias("Tweets"), col('_c0').cast("Int").alias("Label"))
```

2. Data cleaning

clean and Prepare the data for analysis with **PySpark.**

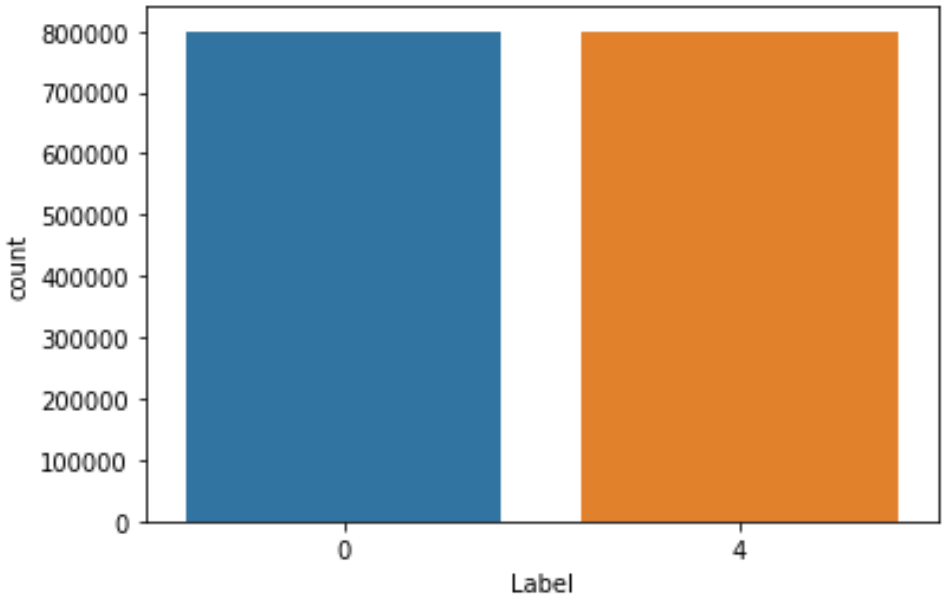
Some other data cleaning steps taken are:

1. **Searches the pattern** in the string and then replaces it with a new given expression.
2. **Replacing URLs:** Links starting with **"http" or "https" or "www"** are replaced by **"URL"**.
3. **Replacing Usernames:** Replace @Usernames with empty string. *(eg: "@Kaggle" to "")*
4. **Removing Non-Alphabets:** Replacing characters except Digits and Alphabets with a space.
5. **Removing Short Words:** Words with lengths of less than 2 are removed

```
[ ]  #Cleaining The Data
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "don't", "do not"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "won't", "will not"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "'re", " are"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "i'm", "i am"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "'m", " am"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "let's", "let us"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "'s", " is"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "'ve", " have"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "can't", "can not"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "shan't", "shall not"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "n't", " not"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "'d", " would"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), "'ll", " will"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), url, "URL"))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), user, " "))
     df2 = df2.withColumn('Tweets', regexp_replace(col('Tweets'), alpha, " "))
     processed_df = df2.withColumn('Tweets', trim(col('Tweets')))
```

3. Data Visualization

visualizing the target column (0:Negative, 4:Positive) so we can find that we have a balanced data



Now we're going to analyze the preprocessed data to get an understanding of it.

We'll plot **Word Clouds** for **Positive and Negative** tweets from our dataset and see which words occur the most.

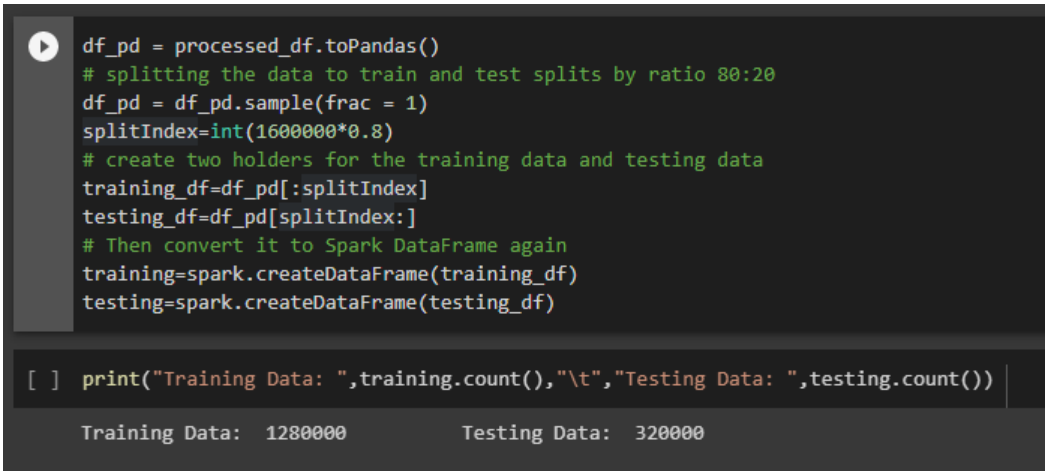
Visualizing the most used words for the negative class

Visualizing the most Used Words for the Positive Class

4. Data Splitting

Labeled data is needed to train a model to recognize patterns in the text that indicate positive or negative sentiment but first, we need to split The Preprocessed Data and divide it into 2 sets:

- **Training Data:** The dataset upon which the model would be trained on. Contains 80% data.
- **Test Data:** The dataset upon which the model would be tested against. Contains 20% data.



```
df_pd = processed_df.toPandas()
# splitting the data to train and test splits by ratio 80:20
df_pd = df_pd.sample(frac = 1)
splitIndex=int(1600000*0.8)
# create two holders for the training data and testing data
training_df=df_pd[:splitIndex]
testing_df=df_pd[splitIndex:]
# Then convert it to Spark DataFrame again
training=spark.createDataFrame(training_df)
testing=spark.createDataFrame(testing_df)
```

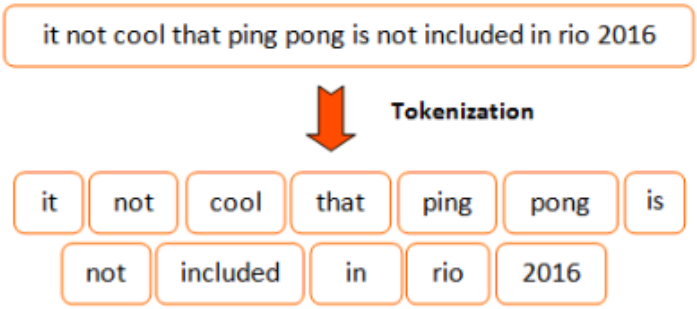
```
[ ] print("Training Data: ",training.count(),"\t","Testing Data: ",testing.count())

    Training Data:  1280000          Testing Data:  320000
```

Then, we Separated Tweets into individual Words using Tokenizer, Then use Stopwords and HashTF For Training and Testing Data using PySpark.

6. Data Preprocessing

A- Tokenization

Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called *tokens*, perhaps at the same time throwing away certain characters, such as punctuation. The process is called **Tokenization.**



Tokenizer creates tokens for every word in the data and maps them to an index

```
[ ]  tokenizer = Tokenizer(inputCol = "Tweets", outputCol = "TokenizedTweets")
     tokenizedTraining = tokenizer.transform(training)

     tokenizedTraining.show(n = 20)

+--------------------+-----+--------------------+
|              Tweets|Label|     TokenizedTweets|
+--------------------+-----+--------------------+
|I do not think sh...|    0|[i, do, not, thin...|
|idk    some rando...|    0|[idk, , , , some,...|
|themirror can lie...|    4|[themirror, can, ...|
|but I will get fi...|    0|[but, i, will, ge...|
|i know but its ju...|    0|[i, know, but, it...|
|Homeeeesie  Back ...|    0|[homeeeesie, , ba...|
|  stop being cryptic|    0|[stop, being, cry...|
|Hey Rob  how do I...|    0|[hey, rob, , how,...|
|me too  im about ...|    0|[me, too, , im, a...|
|Oh god     I have...|    0|[oh, god, , , , ,...|
|It is time to try...|    4|[it, is, time, to...|
|waiting in the li...|    0|[waiting, in, the...|
|NO WAY   I am in ...|    4|[no, way, , , i, ...|
|Thanks again  I w...|    4|[thanks, again, ,...|
|heading to the la...|    4|[heading, to, the...|
|Damn it all  I do...|    0|[damn, it, all, ,...|
|my pleasure  hope...|    4|[my, pleasure, , ...|
|trying to sleep e...|    4|[trying, to, slee...|
|i hate mopping   ...|    4|[i, hate, mopping...|
|getting materials...|    0|[getting, materia...|
+--------------------+-----+--------------------+
only showing top 20 rows
```

B- Stopwords

Stopwords are commonly used words in English which have no contextual meaning in a sentence. So therefore we remove them before classification. Some stopwords are…

```
> stopwords("english")
  [1] "i"         "me"        "my"        "myself"    "we"
  [6] "our"       "ours"      "ourselves" "you"       "your"
 [11] "yours"     "yourself"  "yourselves" "he"       "him"
 [16] "his"       "himself"   "she"       "her"       "hers"
 [21] "herself"   "it"        "its"       "itself"    "they"
 [26] "them"      "their"     "theirs"    "themselves" "what"
 [31] "which"     "who"       "whom"      "this"      "that"
 [36] "these"     "those"     "am"        "is"        "are"
 [41] "was"       "were"      "be"        "been"      "being"
 [46] "have"      "has"       "had"       "having"    "do"
 [51] "does"      "did"       "doing"     "would"     "should"
 [56] "could"     "ought"     "i'm"       "you're"    "he's"
 [61] "she's"     "it's"      "we're"     "they're"   "i've"
 [66] "you've"    "we've"     "they've"   "i'd"       "you'd"
 [71] "he'd"      "she'd"     "we'd"      "they'd"    "i'll"
 [76] "you'll"    "he'll"     "she'll"    "we'll"     "they'll"
 [81] "isn't"     "aren't"    "wasn't"    "weren't"   "hasn't"
 [86] "haven't"   "hadn't"    "doesn't"   "don't"     "didn't"
 [91] "won't"     "wouldn't"  "shan't"    "shouldn't" "can't"
 [96] "cannot"    "couldn't"  "mustn't"   "let's"     "that's"
[101] "who's"     "what's"    "here's"    "there's"   "when's"
[106] "where's"   "why's"     "how's"     "a"         "an"
```

```
[ ] stopWords = StopWordsRemover(inputCol = "TokenizedTweets", outputCol = "RemovedTweets")
    stopWordsTraining = stopWords.transform(tokenizedTraining)

[ ] stopWordsTraining.show(n = 10)

+--------------------+-----+--------------------+--------------------+
|              Tweets|Label|     TokenizedTweets|       RemovedTweets|
+--------------------+-----+--------------------+--------------------+
|I do not think sh...|    0|[i, do, not, thin...|[think, anymore, ...|
|idk    some rando...|    0|[idk, , , , some,...|[idk, , , , rando...|
|themirror can lie...|    4|[themirror, can, ...|[themirror, lie, ...|
|but I will get fi...|    0|[but, i, will, ge...|  [get, fiiiiiired]|
|i know but its ju...|    0|[i, know, but, it...|[know, rude, , ah...|
|Homeeeesie  Back ...|    0|[homeeeesie, , ba...|[homeeeesie, , ba...|
|  stop being cryptic|    0|[stop, being, cry...|     [stop, cryptic]|
|Hey Rob  how do I...|    0|[hey, rob, , how,...|[hey, rob, , get,...|
|me too  im about ...|    0|[me, too, , im, a...|[, im, travel, 7,...|
|Oh god     I have...|    0|[oh, god, , , , ,...|[oh, god, , , , ,...|
+--------------------+-----+--------------------+--------------------+
only showing top 10 rows
```

C- Hashing

HashingTF is a Transformer that takes sets of terms and converts those sets into fixed-length feature vectors. In text processing, a “set of terms” might be a bag of words.

```
hashTF = HashingTF(inputCol = "RemovedTweets", outputCol = "FeaturesTweets")
numericTraining = hashTF.transform(stopWordsTraining).select("Label", "RemovedTweets", "FeaturesTweets")
```

```
[ ] numericTraining.show(n = 10)
```

```
+-----+--------------------+--------------------+
|Label|       RemovedTweets|      FeaturesTweets|
+-----+--------------------+--------------------+
|    0|[think, anymore, ...|(262144,[5381,853...|
|    0|[idk, , , , rando...|(262144,[31015,43...|
|    4|[themirror, lie, ...|(262144,[15539,19...|
|    0|  [get, fiiiiiired]|(262144,[240634,2...|
|    0|[know, rude, , ah...|(262144,[22346,24...|
|    0|[homeeeesie, , ba...|(262144,[59243,77...|
|    0|     [stop, cryptic]|(262144,[85673,20...|
|    0|[hey, rob, , get,...|(262144,[21653,34...|
|    0|[, im, travel, 7,...|(262144,[9129,178...|
|    0|[oh, god, , , , ,...|(262144,[18184,38...|
+-----+--------------------+--------------------+
only showing top 10 rows
```

6. Model Building

After preparing data we train the model and then predict using a **logistic regression model** from **Pyspark ML Classification Library** to classify tweets as positive or negative, an approach using supervised learning.

1. Model Training

```
lr = LogisticRegression(labelCol = "Label", featuresCol = "FeaturesTweets", maxIter = 10, regParam = 0.01)
model = lr.fit(numericTraining)
```

2. Model Prediction

```
predict = model.transform(numericTesting)
prediction = predict.select("Label", "prediction", "FeaturesTweets")
prediction.show(n = 10)
```

```
+-----+----------+--------------------+
|Label|prediction|      FeaturesTweets|
+-----+----------+--------------------+
|    0|       0.0|(262144,[95697,14...|
|    0|       0.0|(262144,[29129,31...|
|    4|       4.0|(262144,[113432,1...|
|    4|       4.0|(262144,[8145,400...|
|    4|       0.0|(262144,[2565,315...|
|    0|       0.0|(262144,[1546,284...|
|    4|       0.0|(262144,[15480,11...|
|    0|       0.0|(262144,[24698,34...|
|    0|       0.0|(262144,[17893,87...|
|    4|       4.0|(262144,[1512,855...|
+-----+----------+--------------------+
only showing top 10 rows
```

7. Model Evaluation

Since our dataset is not skewed, i.e. it has an equal number of Positive and Negative Predictions. We're choosing Accuracy as our evaluation metric. Furthermore, we're plotting the Confusion Matrix to get an understanding of how our model is performing on both classification types.

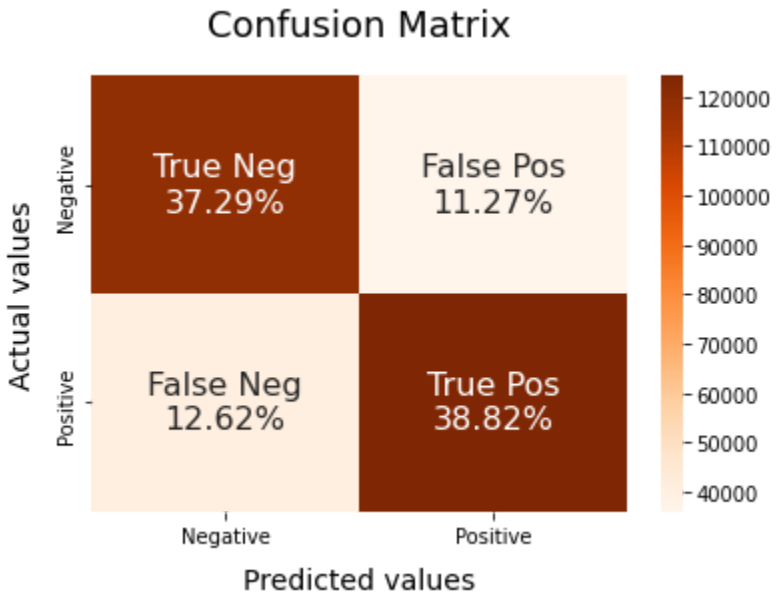
The Logistic Regression Model achieves nearly 76.2% accuracy while classifying the sentiment of a tweet. The evaluation of the model was made by Importing classification_report, confusion_matrix, and accuracy_score from sklearn.metrics to calculate and visualize the performance of the model.

1. Classification Report:

	precision	recall	f1-score	support
0	0.77	0.75	0.76	159736
4	0.75	0.78	0.76	160264
accuracy			0.76	320000
macro avg	0.76	0.76	0.76	320000
weighted avg	0.76	0.76	0.76	320000

Accuracy 76.11%

2. Confusion Matrix:



8. Participation

Task	Name
Data Collection and Setting Up the Environment	Areeg Mansour
Getting The Data and Import Libraries and Data Preparation and Visualization	Ahmed Basha, Amal Fawzy
Model Building and Model Evaluation	Galena Wagdy Zareef, Ahmed Mohamed