Social Media Analytics

[25/02/2023]

[Students]

Name	ID
Mostafa Mohamed Amer	**20398564**
Mohamed Salim	**20399127**
Mohamed Elsetohy	**20398546**
Mohammed Abdelghany	**20398561**

[SUPERVISOR]

Dr: Anwar Hossain

Abstract

This project handles social media analytics using spark to analyze social media data. The aim of the project is to predict the sentiment analysis for any tweet & post on social media.
We used a trusted dataset from Kaggle, and it is labeled data, it is about ~162,980 tweets before the cleaning process.
In order to use PySpark as an interface for Apache Spark, we began by looking at the spark tools, platforms, and machine learning algorithms that are available on the MLIB.

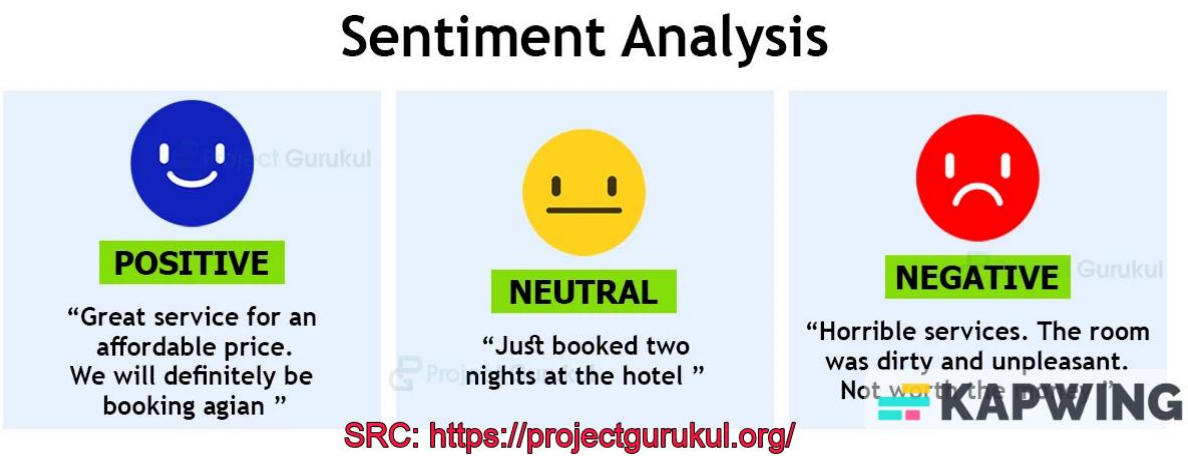
After a preliminary study of the available algorithms and reviewing our dataset, we found that the problem is classification, and we can use supervised learning in this project.

After much data cleaning, preprocessing, and trying multiple algorithms, we achieved an accuracy 85% because we tried more than one algorithm like logistic regression, naïve bayes, and random forest and finally we used the highest accuracy (logistic regression).
The process of achieving this accuracy was not easy but after much work we did it.

Contents

Introduction

Sentiment analysis is a method of reasoning that evaluates writings for polarity, from positive to negative, using statistics, natural language processing, and machine learning. Machine learning tools can automatically learn how to identify sentiment without humans by being trained with examples of feelings in text [1].



The data, being labeled by humans, has a lot of noise, and it is hard to achieve good accuracy.

The best results we achieved by logistic regression algorithm was 85%. We will compare the algorithms used in the upcoming sections.

Data collection

Data source

'Twitter Sentiment Dataset' [2]:
https://www.kaggle.com/datasets/saurabhshahane/twitter-sentiment-dataset?select=Twitter_Data.csv

We used the Kaggle Twitter Sentiment Data Set collection. The majority of the data set's entries are 162,980. This data collection was gathered over a variety of time periods from various users.

We began the coding by importing the necessary libraries, including (SparkSession, pandas, plotly)

```
In [1]: from pyspark.sql import SparkSession
        from pyspark.sql.functions import regexp_replace
        from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, CountVectorizer
        from pyspark.ml import Pipeline
        from pyspark.ml.feature import  StringIndexer, VectorAssembler
        from pyspark.ml.evaluation import MulticlassClassificationEvaluator
        from pyspark.sql.functions import size
        from pyspark_dist_explore import hist # install
        import numpy as np
        import matplotlib.pyplot as plt
        import plotly.express as px
        import pandas as pd
        import math

        %matplotlib inline
```

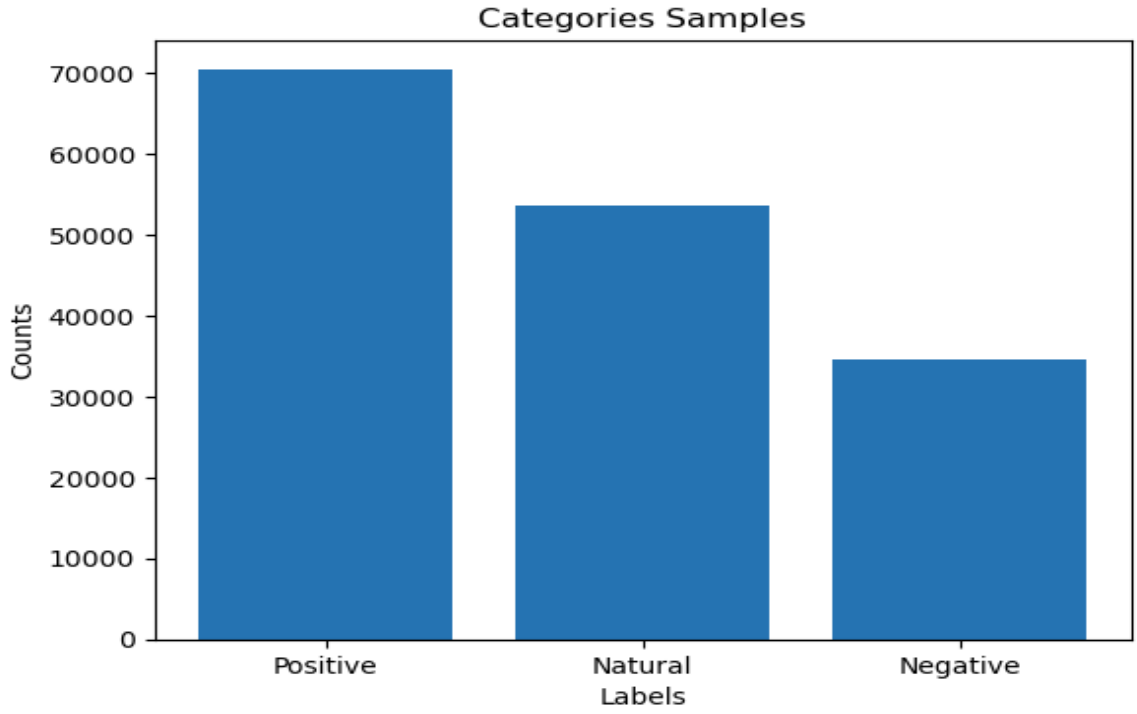
After importing the required libraries, we began downloading data from Kaggle in the form of a "CSV" file and saving it to the project's default folder.

```
# This cell runs once to download the data

import opendatasets as od  # install

od.download(
    "https://www.kaggle.com/datasets/saurabhshahane/twitter-sentiment-dataset", force=True)
```

There are two columns in the trained data (tweets, category)

- Tweets: contains the content of messages.
- Category: the label data contains three labels (positive, negative, and neutral)

```
plt.bar(['Positive', "Natural", "Negative"], df["category"].value_counts())
plt.title("Categories Samples")
plt.xlabel("Labels")
plt.ylabel("Counts")
plt.show()
```



Data representation

The positive value represented by 1
The negative value represented by -1
The neutral value represented by 0

```
xData.groupBy('Category').count().show()
```

```
+--------+-----+
|Category|count|
+--------+-----+
|      -1|34665|
|       0|53553|
|       1|70476|
+--------+-----+
```

Data Preprocessing

Data preprocessing is a data mining technique that entails converting raw data into a usable format. Data preprocessing is a tried-and-true method for dealing with such problems. Data preprocessing is the process of preparing raw data for further processing.

Data preprocessing consists of two steps

- cleaning the data
- tokenization and vectorization

Cleaning the data

We purify the data by

- Remove URLs
- Removing special characters
- Removing numbers

- Removing stop words

We need to remove links , special caracters and degits from our data

```
df = data_df.select("label", regexp_replace("Tweet", "http\S+", ""
                                    ).alias('Tweet1'))
df1= df.select("label", regexp_replace("Tweet1", "@[^\s]+[\s]?", ""
                                    ).alias('Tweet2'))
df2= df1.select("label", regexp_replace("Tweet2", "[^ a-zA-Z0-9]", ""
                                    ).alias('Tweet3'))
df3= df2.select("label", regexp_replace("Tweet3", "[0-9]", ""
                                    ).alias('Tweet'))
df3.show()
```

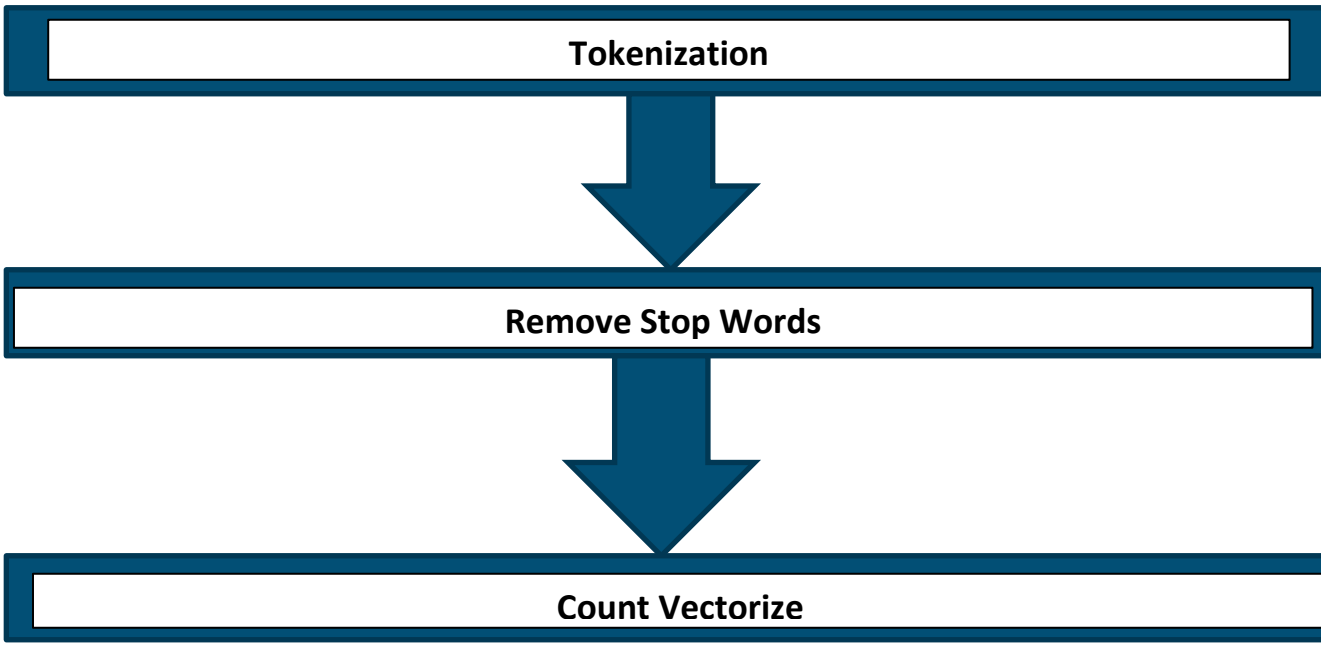
```
+-----+--------------------+
|label|               Tweet|
+-----+--------------------+
|    2|when modi promise...|
|    1|talk all the nons...|
|    0|what did just say...|
|    0|asking his suppor...|
|    0|answer who among ...|
|    1|kiya tho refresh ...|
|    1|this comes from c...|
|    0|with upcoming ele...|
|    0|gandhi was gay do...|
|    0|things like demon...|
|    0|hope tuthukudi pe...|
|    0|calm waters where...|
|    1|one vote can make...|
|    1|one vote can make...|
|    2|vote such party a...|
|    1|vote modi who has...|
|    1|through our vote ...|
|    2|didn write chowki...|
|    0|was the one who r...|
|    2|with firm belief ...|
+-----+--------------------+
only showing top 20 rows
```

Preparing the data

- First, we tokenize data based on white space.
- Second, we drop stop words (I.e., The, they, etc.).
- Finally, we make a count vectorization that will collect words in each sentence based on a pre-defined number of words.

Then we integrate these steps into a pipeline to fit it on the data

```
regexTokenizer = RegexTokenizer(inputCol="Tweet", outputCol="Tweet1", pattern="\\W")
stopwordsRemover = StopWordsRemover(inputCol="Tweet1", outputCol="Tweet2").setStopWords(add_stopwords)
countVectors = CountVectorizer(inputCol="Tweet2", outputCol="features", vocabSize=6000, minDF=5)

pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, countVectors])

pipelineFit = pipeline.fit(df3)
dataset = pipelineFit.transform(df3)
```



Then we printed the data

```
dataset.show(5)
```

```
+-----+--------------------+--------------------+--------------------+--------------------+
|label|               Tweet|              Tweet1|              Tweet2|            features|
+-----+--------------------+--------------------+--------------------+--------------------+
|    2|when modi promise...|[when, modi, prom...|[modi, promised, ...|(6000,[0,14,22,23...|
|    1|talk all the nons...|[talk, all, the, ...|[talk, nonsense, ...|(6000,[0,8,199,59...|
|    0|what did just say...|[what, did, just,...|[say, vote, modi,...|(6000,[0,2,8,16,4...|
|    0|asking his suppor...|[asking, his, sup...|[asking, supporte...|(6000,[0,32,51,63...|
|    0|answer who among ...|[answer, who, amo...|[answer, among, p...|(6000,[0,59,64,68...|
+-----+--------------------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Splitting the data

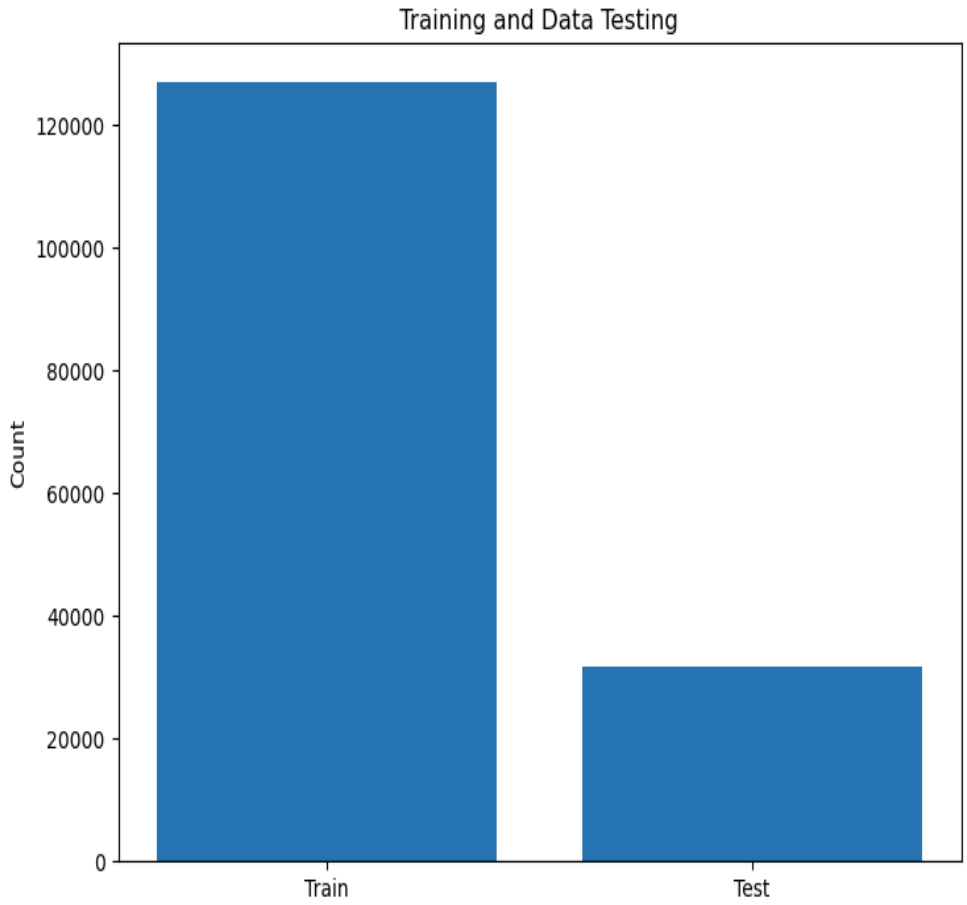
We split the data into train and test split

1- The rate of train data is 80%

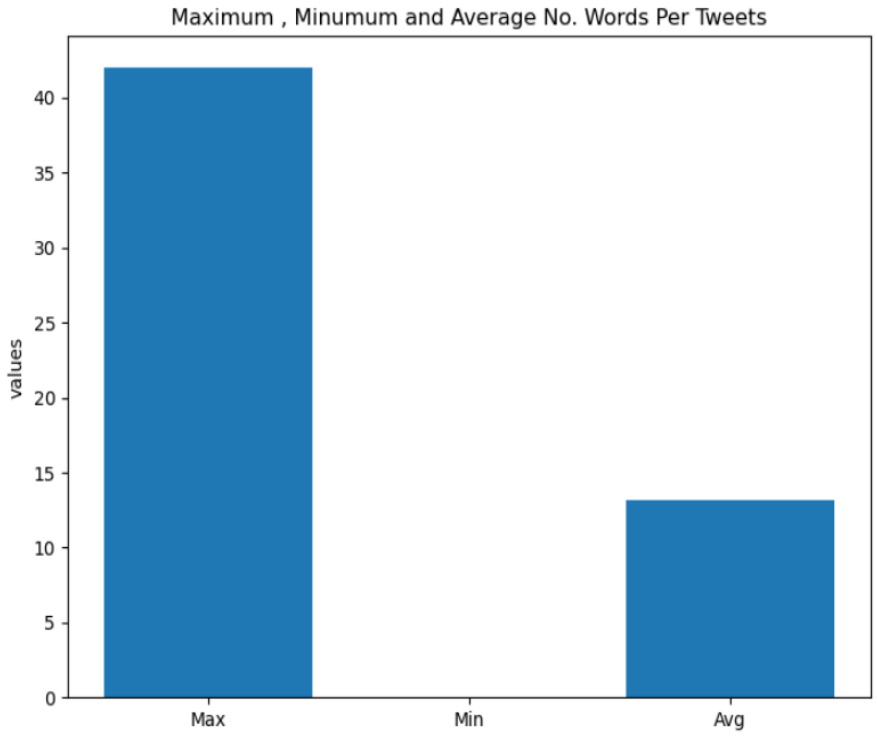
2- The rate of test data is 20%

```
In [23]: (trainingData, testData) = dataset.select("features","label").randomSplit([0.8, 0.2], seed = 20)
         print("Training Dataset Count: " + str(trainingData.count()))
         print("Test Dataset Count: " + str(testData.count()))

         Training Dataset Count: 126887
         Test Dataset Count: 31803
```

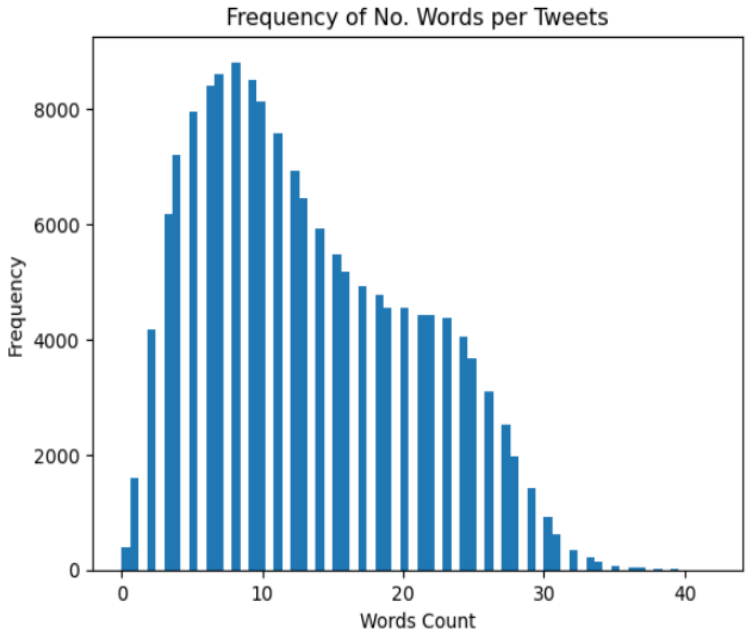
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
data_type = ['Train','Test']
counts = [trainingData.count(),testData.count()]
ax.bar(data_type,counts)
plt.title(" Training and Data Testing ")
plt.ylabel("Count")
plt.show()
```



Then we want to find the Maximum, Minimum, and Average number of words in each tweet in tweets



Finally, we want the distribution of frequencies of the number of words in tweets

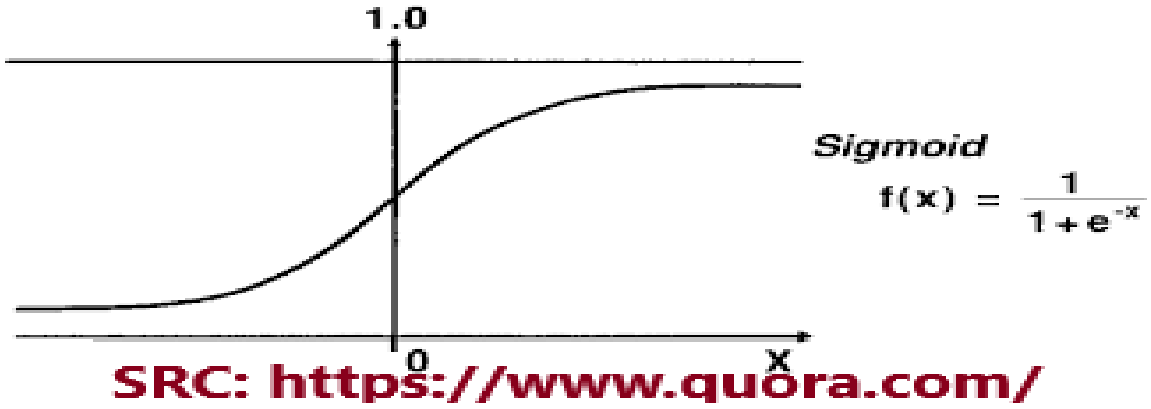


Model Building

We worked on the data through three types of Machine Learning algorithms, **logistic regression, naïve bayes and random forest.**

Logistic Regression

-Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring [1].



We trained logistic regression model on the training data:

```
from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression( regParam=.12)
lrModel = lr.fit(trainingData)

predictions_test_log = lrModel.transform(testData)
predictions_train = lrModel.transform(trainingData)
```

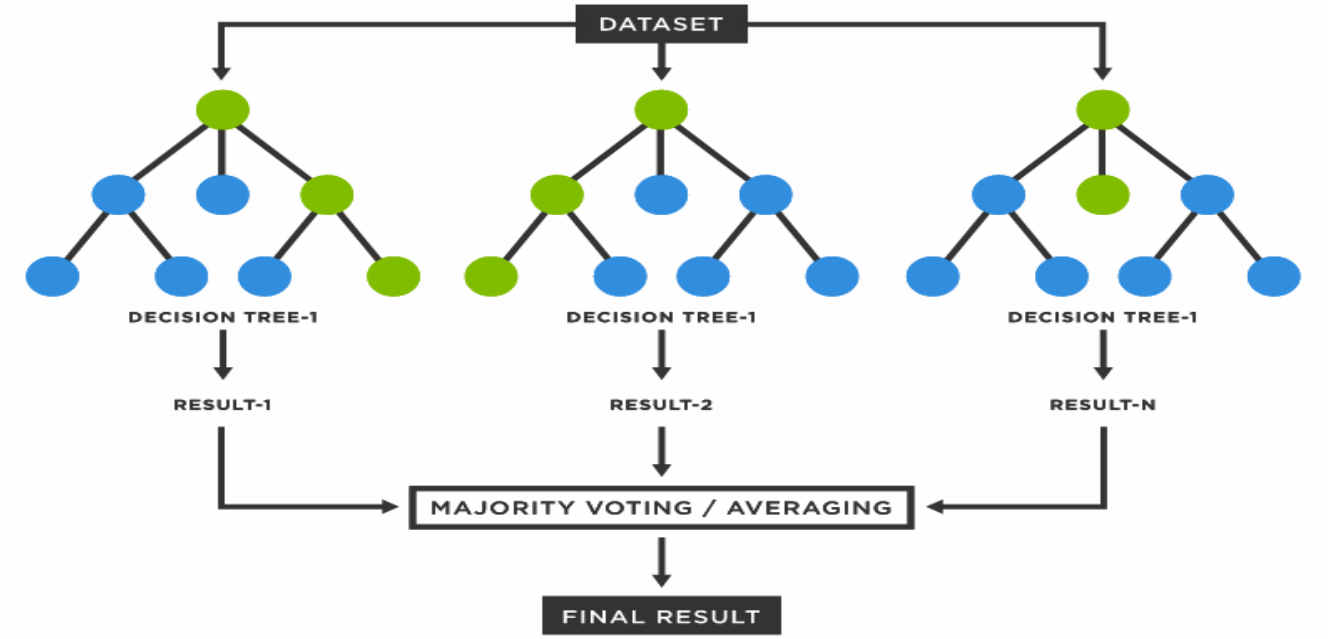
When we used a regularization parameter “0.12,” we got a little bit of good test accuracy, it is about “85%”, overall, it is doing well with this data.

```
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="acc
train_logr_acc = evaluator.evaluate(predictions_train)
test_logr_acc = evaluator.evaluate(predictions_test_log)
print('Training Accuracy on LogisticRegression : {0}'.format(train_logr_acc))
print('Testing Accuracy on LogisticRegression : {0}'.format(test_logr_acc))
```

```
Training Accuracy on LogisticRegression : 0.8684892857424322
Testing Accuracy on LogisticRegression : 0.8447001855170896
```

Random Forest

The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees [2].



SRC: https://www.tibco.com/

We trained random forest classifier model on the training data:

```
from pyspark.ml.classification import RandomForestClassifier

rf = RandomForestClassifier( labelCol="label", seed=42,
    leafCol="label1")
model = rf.fit(trainingData)
```

When we used a parameter of number of seeds “42”, We got a test accuracy about “44%”, overall, it is not suitable for this data.

```
predictions_test = model.transform(testData)
predictions_train = model.transform(trainingData)

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accur
train_ranf_acc = evaluator.evaluate(predictions_train)
test_ranf_acc = evaluator.evaluate(predictions_test)
print('Training Accuracy on RandomForestClassifier : {0}'.format(train_ranf_acc))
print('Testing Accuracy on RandomForestClassifier : {0}'.format(test_ranf_acc))
```

```
Training Accuracy on RandomForestClassifier : 0.4450337702049855
Testing Accuracy on RandomForestClassifier : 0.44134201175989685
```

Naïve Bayes

A Naive Bayes classifier is a probabilistic machine learning model used for classification tasks. The crux of the classifier is based on the Bayes theorem [3].

Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

SRC: https://towardsdatascience.com/

We trained naïve bayes model on the training data:

```
from pyspark.ml.classification import NaiveBayes

nb = NaiveBayes(smoothing=2)
model_nb = nb.fit(trainingData)
predictions_test = model_nb.transform(testData)
predictions_train = model_nb.transform(trainingData)
```

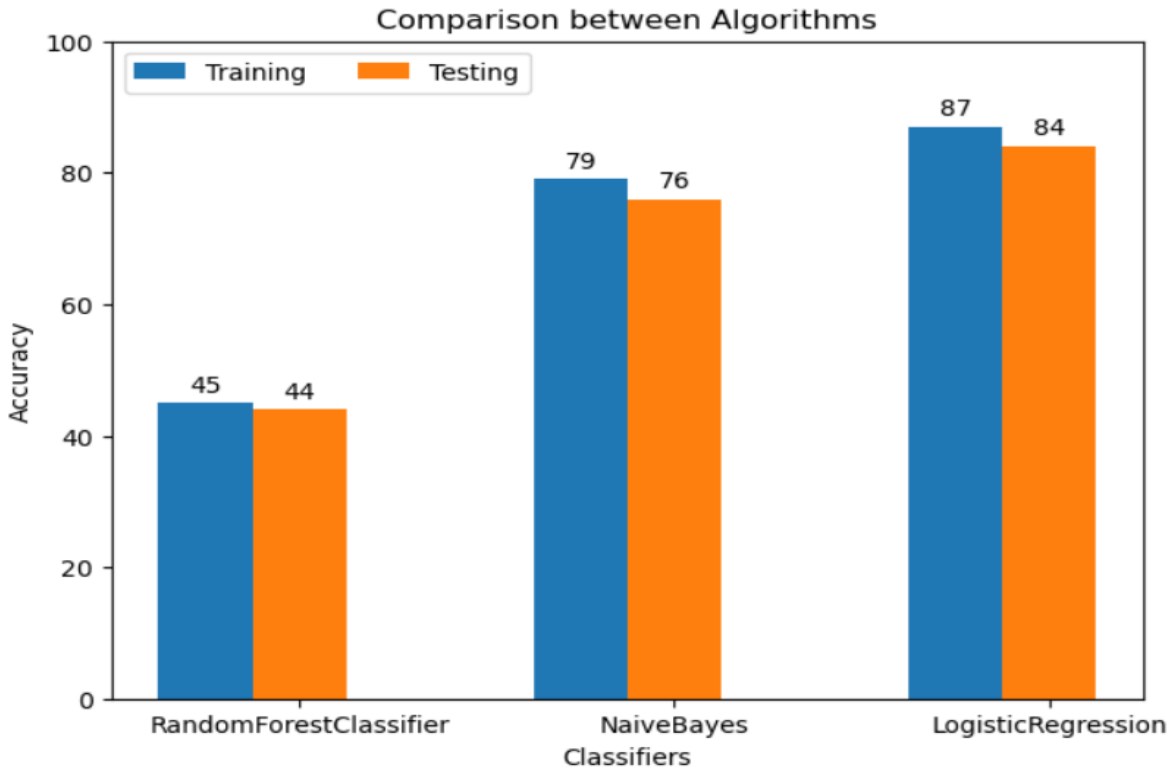
When we used the hyperparameter “smoothing” with the value of “2” after training different values we got a test accuracy that is better than Random Forest, but less than Logistic Regression, it is about “76%”.

```
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
# evaluator_train = MulticlassClassificationEvaluator(predictionCol="prediction")
train_naiveb_acc = evaluator.evaluate(predictions_train)
test_naiveb_acc = evaluator.evaluate(predictions_test)
print('Training Accuracy on NaiveBayes : {0}'.format(train_naiveb_acc))
print('Testing Accuracy on NaiveBayes : {0}'.format(test_naiveb_acc))
```

```
Training Accuracy on NaiveBayes : 0.787597424296983
Testing Accuracy on NaiveBayes : 0.7619933601174785
```

Accepted Algorithm

Finally, we compared the three different results of Logistic Regression, Random Forest, and Naïve Bayes. We got that Logistic Regression fits our data better than the other two algorithms.



As we can see Logistic Regression is doing well with the other classification metrics, such as; precision, recall, and F1-score.

```
eval_accuracy = MulticlassClassificationEvaluator(labelCol="label",
                                                  predictionCol="prediction",
                                                  metricName="accuracy")

eval_precision = MulticlassClassificationEvaluator(labelCol="label",
                                                   predictionCol="prediction",
                                                   metricName="precisionByLabel")

eval_recall = MulticlassClassificationEvaluator(labelCol="label",
                                                predictionCol="prediction",
                                                metricName="recallByLabel")

eval_f1 = MulticlassClassificationEvaluator(labelCol="label",
                                            predictionCol="prediction",
                                            metricName="f1")

# get the evaluation metrics values
accuracy = eval_accuracy.evaluate(predictions_test_log)
precision = eval_precision.evaluate(predictions_test_log)
recall = eval_recall.evaluate(predictions_test_log)
f1score = eval_f1.evaluate(predictions_test_log)
```

```
print(f"the Accuracy of LogisticRegression: {round(accuracy * 100, 2)}%")
print(f"the precision of LogisticRegression: {round(precision* 100, 2)}%")
print(f"the recall of LogisticRegression: {round(recall* 100, 2)}%")
print(f"the f1score of LogisticRegression: {round(f1score* 100, 2)}%")
```

```
the Accuracy of LogisticRegression: 84.47%
the precision of LogisticRegression: 87.29%
the recall of LogisticRegression: 85.75%
the f1score of LogisticRegression: 84.23%
```

Visualization

Visualization is a simple technique for creating a strong mental image of data, visualization helps us to make a feature prediction. As a result, we employ a set of tools to create data visualizations such as SparkSQL, Histogram, Plotly express and use a set of libraries such as Matplotlib.

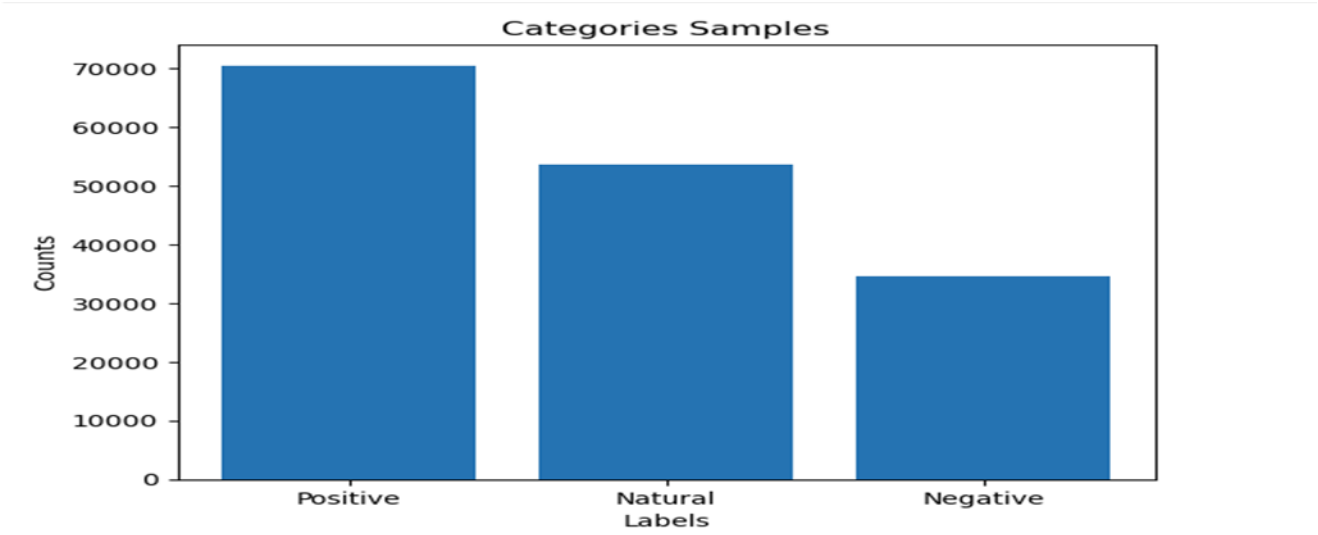


Figure (1): Represent the sentiment types distribution in the data

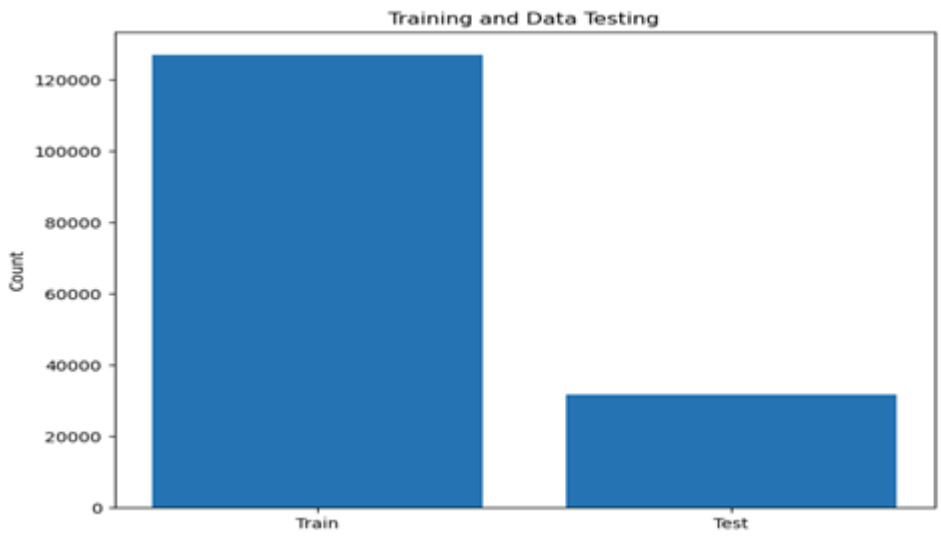


Figure (2): Represent the number of training and testing samples

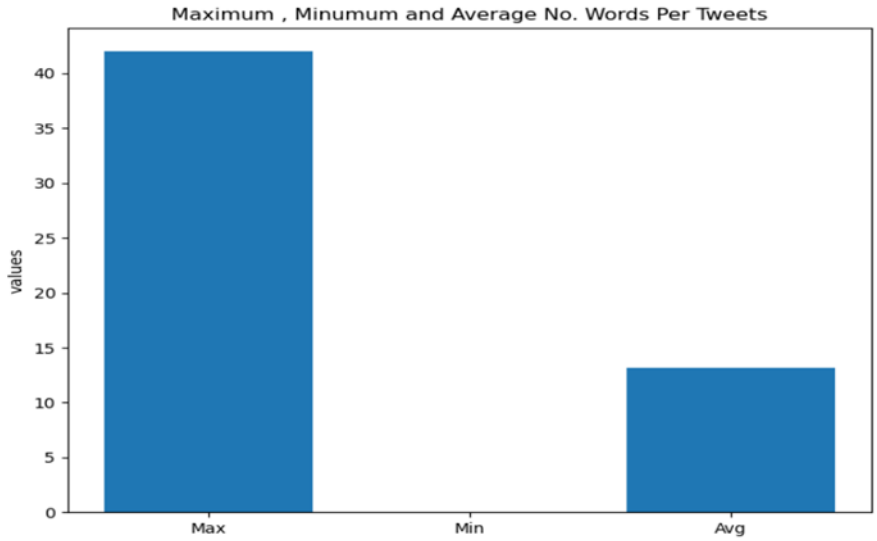


Figure (3): Represent the minimum, average, and maximum of the number of words in tweets

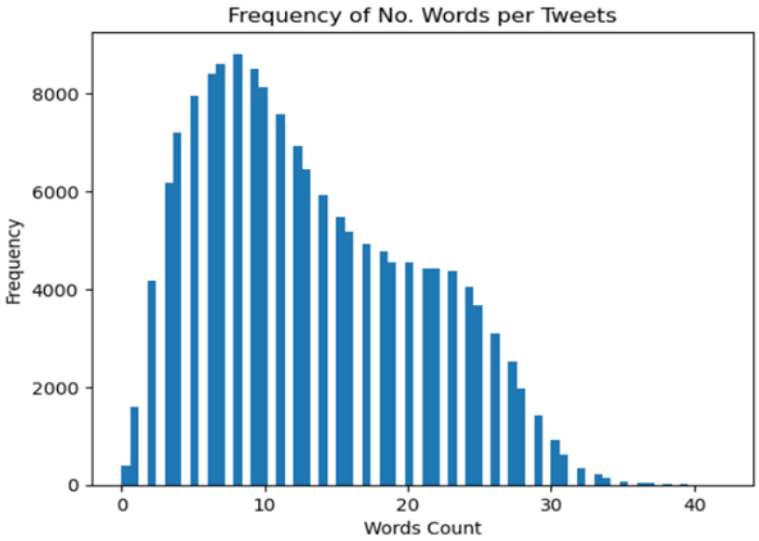


Figure (4): Represent the frequency of the number of words in tweets

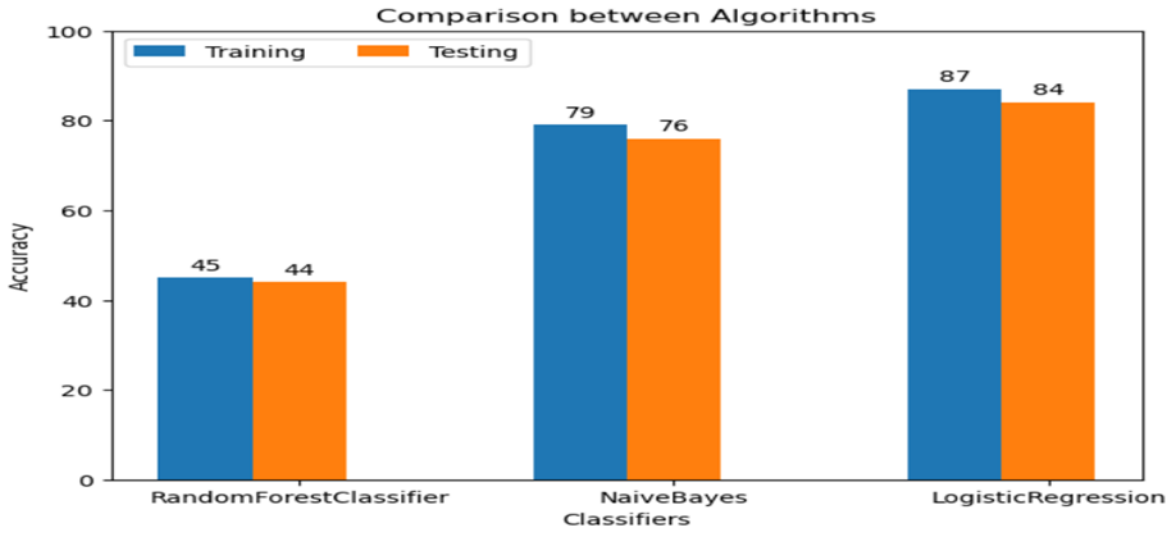


Figure (5): Represent the comparison between the performance of algorithms

Conclusion

Results

The primary objective of the project was to use Spark and other Big Data tools to evaluate social media in order to gather insights. We did this using PySpark and a Kaggle dataset.
When we look at the report's findings, which show that the process had progressed from data collection to preprocessing, cleaning, model building, and finally to the selection of the algorithm that provided us with a high level of accuracy. This algorithm, logistic regression, had an accuracy rate of about 85%, and this was possible because we had carefully chosen, cleaned, prepared the dataset, and adhered to best practices when using the logistic regression algorithm.

Future Work

And lastly, we have a strategy for the project's support in the future, including the implementation of a full system from this seed to a web portal showing human insights and accessibility to integration with other projects' APIs.

Project Tasks Distribution

Tasks		Mostafa Mohamed	Mohamed Salim	Mohamed Elsetohy	Mohammed Abdelghany
Data collection	Data Collection	√	√	√	√
	Data API consuming	√	√	√	√
	Visualizing the consumed data	√	√	√	√
Data Preprocessing	Cleaning Data	√	√	√	√
	Creating Pipeline	√	√	√	√
	Visualizing cleaned data	√	√	√	√
Modeling	Investigating in MLLib of spark	√	√	√	√
	Fitting data with Logistic regression algorithm and enhance	√	√	√	√
	Fitting data with random forest algorithm and enhance	√	√	√	√
	Fitting data with naïve bayes algorithm and enhance	√	√	√	√
	Algorithms performance visualization	√	√	√	√
Spark visualization tools	Investigate in Spark visualization tools	√	√	√	√
Report Writing	Preparing and writing the project report	√	√	√	√

References

[1]: https://projectgurukul.org/

[2]: https://www.kaggle.com/

[3]: https://www.quora.com/

[4]: https://www.tibco.com/

[5]: https://towardsdatascience.com/