Twitter Sentiment Analysis

Hakeem Hinds

Lan Vu

Shawn Yang

hhinds3@students.kennesaw.edu

yyuxuan@students.kennesaw.edu

lvu4@students.kennesaw.edu

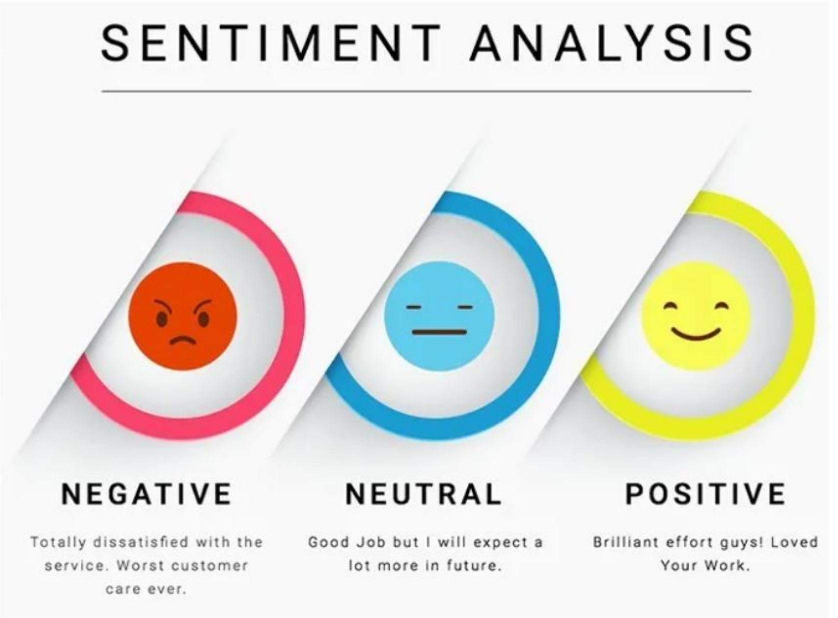
Sentiment Analysis

- Allows us to extract insights from text data

- Determining whether an opinion or text is negative, neutral, or positive

- Neutral text usually have no significance

- Helping understand public opinion, customer feedback, and trends in real-time

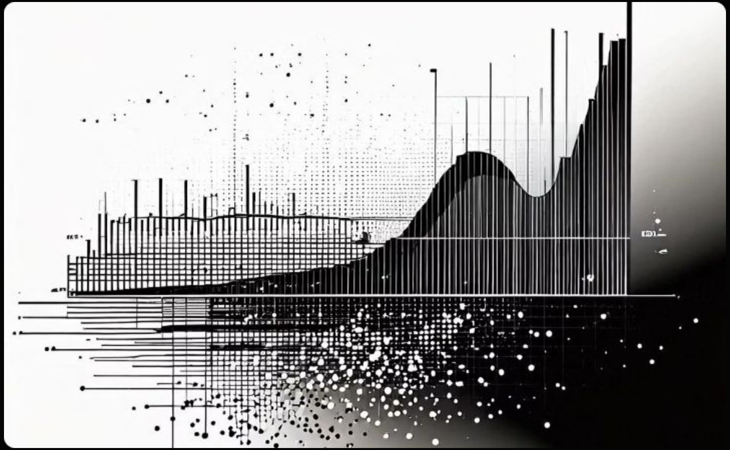


Totally dissatisfied with the service. Worst customer care ever.

Good Job but I will expect a lot more in future.

Brilliant effort guys! Loved Your Work.

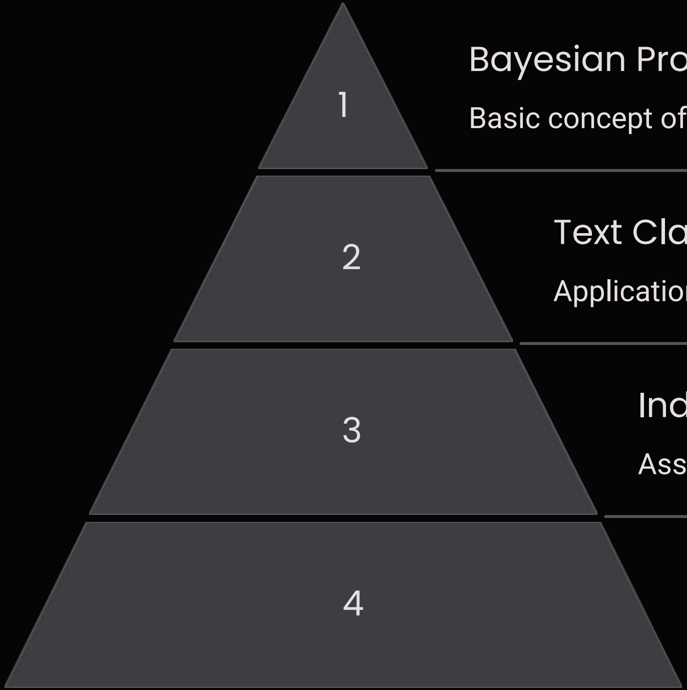
Logistic Regression in Sentiment Analysis



Machine learning algorithms, such as logistic regression, are leveraged to assess sentiment accuracy and make informed decisions.

Logistic regression is applied to classify sentiment in Twitter data, providing statistical insights into public opinions.

Naive Bayes Algorithm in Sentiment Analysis



1

Bayesian Probability

Basic concept of conditional probability

2

Text Classification

Application to categorize sentiment in text data

3

Independence Assumption

Assuming independence of features in the data

4

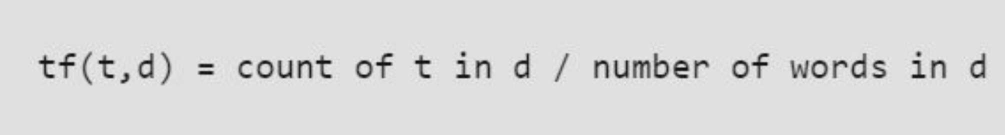
Classification Accuracy

Evaluating accuracy of sentiment classification

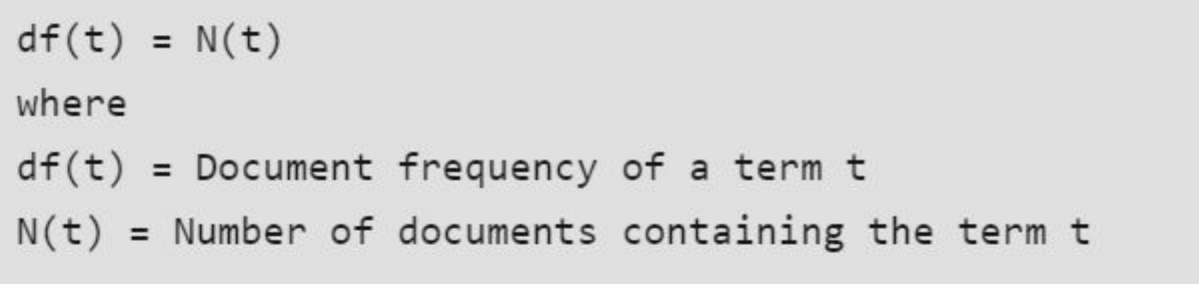
TF-IDF

Term Frequency: Number of times t occurs in a sentence

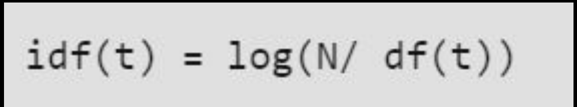
TF-IDF: measures the importance of a word



```
tf(t,d) = count of t in d / number of words in d
```

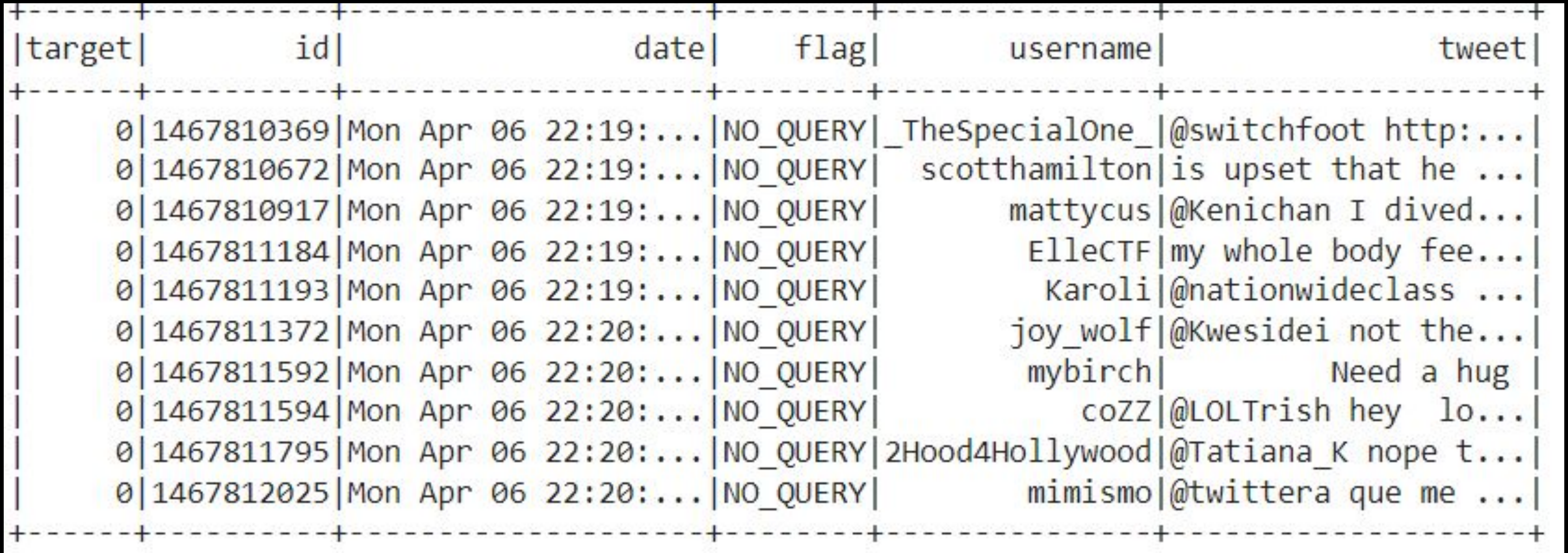


```
df(t) = N(t)
where
df(t) = Document frequency of a term t
N(t) = Number of documents containing the term t
```



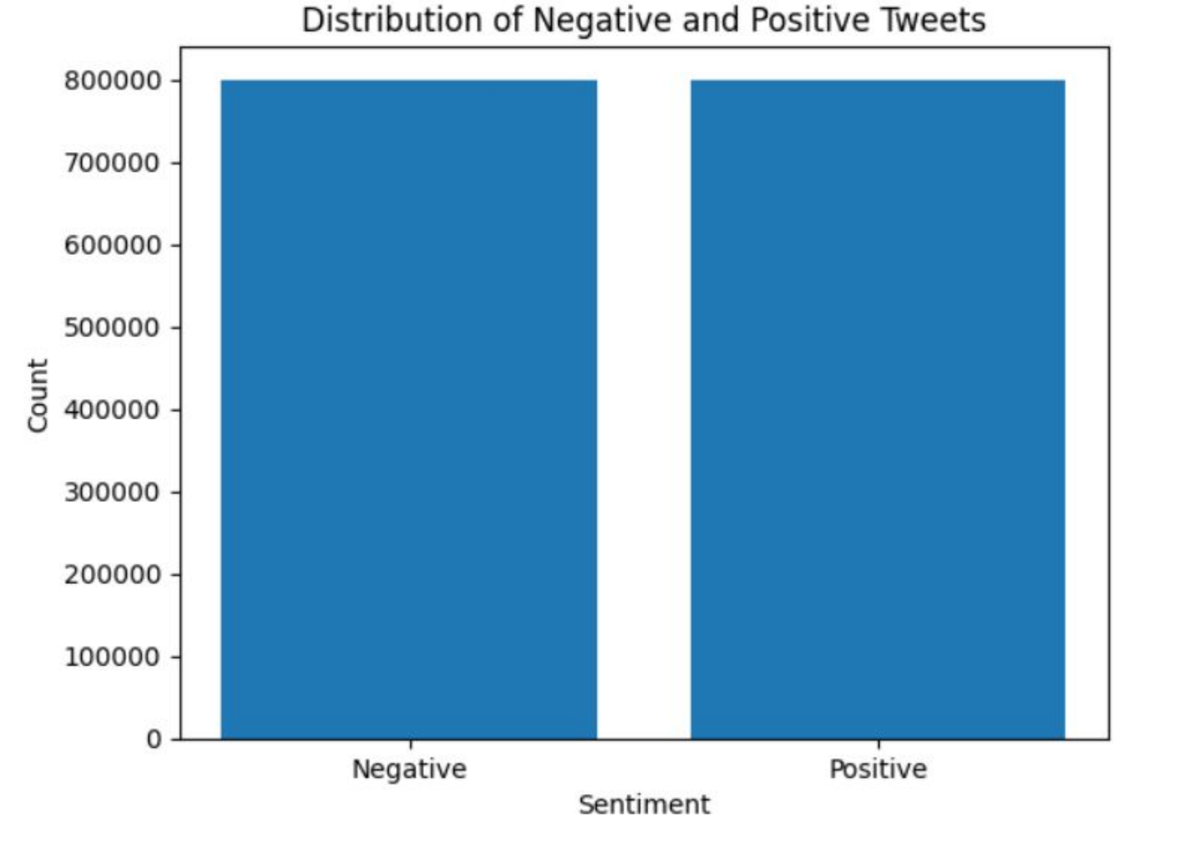
```
idf(t) = log(N/ df(t))
```

Unprocessed Data



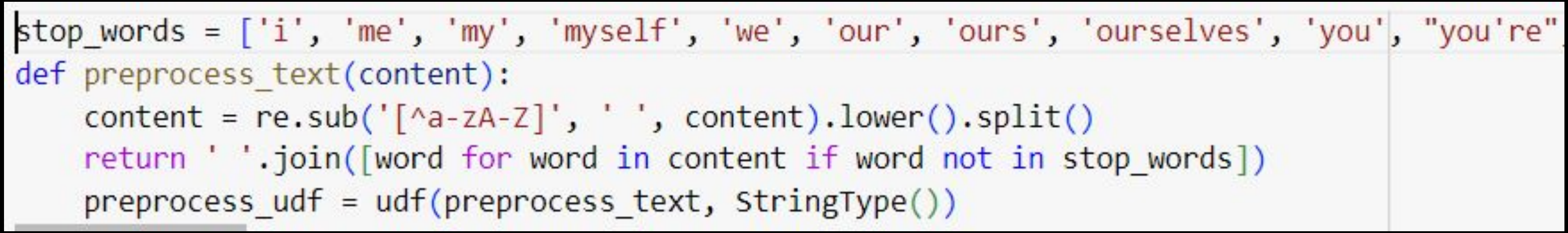
```
+------+----------+--------------------+--------+---------------+--------------------+
|target|        id|                date|    flag|       username|               tweet|
+------+----------+--------------------+--------+---------------+--------------------+
|     0|1467810369|Mon Apr 06 22:19:...|NO_QUERY|_TheSpecialOne_|@switchfoot 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 ...|
+------+----------+--------------------+--------+---------------+--------------------+
```

Bar chart of positive and negative tweets

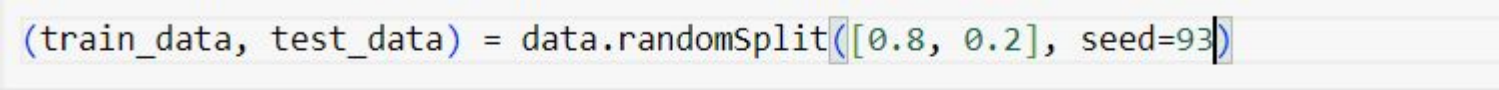


Preprocess/ Data Splitting

- Stemming is the process of eliminating characters, prefixes, suffix resulting in the root word
- Splitting the data into 80% training data and 20% testing data

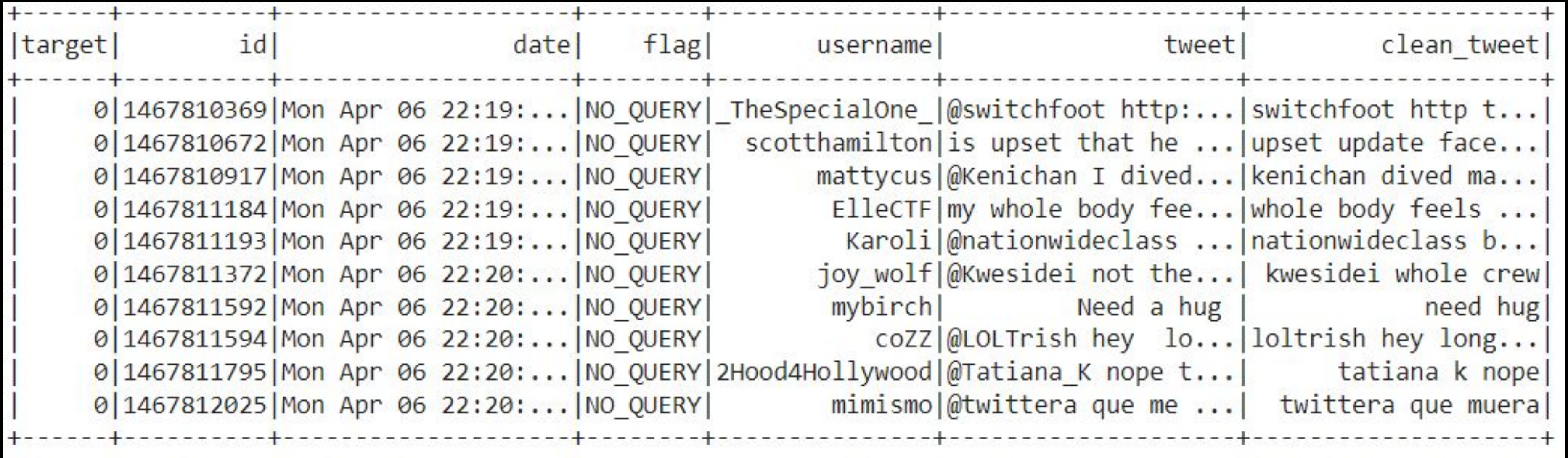


```
stop_words = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"
def preprocess_text(content):
    content = re.sub('[^a-zA-Z]', ' ', content).lower().split()
    return ' '.join([word for word in content if word not in stop_words])
    preprocess_udf = udf(preprocess_text, StringType())
```



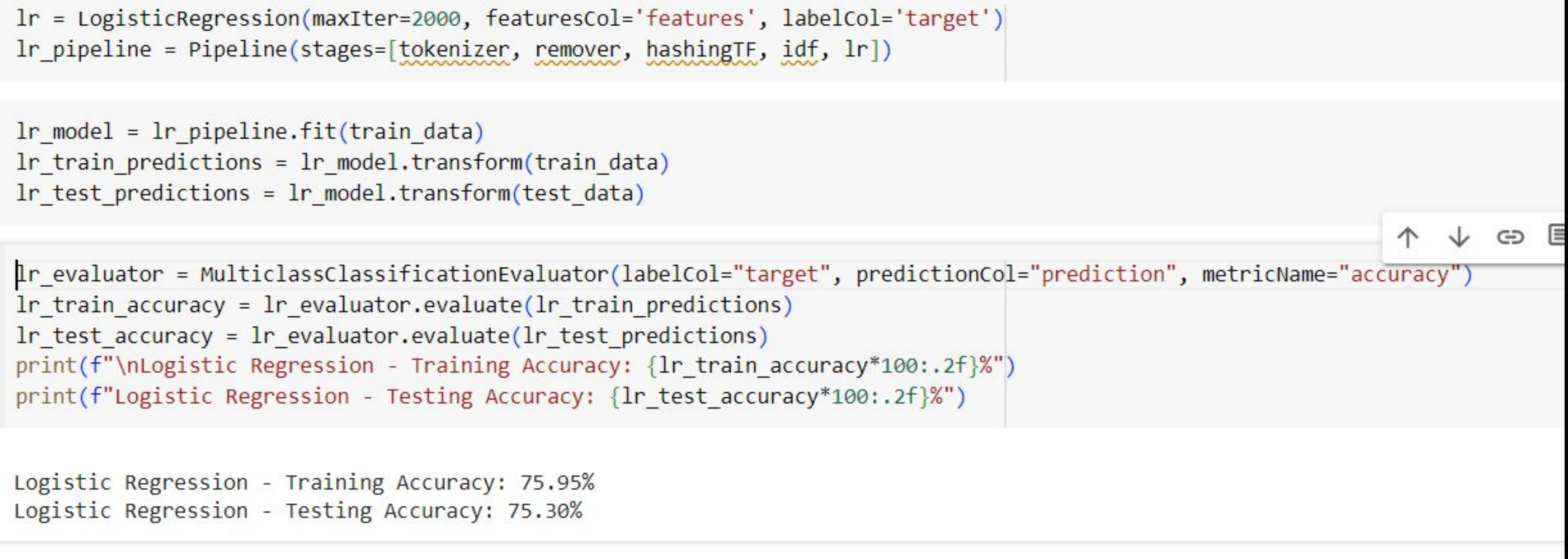
```
(train_data, test_data) = data.randomSplit([0.8, 0.2], seed=93)
```

Image after Stemming



```
+------+----------+--------------------+--------+---------------+--------------------+--------------------+
|target|        id|                date|    flag|       username|               tweet|         clean_tweet|
+------+----------+--------------------+--------+---------------+--------------------+--------------------+
|     0|1467810369|Mon Apr 06 22:19:...|NO_QUERY|_TheSpecialOne_|@switchfoot http:...|switchfoot http t...|
|     0|1467810672|Mon Apr 06 22:19:...|NO_QUERY|  scotthamilton|is upset that he ...|upset update face...|
|     0|1467810917|Mon Apr 06 22:19:...|NO_QUERY|       mattycus|@Kenichan I dived...|kenichan dived ma...|
|     0|1467811184|Mon Apr 06 22:19:...|NO_QUERY|        ElleCTF|my whole body fee...|whole body feels ...|
|     0|1467811193|Mon Apr 06 22:19:...|NO_QUERY|         Karoli|@nationwideclass ...|nationwideclass b...|
|     0|1467811372|Mon Apr 06 22:20:...|NO_QUERY|       joy_wolf|@Kwesidei not the...| kwesidei whole crew|
|     0|1467811592|Mon Apr 06 22:20:...|NO_QUERY|        mybirch|         Need a hug |            need hug|
|     0|1467811594|Mon Apr 06 22:20:...|NO_QUERY|           coZZ|@LOLTrish hey  lo...|loltrish hey long...|
|     0|1467811795|Mon Apr 06 22:20:...|NO_QUERY|2Hood4Hollywood|@Tatiana_K nope t...|      tatiana k nope|
|     0|1467812025|Mon Apr 06 22:20:...|NO_QUERY|        mimismo|@twittera que me ...|  twittera que muera|
+------+----------+--------------------+--------+---------------+--------------------+--------------------+
```

Logistic Regression Model



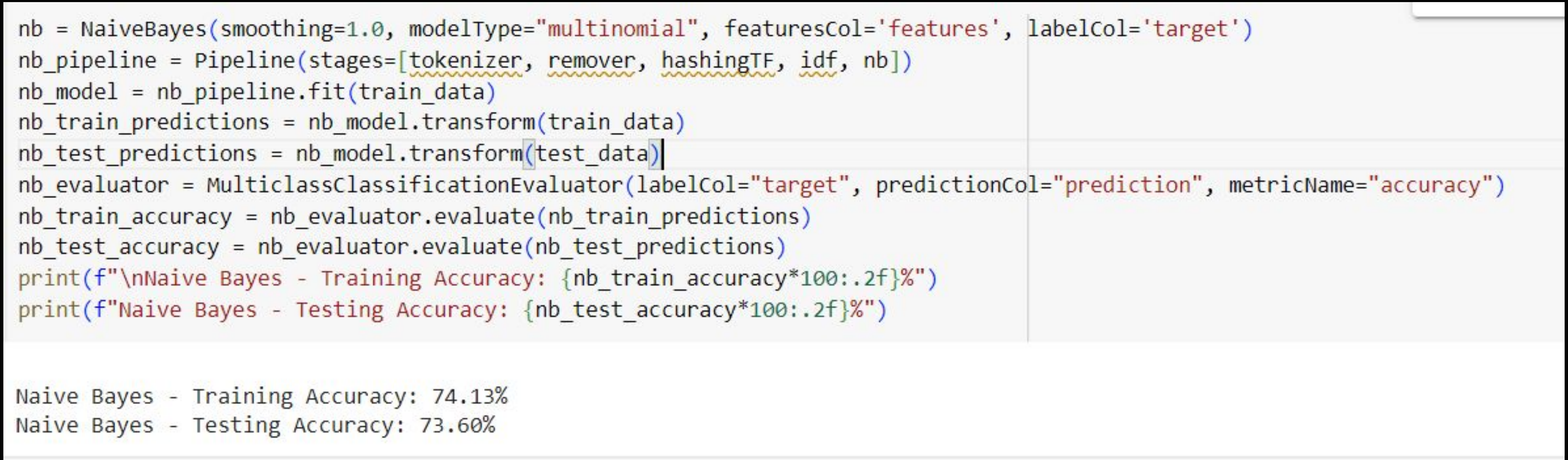
```
lr = LogisticRegression(maxIter=2000, featuresCol='features', labelCol='target')
lr_pipeline = Pipeline(stages=[tokenizer, remover, hashingTF, idf, lr])
```

```
lr_model = lr_pipeline.fit(train_data)
lr_train_predictions = lr_model.transform(train_data)
lr_test_predictions = lr_model.transform(test_data)
```

```
lr_evaluator = MulticlassClassificationEvaluator(labelCol="target", predictionCol="prediction", metricName="accuracy")
lr_train_accuracy = lr_evaluator.evaluate(lr_train_predictions)
lr_test_accuracy = lr_evaluator.evaluate(lr_test_predictions)
print(f"\nLogistic Regression - Training Accuracy: {lr_train_accuracy*100:.2f}%")
print(f"Logistic Regression - Testing Accuracy: {lr_test_accuracy*100:.2f}%")
```

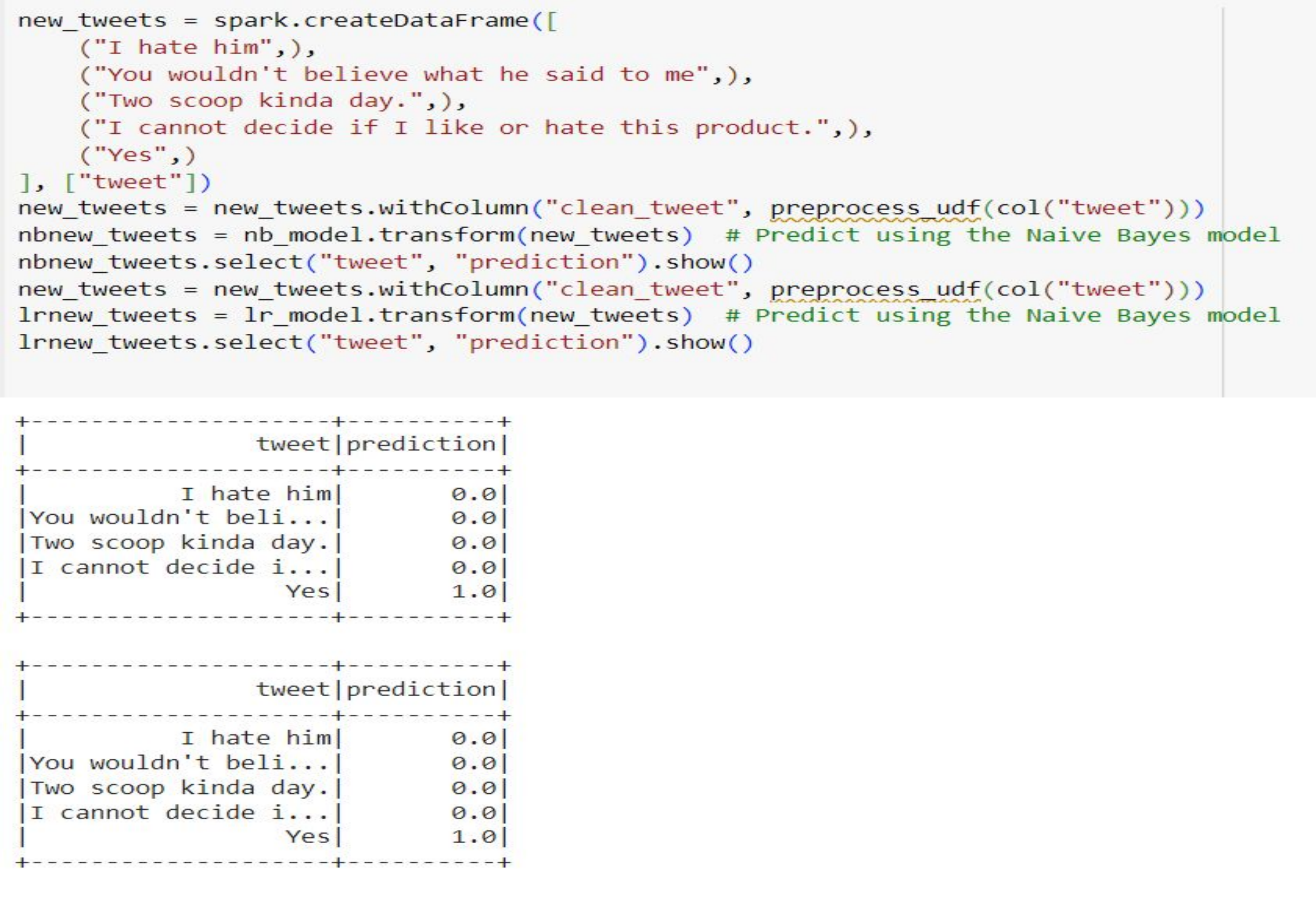
```

Logistic Regression - Training Accuracy: 75.95%
Logistic Regression - Testing Accuracy: 75.30%
```

Naive Bayes Multinomial Model



```
nb = NaiveBayes(smoothing=1.0, modelType="multinomial", featuresCol='features', labelCol='target')
nb_pipeline = Pipeline(stages=[tokenizer, remover, hashingTF, idf, nb])
nb_model = nb_pipeline.fit(train_data)
nb_train_predictions = nb_model.transform(train_data)
nb_test_predictions = nb_model.transform(test_data)
nb_evaluator = MulticlassClassificationEvaluator(labelCol="target", predictionCol="prediction", metricName="accuracy")
nb_train_accuracy = nb_evaluator.evaluate(nb_train_predictions)
nb_test_accuracy = nb_evaluator.evaluate(nb_test_predictions)
print(f"\nNaive Bayes - Training Accuracy: {nb_train_accuracy*100:.2f}%")
print(f"Naive Bayes - Testing Accuracy: {nb_test_accuracy*100:.2f}%")
```

```
Naive Bayes - Training Accuracy: 74.13%
Naive Bayes - Testing Accuracy: 73.60%
```



```
new_tweets = spark.createDataFrame([
    ("I hate him",),
    ("You wouldn't believe what he said to me",),
    ("Two scoop kinda day.",),
    ("I cannot decide if I like or hate this product.",),
    ("Yes",)
], ["tweet"])
new_tweets = new_tweets.withColumn("clean_tweet", preprocess_udf(col("tweet")))
nbnew_tweets = nb_model.transform(new_tweets)  # Predict using the Naive Bayes model
nbnew_tweets.select("tweet", "prediction").show()
new_tweets = new_tweets.withColumn("clean_tweet", preprocess_udf(col("tweet")))
lrnew_tweets = lr_model.transform(new_tweets)  # Predict using the Naive Bayes model
lrnew_tweets.select("tweet", "prediction").show()
```

```
+--------------------+----------+
|               tweet|prediction|
+--------------------+----------+
|          I hate him|       0.0|
|You wouldn't beli...|       0.0|
|Two scoop kinda day.|       0.0|
|I cannot decide i...|       0.0|
|                 Yes|       1.0|
+--------------------+----------+

+--------------------+----------+
|               tweet|prediction|
+--------------------+----------+
|          I hate him|       0.0|
|You wouldn't beli...|       0.0|
|Two scoop kinda day.|       0.0|
|I cannot decide i...|       0.0|
|                 Yes|       1.0|
+--------------------+----------+
```

Sources

https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/