SENTIMENT ANALYSIS THROUGH SPARK

Prepared by:

Anupam Misra
C21001
PGP in Data Science
Praxis Business School

CONTENTS

BUSINESS PROBLEM

1. Background
2. About the project
3. Stakeholders
4. The case for Spark

Background

Sentiment analysis is a branch of Natural Language Processing. It gauges an incoming stream of text to interpret whether the sentiment about the context is positive, negative or neutral. It's estimated that 90% of the world's data is unstructured, in other words it's unorganized. Huge volumes of unstructured business data are created every day: emails, support tickets, chats, social media conversations, surveys, articles, documents, etc). But it's hard to analyze for sentiment in a timely and efficient manner.

Sentiment analysis is extremely important because it allows businesses to understand the sentiment of their customers towards their business and products. By automatically sorting the sentiment behind social media conversations, reviews, and more, businesses can make better and more informed decisions. They can also tweak their products to make better offerings.

About the project

In this project, we propose building an application to interpret the customer sentiments about the business and products from the product reviews real-time. The application will interpret whether the sentiment polarity in a review is positive or negative. This intelligence can be used to adjust the price, visibility, and variety of offerings provided on the digital shelves.

Stakeholders

All e-commerce companies which collect product reviews will benefit from the implementation of review sentiment analysis. This will enable them to remain competitive by providing better offerings and making better business decisions. Better product offerings will boost sales and improve revenue. Trimming poor products and bad sellers will improve operational efficiency and improve business profitability. By understanding what the customers are thinking, the company will also be in a better position to frame long-term goals to grow sustainably and widen the competitive advantage.

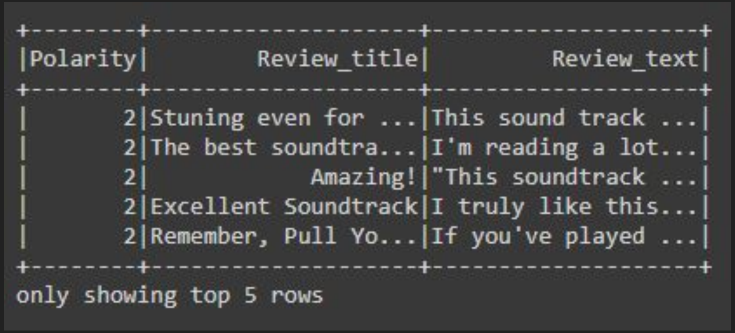
The case for Spark

We want to build a real-time application to track customer sentiments. The volume and velocity of incoming data from reviews for an established e-commerce company cannot be handled by traditional applications backed by machine learning.

Spark is suited for this application because the execution can be distributed across a cluster of commodity machines.

DATA DESCRIPTION

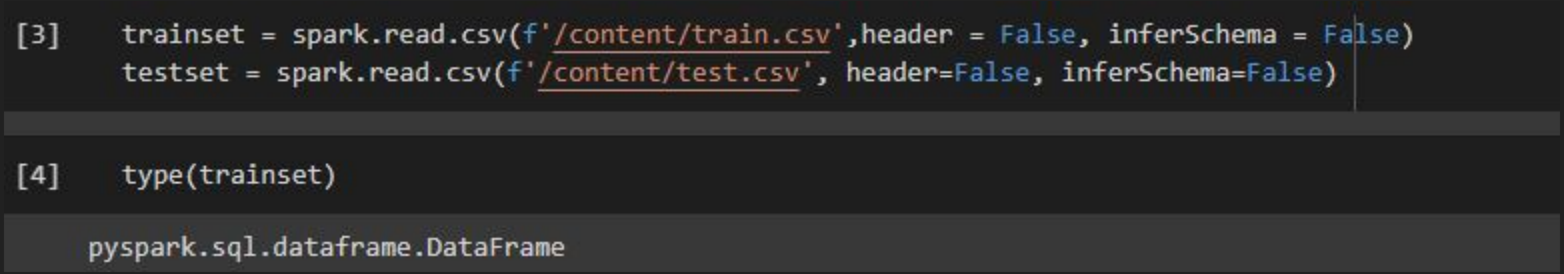
- The dataset is the Amazon Review Polarity Dataset taken from Kaggle.
- It contains 18,00,000 training samples and 2,00,000 testing samples against positive and negative polarity.
- Polarity: 1 = negative; 2 = positive



```
+--------+--------------------+--------------------+
|Polarity|        Review_title|         Review_text|
+--------+--------------------+--------------------+
|       2|Stuning even for ...|This sound track ...|
|       2|The best soundtra...|I'm reading a lot...|
|       2|            Amazing!|"This soundtrack ...|
|       2|Excellent Soundtrack|I truly like this...|
|       2|Remember, Pull Yo...|If you've played ...|
+--------+--------------------+--------------------+
only showing top 5 rows
```

Data loading

- The data was loaded from google drive into the colab environment
- The train and test files were read from the colab machine directory into Spark dataframes



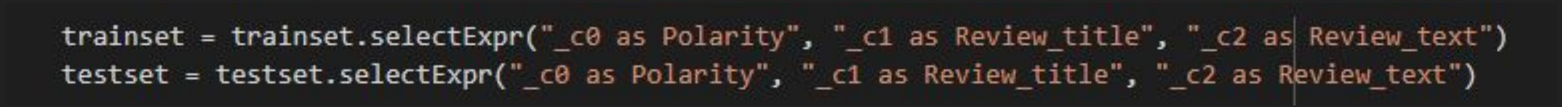
```
[3]   trainset = spark.read.csv(f'/content/train.csv',header = False, inferSchema = False)
      testset = spark.read.csv(f'/content/test.csv', header=False, inferSchema=False)

[4]   type(trainset)

    pyspark.sql.dataframe.DataFrame
```

DATA PREPROCESSING

- Some rows containing missing information were dropped.

	Training set	**Testing set**
Original no. of rows	36,00,000	4,00,000
No. of rows dropped	61	5
Reduced no. of rows	35,99,939	3,99,995

- The columns were renamed in the Spark dataframe



```
trainset = trainset.selectExpr("_c0 as Polarity", "_c1 as Review_title", "_c2 as Review_text")
testset = testset.selectExpr("_c0 as Polarity", "_c1 as Review_title", "_c2 as Review_text")
```

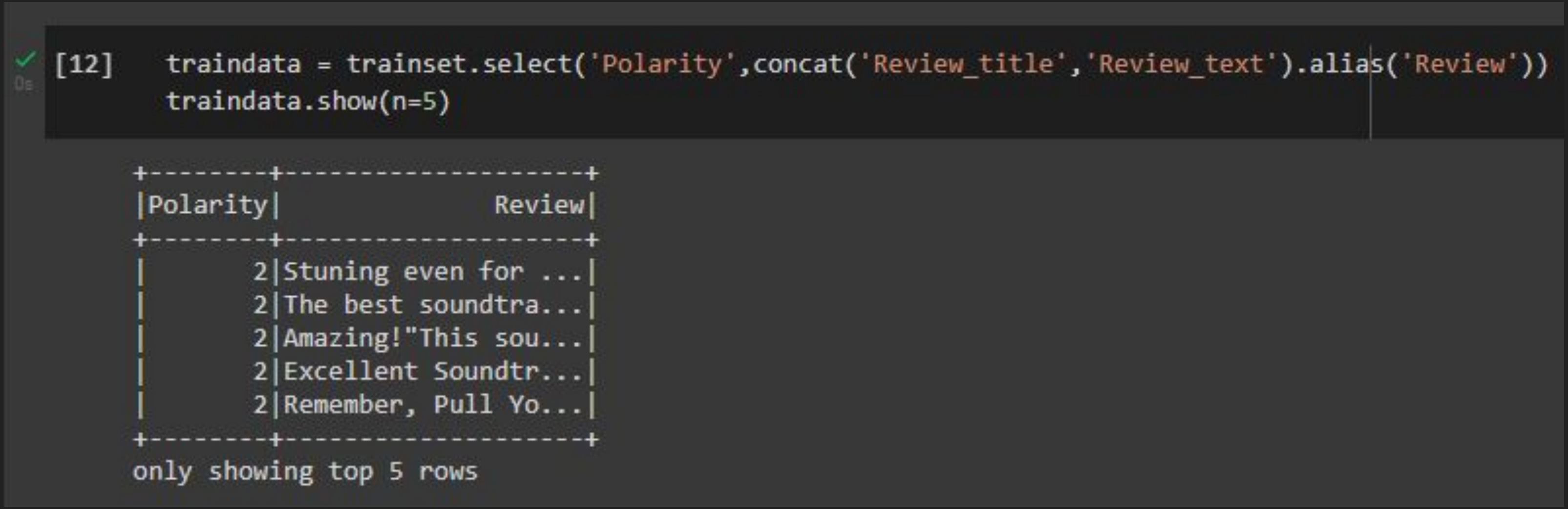
TEXT PREPROCESSING

- Concating review title and text columns to obtain single review column
- Removing numeric characters using `regexp_replace`
- Splitting each review into a list of words using `RegexTokenizer`
- Removing stop words using `StopWordsRemover`
- Generating feature matrix using `CountVectorizer`
- Casting the **label** to Integer datatype and binarizing it by subtracting 1

NOTE:

`CountVectorizer` was fit on the training dataset and used to transform the data. This prevented data leakage.

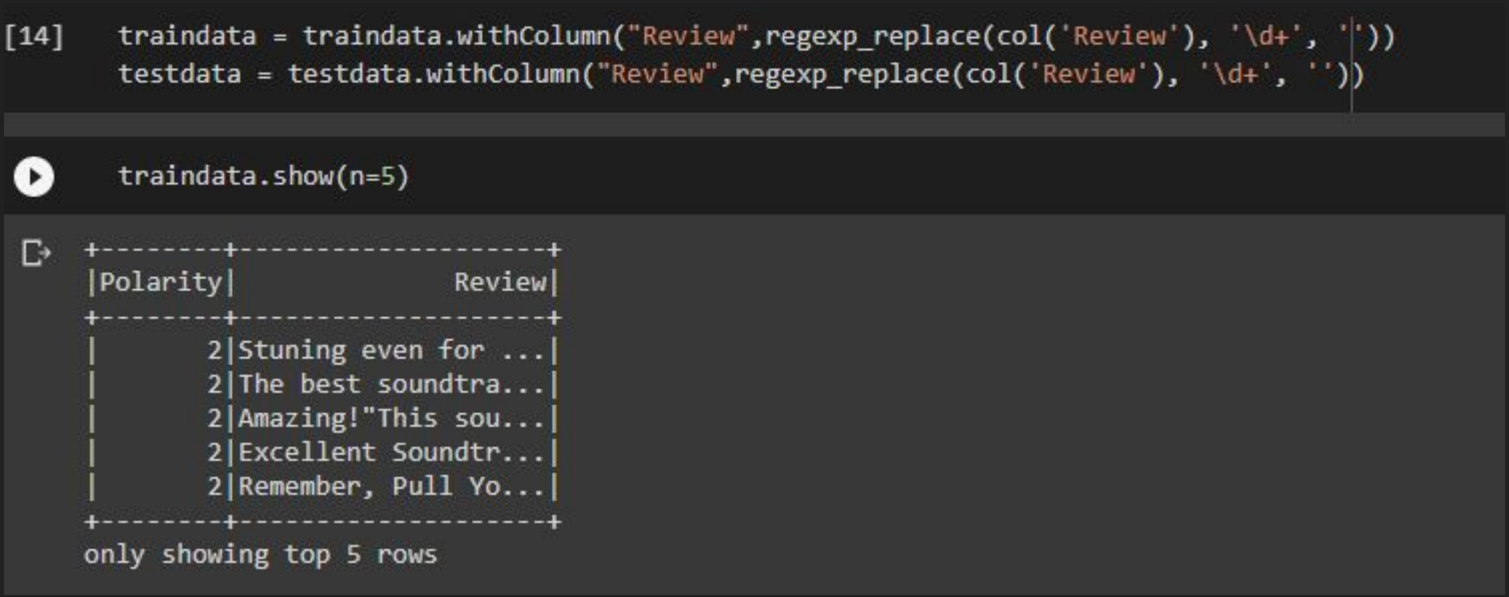
- Concating review title and text columns to obtain single review column



```
[12]  traindata = trainset.select('Polarity',concat('Review_title','Review_text').alias('Review'))
      traindata.show(n=5)
```

```
+--------+--------------------+
|Polarity|              Review|
+--------+--------------------+
|       2|Stuning even for ...|
|       2|The best soundtra...|
|       2|Amazing!"This sou...|
|       2|Excellent Soundtr...|
|       2|Remember, Pull Yo...|
+--------+--------------------+
only showing top 5 rows
```

- Removing numeric characters

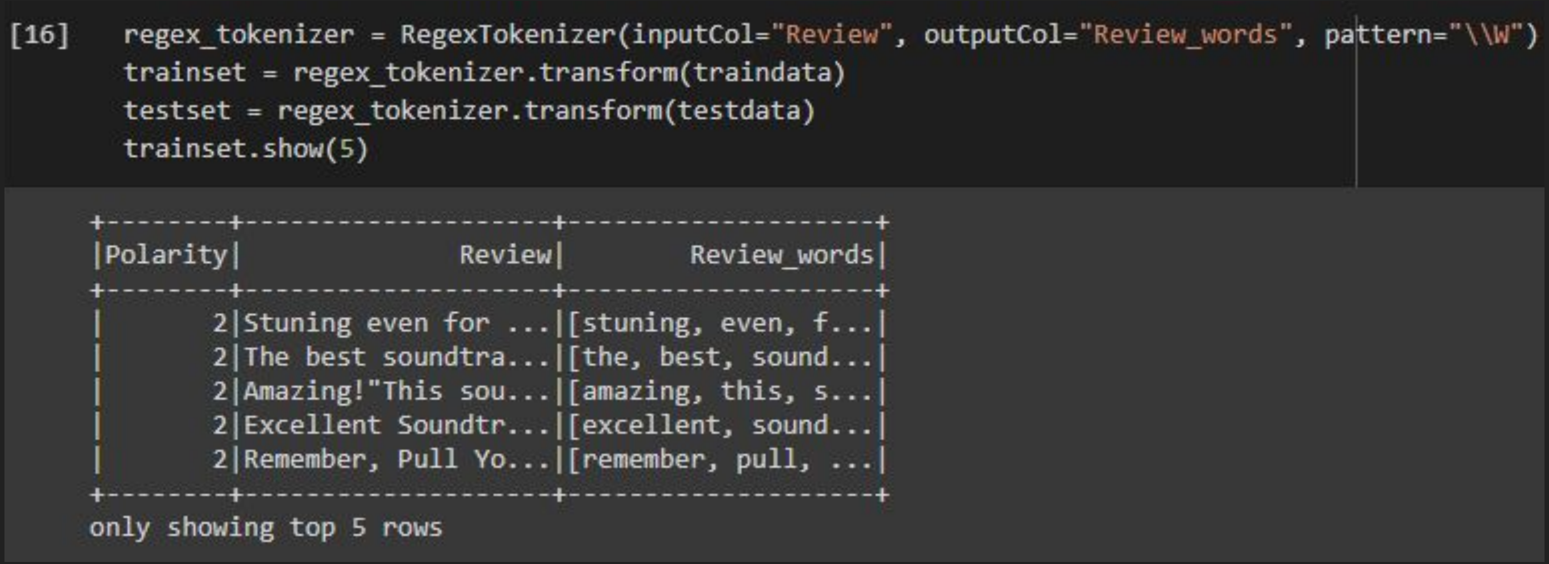


```
[14]  traindata = traindata.withColumn("Review",regexp_replace(col('Review'), '\d+', ''))
      testdata = testdata.withColumn("Review",regexp_replace(col('Review'), '\d+', ''))
```

```
traindata.show(n=5)
```

```
+--------+--------------------+
|Polarity|              Review|
+--------+--------------------+
|       2|Stuning even for ...|
|       2|The best soundtra...|
|       2|Amazing!"This sou...|
|       2|Excellent Soundtr...|
|       2|Remember, Pull Yo...|
+--------+--------------------+
only showing top 5 rows
```

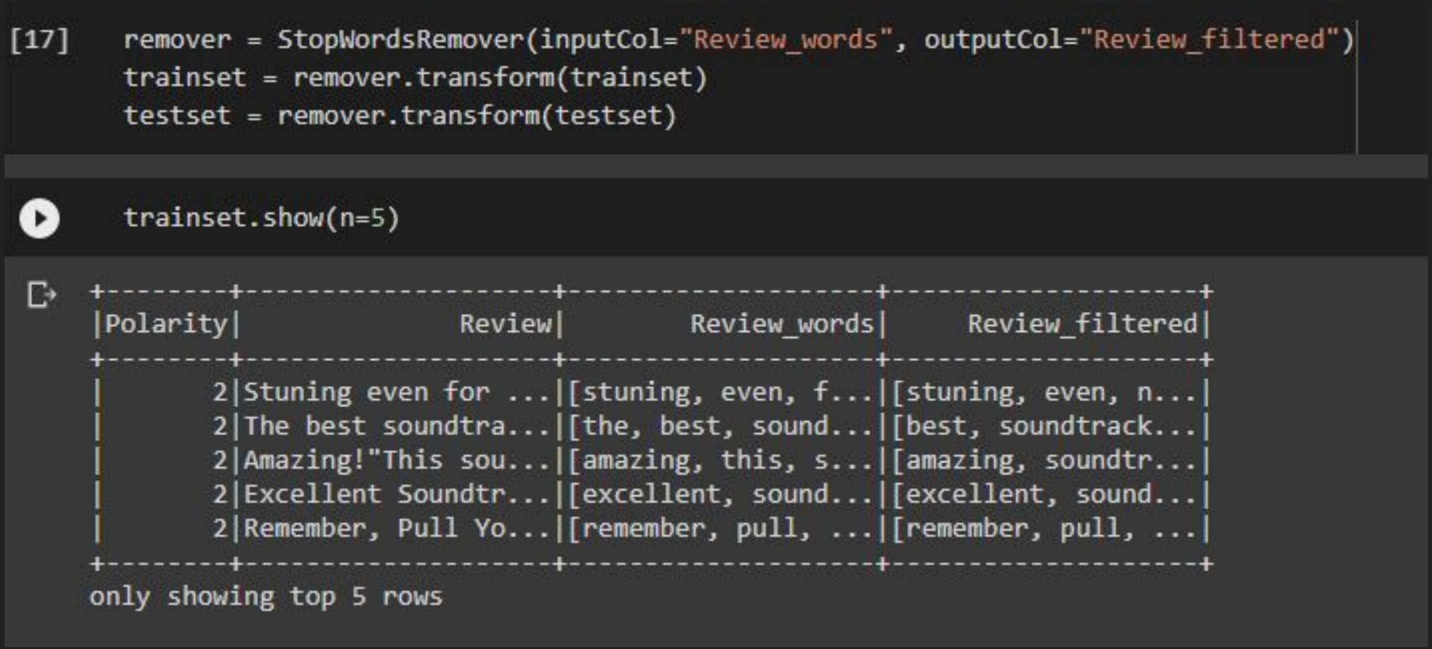
- Splitting each review into a list of words using `RegexTokenizer`



```
[16]  regex_tokenizer = RegexTokenizer(inputCol="Review", outputCol="Review_words", pattern="\\W")
      trainset = regex_tokenizer.transform(traindata)
      testset = regex_tokenizer.transform(testdata)
      trainset.show(5)
```

```
+--------+--------------------+--------------------+
|Polarity|              Review|        Review_words|
+--------+--------------------+--------------------+
|       2|Stuning even for ...|[stuning, even, f...|
|       2|The best soundtra...|[the, best, sound...|
|       2|Amazing!"This sou...|[amazing, this, s...|
|       2|Excellent Soundtr...|[excellent, sound...|
|       2|Remember, Pull Yo...|[remember, pull, ...|
+--------+--------------------+--------------------+
only showing top 5 rows
```

- Removing stop words using `StopWordsRemover`

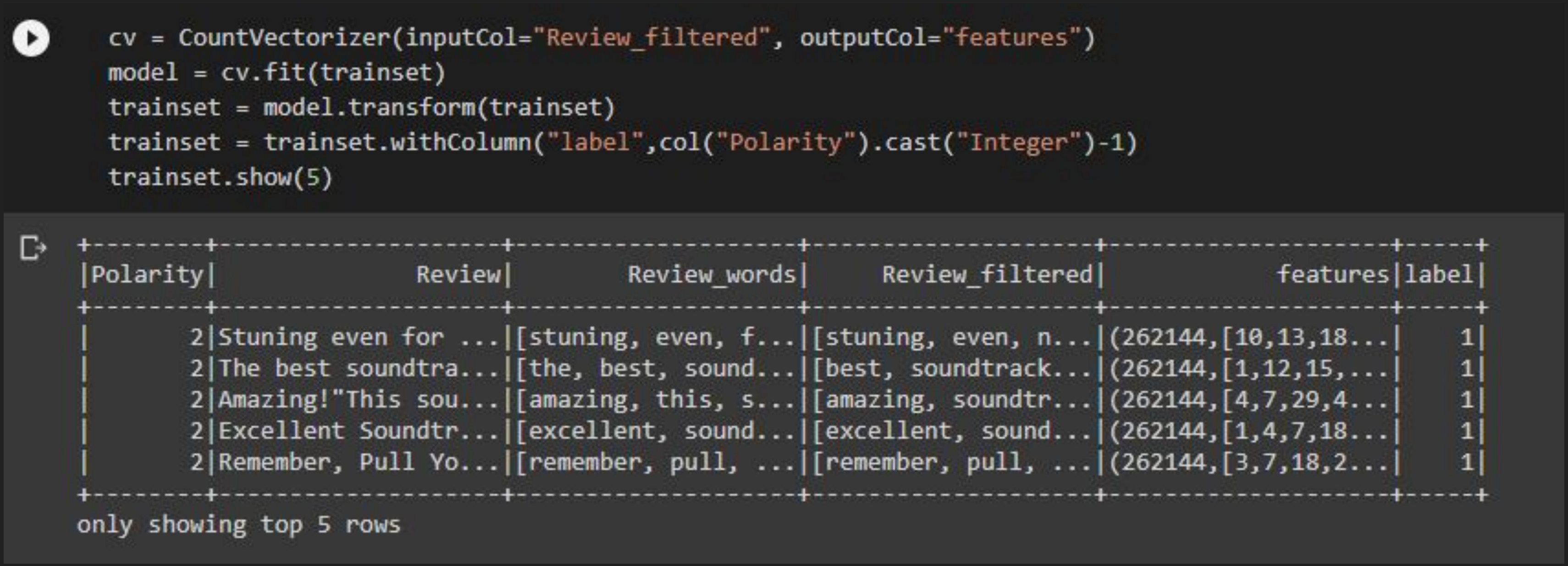


```
[17]  remover = StopWordsRemover(inputCol="Review_words", outputCol="Review_filtered")
      trainset = remover.transform(trainset)
      testset = remover.transform(testset)
```

```
trainset.show(n=5)
```

```
+--------+--------------------+--------------------+--------------------+
|Polarity|              Review|        Review_words|     Review_filtered|
+--------+--------------------+--------------------+--------------------+
|       2|Stuning even for ...|[stuning, even, f...|[stuning, even, n...|
|       2|The best soundtra...|[the, best, sound...|[best, soundtrack...|
|       2|Amazing!"This sou...|[amazing, this, s...|[amazing, soundtr...|
|       2|Excellent Soundtr...|[excellent, sound...|[excellent, sound...|
|       2|Remember, Pull Yo...|[remember, pull, ...|[remember, pull, ...|
+--------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

- Generating features using `CountVectorizer`
- Casting the label to Integer datatype and binarizing it by subtracting 1



```
cv = CountVectorizer(inputCol="Review_filtered", outputCol="features")
model = cv.fit(trainset)
trainset = model.transform(trainset)
trainset = trainset.withColumn("label",col("Polarity").cast("Integer")-1)
trainset.show(5)
```

```
+--------+--------------------+--------------------+--------------------+--------------------+-----+
|Polarity|              Review|        Review_words|     Review_filtered|            features|label|
+--------+--------------------+--------------------+--------------------+--------------------+-----+
|       2|Stuning even for ...|[stuning, even, f...|[stuning, even, n...|(262144,[10,13,18...|    1|
|       2|The best soundtra...|[the, best, sound...|[best, soundtrack...|(262144,[1,12,15,...|    1|
|       2|Amazing!"This sou...|[amazing, this, s...|[amazing, soundtr...|(262144,[4,7,29,4...|    1|
|       2|Excellent Soundtr...|[excellent, sound...|[excellent, sound...|(262144,[1,4,7,18...|    1|
|       2|Remember, Pull Yo...|[remember, pull, ...|[remember, pull, ...|(262144,[3,7,18,2...|    1|
+--------+--------------------+--------------------+--------------------+--------------------+-----+
only showing top 5 rows
```

MACHINE LEARNING

	No. of observations	**No. of features**
Training data	35,99,939	2,62,144
Testing data	3,99,995	

The training and testing feature matrices are sparse matrices. Tree models perform poorly with sparse matrices. Generally Naive Bayes perform well with textual feature matrices. Statistical models like Logistic Regression also perform well when there are a large number of features and if the classes are linearly separable.

I fitted the Logistic Regression and Naive Bayes models on the training data and tested it on the test data.

EVALUATION

The results of the models on the test data:

Model	**AUC_ROC**	**Accuracy**
`NaiveBayes`	0.54	84.63%
`LogisticRegression`	0.97	91.67%

AUC_ROC (Area under the curve; Receiver Operating Characteristics):

This metric shows the discriminatory power of the classifier across various thresholds. A value of 0.5 means no discriminatory power. A value of 1 implies perfect discriminatory power at all thresholds.

Accuracy :

As our test set was balanced, we could evaluate the model with a balanced classification metric.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

CONCLUSION

We can see that Logistic regression has very good performance on unseen data. This implies two things:

1. The classes are linearly separable.
2. There are a lot of words(features) which are absent in the test set and have to be Laplace smoothened in the Naive Bayes model.

We can deploy an application backed by this Logistic Regression classifier to classify sentiments of incoming reviews real time.

REFERENCES

- Suraj Malpani: *Natural Language Processing with Spark*
- Shashank Gupta: Sentiment Analysis: *Concept, Analysis and Applications*
- *Sentiment Analysis, A Definitive Guide*

Thank you