BIA 678 – C: BIG DATA TECHNOLOGIES

PROJECT REPORT ON

AMAZON FOOD REVIEWS

UNDER THE GUIDANCE OF

LATE. PROF DAVID BELANGER

SUBMITTED BY:

KULDEEPSINH VAGHELA

SAMAN FATIMA

MITRA MORE

RITIKA DAVE

Table of Contents

INTRODUCTION

A massive amount of data is generated every second on the internet. Big Data is characterized as the data sources that are too vast and complex for typical relational databases that gather, maintain, and process with minimal latency. The Data sources are becoming more complicated than conventional data sources because of the existence of AI (Artificial Intelligence), Mobiles, Social Media platforms, etc. There is a ton of volume, speed, and variety in big data. Considering open-source technologies like Apache Spark for building big data solutions proves to be cost-effective, helpful to society and provides flexible computational and storage tools.

In this project, we include up to 5,00,000 food reviews. These reviews include product information, User information, ratings, and reviews from all other categories. This will help the User in a way as to they can make informed decisions before ordering something by reading the reviews and eliminate the potential risks involved. *The aim of our project is to classify the reviews given by the customers as positive or negative*

DATASET

The dataset is collected from the Kaggle Website. It consists of reviews of fine food from Amazon. The data span over 10 years, including all 500,000 reviews up to October 2012. Reviews include product and user information, ratings and a plaintext review.

The Amazon Fine Food Review dataset contains:

1. Number of Reviews: 568,454
2. Number of Users: 256,059

3. Number of Products: 74,258
4. Timespan: October 1999 – October 2012
5. Number of Attributes/Columns in Data: 10

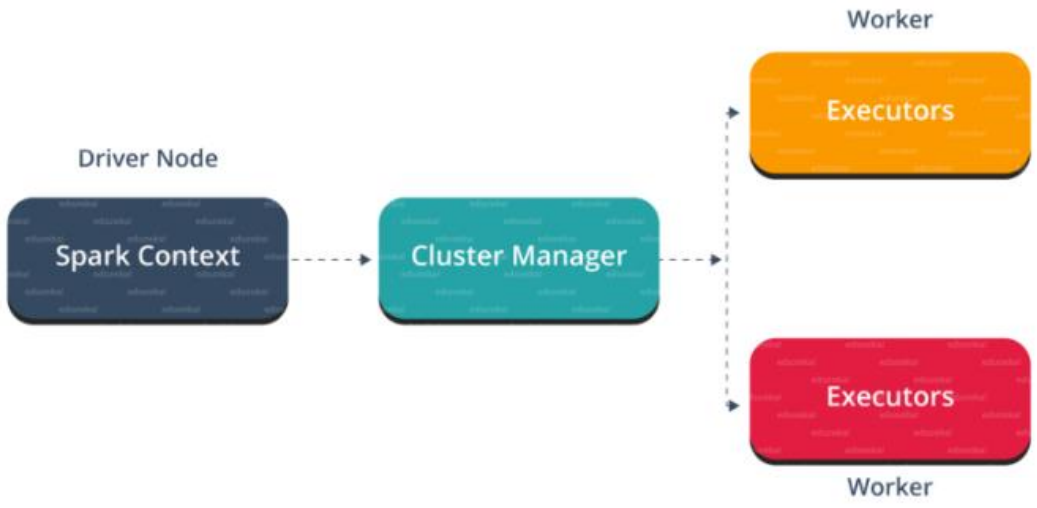
Attribute Information:

1. Id
2. ProductId
3. UserId
4. ProfileName
5. HelpfulnessNumerator – number of users who found the review helpful
6. HelpfulnessDenominator – number of users who indicated whether they found the review helpful or not
7. Score – rating between 1 and 5
8. Time – timestamp for the review
9. Summary – Brief summary of the review
10. Text – text of the review

SOFTWARE & TECHNOLOGIES USED

Apache Spark is a distributed processing system used to perform big data and machine learning tasks on large datasets. Distributed Processing is a setup in which multiple processors are helpful in running the application. The tasks are divided between multiple devices that communicate with each other.

With the help of Apache, we can do more like running big queries and machine learning workflows on petabytes of data. The Apache Spark is known to be fastest and powerful framework that offers an API to perform massive, distributed processing over sets of data. The ability to analyze data and train machine learning models on a large-scale database is a crucial task to do.



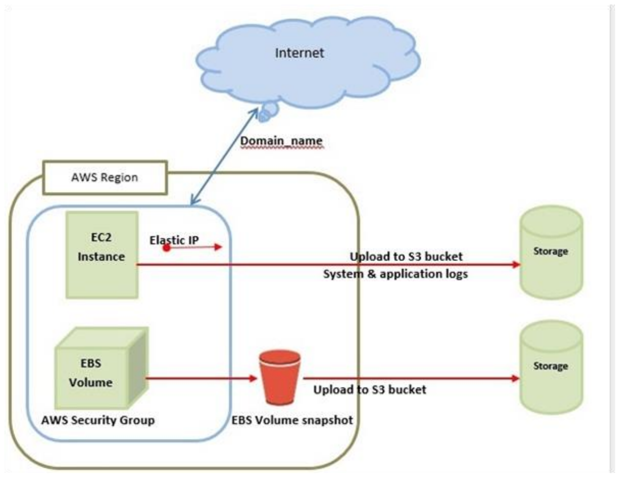
PYSPARK:

Companies that collect massive amounts of will have a big data framework like Apache Spark. The reason that PySpark works is, it can handle real-world data. With the help of PySpark it’s easy to write a code and collect data from various sources that is continuously updated. Furthermore, PySpark has the capability to recover loss after a failure occurs. The framework has in-memory computation and is stores in Random Access Memory (RAM). PySpark is an interface used for Apache Spark which is a distributed engine used to perform big data analytics

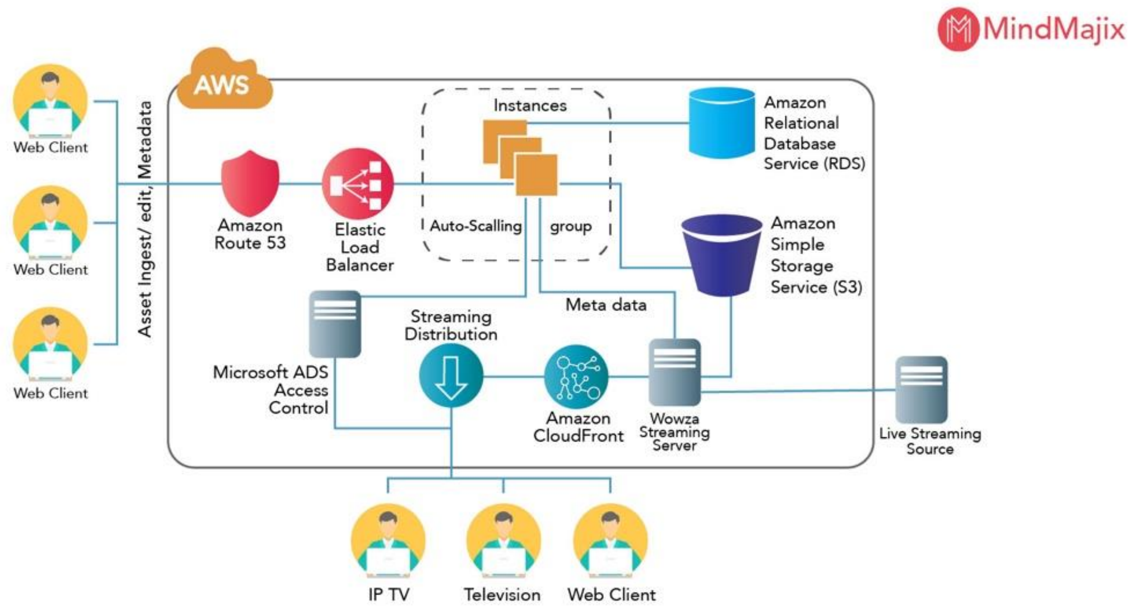
AWS:

AWS offers a wide range of storage services that can be provisioned depending on the project requirements and user cases. AWS Storage Services have different areas for highly confidential data, frequently accessed data, and the not so frequently data. Also, there are different storage types namely, object storage, file storage, block storage services, backups, and data migration options. All these categories fall under the AWS Storage Services List.

AWS S3:



Amazon Simple Storage Service provides object storage, which is built for storing and recovering any amount of information or data from anywhere over the internet. It provides this storage through a web services interface.



Exploratory Data Analysis

1. Understanding the data

Data Description

The Dataset consists of amazon fine food reviews including information about product id, user id, profile name, helpfulness numerator, helpfulness denominator, Score given by the customer, time, summary, and the text which is the review given by the customer. The dataset contains information about 5,68,454 customer reviews and it contains 10 variables. Detailed data description is as follows:

```
df.printSchema()
```

```
root
 |-- Id: integer (nullable = true)
 |-- ProductId: string (nullable = true)
 |-- UserId: string (nullable = true)
 |-- ProfileName: string (nullable = true)
 |-- HelpfulnessNumerator: string (nullable = true)
 |-- HelpfulnessDenominator: string (nullable = true)
 |-- Score: string (nullable = true)
 |-- Time: string (nullable = true)
 |-- Summary: string (nullable = true)
 |-- Text: string (nullable = true)
```

2. Data Preprocessing

2.1 Data Cleaning:

The original dataset consisted of 10 variables. Of those variables we selected "Text" and "Score" because "Text" contained the review in the form of text and "Score" was the review score given by the customer.

```
df = df.select('Text','Score')
```

```
df.printSchema()
```

```
root
 |-- Text: string (nullable = true)
 |-- Score: string (nullable = true)
```

```
|If you are lookin...|    2|
|Great taffy at a ...|    5|
|I got a wild hair...|    4|
|This saltwater ta...|    5|
|This taffy is so ...|    5|
|Right now I'm mos...|    5|
|This is a very he...|    5|
|I don't know if i...|    5|
|One of my boys ne...|    5|
|My cats have been...|    1|
|good flavor! thes...|    4|
|The Strawberry Tw...|    5|
|My daughter loves...|    5|
|I love eating the...|    2|
|I am very satisfi...|    5|
```

After looking at the above data frame, we checked for null values in the data frame, and dropped those values. There were about 10 rows which contained null values, so they were removed.

```
df = df.dropna(how='any')
```

```
+--------------------+-----+
only showing top 20 rows
```

2.2. Data Preparation

After getting the required columns and removing the null values, we plot the graph to see the distribution among the different score values present in the dataset and we get the following graph:

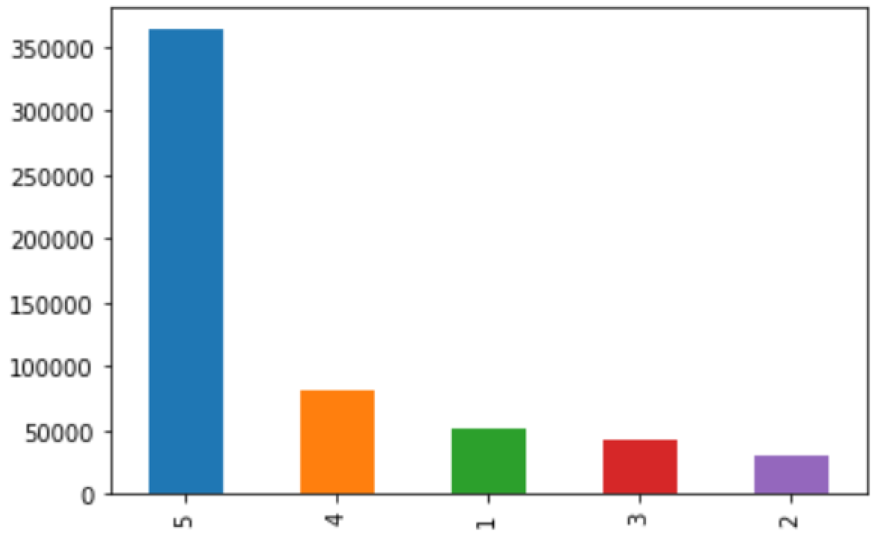


Fig.1.

From the above graph, we can see that we have a 5-star rating system and for rating 5 we have more reviews, to balance the classes we considered rating 4 and 5 as positive and the rest as negative reviews.

```
from pyspark.sql.functions import when
from pyspark.sql.functions import col
df_3 = df.withColumn("new_column",
              when((col("Score") == "3") | (col("Score") == "2") | (col("Score") == "1"), 0)
        .when((col("Score") == "4") | (col("Score") == "5"), 1)
        .otherwise("new_df.Score"))
df_4 = df_3.drop(df_3.Score)
```

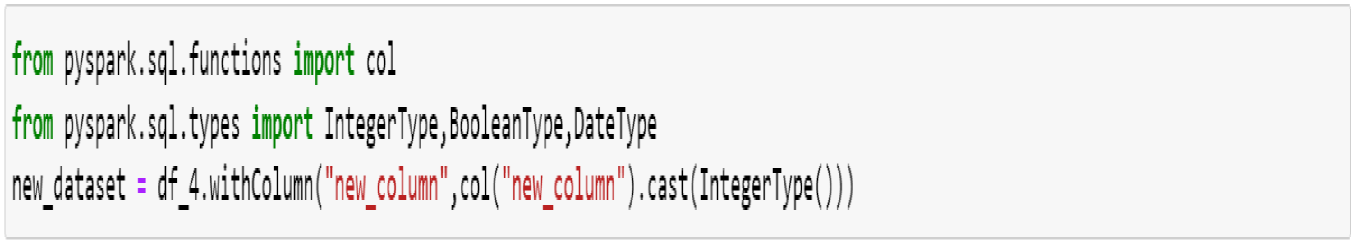
```
df_4.show()
```

```
+--------------------+----------+
|                Text|new_column|
+--------------------+----------+
|I have bought sev...|         1|
|"Product arrived ...|         0|
|"This is a confec...|         1|
|If you are lookin...|         0|
|Great taffy at a ...|         1|
|I got a wild hair...|         1|
|This saltwater ta...|         1|
|This taffy is so ...|         1|
|Right now I'm mos...|         1|
|This is a very he...|         1|
|I don't know if i...|         1|
|One of my boys ne...|         1|
|My cats have been...|         0|
|good flavor! thes...|         1|
|The Strawberry Tw...|         1|
|My daughter loves...|         1|
|I love eating the...|         0|
|I am very satisfi...|         1|
```

After getting the above columns, we plot again, and the result are as follows:

400000
300000
200000

The data type of the column



```
from pyspark.sql.functions import col
from pyspark.sql.types import IntegerType,BooleanType,DateType
new_dataset = df_4.withColumn("new_column",col("new_column").cast(IntegerType()))
```

"new_column" was string. That column classified the reviews as positive or negative and we needed the datatype to be an integer type, so we changed the datatype of that column.

2.2 Splitting the dataset into train and test

Then we split the data into 70 percent train and 30 percent test data. The train data consists of 3,97,943 reviews and the test data contains 1,70,501 reviews.

Methodology

We used Spark ML to build an ML pipeline and we used Logistic Regression, Naïve Bayes, and Linear SVC to build classifiers for the text classification. Then we compared the performance of the three ML algorithms to see which one gave the maximum accuracy.

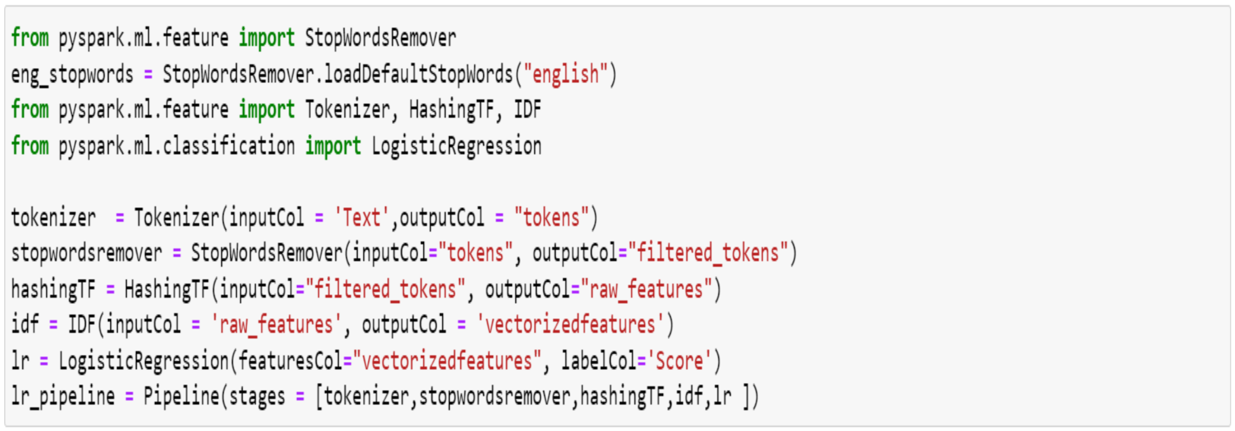
We briefly discuss the models we have used:

Logistic Regression –

Naïve Bayes –

Linear SVC -

ML Pipeline for Logistic Regression:



```
from pyspark.ml.feature import StopWordsRemover
eng_stopwords = StopWordsRemover.loadDefaultStopWords("english")
from pyspark.ml.feature import Tokenizer, HashingTF, IDF
from pyspark.ml.classification import LogisticRegression

tokenizer  = Tokenizer(inputCol = 'Text',outputCol = "tokens")
stopwordsremover = StopWordsRemover(inputCol="tokens", outputCol="filtered_tokens")
hashingTF = HashingTF(inputCol="filtered_tokens", outputCol="raw_features")
idf = IDF(inputCol = 'raw_features', outputCol = 'vectorizedfeatures')
lr = LogisticRegression(featuresCol="vectorizedfeatures", labelCol='Score')
lr_pipeline = Pipeline(stages = [tokenizer,stopwordsremover,hashingTF,idf,lr ])
```

As can be seen from the above figure the following steps were included in the pipeline:

- **Tokenization**

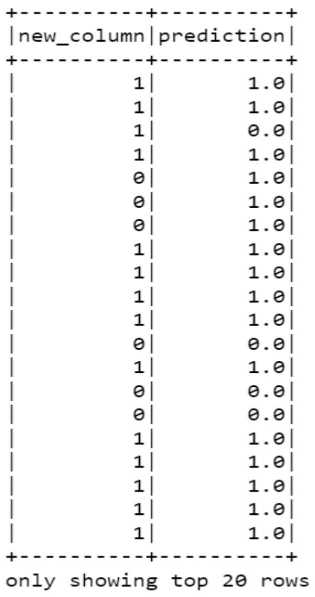
  The first step was to tokenize the "Text" column as an input, we get the output column "token" that will contain the tokens.

- **Removing Stop Words**

  In the next step, we remove the stop words so the input column for that step would be "tokens" and after removing the stop words we will receive an output column named "filtered_tokens".

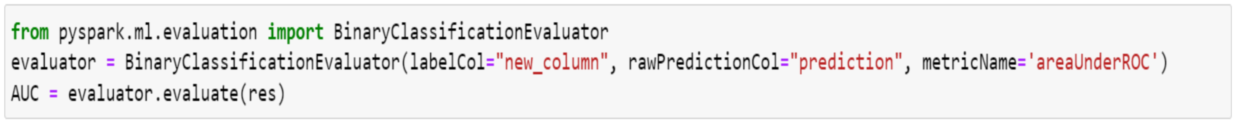
- After that, we will create the term frequency using hashingTF and in that step, the input column will be "filtered_tokens" and as an output, we will get the column "vectorizedfeatures".
- In the next step, we will create a logistic regression model.
- The model is then trained on the training dataset and once it is trained, we predict the performance of the trained model on the test set using the transform method.

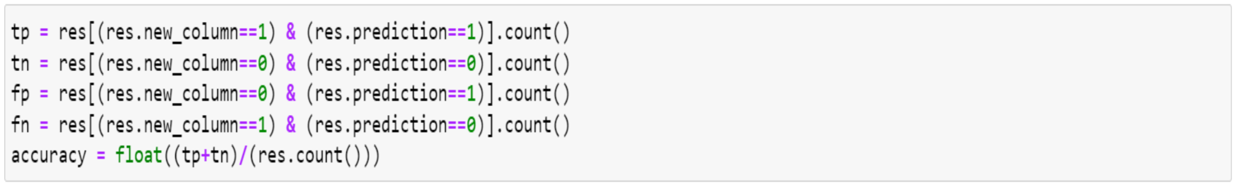
MODEL EVALUATION


```
+----------+----------+
|new_column|prediction|
+----------+----------+
|         1|       1.0|
|         1|       1.0|
|         1|       0.0|
|         1|       1.0|
|         0|       1.0|
|         0|       1.0|
|         0|       1.0|
|         1|       1.0|
|         1|       1.0|
|         1|       1.0|
|         1|       1.0|
|         0|       0.0|
|         1|       1.0|
|         0|       0.0|
|         0|       0.0|
|         1|       1.0|
|         1|       1.0|
|         1|       1.0|
|         1|       1.0|
|         1|       1.0|
+----------+----------+
only showing top 20 rows
```

The above figure shows the predictions obtained when we used the trained model on our test dataset.

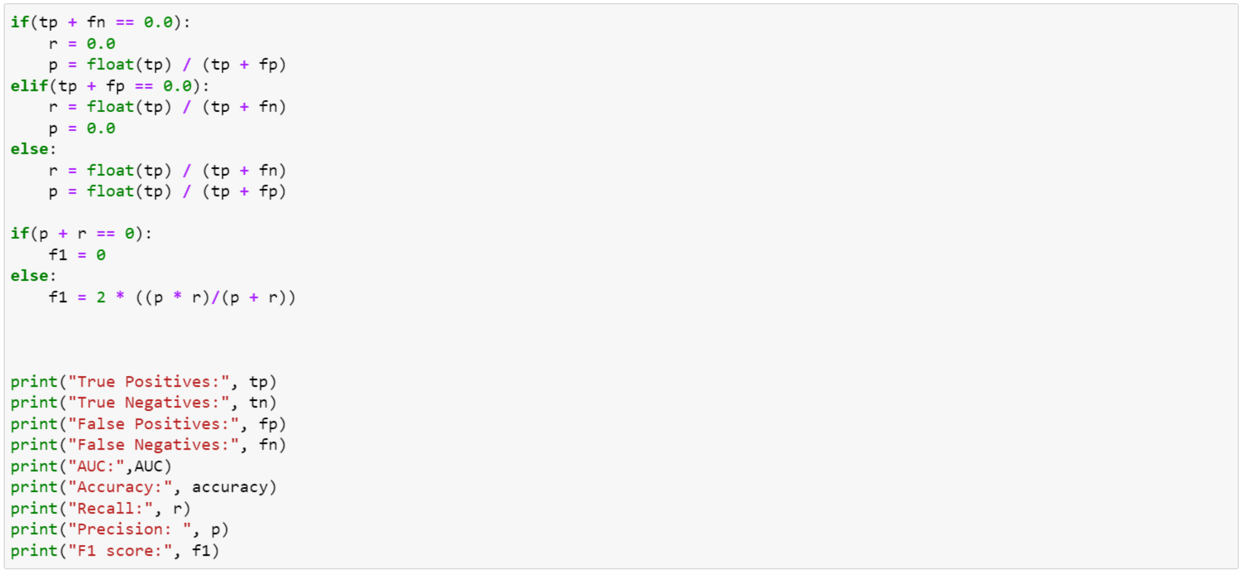
The next task was to evaluate the model and get the AUC score. We used the 'BinaryClassificationEvaluator' from 'pyspark.ml.evaluation'. As it is a binary classification problem, to get the Performance Metrics, we calculated the value of True Positive, True Negatives, False


```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator(labelCol="new_column", rawPredictionCol="prediction", metricName='areaUnderROC')
AUC = evaluator.evaluate(res)
```


```
tp = res[(res.new_column==1) & (res.prediction==1)].count()
tn = res[(res.new_column==0) & (res.prediction==0)].count()
fp = res[(res.new_column==0) & (res.prediction==1)].count()
fn = res[(res.new_column==1) & (res.prediction==0)].count()
accuracy = float((tp+tn)/(res.count()))
```
Positives, False Negatives, Accuracy, Recall, Precision, and F1 score as shown in the figure below:

By doing this we get the following values:

1. True Positives: 117734
2. True Negatives: 28059
3. False Positives: 9529
4. False Negatives: 14684
5. AUC: 0.8177984861846503



```
if(tp + fn == 0.0):
    r = 0.0
    p = float(tp) / (tp + fp)
elif(tp + fp == 0.0):
    r = float(tp) / (tp + fn)
    p = 0.0
else:
    r = float(tp) / (tp + fn)
    p = float(tp) / (tp + fp)

if(p + r == 0):
    f1 = 0
else:
    f1 = 2 * ((p * r)/(p + r))



print("True Positives:", tp)
print("True Negatives:", tn)
print("False Positives:", fp)
print("False Negatives:", fn)
print("AUC:",AUC)
print("Accuracy:", accuracy)
print("Recall:", r)
print("Precision: ", p)
print("F1 score:", f1)
```

6. Accuracy: 0.8575756149782949
7. Recall: 0.8891087314413448
8. Precision: 0.9251235630151733
9. F1 score: 0.9067586769921557

In a similar manner, we created pipelines for LinearSVC, and Naïve Bayes and achieved the following results:

For LinearSVC:

1. True Positives: 121396

2. True Negatives: 29007
3. False Positives: 8371
4. False Negatives: 11020
5. AUC: 0.8464110804524652
6. Accuracy: 0.8857969068400532
7. Recall: 0.9167774287095215
8. Precision: 0.9354920742561669
9. F1 score: 0.9260402085566188

For Naïve Bayes:

1. True Positives: 116797
2. True Negatives: 28505
3. False Positives: 9121
4. False Negatives: 15764
5. AUC: 0.8193345004157759
6. Accuracy: 0.8537784907190326
7. Recall: 0.8810811626345607
8. Precision: 0.9275639702028304
9. F1 score: 0.9037252542759759

The three classification models gave us the following results:

Classification Models	AUC	Accuracy	F-1 Score
Logistic Regression	0.8177984861846503	0.8575756149782949	0.9067586769921557

LinearSVC	0.8464110804524652	0.8857969068400532	0.9260402085566188
Naïve Bayes	0.8193345004157759	0.8537784907190326	0.9037252542759759

RESULTS

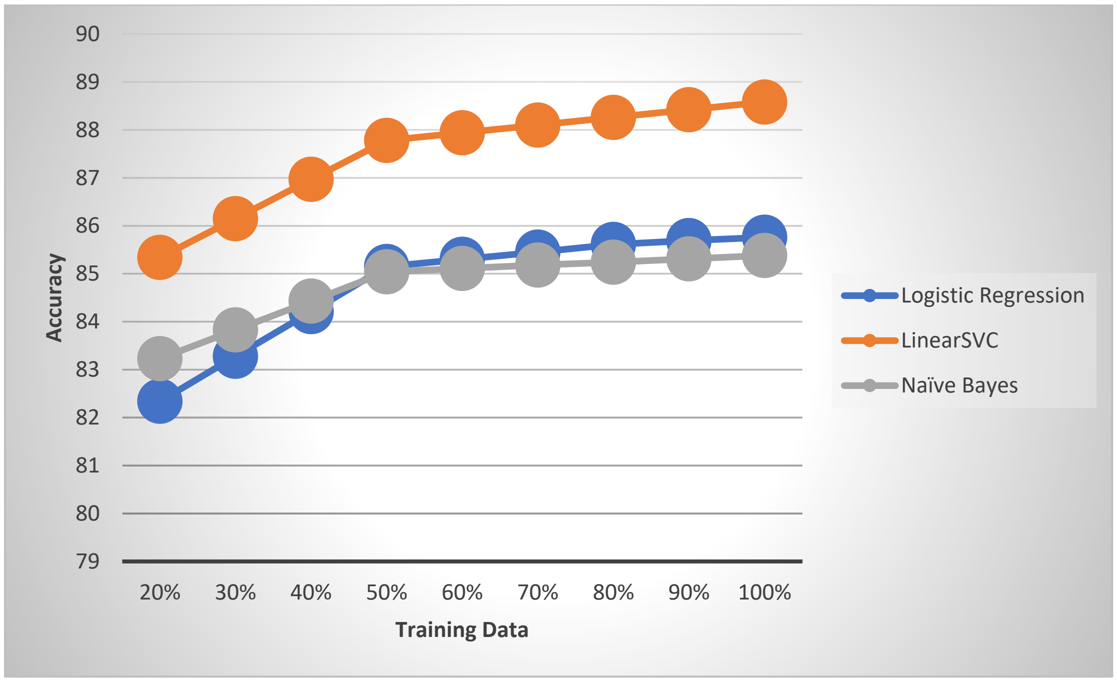
From the above table, we can see that LinearSVC provided better results with better AUC score, Accuracy and F-1 Score compared to the other two models.

Performance Measurement

SCALING

In order to see the performance based on scaling we tried to run the Spark Application using small subsets of data and in the end, we used whole data. Following is the graph which shows the impact of scalability on Accuracy.

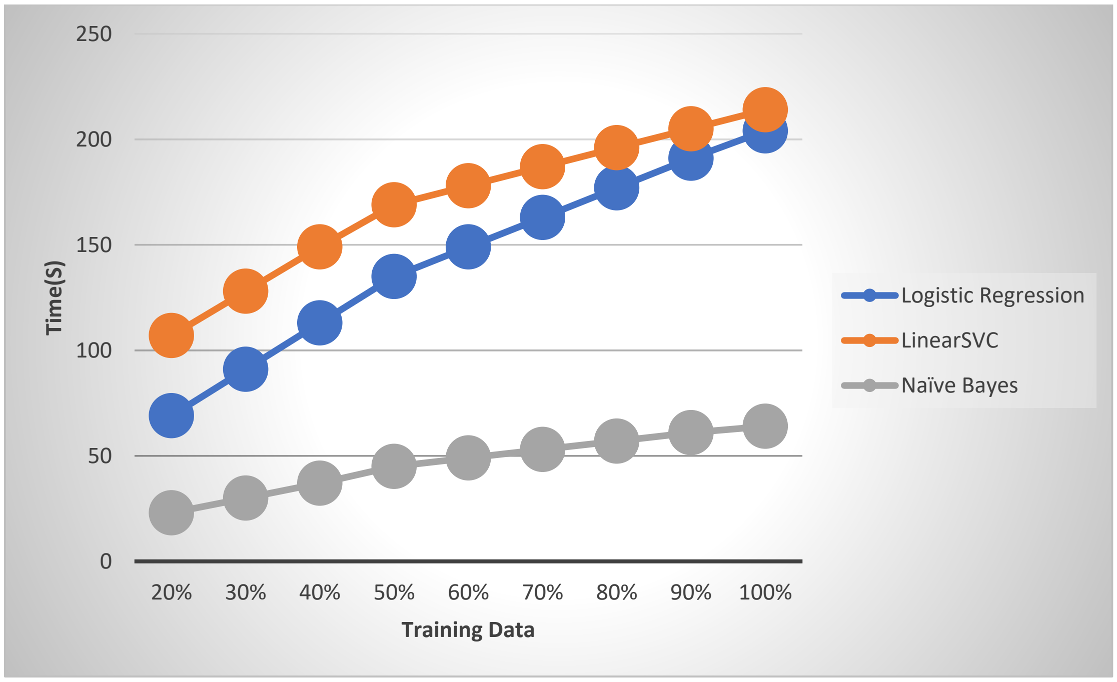
1. Impact of scalability on Accuracy



It shows that the Accuracy increases as we increase the training data. It is the highest for LinearSVC model and it's almost same for Logistic Regression and Naïve Bayes model.

2. Impact of Scale on Performance (Time)

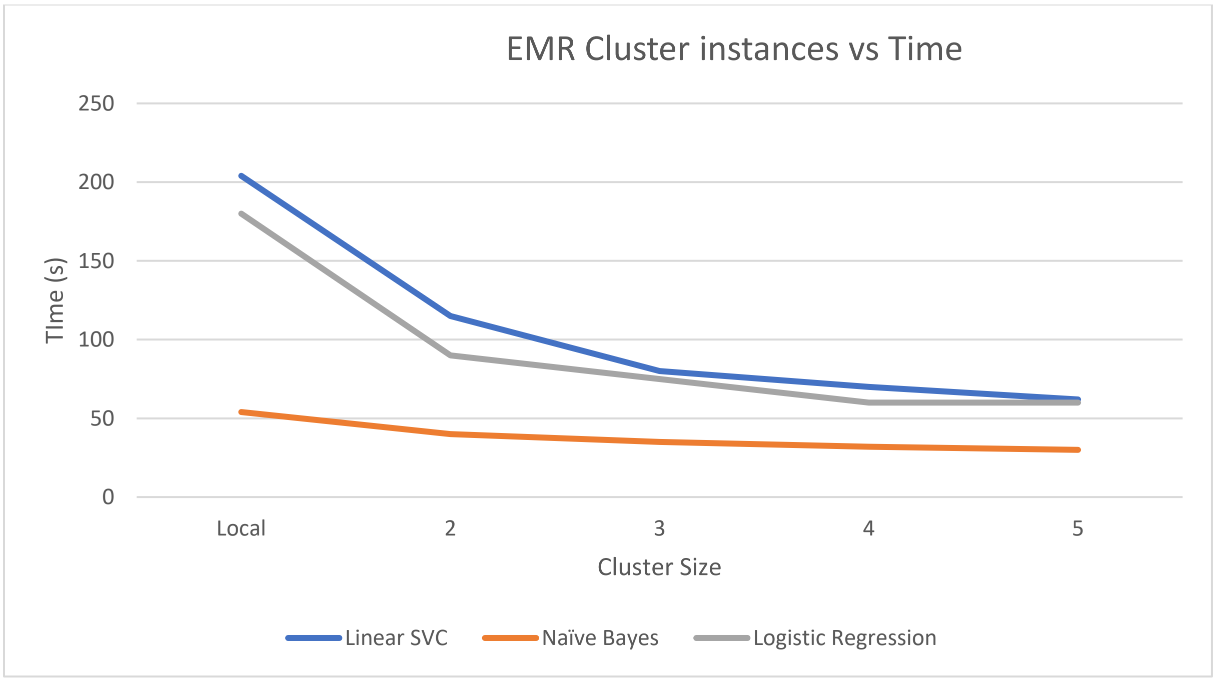
To further check our scalability performance, we check what has been its impact on the time taken for training our models.



The above chart shows how time increases as we scale the size of our data. Naïve Bayes takes the lowest time while the Logistic Regression and the LinearSVC models take almost the same time. For all the models, the time taken increases as the size of the data increases.

3. Impact of parallel computation on performance

To test the impact of parallel computation on the performance of our models we use AWS EMR clusters with 2 instances (1 Master 1 Slave), 3 instances (1 Master 2 slaves), and 4 instances (1 Master 3 Slaves). These were used to compare the performance with the local environment (Specs: 8GB RAM, Quad-Core CPU). The following graph shows the results obtained.



We can see from the above graph that the local environment takes a longer time than the AWS EMR parallel environment for all the clusters. The time decreases substantially when we switch from the local environment to the AWS clusters.

CONCLUSION

In this project, we were working with a huge amount of data which was about the Amazon food reviews given by customers. We created an ML pipeline which would do the text classification of the reviews and label the review as positive or negative review. Out of the three models we received the best accuracy when we used LinearSVC and the accuracy we received was around 88.6%. On the other hand, the accuracy obtained from Logistic Regression and Naïve Bayes model is quite similar.

When we did run our model on a local machine vs on an AWS cluster, we learned that the execution time taken by the model decreases as we increase the cluster size. Also, as we did the scaling, we learned that the accuracy of all three models increased as we increased the amount of data used and the execution time also increased with the increase in data size.

In future we can tune some hyperparameters and can further increase the accuracy of the models. We can also think about implementing these models on other cloud platforms like Micrisoft Azure and Google Cloud platform etc. In future we can also explore other machine learning models like random forest, SDC, or some deep learning models to increase the accuracy of the models.

REFERENCES

https://github.com/osemrt/Amazon-Fine-Food-Reviews

https://medium.com/analytics-vidhya/amazon-fine-food-reviews-featurization-with-natural-language-processing-a386b0317f56

https://medium.com/analytics-vidhya/amazon-fine-food-reviews-featurization-with-natural-language-processing-a386b0317f56

https://towardsdatascience.com/text-classification-of-amazon-fine-food-reviews-ed27948fc1a2

https://ai.plainenglish.io/semantic-analysis-of-amazon-fine-food-reviews-8225fe11ddb1

https://towardsdatascience.com/text-classification-of-amazon-fine-food-reviews-ed27948fc1a2