Tripartite Text Sentiment Analysis in Online Food Reviews: A PySpark and Spark ML Approach

Group-4

Group Members	Student id
Nikathyasmeen Shaik	12592648
Mohammed Feroz Shaik	16326315
Sowmini Raavi	16336112
Rakesh Naini	16341798
Abhinaya Pailla	16344740
Avula Sri Hari	16344742

Abstract— This paper details a sentiment analysis conducted on the Amazon Fine Food Reviews dataset using PySpark and Spark ML libraries in a Databricks environment. Leveraging a high-performance multi-node cluster, the study implements Logistic Regression, Naïve Bayes, Linear Support Vector Machine and Random Forest Classifiers coupled with Count Vectorizer for bi-gram feature engineering. The model successfully differentiates between positive, negative, and neutral reviews, achieving over an accuracy of 76% in a multiclass classification setup. This analysis underscores the potential of advanced big data tools in extracting meaningful insights from large-scale customer feedback.

I. INTRODUCTION

This project focuses on the Amazon Fine Food Reviews dataset, a rich repository of customer feedback on a diverse range of food products. Comprising ~500,000 reviews, this data set presents a unique opportunity to explore consumer sentiments in the food sector. It offers the complexity and volume necessary to test and refine sentiment analysis models effectively.

At the core of our analysis lies Apache Spark, a powerful distributed computing system renowned for its ability to handle large-scale data processing. Leveraging the PySpark interface, this project harnesses Spark's robust capabilities to manage and analyze extensive data from Amazon reviews. This framework is particularly suited for this task due to its efficiency in processing large datasets and its sophisticated machine learning libraries.

Our methodology encompasses utilizing machine learning techniques to classify reviews into distinct sentiment categories: positive, negative, and neutral. This classification not only aids in understanding consumer attitudes but also demonstrates the practical application of big data tools in extracting meaningful patterns from complex datasets.

The objectives of this project are multifaceted: to showcase the effectiveness of PySpark in big data sentiment analysis, to provide insights into customer opinions in the food industry, and to explore the nuances of sentiment classification using advanced computational techniques. By navigating these objectives, the project aims to contribute to the growing field of sentiment analysis and offer valuable perspectives for businesses leveraging customer review data.

Through this endeavor, we anticipate uncovering detailed insights into consumer sentiment trends, thereby offering a model which can be modified for analogous studies in different contexts. This project stands as a testament to the evolving landscape of big data analytics and its pivotal role in understanding and utilizing customer feedback in the digital era.

A. Project Goals and Objectives

In the realm of big data analytics, the ability to effectively process and analyze extensive sets of textual data is not just a technical challenge but a gateway to unlocking the vast potential of consumer insights. This project, centered around the comprehensive dataset, sets out to address this challenge head-on. Our primary ambition is to harness the power of distributed computing and advanced machine learning to develop a model that not only scales with the increasing volume of data but also provides accurate and nuanced insights into consumer sentiments. This endeavor is not just an exercise in technical proficiency but a significant step towards elevating the standards of sentiment analysis in the context of consumer feedback. Detailed goals and objectives are as follows:

- Construct an analytical model designed to process a substantial corpus of textual data, emphasizing scalability to adapt to increasing data volumes while maintaining computational efficiency, reflecting an advanced application of big data analytics.
- Attain a good level of accuracy in categorizing consumer reviews into defined sentiment classes (positive, negative, neutral), underscoring the model's ability to discern intricate sentiment nuances inherent in consumer feedback.
- Leverage the capabilities of PySpark for distributed data processing, harnessing its potential to conduct extensive data analysis tasks efficiently across a cluster computing environment.
- This goal involves utilizing the distributed nature of PySpark to its fullest potential, enabling the processing of large-scale data with optimal resource utilization.

- Systematically evaluate and contrast various machine learning algorithms within the Spark ML library, with an emphasis on identifying the most effective model for sentiment analysis in large datasets.

B. Project Scope

Dataset: The Amazon Fine Food Reviews dataset serves as the cornerstone of our analysis. It comprises approximately ~500,000 food product reviews, contributed by various users over a span of more than 12 years. Each review includes information such as the product ID, user ID, profile name, helpfulness votes, score (rating), time of review, summary, and the full text of the review. The dataset presents a rich amalgamation of consumer sentiments, ranging from succinct summaries to detailed descriptions, providing an extensive spectrum of opinions on a wide array of food products. The diversity and volume of this dataset make it an ideal candidate for applying and testing advanced sentiment analysis models. There are also a few parts of the dataset that are not utilized in this research, such as metadata that is not relevant to sentiment analysis.

Technological Boundaries: Our project is framed within a robust technological ecosystem, primarily utilizing PySpark and Spark ML, operated within a Databricks environment. PySpark, the Python API for Apache Spark, is a key component, enabling efficient handling of large-scale data processing. Spark ML, a part of Apache Spark's scalable machine learning library, offers a range of algorithms and utilities for machine learning tasks. Databricks provides an optimized environment for running these tools, offering a high-performance multi-node cluster that enhances computational capabilities. This combination of technologies ensures not only efficient data processing and analysis but also scalability and reliability in handling the complexities of large datasets .

Analysis Limitations: The scope of our sentiment analysis is confined to textual data available in the dataset. While this provides substantial grounds for analysis, there are inherent limitations. Firstly, the analysis is restricted to the sentiment classification of review texts and does not extend to other elements like user ratings or metadata. Additionally, the sentiment analysis is focused on classifying sentiments into three categories (positive, negative, and neutral) and does not delve into more nuanced aspects of emotions or specific attributes of the food products. Furthermore, the reliability of sentiment analysis is contingent on the inherent biases and subjectivity of the user reviews, which might affect the overall accuracy and insights derived from the data.

C. **Project Limitation and Constraints**

Some of the Limitations that needs to be considered are:

- The Food Reviews dataset may contain biases from subjective user experiences, potentially skewing the sentiment analysis.
- The analysis was confined to textual data, omitting multimodal elements such as user-provided images.
- Computational limitations were encountered during intensive machine learning processes such as hyperparameter tuning.
- Power BI showed limitations in processing large volumes of data, necessitating usage of subset of dataset for effective visualization and feasibility.
- The sentiment analysis was limited to broad categorizations (positive, negative, neutral), which may not capture the complex spectrum of human emotions.
- The selection and optimization of machine learning models were bounded by the available computational resources.
- The findings and insights are contextually tied to the dataset, limiting their applicability when extrapolated to other domains or datasets.
- The project exclusively utilized Spark ML classifiers for sentiment analysis, precluding the exploration of deep learning models, which could potentially enhance the accuracy and richness of sentiment analysis.

These limitations provide valuable insights into future improvements and expansions of the project.

C. Feasibility Study

This section evaluates the feasibility of the sentiment analysis project, considering both technical and resource aspects, to ensure the project's viability and success.

Technical Feasibility:

Apache Spark Capabilities: The use of Apache Spark, specifically its PySpark interface, was deemed technically feasible for handling the large-scale data processing required for this project. Spark's in-memory data processing capabilities and

its efficient handling of big data analytics were key factors in this assessment.

Machine Learning Libraries: The Spark ML library, known for its diverse range of machine learning algorithms and tools, was evaluated for its suitability in implementing the required models (Logistic Regression, Naïve Bayes, Random Forest Classifier, SVM) and for conducting complex computations efficiently.

Data Handling and Processing: The technical feasibility of managing and processing the dataset, which is extensive in size and complexity, was confirmed. This included the ability to perform data preprocessing, feature extraction, and model training effectively within the Spark environment.

Integration with Databricks: Utilizing Databricks for this project was found to be technically feasible, offering an enhanced environment for running Spark applications. Databricks' optimized setup for big data processing significantly contributed to the project's technical viability.

Resource Feasibility:

Computational Resources: The project required substantial computational resources, including a multi-node cluster with sufficient processing power and memory to handle large-scale data analytics. The availability of these resources was confirmed, ensuring the smooth execution of data processing and model training tasks.

Data Storage and Management: Adequate data storage facilities were required to store the large dataset and intermediate processing data. The project's resource assessment confirmed the availability of sufficient storage space and robust data management systems.

In conclusion, the feasibility study affirmed that both technical and resource requirements for the sentiment analysis project were met, paving the way for its successful implementation and execution. This study provided the foundational confidence to proceed, ensuring that the project was viable in terms of technology, resources, and practical constraints.

II. SYSTEM REQUIREMENT SPECIFICATIONS

A. Local Machine Specs

- System: 12th Gen Intel(R) Core (TM) i7-1255U 1.70 GHz
- Ram: 16.0 GB

1. Databricks Environment Specifications:

Cluster Configuration: The project utilized a Databricks cluster configuration with the ability to scale between 2 to 8 worker nodes. Each node is of type i3. xlarge, which comes with 30.5 GB of memory and 4 CPU cores. This configuration supports the distribution of the data processing workload and can handle the large-scale data analytics required by the project.

Databricks Runtime Version: The Databricks runtime for the project is version 13.3 LTS, which includes Apache Spark 3.4.1 and Scala 2.12, ensuring a stable and long-term support environment for the analysis tasks.

2. **Software Requirements:**

Apache Spark & PySpark -The project leveraged Apache Spark version 3.4.1 through the PySpark interface, fully integrated into the Databricks runtime 13.3 LTS environment. This version supports advanced data processing and machine learning capabilities.

Spark ML Libraries: Within the Spark MLlib used, essential libraries such as ML Pipelines, feature transformers, and various classification algorithms were employed to develop the sentiment analysis models.

Additional Libraries and Tools: The environment was augmented with additional Python libraries for tasks that complement Spark's capabilities. Libraries like Pandas for data manipulation, Matplotlib and Seaborn for data visualization, and other specialized libraries were used as part of the analysis pipeline.

III. SYSTEM DESIGN

A. System Architecture

This architecture illustrates the overall architecture of your sentiment analysis system, including data flow and component interaction.

Fig. 1. Architecture

B. Use case diagram

A use case diagram serves to illustrate the functional capabilities of a system through a visual representation. It outlines the interactions between various actors and the system, detailing the specific objectives (or use cases) these interactions achieve. In this context, the diagram is instrumental in demonstrating how different users or roles interact with the sentiment analysis system. Each interaction is mapped out to emphasize the diverse roles and their engagement with the system, reflecting the workflow and dependencies among the use cases.

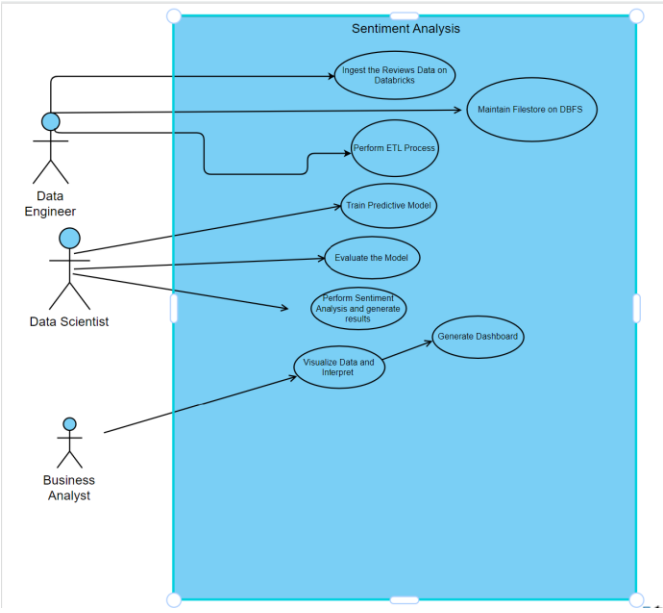


Fig. 2. Usecase diagram

C. **Sequence Diagram**

The sequence diagram is a critical tool that illustrates the chronological order of interactions between various processes within a system. It's essentially a graphical representation of the flow of operations, showcasing how these operations communicate through messages in a sequential manner. In the context of our project on sentiment analysis of online food reviews, the sequence diagram serves as a detailed guide to understanding the operational flow.

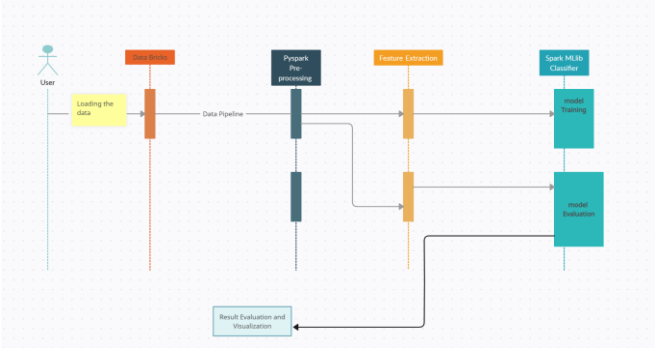


Fig. 3. Sequence Diagram

It visualizes the sequence of events, from data ingestion to processing and analysis using PySpark and Spark ML, and finally to output generation.

IV. DATA DESIGN

A. ETL Process

In the sentiment analysis project utilizing the Food Reviews dataset, the ETL (Extract, Transform, Load) process is a pivotal component, meticulously orchestrated to prepare the dataset for analytical and machine learning purposes. This process is structured into three distinct phases: Extract, Transform, and Load, each tailored to the specific needs and objectives of the project.

Extract:

Data Source Extraction: The initial phase involves extracting the Reviews dataset(.csv), a substantial compilation of user-generated product reviews, which serves as the foundational data source for the project.

Data Retrieval Mechanism: Leveraging the Databricks platform, the dataset is fetched from its File Store location in the Databricks File System(DBFS). This extraction is facilitated through PySpark's data reading capabilities, wherein the CSV file is ingested into a DataFrame, a versatile data structure apt for handling large datasets in PySpark.

Transform:

Data Cleansing and Preprocessing: The transformation phase encompasses a series of rigorous data cleaning and preprocessing steps:

Data Deduplication and Null Value Treatment: Employing methods such as dropDuplicates() and na.drop(), the dataset is cleansed of duplicate entries and missing values, ensuring data quality and consistency.

Textual Data Transformation: This involves several critical operations, including:

Text normalization (e.g., converting text to lowercase).

Purging of irrelevant characters and hyperlinks using regular expressions, enhancing the textual data quality.

Tokenization and removal of stop words to structure the textual content.

Generation of n-grams and application of CountVectorizer, transforming the text into a numerical format conducive to machine learning algorithms.

Label Formulation: An essential part of the transformation process is the categorization of review scores into sentiment labels (positive, negative, neutral). This step is fundamental for the subsequent supervised learning tasks.

Load:

Storage in DBFS: Upon transformation, the processed dataset is stored back into DBFS, Databricks' integrated file system. This storage strategy not only ensures the persistence of the transformed dataset but also facilitates efficient data retrieval for analysis and model training.

Concurrently, an in-memory version of the dataset has been maintained within the Databricks environment. This approach facilitated instantaneous data access for ongoing analysis, allowing for seamless continuation of data processing and exploratory analysis within the same computational session.

For future computational sessions or subsequent analyses, the stored dataset could be efficiently retrieved from DBFS.

The ETL process is meticulously designed to refine the raw review data into a structured and analysis-ready format. This process is integral to transforming the dataset into an asset for machine learning modeling and data visualization, ultimately driving the extraction of meaningful insights from the extensive consumer feedback encapsulated in the dataset.

B. Data Management

In the sentiment analysis project, the data collection phase is critical for establishing a robust foundation for subsequent analysis.

Primary Data Source: The project's main data source is the Amazon Fine Food Reviews dataset. This dataset encompasses a vast array of consumer feedback on various food products, offering a rich tapestry of insights into customer sentiments and preferences.

Data Types and Structure: The dataset primarily consists of structured data, including textual reviews and numerical ratings. Each review encompasses multiple attributes such as product ID, user ID, profile name, helpfulness votes, score, time of review, summary, and the review text itself.

Additional Data Sources: While the Amazon dataset forms the core of the analysis, additional data sources, such as metadata or external benchmarks for sentiment analysis, may be incorporated to enrich the analysis, though these are not explicitly mentioned in the provided context.

Storage (Format, Replication, Security)

The storage strategy for the project is tailored to manage the large-scale dataset efficiently, ensuring data integrity, accessibility, and security.

Storage Format: The dataset is stored in CSV (Comma-Separated Values) format, a common and widely used format for storing tabular data. This format is particularly suitable for large datasets due to its simplicity and ease of handling in data processing frameworks like PySpark.

Storage in DBFS: The data is stored in the DBFS, a distributed file system mounted into the Databricks workspace. DBFS facilitates easy and efficient data management within the Databricks ecosystem, allowing seamless access to the dataset across various clusters and notebooks.

Data Replication: DBFS inherently provides data replication, ensuring data redundancy and high availability. This feature is crucial for safeguarding the data against potential loss due to hardware failures or other unforeseen incidents.

Security Measures: The Databricks platform, including DBFS, incorporates robust security measures to protect data. These measures include data encryption at rest and in transit, rigorous access controls, and compliance with industry-standard security protocols. Such security mechanisms are pivotal in maintaining the confidentiality and integrity of sensitive data, especially when dealing with user-generated content.

The data management approach in the project is thus characterized by meticulous attention to data collection and storage, ensuring that the dataset's richness is preserved and leveraged effectively while maintaining high standards of data security and integrity.

C. Data Engineering

The data engineering phase for the sentiment analysis project, utilizing the Food Reviews dataset, was strategically planned with specific objectives to ensure the efficacy of the analysis.

Data Preparation for Sentiment Analysis: The primary goal was to prepare the dataset for effective sentiment analysis. This included data cleaning, transformation, feature engineering, and balancing the dataset for unbiased model training.
Scalability and Processing Efficiency: Emphasis was placed on designing a scalable and efficient data processing pipeline, leveraging the distributed computing power of Databricks and PySpark, especially crucial given the dataset's substantial volume.
Quality and Integrity Assurance: Ensuring the high quality and integrity of the data was a critical objective, given the implications for the accuracy of sentiment analysis.
Processing (Cleaning, Transforming, Balancing)
The processing stage involved several steps to refine and balance the dataset:
Data Cleaning:
Removing duplicates and handling missing values to improve data quality and consistency.
Data Transformation:

- Normalizing text i.e., converting to lowercase.
- Removing noise such as irrelevant characters and hyperlinks.
- Tokenizing text and removing stop words for structured analysis.
- Creating n-grams and applying Count Vectorizer for feature extraction.

Data Balancing:
The dataset was balanced to address the skewness in sentiment distribution. Given the different counts of positive, negative, and neutral reviews, balancing was crucial to avoid bias in model training.
Techniques like under sampling the overrepresented classes and oversampling the underrepresented ones were employed. This ensured that each sentiment class (positive, negative, neutral) had an approximately equal presence in the dataset, facilitating a more unbiased and accurate machine learning model training.
Validation
Validation ensured the effectiveness and accuracy of the data engineering process:
Data Quality Checks: Post-cleaning and transformation, the dataset was scrutinized to ensure data integrity and that key sentiments and meanings within the reviews were preserved.
Balancing Verification: The effectiveness of the dataset balancing process was validated, ensuring that each sentiment class was adequately represented. This step was critical to confirm the removal of any inherent biases in the dataset that could affect the sentiment analysis.
Model Input Assessment: The suitability of the transformed and balanced data as input for the machine learning models was validated. This involved ensuring that the feature vectors correctly encapsulated the necessary information for sentiment classification.
Iterative Refinement: The data engineering process was iteratively refined based on feedback from initial model results, aligning the data preparation steps with the overarching goal of conducting a comprehensive and unbiased sentiment analysis.
Incorporating dataset balancing into the data engineering phase significantly enhanced the reliability of the sentiment analysis, ensuring a more equitable representation of different sentiments and thereby contributing to the robustness of the machine learning models.

D. Data Analysis and Modelling

The project leverages both descriptive and predictive analytics to extract insights from the dataset and to forecast sentiment trends.

Descriptive Analytics:
Initial Data Exploration: Involved summarizing the dataset to understand its characteristics, such as the distribution of review sentiments, frequency of words in reviews, and patterns in customer feedback. Techniques like generating word clouds and

conducting basic statistical analysis were employed to grasp the underlying structure of the data.

Visualization: Tools like Matplotlib and plotpy express were used for visualizing various aspects of the data, such as review score distributions, which provided intuitive insights into customer sentiments.

Predictive Analytics:

Sentiment Classification Models: The core of predictive analytics in this project was building machine learning models to classify the sentiments of reviews into categories like positive, negative, and neutral.

Model Training and Evaluation: Models such as Logistic Regression, Random Forest Classifier, SVM, and NaiveBayes from the PySpark MLlib were trained using the balanced dataset. The evaluation of the models was conducted by examining their performance across various metrics, including accuracy, precision, recall, and F1 scores to determine their effectiveness in predicting sentiments.

Statistical / ML

The project's approach to data analytics was grounded in both statistical methods and machine learning techniques.

Statistical Analysis:

Exploratory Data Analysis (EDA): This involved statistical methods to explore and understand the data, including measures of central tendency, dispersion, and correlation analysis between different features.

Hypothesis Testing: Statistical tests have been used to validate certain hypotheses about the data, such as the significance of certain words or phrases in predicting sentiment.

Machine Learning:

Feature Engineering: Critical in transforming raw text into actionable insights. Techniques Count Vectorization, and n-gram models were employed to convert text data into a format suitable for machine learning models.

Model Development and Tuning: The project involved developing various classification models, tuning their hyperparameters, and employing cross-validation techniques to enhance model performance and reduce the risk of overfitting.

Model Interpretation and Evaluation: Post-training, the models were interpreted to understand their performance. Metrics including accuracy, precision, recall, and F1 score were utilized to assess the performance of the models and model's effectiveness in correctly classifying sentiments.

The integration of descriptive and predictive analytics, supplemented by a blend of statistical methods and machine learning techniques, culminated in a comprehensive dataset analysis. This approach illuminated the current data landscape and enabled precise predictions of customer sentiments. Notably, our models achieved high accuracy levels, with SVM and Logistic Regression with cross-validation both reaching an accuracy of 76%, outperforming the standard Logistic Regression and Naive Bayes models. These findings provided valuable insights for strategic decision-making and highlighted the potential of advanced analytics in extracting meaningful insights.

E. Data Visualization

The Data visualization component of the sentiment analysis project plays a crucial role in translating complex analytical findings into understandable and actionable insights. Utilizing a range of visualization tools and techniques, this phase aimed to effectively communicate the patterns, trends, and results derived from the Food Reviews dataset.

Sentiment Distribution in Balanced Dataset

Bar charts were utilized to depict sentiment distribution across the balanced dataset, effectively illustrating the proportion of positive, negative, and neutral reviews. This approach was integral in assessing the sentiment landscape and validating the dataset balancing process.

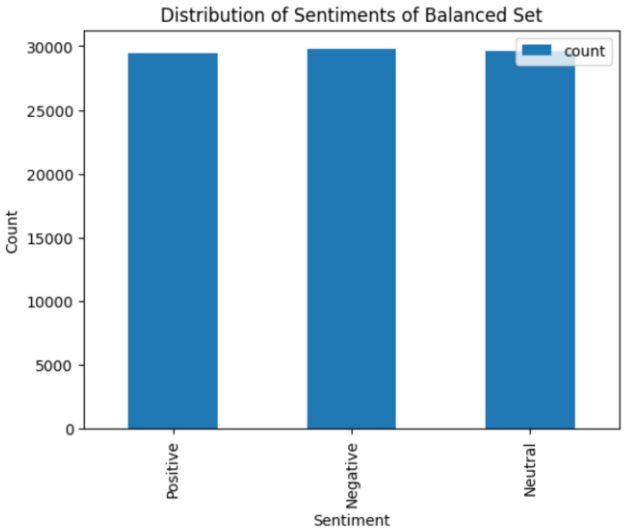


Fig. 04. Sentiment distribution of balanced dataset

Word Clouds for Sentiment Categories

Word clouds were generated for each sentiment category – positive, neutral, and negative. These visual representations highlighted the most frequent words within each category, offering insights into the predominant themes and linguistic nuances associated with different sentiments.

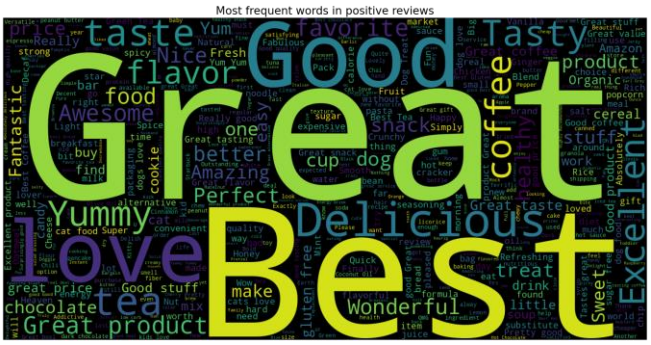


Fig. 05. Frequent terms against positive reviews

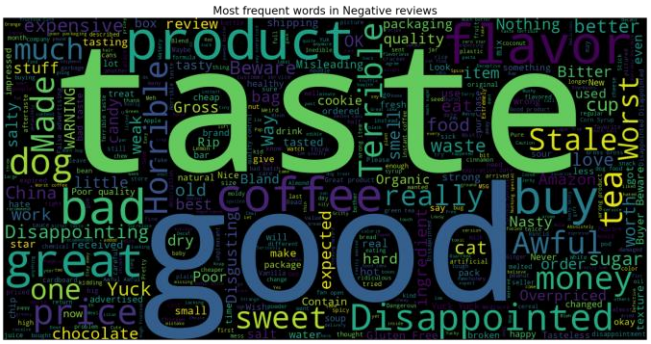
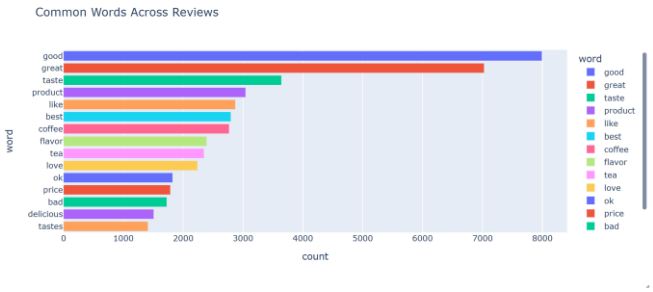


Fig. 06. Frequent terms against negative reviews

Fig. 07. Frequent terms against neutral reviews

Common Words Frequency Analysis

A Counter object was utilized to identify and quantify the most frequent words across reviews. This analysis shed light on prevalent linguistic patterns and topics within customer feedback, providing quantitative insights into the language used across the dataset.



V. Code

Hereby sharing the snippets of the code that have been paramount as part of the analysis.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, udf, lower, regexp_replace, when
from pyspark.sql.types import StringType, IntegerType
from pyspark.ml import Pipeline
from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, HashingTF, IDF, StringIndexer, CountVectorizer
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Initialize Spark Session
spark = SparkSession.builder.appName("SentimentAnalysis").getOrCreate()

# File location and type
file_location = "/FileStore/tables/Reviews.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# Read the CSV file to DataFrame
df = spark.read.csv(file_location, inferSchema=True, header=True, sep=delimiter)

# Dropping duplicates and rows with missing values in 'Text' or 'Score' columns
df = df.dropDuplicates(['Text']).na.drop(subset=['Text', 'Score'])

# Converting the score to a sentiment label using withColumn and when
df = df.withColumn("Sentiment", when(col("Score") < 3, 'Negative')
                                .when(col("Score") == 3, 'Neutral')
                                .otherwise('Positive'))

# Remove punctuation, hashtags, and links
df = df.withColumn("text", lower(col("Text")))
df = df.withColumn("text", regexp_replace(col("text"), r"http\S+", ""))
df = df.withColumn("text", regexp_replace(col("text"), "[^a-zA-Z\s]", ""))
```

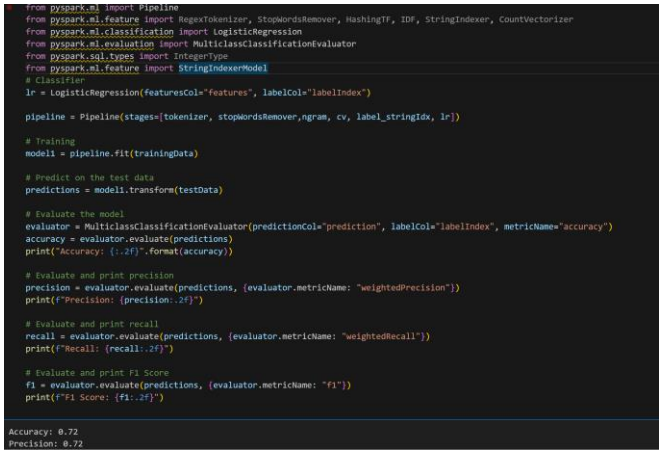
Fig. 08 – Data fetched from Filestore into spark data frame and Data Cleansing

```
from pyspark.sql import functions as F

from pyspark.sql.functions import col

# Calculating the counts of each class
class_counts = df.groupBy("Sentiment").count()
class_counts.show()

# Extracting the counts for each sentiment
positive_count = class_counts.filter(col("Sentiment") == "Positive").collect()[0]["count"]
neutral_count = class_counts.filter(col("Sentiment") == "Neutral").collect()[0]["count"]
negative_count = class_counts.filter(col("Sentiment") == "Negative").collect()[0]["count"]


# Targeting count for balancing (using the smallest class count)
target_count = min(positive_count, neutral_count, negative_count)

# Calculate sampling ratios
positive_sample_ratio = target_count / positive_count
negative_sample_ratio = target_count / negative_count


# Undersampling the 'Positive' class and oversampling 'Negative' class
positive_df = df.filter(col("Sentiment") == "Positive").sample(False, positive_sample_ratio)
negative_df = df.filter(col("Sentiment") == "Negative").sample(True, negative_sample_ratio)

# Keeping the 'Neutral' class as is
neutral_df = df.filter(col("Sentiment") == "Neutral")

# Combining the datasets
balanced_df = positive_df.unionAll(negative_df).unionAll(neutral_df)
```

Fig. 09 – Transformation into Balanced data set

```
# Preprocessing Pipeline
from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, HashingTF, IDF, StringIndexer, CountVectorizer
from pyspark.sql.types import IntegerType
from pyspark.ml.feature import StringIndexerModel
from pyspark.sql.functions import col, udf, lower, regexp_replace
from pyspark.ml.feature import NGram

tokenizer = RegexTokenizer(inputCol="text", outputCol="words", pattern="\\W")
stopWordsRemover = StopWordsRemover(inputCol="words", outputCol="filtered")
ngram = NGram(n=2, inputCol="filtered", outputCol="ngrams")  # Set n=2 for bi-grams, n=3 for tri-grams

cv = CountVectorizer(inputCol="ngrams", outputCol="features")
label_stringIdx = StringIndexer(inputCol="Sentiment", outputCol="labelIndex")
```

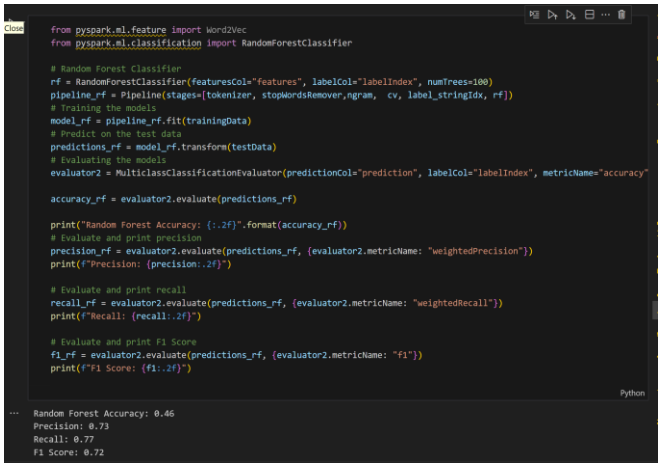
Fig. 10 - Preprocessing Pipeline



```
from pyspark.ml import Pipeline
from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, HashingTF, IDF, StringIndexer, CountVectorizer
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql.types import IntegerType
from pyspark.ml.feature import StringIndexerModel
# Classifier
lr = LogisticRegression(featuresCol="features", labelCol="labelIndex")

pipeline = Pipeline(stages=[tokenizer, stopWordsRemover,ngram, cv, label_stringIdx, lr])

# Training
model1 = pipeline.fit(trainingData)

# Predict on the test data
predictions = model1.transform(testData)

# Evaluate the model
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="labelIndex", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Accuracy: {:.2f}".format(accuracy))

# Evaluate and print precision
precision = evaluator.evaluate(predictions, {evaluator.metricName: "weightedPrecision"})
print(f"Precision: {precision:.2f}")

# Evaluate and print recall
recall = evaluator.evaluate(predictions, {evaluator.metricName: "weightedRecall"})
print(f"Recall: {recall:.2f}")

# Evaluate and print F1 Score
f1 = evaluator.evaluate(predictions, {evaluator.metricName: "f1"})
print(f"F1 Score: {f1:.2f}")
```

```
Accuracy: 0.72
Precision: 0.72
```

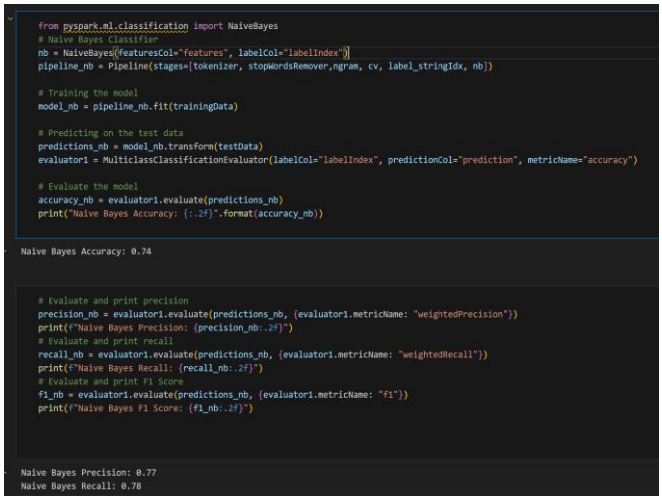
Fig. 11 – Logistic Regressive model evaluation



```
from pyspark.ml.feature import Word2Vec
from pyspark.ml.classification import RandomForestClassifier

# Random Forest Classifier
rf = RandomForestClassifier(featuresCol="features", labelCol="labelIndex", numTrees=100)
pipeline_rf = Pipeline(stages=[tokenizer, stopWordsRemover,ngram,  cv, label_stringIdx, rf])
# Training the models
model_rf = pipeline_rf.fit(trainingData)
# Predict on the test data
predictions_rf = model_rf.transform(testData)
# Evaluating the models
evaluator2 = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="labelIndex", metricName="accuracy"

accuracy_rf = evaluator2.evaluate(predictions_rf)

print("Random Forest Accuracy: {:.2f}".format(accuracy_rf))
# Evaluate and print precision
precision_rf = evaluator2.evaluate(predictions_rf, {evaluator2.metricName: "weightedPrecision"})
print(f"Precision: {precision:.2f}")

# Evaluate and print recall
recall_rf = evaluator2.evaluate(predictions_rf, {evaluator2.metricName: "weightedRecall"})
print(f"Recall: {recall:.2f}")

# Evaluate and print F1 Score
f1_rf = evaluator2.evaluate(predictions_rf, {evaluator2.metricName: "f1"})
print(f"F1 Score: {f1:.2f}")
```

```
Random Forest Accuracy: 0.46
Precision: 0.73
Recall: 0.77
F1 Score: 0.72
```

Fig. 12 – Random Forest model evaluation



```
from pyspark.ml.classification import NaiveBayes
# Naive Bayes Classifier
nb = NaiveBayes(featuresCol="features", labelCol="labelIndex")
pipeline_nb = Pipeline(stages=[tokenizer, stopWordsRemover,ngram, cv, label_stringIdx, nb])

# Training the model
model_nb = pipeline_nb.fit(trainingData)

# Predicting on the test data
predictions_nb = model_nb.transform(testData)
evaluator1 = MulticlassClassificationEvaluator(labelCol="labelIndex", predictionCol="prediction", metricName="accuracy")

# Evaluate the model
accuracy_nb = evaluator1.evaluate(predictions_nb)
print("Naive Bayes Accuracy: {:.2f}".format(accuracy_nb))
```

```
Naive Bayes Accuracy: 0.74
```

```
# Evaluate and print precision
precision_nb = evaluator1.evaluate(predictions_nb, {evaluator1.metricName: "weightedPrecision"})
print(f"Naive Bayes Precision: {precision_nb:.2f}")
# Evaluate and print recall
recall_nb = evaluator1.evaluate(predictions_nb, {evaluator1.metricName: "weightedRecall"})
print(f"Naive Bayes Recall: {recall_nb:.2f}")
# Evaluate and print F1 Score
f1_nb = evaluator1.evaluate(predictions_nb, {evaluator1.metricName: "f1"})
print(f"Naive Bayes F1 Score: {f1_nb:.2f}")
```

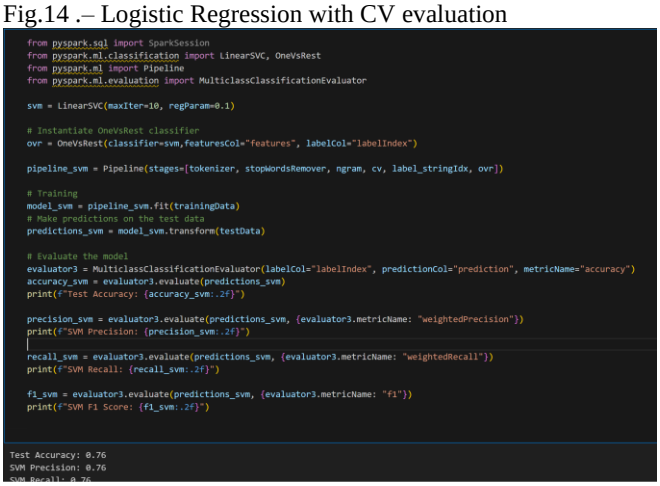
```
Naive Bayes Precision: 0.77
Naive Bayes Recall: 0.78
```

Fig.13 – Naïve bayes model evaluation

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml import Pipeline

# Logistic Regression
lr = LogisticRegression(featuresCol="features", labelCol="labelIndex")

# Pipeline
pipeline_lr = Pipeline(stages=[tokenizer, stopWordsRemover,ngram, cv, label_stringIdx, lr])

# Hyperparameter Tuning
paramGrid_lr = ParamGridBuilder() \
    .addGrid(lr.regParam, [0.1, 0.01]) \
    .addGrid(lr.maxIter, [10, 20]) \
    .build()

crossval_lr = CrossValidator(estimator=pipeline_lr,
                             estimatorParamMaps=paramGrid_lr,
                             evaluator=MulticlassClassificationEvaluator(labelCol="labelIndex", predictionCol="prediction", metricName="accuracy"),
                             numFolds=5)

# Training and Evaluation
cvModel_lr = crossval_lr.fit(trainingData)
prediction_lr = cvModel_lr.transform(testData)
accuracy_lr = evaluator.evaluate(prediction_lr)
print("Logistic Regression (with CV) Accuracy: {:.2f}".format(accuracy_lr))
```

```
Logistic Regression (with CV) Accuracy: 0.76
```

Fig.14 .– Logistic Regression with CV evaluation



```
from pyspark.sql import SparkSession
from pyspark.ml.classification import LinearSVC, OneVsRest
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

svm = LinearSVC(maxIter=10, regParam=0.1)

# Instantiate OneVsRest classifier
ovr = OneVsRest(classifier=svm,featuresCol="features", labelCol="labelIndex")

pipeline_svm = Pipeline(stages=[tokenizer, stopWordsRemover, ngram, cv, label_stringIdx, ovr])

# Training
model_svm = pipeline_svm.fit(trainingData)
# Make predictions on the test data
predictions_svm = model_svm.transform(testData)

# Evaluate the model
evaluator3 = MulticlassClassificationEvaluator(labelCol="labelIndex", predictionCol="prediction", metricName="accuracy")
accuracy_svm = evaluator3.evaluate(predictions_svm)
print(f"Test Accuracy: {accuracy_svm:.2f}")

precision_svm = evaluator3.evaluate(predictions_svm, {evaluator3.metricName: "weightedPrecision"})
print(f"SVM Precision: {precision_svm:.2f}")

recall_svm = evaluator3.evaluate(predictions_svm, {evaluator3.metricName: "weightedRecall"})
print(f"SVM Recall: {recall_svm:.2f}")

f1_svm = evaluator3.evaluate(predictions_svm, {evaluator3.metricName: "f1"})
print(f"SVM F1 Score: {f1_svm:.2f}")
```

```
Test Accuracy: 0.76
SVM Precision: 0.76
```

Fig .15. – Support Vector Machine model evaluation

The whole working notebook shall be attached which clearly depicts each of code blocks run on databricks.

VI. Analytical Outcomes

This study's analytical phase utilized Power BI to transform the raw sentiment data into a series of insightful visualizations. The resulting dashboard, a snapshot of which is included as an appendix, offers a granular view of consumer sentiment across a spectrum of food products based on reviews from the dataset. The dashboard displays several key metrics: the distribution of sentiments across reviews, the average polarity of sentiments, and the occurrence of specific named entities within the texts. The sentiment distribution is represented through a decomposition tree, which allows for immediate visual interpretation of the predominant sentiment trends within the

dataset.

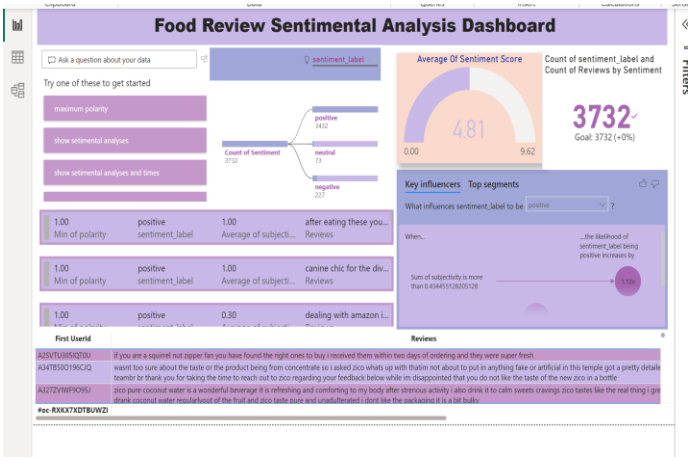


Fig .16. Analysis Dashboard from Power BI

Polarity scores and subjectivity scores, which could indicate the level of objectivity or bias in the reviews. These entities, often pertaining to specific reviews, are critical for understanding the context of sentiments expressed.

The insights gleaned from this analysis could have practical implications for business strategy. By understanding the sentiments expressed in consumer reviews, Companies can more effectively tailor their products and services to meet customer preferences. The pertaining .pbix file where the dashboard built on the report view shall be attached as well.

VII. TEST CASES

In the sentiment analysis project, have adopted a testing approach tailored to the nuances of machine learning and data processing. Our primary focus was on model validation, where we meticulously assessed the accuracy of classifiers like Logistic Regression, Naive Bayes, and SVM on separate test datasets. This was complemented by rigorous cross-validation to ensure model stability and to prevent overfitting, alongside hyperparameter tuning to optimize performance metrics. We also conducted thorough data integrity tests, verifying each preprocessing step and the effectiveness of dataset balancing methods to maintain unbiased training data. Additionally, we scrutinized the models' responses to varied sentiment distributions and different feature extraction techniques, assessing their adaptability and interpretability. This comprehensive testing regime was pivotal in refining our models, ensuring their robustness and applicability in practical sentiment analysis scenarios, thereby guaranteeing their efficacy in deriving meaningful insights from extensive customer feedback data.

VIII. CONCLUSION

In terms of code, the project's successful completion is a testament to the efficacy of leveraging a high-performance analytics platform like Databricks and the versatility of PySpark in processing and analyzing big data for sentiment analysis. The code utilized Spark's MLlib to implement and evaluate multiple machine learning algorithms, ensuring robust model selection and a comprehensive approach to understanding consumer sentiment.

Future Directions: While the current model achieves a respectable 76% accuracy, there are several avenues for enhancement. Future iterations could benefit from advanced machine learning techniques such as deep neural networks and ensemble models for richer feature extraction and more nuanced sentiment detection. Incorporating contextual and linguistic analysis could also refine the accuracy of sentiment classification.

REFERENCES

[1] Ahmed, H. M., Awan, M. J., Khan, N. S., Yasin, A., & Shehzad, H. M. F. (2021). Sentiment Analysis of Online Food Reviews using Big Data Analytics. Ilkogretim Online - Elementary Education Online, 20(2), 827-836. [doi:10.17051/ilkonline.2021.02.93]

[2] Mutanov, G., Karyukin, V., & Mamykova, Z. (2021). Multi-Class Sentiment Analysis of Social Media Data with Machine Learning Algorithms. Computers, Materials & Continua. doi:10.32604/cmc.2021.017827

[3] Salloum, S., Dautov, R., Chen, X. et al. Big data analytics on Apache Spark. Int J Data Sci Anal 1, 145–164 (2016). https://doi.org/10.1007/s41060-016-0027-9

[4] M. Assefi, E. Behravesh, G. Liu and A. P. Tafti, "Big data machine learning using apache spark MLlib," 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 2017, pp. 3492-3498, doi:10.1109/BigData.2017.8258338.

[5] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D. B., Amde, M., Owen, S., Xin, D., Franklin, M. J., Zadeh, R., Zaharia, M., & Talwalkar, A. (2016). MLlib: Machine Learning in Apache Spark. Journal of Machine Learning Research

[6] Databricks. (n.d.). Train a model with Apache Spark MLlib.Retrieved[2023],fromhttps://docs.databricks.com/en/machine-learning/train-model/mllib.html