**Scalable Analytics Final Project Report**

**Group 4**

**Sentiment Analysis Of Yelp Reviews**   
**with**

**Spark Ml And Spark Streaming**

**Team Members**

Subramanian Arumugam

Agash Sekar

Abinesh Ganesan

Hari Priya Avarampalayam Manoharan

Rida Fathima

**h**   
**Spark ML and Spark Streaming**

# **Motivation and Problem Statement:**

Sentiment analysis is becoming increasingly important in today's digital age where customers can easily share their opinions and experiences with others through online reviews, social media platforms, and other digital channels. With the increasing volume of data being generated by customers, it has become more important for businesses to be able to analyze customer sentiment and increase customer satisfaction quickly and accurately. This can help businesses to improve their products or services and increase customer satisfaction.

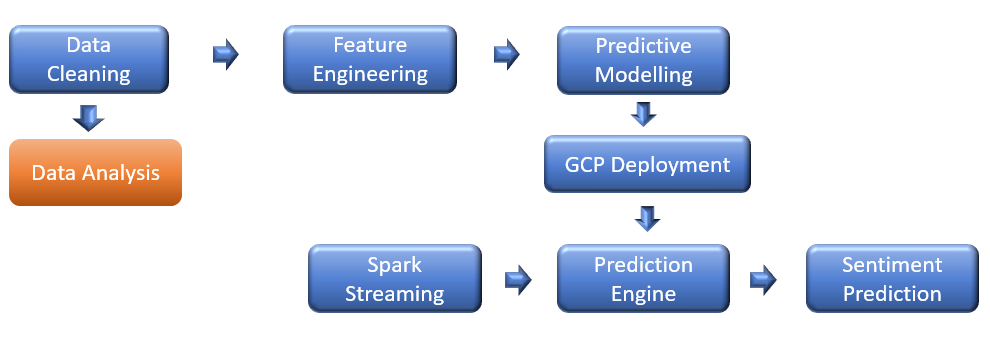
However, the main challenge in sentiment analysis is processing large volumes of unstructured data in real time. Traditional sentiment analysis techniques are not efficient enough to process and analyze large volumes of data in real time. Moreover, traditional techniques are not scalable and require significant computational resources.

# **Proposed Solution:**

To overcome the challenges in sentiment analysis, we propose to develop a sentiment analysis model that can predict the sentiment of reviews in real-time using Spark ML and Spark Streaming, which can analyze large volumes of customer feedback data. The proposed model is developed using natural language processing techniques and machine learning algorithms.

To further enhance the scalability and efficiency of the sentiment analysis model, we propose to deploy the model on a Google Cloud Platform (GCP) cluster. The GCP cluster will provide the computational resources required to process large volumes of customer feedback data in real time.

# **Process Overview:**



The project mainly comprises the following pipelines:

* **Data Visualization Pipeline:**

The data visualization pipeline starts with combining and cleaning the reviews, users, and business datasets on a cluster. The data is then transferred to a local environment using the SCP command. PySpark and Python codes are written on a Jupyter Notebook to generate various plots and visualizations.

* **Predictive Modelling Pipeline:**

The ML pipeline is used to train models and store them in HDFS. Once the model is deployed, PySpark streaming code can be written to make predictions on real-time review data.

* **Spark Streaming Pipeline:**

The streaming pipeline allows for the processing of real-time data. It involves using PySpark streaming code to take input from a streaming source, such as Kafka or Flume, and apply the trained model to make predictions. The output can then be stored in a database or displayed on a dashboard in real time.

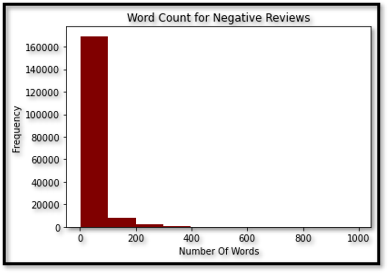
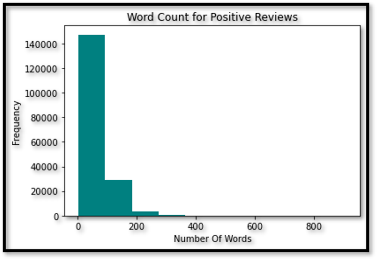
Overall, these three pipelines work together to handle data cleaning, visualization, machine learning, and real-time streaming processing, making it easier to analyze and make predictions on large datasets.

# **Data Visualization:**

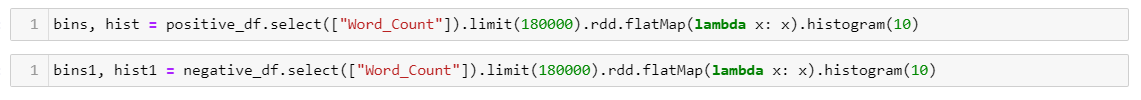
* **Data Preprocessing:**

Data preprocessing for the data visualization pipeline involves the data cleaning of the 3 main csv files: reviews, user, and business. The following points illustrate the data cleaning activities we performed:

* **Yelp Reviews File**: The rows with missing values were first removed and then the unnecessary characters and spaces were removed from the text column. Next, a new “label” column is created based on the ratings column, where a rating less than 4 is considered 0 and all other ratings are considered as 1. Then, the values in text column is left trimmed, to remove leading spaces.
* **Yelp Users :** In the “users” file, we keep only the user\_id and elite columns. The null values and values with ‘None’ strings in the elite column are considered to be “Non-Elite” whereas all the other values are considered as “Elite”.
* **Yelp Business:** In the “business” file, we select the business\_id, state, and categories column.
* **Preprocessed Data:** The Reviews data is joined with users data based on the user\_id and the resultant data frame is joined with the Business data frame based on the business\_id. The prepared data frame is written to HDFS as a CSV file (“PreProcessedDataSet”). This file is later transferred to the local computer where we perform various visual analysis in a Jupyter notebook.
* **Word Count Frequency Analysis:**



The above exploratory data analysis reveals that the word count in reviews differs between positive and negative sentiments. Positive reviews tend to have a higher word count, particularly exceeding 100 words, compared to negative reviews. This finding suggests that customers who have a positive experience are more likely to share a detailed review compared to those who have a negative experience.



This PySpark code calculates the histogram of the "Wordcount" column in the "positive\_df" and “negative\_df” data frames. The histogram function is applied to an RDD (Resilient Distributed Dataset) of the "Word\_Count" column, which is created by calling the "flat map" function on the "positive\_df" and “negative\_df” data frames.

From a business perspective, this insight highlights the importance of providing a positive experience to customers. Satisfied customers are more likely to write detailed positive reviews, which can attract new customers and boost the company's reputation. Therefore, companies should strive to provide an exceptional customer experience to increase the chances of receiving positive reviews.

* **Sentiment Analysis of Yelp Reviews using NLTK library:**

This section of the analysis focuses on performing sentiment analysis of Yelp reviews using the VADER (Valence Aware Dictionary and Sentiment Reasoner) sentiment analysis tool from the Natural Language Toolkit (NLTK). To achieve this, a user-defined function (UDF) is created to apply the VADER analyzer to each text in the reviews and return a compound score that represents the overall sentiment of the text. The UDF is then used to create a new column called 'sentiment' in the Data Frame based on the 'text' column. To visualize the sentiment scores of the reviews, a histogram plot is generated. The results of this analysis can provide insights into the overall sentiment of the reviews and identify factors that contribute to positive or negative sentiment.

Chart, histogram

Description automatically generated with medium confidenceThe sentiment intensity scores' histogram plot reveals that the distribution of scores is skewed towards the positive side, with most of the reviews having a compound sentiment score ranging from 0.5 to 1.0. This finding suggests that most of the reviews on Yelp have a positive sentiment, while only a small portion of the reviews express a negative sentiment.

Moreover, the plot shows that a considerable number of reviews have a compound score of 0, indicating a neutral sentiment. This outcome might be due to reviews with unclear sentiments or ambiguous language that is difficult to interpret. *Overall, this sentiment analysis provides an overview of the overall sentiment of Yelp reviews, where most of the reviews express positive sentiment.*

* **Comparative Analysis of Elite Users and Non-Elite Users:**

This section of the analysis examines the correlation between elite status and star ratings on Yelp. The Elite Squad on Yelp comprises users who are acknowledged for their contributions to the platform, and in this study, we compare the star ratings provided by elite users and non-elite users. The findings reveal that elite users generally give higher star ratings than non-elite users. To visually represent the results, four graphs were generated. The first graph displays the number of elite and non-elite users, while the second graph illustrates the distribution of stars for the entire Yelp review dataset. The last two graphs show the distribution of stars by user type.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

The bar graph shows the distribution of star ratings for all users, elite users, and non-elite users on Yelp. Some useful inferences that can be drawn from this graph are:

Most Yelp reviews (about 57.9%) have a 5-star rating. This suggests that most Yelp users are satisfied with the businesses they review.

* *Elite users tend to give higher star ratings than non-elite users*. For example, 80.5% of the reviews written by elite users have a 4 or 5-star rating, while only 66.6% of the reviews written by non-elite users have a 4 or 5-star rating.
* *Non-elite users are more likely to give 1 or 2-star ratings compared to elite users*. For example, 35.8% of the reviews written by non-elite users have a 1 or 2-star rating, while only 15.3% of the reviews written by elite users have a 1 or 2-star rating.
* The number of elite users on Yelp is relatively small (about 4.6% of all users), but they contribute a significant number of reviews (about 8.2% of all reviews). *This suggests that elite users are more engaged and active in the Yelp community compar ed to non-elite users.*
* **Businesses with Top Reviews**

Chart, funnel chart

Description automatically generatedChart, funnel chart

Description automatically generated

* This bar chart shows the Top ten Businesses Categories with the most positive reviews on the left and Top ten Businesses with most negative reviews on the right.
* Restaurants bags the first place in both Top positive (800k+ positive reviews) and negative (400k+ negative reviews) reviews section. This implies that users of yelp utilize this platform to review restaurants more than any other businesses.
* However, the business category “Hotels&Travels” appears only in the negative reviews section. This indicates that customers are generally dissatisfied with this category and have more expectations and standards to satisfy their needs.
* **Word Cloud Analysis**

A word cloud in sentiment analysis is a visual representation of the most frequently occurring words within a text or a set of texts, with the size of each word proportional to its frequency. In sentiment analysis, the word cloud can be used to identify the most common positive, negative, or neutral sentiment words, allowing analysts to gain insights into the overall sentiment of the text. The word cloud can also be used to identify trends, patterns, and themes within the text.

A picture containing text

Description automatically generatedBased on the common words mentioned in the word cloud from 50000 records, it can be inferred that the sentiment of the text is generally positive towards a place, possibly a restaurant or a cafe, and the experience of the food and service provided. The frequent occurrence of words such as "great", "good", and "food" indicates a positive sentiment towards the quality of the food. The word "service" also suggests that the sentiment is positive towards the customer service provided at the place. Overall, it seems that the sentiment of the text is positive and favorable towards the place in question.

# **Sentiment Prediction using Spark-ML**

The Sentiment Prediction using Spark ML can be summarized using the 4 sub-modules listed below:

* Data Preprocessing
* Vectorization
* Feature Engineering using tf-IDF
* Model Training using GCP cluster.
* **Data Preprocessing**

Data preprocessing for sentiment analysis involves cleaning and transforming the text data by removing irrelevant information, normalizing the text, and converting it to a numerical format to prepare it for analysis. Data Preprocessing involves four steps.

* **Removing null values**: This involves identifying any missing or null data points in the dataset and either removing them or filling them in with appropriate values.
* **Removing unnecessary characters:** This involves removing any punctuation and extra spaces from the text data in the dataset using the “regex” library in python, which can sometimes interfere with natural language processing and machine learning algorithms. (ref.: “remove\_punct” and “remove\_spaces” functions in our python scripts)
* **Converting rating to positive or negative label:** This involves assigning a binary label to the rating data based on a specific threshold. In this case, any star rating less than 4 is considered negative and any rating greater than or equal to 4 is considered positive. (ref.: convert\_rating function in our python scripts)
* **Vectorization**

Vectorization in sentiment analysis is the process of converting text data into a numerical format that can be understood by machine learning algorithms, by assigning numerical values to each word or phrase in the text data and representing them as vectors in a high-dimensional space, which can then be used for analysis and modeling.

A picture containing graphical user interface

Description automatically generated This allows for the application of various machine learning algorithms to accurately classify and predict the sentiment of text data. We have used the “CountVectorizer” from pyspark library to perform vectorization. Once the vectorization model is fit using “fit and transform methods”, we will save the fitted object onto **HDFS:**

* **Feature Engineering using TF-IDF**

A screenshot of a computer

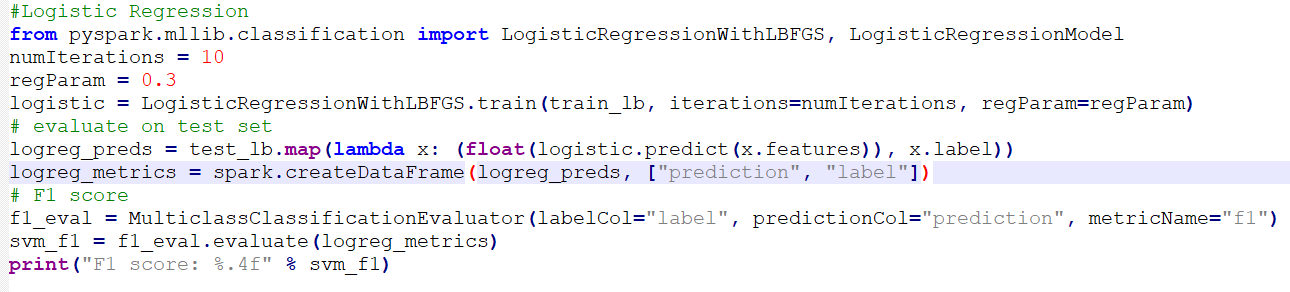
Description automatically generated with medium confidenceTF-IDF (Term Frequency-Inverse Document Frequency) is a technique used in text mining and NLP to represent text data in a numerical format. It assigns weights to each word in the text based on its frequency within a document and across the entire corpus of documents. Words with higher weights are considered more important and relevant to the sentiment of the text. This allows for more accurate and effective sentiment analysis, as well as other text analysis tasks such as information retrieval and text classification. We have used the IDF module from the pyspark library to implement the same. Once the IDF object is fit onto our data, we will store the final model onto **HDFS** as shown in the figure.

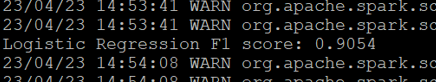
* **Training ML Models using GCP Cluster**

Three models including naive bayes, logistic regression, and SVM were implemented on the Google Cloud Platform Cluster using SparkML for sentiment analysis, and F1-score was used as a metric to evaluate their overall accuracy. The data was divided into a training set and a testing set with a split ratio of 80:20. The results showed that both logistic regression and SVM outperformed the naive bayes model, indicating their superior ability to classify and predict the sentiment of text data.

This suggests that logistic regression and SVM models may be more suitable for practical applications of sentiment analysis. Regularization parameter (regParam) is used in machine learning algorithms to prevent overfitting of the model to the training data by adding a penalty term to the objective function being optimized. The number of iterations in the algorithm (numIterations) determines how many times the entire training dataset is processed during the optimization process.

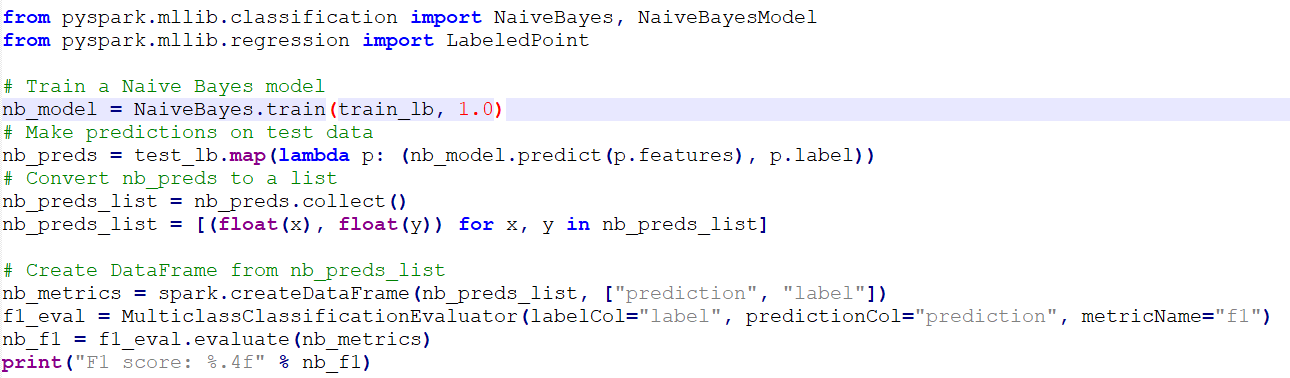
1. **Logistic Regression:**

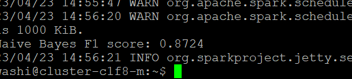
Logistic regression model is effective for sentiment analysis using Spark due to its ability to handle large datasets, its efficient computation, and its ability to provide probabilistic outputs for classification.



1. **Naïve Bayes:**

Naive Bayes model is suitable for sentiment analysis using Spark due to its simplicity, scalability, and ability to handle high-dimensional data with relatively less computational resources.





**SVM model:**

We implemented the SVM model in the pyspark ML library and found the best parameters for our training data with number of iterations as 50 and Regularization Parameter as 0.3. Once the model was trained, we saved the final model object onto HDFS.

Fig 1
***Implementation:***

Text

Description automatically generated

Graphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generatedTo implement the Spark Streaming pipeline, we chose the SVM model for its fast prediction capabilities. Once the SVM model was trained, we saved the model object onto HDFS by using the ‘save’ method as shown.

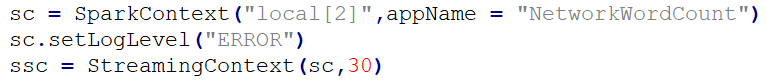
We experimented with three models and have tabulated the F1 Scores as shown. We implemented the Spark Streaming Pipeline using the SVM model as it is advantageous while dealing with high dimensional data sets. The screenshot of the cluster shows us the objects available in the HDFS. We have svm\_model which the trained ML model and the other objects required for tokenization, vectorization, and IDF.

# **Spark Streaming Pipeline**

In this project, we successfully set up a spark streaming pipeline where we were able to obtain real-time predictions of sample review data. A Spark Streaming context was established and input data streams from the terminal were used as input to the already trained ML models and the output of the ML model was displayed as output on another terminal window.

The Spark Streaming Pipeline can be summarized using the following steps:

* Streaming Data Input
* Data Cleaning
* Tokenization, Vectorization, tf-IDF using the saved model objects in HDFS.
* Model Prediction using the saved ML model in HDFS.
* **Streaming Data Input**

The streaming data input is obtained by establishing a Spark Streaming Context within a Spark Session which is carried out as follows:

Here, we set the timer to 30 seconds. We will be scanning the input port every 30 seconds.

Next, we need to set up a Discretized Stream by reading input text data from a TCP/IP socket.

“socketTextStream’ is a method in Apache Spark's Streaming API that helps us to create a Discretized Stream as shown below:

A screenshot of a computer

Description automatically generated with low confidenceHere, we are listening to the port with port number 65395. The lines object is a socket stream object and is close to an RDD object.

To perform predictions on the input text from a user, we need to convert this RDD object to a spark data frame which will be later consumed by the predictive model. The next step is to perform data frame like operations on the lines object for which we use the “foreachRDD” method as shown below:

Text

Description automatically generated with low confidenceHere, “get\_prediction” is the function we have defined to perform the data frame operations on the received user input/ RDD. In the “get\_prediction” function, we are first converting the RDD to a data frame using “createDataFrame” function in spark as shown below:

Text, letter

Description automatically generatedHere, we use a “map” function on the tweet\_text object to convert each item in the RDD as a Row Object with column name “text”. Then we utilize the “createDataFrame” method on the rowRdd object to create the new test data frame object called “df”.

Next, we create a dummy column called "label” on the data frame with values 0 to pass to the spark ML model. Then, we perform the “remove\_punct” function mentioned above on the ‘text” column to remove unwanted punctuation. Next we define a UDF (user defined function) with “remove\_punct” function and name the UDF as “punct\_remover”. Then, we apply this UDF on our data frame as follows:

Text, letter

Description automatically generatedAfter applying the UDF on the “text” column, we later obtain the same “text” column after renaming the column. Then we trim the leading spaces in the text column using the “ltrim” function. Next, we load the saved Tokenizer object in **HDFS** and perform tokenization on our text column as follows: Graphical user interface, text, application

Description automatically generated

Text, letter

Description automatically generatedWe also remove the Stop Words from the tokenized column as shown above.

Next, we perform Count Vectorization and perform tf-IDF calculations using the saved objects in **HDFS** as shown. Finally, we only select the “tfidf” and “label” columns to obtain the predictions. Next, we load the saved ML model from **HDFS** and **Graphical user interface, text, application, Word

Description automatically generated**use the “predict” method to predict on the data frame as shown below.

The final prediction for the input is printed out as Positive or Negative based on the predicted value. Now, the function “get\_prediction” returns the output to the main code.

Text

Description automatically generated with medium confidenceNext, we need to start the streaming context using “start” method. The “awaitTermination” method waits for termination. We need to run this entire python script using spark-submit in the terminal.

**Model Output:**

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# **Conclusion**

Our team was able to effectively apply natural language processing and machine learning techniques to analyze the Yelp data set using the GCP cluster. Through exploratory data analysis, we gained valuable insights on the differences between elite and non-elite users. Additionally, we developed three machine learning models using the pyspark ML library, achieving a high accuracy rate of 90% on the validation data set. Finally, we established a Spark Streaming pipeline to predict sentiment in real-time as reviews were submitted. Overall, our project successfully demonstrated the power of NLP and ML in processing and analyzing large data sets.