**National University of Computer and Emerging Sciences**



Fundamentals of Big Data Analytics

**Project Report**

29th April 2024

Fundamentals of Big Data Analytics

Spring 2024

Department of Computer Science

FAST-NU, Lahore, Pakistan

**Overview**

This project involves a comprehensive pipeline for sentiment analysis using Apache Spark. It demonstrates data preprocessing, sentiment classification using logistic regression, and optimization techniques involving gradient descent methods. The key components covered in this report include:

Data Preprocessing

Sentiment Analysis

Logistic Regression with Spark

Confusion Matrix and Model Evaluation

K-Means Clustering and K-Nearest Neighbors

Gradient Descent for Logistic Regression Optimization

**1. Data Preprocessing**

Data preprocessing is critical for ensuring data quality and preparing it for analysis. In this project, Apache Spark is used to preprocess a large dataset, focusing on cleaning, transforming, and tokenizing text data:

**Loading the Dataset:** The dataset is loaded from a CSV file into a Spark DataFrame.

**Cleaning the Data:** Removing Retweets: Any text starting with "RT" or '"RT' is filtered out to remove retweets.

**Dropping Duplicates:** Duplicates are removed based on unique tweet IDs.

**Filtering Relevant Data:** Only rows with "relevant\_yn" marked as "yes" are retained.

**Tokenization:** The Tokenizer class splits text into words, preparing it for feature extraction.

**Stop Words Removal:** Common English stop words are removed using StopWordsRemover.

**Text Cleanup:** Emoji Replacement: The emoji package replaces emojis with text descriptions.

**URL Removal:** URLs are removed from text to clean the data.

**Special Character Removal:** Special characters are removed, leaving only alphabetic characters and spaces.

**2. Sentiment Analysis**

Sentiment analysis aims to determine the emotional tone of text. This project uses the TextBlob library to perform sentiment analysis and classify text as "Positive," "Neutral," or "Negative":

**Sentiment Labeling:** A User Defined Function (UDF) is created to assign sentiment labels based on TextBlob's polarity score.

This UDF is applied to the cleaned text, resulting in a new "sentiment\_label" column in the DataFrame.

**3. Logistic Regression with Spark**

Logistic regression is used to classify text into categorical sentiment labels. Here's the workflow for implementing logistic regression in Spark:

**Pipeline Creation:** A StringIndexer converts categorical sentiment labels into numeric values.

CountVectorizer is used to transform words into numerical feature vectors (bag-of-words approach).

A Pipeline object is used to encapsulate these transformations and the logistic regression model.

**Model Training:** The data is split into training and test sets (80% for training, 20% for testing).

The pipeline is fitted on the training set, creating a logistic regression model.

**4. Confusion Matrix and Model Evaluation**

After training, the model is evaluated to determine its accuracy and other performance metrics:

**Making Predictions:** The model is applied to the test set to predict sentiment.

**Calculating Accuracy:** A MulticlassClassificationEvaluator calculates the accuracy of predictions against actual labels.

**Confusion Matrix:** MulticlassMetrics is used to create a confusion matrix, which visualizes the accuracy of predictions.

A heatmap is generated to display the confusion matrix, showing how often predicted sentiments match the actual sentiments.

**5. K-Means Clustering**

K-Means clustering is employed to identify natural groupings within the preprocessed data. Initially, the necessary columns are selected for K-Means, typically involving features that exhibit clustering patterns. The data is then split into training and testing sets, with the majority allocated for training. In this implementation, a subset of the training data (1500 samples) is utilized due to computational constraints. Next, the K-Means algorithm is applied, specifying the desired number of clusters (k=2 in this case). Through iterative optimization, cluster centers are determined, representing the centroids of each cluster. To visualize the clustering results, a scatter plot is generated, with data points colored according to their assigned cluster. Additionally, cluster centers are marked on the plot to provide insights into the distribution of data points across clusters.

**6. K-Nearest Neighbors**

The K-Nearest Neighbors (KNN) algorithm is employed to find the nearest neighbors for data points based on their feature vectors. Initially, feature vectors are assembled from the preprocessed data, preparing them for distance calculations. Euclidean distance is then computed between data points to quantify their similarity or dissimilarity. The parameter k, representing the number of neighbors to consider, is specified (set to 5 in this example). Nearest neighbors are identified by sorting data points based on their calculated distances and selecting the top k neighbors. In this implementation, if the 'predicted label' column exists in the Data Frame containing the nearest neighbors, it is selected for further analysis. Otherwise, appropriate adjustments are suggested to ensure the inclusion of predicted labels for subsequent steps, such as model evaluation or visualization.

**7. Gradient Descent for Logistic Regression Optimization**

Gradient descent is an optimization technique to refine model parameters, aiming to minimize a loss function. This project implements both Batch Gradient Descent and Stochastic Gradient Descent for logistic regression:

**Logistic Regression Function:** A custom function computes logistic regression using the sigmoid function to convert features to a binary outcome.

**Batch Gradient Descent:** This method updates model weights by averaging gradients across the entire dataset.

It provides stable convergence as it uses the entire dataset to update the weights.

**Stochastic Gradient Descent:** This method updates weights incrementally using individual data points or mini-batches.

It often converges faster than batch gradient descent but may exhibit more oscillations.

**Conclusion**

This project demonstrates the complete pipeline for sentiment analysis, from data preprocessing to model evaluation and optimization. It shows the versatility of Apache Spark in handling large-scale data processing and logistic regression. The implementation of gradient descent methods allows for fine-tuning the logistic regression model, potentially improving accuracy and convergence speed. Further improvements could involve hyperparameter tuning, advanced feature engineering, or exploring different gradient descent variations.