Sentiment Analysis on US 2020 Elections using Twitter Data on Apache Spark

Abel Johny
Student number: 20595153
Email: psxaj3@nottingham.ac.uk

Mathew Aniyan
Student number: 20194326
Email: psxma18@nottingham.ac.uk

Jacob Aniyan
Student number: 20164772
Email: psxja14@nottingham.ac.uk

Jui Ting Tsai
Student number: 20538358
Email: psxjt6@nottingham.ac.uk

Brendan Ezekiel Agnelo Vaz
Student number: 20610206
Email: psxbv1@nottingham.ac.uk

Kwabena Asafo-Adjei
Student number: 20610776
Email: psxka11@nottingham.ac.uk

***Abstract*—This study examines the influence of social media sentiment on political outcomes, particularly during the US 2020 elections, using Twitter data analyzed through the Apache Spark framework. We investigated how the sentiments expressed on social media impact election results and public opinion regarding political events. Utilizing a dataset comprising of approximately 1.8M tweets, our research employed advanced machine learning models and natural language processing techniques to perform sentiment analysis. We specifically explored the effectiveness of Logistic Regression, Naive Bayes, and K-Means clustering in analyzing the data, with Logistic Regression providing the most accurate results. Our findings highlight the significant role of Twitter in shaping political discourse and outcomes, demonstrating that sentiments on social media platforms can correlate with political events. The study emphasises the potential of using sophisticated big data analytics to understand the dynamics of digital communication and its impact on politics. This research not only contributes to academic discussions but also aids policymakers and political strategists in harnessing the power of social media insights.**

I. Introduction and Background

The development of sentiment analysis techniques for tweets, especially in the context of political analysis, has evolved significantly over the years, leveraging both Machine Learning and Natural Language Processing (NLP) methodologies. Initially, sentiment analysis focused on basic text mining and classification algorithms to categorize opinions into positive or negative sentiments. Techniques such as Naive Bayes, Logistic Regression, and Support Vector Machines were commonly employed to analyze textual data from social media platforms like Twitter, aiming to extract public opinions towards political entities or events [1]. As the field progressed, the complexity and volume of data on platforms like Twitter, with its rapid expansion and the vast number of active users, necessitated more advanced approaches. Deep learning methodologies, including recurrent neural networks (RNNs) and convolutional neural networks (CNN), began to be applied, achieving higher success rates in sentiment classification. These models were trained on large datasets of tweets, significantly improving the accuracy of sentiment analysis [2]. Moreover, the analysis of sentiment in tweets has expanded to include not just binary classifications (positive or negative) but also the detection of political bias and identification of specific political ideologies.

Advanced pre-processing techniques and a variety of classification algorithms have been utilized to predict political leanings from tweets, demonstrating the versatility of sentiment analysis in political discourse [3][4]. The inclusion of deep learning models, such as CNN-LSTM combinations, further exemplifies the advancements in handling more complex linguistic features and achieving higher accuracy in sentiment classification, even in languages with intricate structures like Persian [5]. Additionally, sentiment analysis has been enriched by hybrid methods that combine lexicon-based approaches with machine learning, offering nuanced insights into public opinions on political matters [6]. This evolution reflects a broader trend towards more sophisticated, context-aware systems capable of understanding the subtleties of human language and sentiment, including the challenges posed by sarcasm and varying emotional expressions [7]. In summary, the development of sentiment analysis techniques for political analysis on Twitter has transitioned from simple classification algorithms to complex models that incorporate deep learning and hybrid methods, reflecting the growing sophistication of NLP and machine learning technologies [8].

Our research focuses on exploring the influence of social media sentiment on political outcomes. Specifically, we aim to determine how sentiments expressed on social media platforms can affect election results and public opinion regarding political events.

II. Methodology

Fig. 1 illustrates a high-level overview of the solution pipeline for sentiment analysis of political tweets. Our methodology involves four primary steps: Data Loading and caching, Data Handling and Visualization, Model Development and Fine-Tuning, and Model Evaluation. The next few subsections explain each of these steps in detail.

From analysing the two datasets provided, we can identify that there are 22 columns of features in each of them. The features are categorised into 14 string columns, 5 numeric columns (float and double), and 3 timestamp columns. The

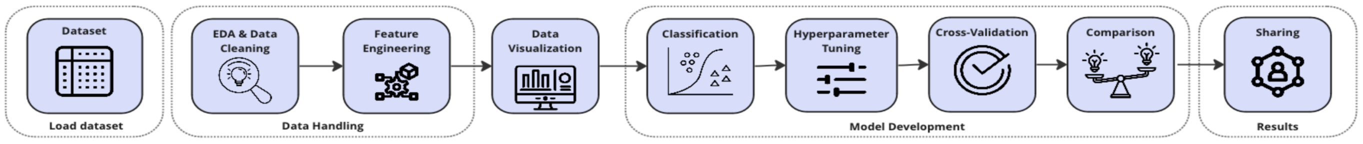

Fig. 1. Big Data Machine Learning Pipeline

datasets together contain a total of 1,865,246 entries and are unlabelled. As our task focuses on understanding if a correlation exists between what's going on in twitter and the actual results in the US elections, our analysis narrows to tweets originating from United States. An overview of tweets matching regex patterns for United States highlight the prevalent occurrence of values 'United States of America' and 'United States'. Scrutinizing the dataset for anomalies, we identified 10 records with invalid timestamps. Moreover, certain features deemed irrelevant to the primary task of sentiment analysis, such as 'user_name', 'user_screen_name', 'user_description', 'country', and 'continent', were observed. Additionally, numerous features exhibit NULL values, indicating missing data points. However, it would be pertinent to drop records with missing data points in features relevant for model training or feature column identification.

A. Data Pre-Processing

In order to facilitate the development of machine learning models, it was imperative to ensure that the data was comprehensive, devoid of inconsistencies, and formatted to meet the requirements of model APIs. As both datasets were identical feature-wise, a union operation is done to merge them and a new column, *candidate*, added to identify which dataset each record originally belonged to. Pre-processing starts by dropping records that contain NULL values in pertinent features for model training, such as, 'tweet_id' and 'tweet', using the pyspark.sql.DataFrame.dropna function. Next, tweets originating from the United States are filtered using the pyspark.sql.DataFrame.where function. Features deemed irrelevant or containing anomalies, as highlighted in EDA, are then eliminated utilizing the pyspark.sql.DataFrame.drop function. Finally, as user engagement has been shown to be influenced heavily by likes [9], records lacking a predefined number of likes, set at 10, are discarded. A custom transformer, *CleanTweet*, is created to amalgamate the pre-processing steps mentioned and for integrating into a 'pyspark.ml.Pipeline' pipeline. Step 1 of Fig. 2 depicts a summary of this stage showing the features transformed to an array of processed words and labels L.

B. Model Data Preparation and Evaluation metrics

Following Exploratory Data Analysis and Pre-Processing, we focus on preparing the data for training our machine learning models. Given the experimentation with both supervised and unsupervised models, it becomes necessary to label the dataset for supervised learning. To accomplish this, a Spark User Defined Function (UDF) is registered, employing a rule-based algorithm provided by the Textblob Python library to classify the polarity/sentiment of tweets. A custom transformer, *polarityCalculator*, is created for applying the UDF to label each record in the dataset. Next, the features are transformed into a vector representation using pyspark.ml.feature.CountVectorizer followed by computing the Inverse Document Frequency on this vector using pyspark.mllib.feature.IDF. For model evaluation, label-encoding is performed on the polarity feature using pyspark.ml.feature.StringIndexer to transform them into numerical labels that our models can understand and compare against the predicted label. A 70-30 train-test split is also applied beforehand for evaluating the model's performance on unseen data. Step 2 of Fig. 2 depicts a summary of this stage showing the features transformed into a sparse vector with encoded labels L'.

The next few sections explain the models explored with estimated measures of the models performance, evaluated using 10-fold cross-validation. We fixed a value of 10 for cross-validation as this value has been empirically shown to yield error rates that do not suffer from high bias and high variance for large datasets [10]. The performance of all classifiers were gauged with pyspark.ml.evaluation.MulticlassClassificationEvaluator to provide metrics such as accuracy, F1 score, Weighted Precision, and Weighted Recall.

- Supervised learning with Logistic Regression:

The initial model implemented was a logistic regression model. Default model hyperparameters were used with the *maxIter* set to 100, allowing ample number of iterations for the model to converge towards an optimal solution. The model's *maxIter* and *regParam* hyperparameters are then fine-tuned using cross-validation. Fig. 3 illustrates the coefficient matrix derived from the tuned model.

- Supervised learning with Naive Bayes:

The Naive Bayes algorithm is based on the concept of Bayes' theorem, which calculates the probability of a label given some observed features The "naive" assumption implies that features are independent and have the same importance given the class label. Mathematically, the Naive Bayes classifier calculates posterior probability of a class A given observed features A using Bayes' theorem:

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

In the context of sentiment analysis, the PySpark Multinomial pyspark.ml.classification.NaiveBayes algorithm

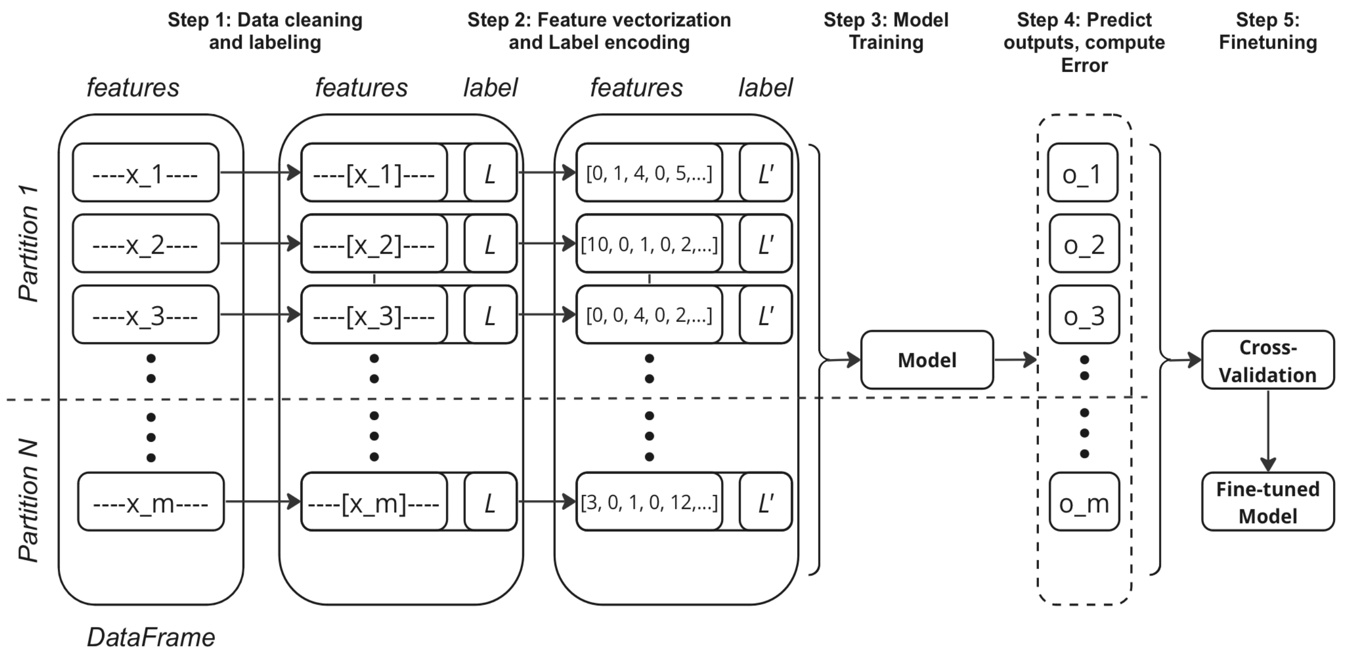

Fig. 2. Workflow of a global design for sentiment analysis with Spark DataFrames.

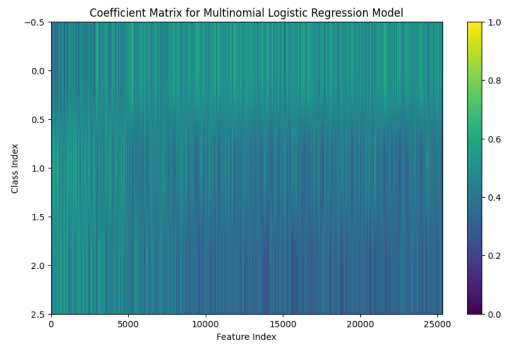

Fig. 3. Coefficient Matrix for Multinomial Logistic Regression Model

is implemented to classify tweets into positive, negative, or neutral sentiments based on a sparse vector of stop words obtained after transforming the *tweet* feature with pyspark.ml.feature.CountVectorizer and pyspark.mllib.feature.IDF. The model's *smoothing* hyperparameter is then fine-tuned using cross-validation.

- Unsupervised learning with K-Means:

The *k*-means algorithm, an unsupervised learning technique, discerns the inherent patterns within a dataset by clustering the data based on their similarities. In this analysis, the PySpark implementation, specifically 'pyspark.ml.clustering.KMeans', is employed. Initially, a cluster configuration of 3 is tested to categorize sentiments as Positive, Negative, and Neutral. Utilizing the previously outlined pipeline in conjunction with the KMeans algorithm, this configuration, yielded subpar results, achieving only a 38.2% accuracy rate. To investigate whether this outcome is attributed to the cluster size, a configuration with 2 clusters is examined, with Neutral sentiment tweets filtered out. This adjustment resulted in a decrease in accuracy, plummeting to 35.78%.

C. Correlation analysis

Correlation analysis is performed in order to explore whether there is a relationship between twitter sentiment and election results. This is achieved by grouping the predictions data frame by candidate and state, and then aggregating the sum of sentiment to produce a new data frame with a column that represents the total positive sentiment for each candidate in each state. This is then merged with another dataframe, which contain features such as the total votes for each candidate in each state, using a Join operation on the state column.The new dataframe containing total sentiment and total votes for each candidate for the individual states, is then further split into two; one for each candidate. For each of the dataframes for the respective candidates, Pearson correlation coefficient is calculated between the total positive sentiment and the total votes variables. Additionally a regression plot is produced for each of the candidates in order to visualise the relationship between total sentiment and total votes.

D. Word Cloud

After tokenizing using pyspark.ml.feature.RegexTokenizer and cleaning the data with the *CleanTweet* transformer to obtain the sparse vector, the dataset is grouped by the vector and its associated sentiment. Next, the tokens in the vector are ranked by the count of positive sentiment to obtain the most important keywords. This is then visualized using the WordCloud library. The size and position of the terms determine its significance; larger and centralized words are most relevant, whereas the surrounding words are related subjects.

III. Experimental set-up

A. Apache Spark Configuration

Our Apache Spark (version 3.5.1) configuration uses 1 to 12 cores and is run on local mode, which can still implement the solution with parallelism. However, the degree of parallelism is limited by the number of CPU cores available on the machine. The machine used to carry out the experiment has 16 GB of memory. The master configuration is set to ’local [*]’. For our experiment, we varied the number of CPU cores and observed differences in runtime. We used the pyspark.sql module and pyspark machine learning library (MLib) in our implementation aswell as the scikit-learn library for preprocessing.

B. Dataset

The dataset was provided by Manch Hui as part of a Kaggle competition [12]. A second dataset, providing data on the US 2020 election result, was used for correlation analysis [11]. Tweets were collected using the Twitter API and the timeframe covers the months leading up to the election, specifically, October 15, 2020 to November 8, 2020. The datasets are in two CSV files that were read into PySpark. A union is then performed on them in a custom extract function to merge them together. All entries were then input into a random train and a test split of 70% and 30%, respectively. Both of the split dataframes were cached in memory for performance. These were then input into the pipelines’ ‘fit’ and ‘transform’ functions as outlined in the Methodology.

C. Experimental Design

The aim of the experimental design is to investigate whether our big data solution to this Twitter sentiment analysis is scalable. For this, we used the scalability metrics speed-up and scale-up. These metrics ensure that our solution would be able to handle increasingly large datasets without producing too much overhead [13].

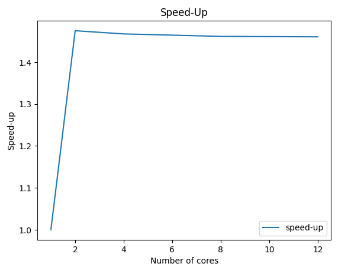

Fig. 4. Speed-up plot according to runtimes

Speed-Up relates to runtime of the solution with respect to the number of cores i.e. how much faster can data be processed with n cores instead of one [13]. The input data is controlled while n is varied and ideally there should be a linear relationship but factors such as network overhead will slow runtime. Fig. 4 displays the speed-up obtained on our dataset with a range of 1, 2, 4, 8, and 12 cores explored.

$$\text{Speed-up}(n) = \frac{\text{runtime in 1 core}}{\text{runtime in } n \text{ cores}}$$

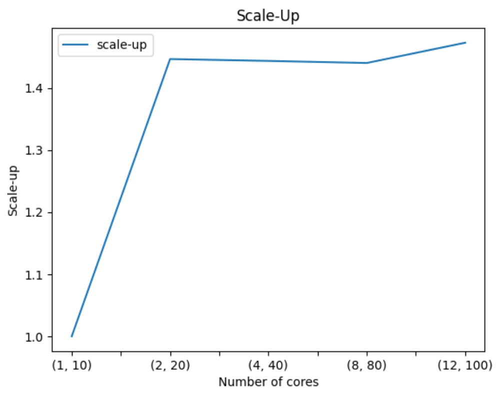

Fig. 5. Scale-Up plot according to runtimes

Scale-Up measures the ability of a system to process data in 1 core versus n-times larger dataset on n cores [13].

$$\text{Scale-up}(n) = \frac{\text{runtime to process } \textit{data} \text{ in 1 core}}{\text{runtime to process } n\ \textit{data} \text{ in } n \text{ cores}}$$

Fig. 5 shows the scale-up obtained in our dataset with a range of 1, 2, 4, 8, and 12 cores explored the number of CPU cores and 10, 20, 40, 80, and 100 percent of the dataset explored.

D. Performance Metrics

For the performance metrics, we imported ‘MulticlassClassificationEvaluator’ from ‘pyspark.ml.evaluation’. The metrics were accuracy, F1-score, precision, and recall. Since we used unsupervised learning with K-means clustering, and these metrics are not typically used to evaluate its performance, we opted to find only its accuracy as a measure. Usually, this would not be possible as K-means clustering does not have access to ground truth labels to compare its predictions with, but since we made use of the TextBlob library to assign sentiment and produce labels independently, we decided to compare accuracy with these labels.

Accuracy is the ratio between the number of correct predictions to the total number of predictions made. However, when classes are imbalanced, accuracy is not as appropriate, since the majority class can be predicted more frequently, leading to an artificially high accuracy measure. Therefore, we also used Precision, Recall and F1-score to account for imbalance and give a better picture of how well the model did.

Precision gives the proportion of positive identification of a particular class (in our case positive, neutral, and negative) that were correct and is calculated as

TABLE I
EVALUATION METRICS

Model	Accuracy	F_1 Score	Precision	Recall
K means	38.20%	-	-	-
Logistic regression	79.41%	78.95%	79.38%	79.41%
Naive Bayes	64.41%	64.31%	71.72%	64.41%

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Where TP = True Positives and FP = False Positives

Recall, on the other-hand, gives the proportion of actual positive values in a class that were identified correctly and is calculated as

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Where FN = False Negatives

Both precision and recall must be evaluated as typically, improving one tends to decreases the other. F_1 score strikes a balance in measure of precision and recall and is calculated as

$$F_1 \text{ Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

E. Validation Procedure

As discussed in the methodology, we performed a 10-fold cross-validation assess the performance of the model and aid in hyperparameter tuning. The data is split into 10 subsets (or folds), each of which is iterated as the test set. We constructed parameter grid using class 'ParamGridBuilder'. The model's maxIter and regParam hyperparameters are then fine-tuned using cross-validation exploring a parameter grid spanning values of [100, 200, 300, 500, 800] for maxIter and [0.1, 0.01] for regParam respectively. The combination that produced the best performance metrics were selected for. The smoothing parameter for Naive Bayes was also tested for values of 0.0, 0.2, 0.4, 0.6, 0.8 and 1.

IV. RESULTS AND DISCUSSION

The logistic regression model outperformed the other models, having the best performance on all of the metrics (Table 1), with K means performing the worst. This may be due to the fact that logistic regression models are more robust to noise in the data features compared to K means and Naive Bayes. Naive Bayes, for example assumes features independence and is sensitive to data quality, this may not have been an appropriate assumption for the data set. Additionally, K means may have performed the most poorly due to its sensitivity to initial cluster centroids, outliers and the assumption that clusters follow a spherical shape with relatively similar size.

The results from the correlation analysis shows a Pearson correlation coefficient of 0.55 between the variables total positive sentiment and total votes won for candidate Trump. For candidate Biden, the correlation coefficient between the same variables was higher, reaching 0.69, indicating a stronger positive relationship between sentiment and votes won than for Trump on the same variables. p-values were also calculated for both correlation coefficients. Both were found to be under the standard significant level of 0.05, showing the correlation can be said to be statistically significant. Overall, this suggests a positive linear correlation with the total positive sentiment of Twitter data to total votes won, and by election results. A visual representation of the correlation analysis can be seen in Fig. 6 and 7.

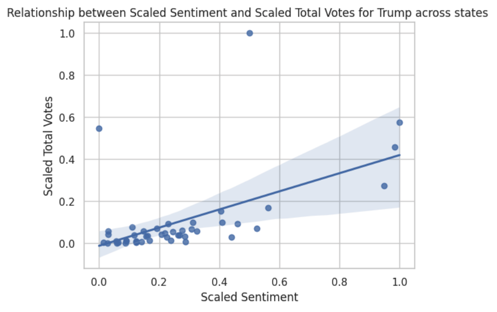

Fig. 6. Correlation between sentiment and votes for Trump

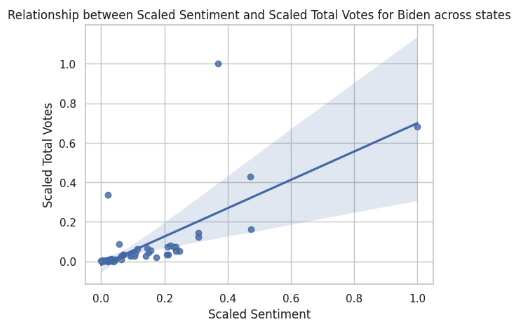

Fig. 7. Correlation between sentiment and votes for Biden

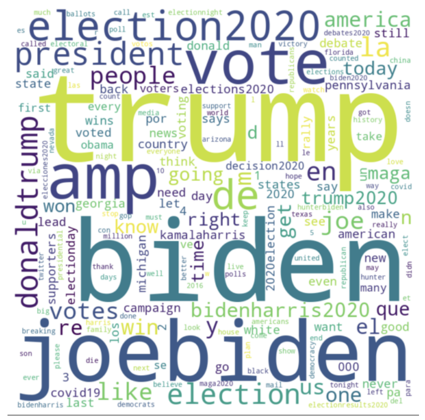

Fig. 8. Word Cloud

The word cloud highlights prominent terms used quite frequently in the tweets. The terms "trump", "biden", "joebiden", "election2020", "vote" and "donaldtrump" were most noticeable in the illustration. This advocated that there is a major focus on conversations about the 2020 US presidential candidates - Donald Trump and Joe Biden.

The main challenges of big data for our research, were the increased scale of instances and features of the dataset. The challenges associated with the volume and veracity of big data, such as processing, were addressed by designing a global approach solution which parallelised the operations via an execution plan, that would otherwise be executed sequentially in a procedural implementation. The strength of this approach allowed the distance computation of the data points and the calculation of centroid in K means to be parallelised, whereas in a sequential implementation, this operation would have been time consuming for large datasets. Furthermore, native PySpark functions were used for all of the operations, not including the user defined function, which optimises the computation using Sparks lazy evaluation strategy. This optimised the execution plan by postponing the data transformation until an action is invoked, minimising data movement and computation. Although most of the operations were designed using native Spark functions and narrow transformations, wide transformations such as Group By, Aggregation and Join were performed when joining data frames and grouping total sentiment and total votes during correlation analysis, which involved data movement through shuffling. Additionally, the solution was executed locally and a very large distributed cluster configuration could not be explored and is a limitation of our research experiment, as the linear scalability of parallelisation of our solution could have been investigated further. In comparison to the sequential approach, when redesigning the distributed implementation, it was necessary to register a Spark UDF in order to label tweet sentiment for supervised classification as there were no built-in Pyspark APIs available for this task, thereby increasing overhead as Spark is unable to optimize it.

V. Conclusion

Our research aimed to explore the influence of social media sentiment, specifically Twitter sentiment, on political outcomes during the US 2020 elections. We focused on the effects of sentiments expressed in tweets on the outcome of the elections at both a national and state-level.

Through our methodology, which incorporated a series of detailed exploratory data analyses, pre-processing steps, and machine learning model applications, we examined a large dataset consisting of approximately 1.8M tweets. Our data handling processes, particularly through the use of PySpark's machine learning libraries, enabled efficient management and analysis of this substantial dataset. Key pre-processing steps included cleaning tweets, handling missing data, and preparing the data for machine learning modeling.

Our model development revealed that Logistic Regression, among other models tested including Naive Bayes and K-Means clustering, provided the highest accuracy. This superior performance likely stemmed from logistic regression's robustness to noisy data features, which were prevalent in our social media dataset. The logistic regression model effectively captured the complexities of sentiment analysis, outperforming other models which showed limitations due to their assumptions about data independence and sensitivity to initial conditions. Both Speed-Up and Scale-Up show a deviation from the ideal linear plot and we observe a marginally small changes over 2 CPU cores.

The analysis highlighted the critical role of popular tweets, identified through retweet counts, which emphasized significant keywords related to the election. Visualization techniques, particularly the use of word clouds, proved invaluable. They not only enhanced our understanding of the common phrases and topics discussed but also demonstrated the effectiveness of PySpark in handling and visualizing large-scale text data. Key terms such as ”Trump,” ”Biden,” and ”election2020” dominated these visualizations, reflecting the intense focus on these topics among Twitter users.

In conclusion, our findings highlight the measurable impact of Twitter sentiments on political outcomes. Sentiments expressed on social media platforms like Twitter do influence public opinion and can sway the results of political events such as elections. Our research contributes to the understanding of digital communication's power in political arenas and underscores the importance of advanced analytical techniques in interpreting vast amounts of social media data. Future studies should consider extending this analysis to other platforms and elections to further explore the dynamics between social media sentiment and political outcomes, enhancing the robustness and applicability of our findings.

References

[1] Jayanta, Kumar, Mishra. (2023). Twitter sentiment analysis. Indian Scientific Journal Of Research In Engineering And Management, doi: 10.55041/ijsrem24071

[2] Yaser, A., Jasim., Mustafa, Ghanem, Saeed., Manaf, B., Raewf. (2023). Analyzing Social Media Sentiment: Twitter as a Case Study. Advances in distributed computing and artificial intelligence journal, doi: 10.14201/adcaij.28394

[3] Mohammad, Hadil, Dehghani. (2023). Political Sentiment Analysis of Persian Tweets Using CNN-LSTM Model.

[4] (2023). Prediction of Political Biasness of Statements. doi:10.1109/icaaic56838.2023.10140937

[5] Nitin, Prakash, Verma. (2023). Prediction of Political Biasness of Statements. doi: 10.1109/ICAAIC56838.2023.10140937

[6] (2023). Sentiment Analysis Techniques –Survey. doi: 10.31185/wjps.152

[7] (2023). Sentiment Analysis of Political Tweets for Israel Using Machine Learning. doi: 10.1007/978-3-031-15175-0_15

[8] (2023). Twitter Sentiment Analysis Using Naive Bayes-Based Machine Learning Technique. Advances in intelligent systems and computing, doi: 10.1007/978-981-19-5443-6_27

[9] Cao, D., Meadows, M., Wong, D., Xia, S., 2021. Understanding consumers' social media engagement behaviour: an examination of the moderation effect of social media context. J. Bus. Res. 122, 835–846.

[10] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, “An Introduction to Statistical Learning : with Applications in R“, New York :Springer, 2013, pp.184.

[11] Kaggle. US Election 2020 election result. URL: https://www.kaggle.com/datasets/callummacpherson14/2020-us-presidential-election-results-by-state

[12] Kaggle. US Election 2020 Tweets. URL: https://www.kaggle.com/datasets/manchunhui/us-election-2020-tweets

[13] Triguero, I., & Galar, M. (2023). Large-Scale Data Analytics with Python and Spark: A Hands-on Guide to Implementing Machine Learning Solutions. Cambridge: Cambridge University Press.