University of Camerino

School of Science and Technologies

Course Master Degree In Computer Science (LM-18)

Technologies for Big Data Management

Case of Study of a Distributed Machine Learning Pipeline

Students

Luca Bianchi

Michele Martini

Angelica Berdini

Supervisor

Prof. Massimo Callisto De Donato

A.A. 2023/2024

Abstract

In contemporary data-driven environments, the volume and complexity of datasets have surged exponentially, necessitating sophisticated solutions for analysis and management. Distributed systems have emerged as indispensable tools in this landscape, offering scalable and efficient frameworks for processing, analyzing, and storing large datasets.

This thesis explores the application of data analytics and machine learning techniques to the Yelp dataset, focusing initially on sentiment analysis but extending to incorporate the evaluation of review reliability. The primary aim is to compare the efficacy of PySpark ML algorithms with other contemporary machine learning methodologies commonly employed in sentiment analysis tasks. Leveraging the scalability and efficiency inherent in PySpark, the research seeks to delve into the intricacies of sentiment analysis across extensive datasets. Employing various machine learning libraries, encompassing transformer models and random forests, the study scrutinizes sentiments within reviews, aiming to discern subtle emotional nuances. Additionally, a review reliability assessment system is developed, investigating the correlation between star ratings and positive sentiments. Performance evaluation metrics, including accuracy, precision, recall, and F1-score, are utilized to assess algorithmic efficacy. Moreover, computational efficiency and scalability considerations are explored, contrasting PySpark's capabilities in handling voluminous data against conventional machine learning frameworks.

Contents

List of Figures

1. Introduction

In today's data-driven world, the ability to extract actionable insights from vast amounts of data is paramount. Data analytics, the science of examining raw data with the purpose of drawing conclusions, plays a pivotal role in various industries ranging from e-commerce to healthcare. Leveraging advanced techniques in data analytics not only enables businesses to gain a competitive edge but also facilitates informed decision-making processes.

In this context, the term "Business Intelligence" refers to the possibility of extracting meaningful information from previously collected data. This information will then be used to enhance business processes and generate profit based on improved "decision-making".

The project at hand is centered around implementing robust data analytics solutions, with a focus on sentiment analysis, utilizing a public dataset. Sentiment analysis, a branch of natural language processing, involves determining the sentiment expressed in textual data and providing valuable insights into customer opinions and preferences. By employing advanced sentiment analysis techniques, businesses can better understand customer sentiments, thereby refining marketing strategies, enhancing product offerings, and ultimately improving customer satisfaction.

In order to tackle the challenges posed by the size and complexity of the dataset, the project makes use of distributed computing frameworks such as Apache Spark. Spark offers unparalleled scalability and performance, enabling efficient processing of large-scale datasets across distributed computing clusters. Additionally, the flexibility and versatility of Spark allow for seamless integration with various data sources and formats, including semi-structured and unstructured data. Thus, necessary is a comparison between the non-distributed and distributed machine learning approaches and technologies.

One of the key objectives of the project is to explore and compare different machine-learning libraries for sentiment analysis. By experimenting with various algorithms and techniques, ranging from traditional statistical models to cutting-edge deep learning architectures such as transformer models, the project aims to identify the most effective approach for sentiment analysis in the given context. Furthermore, the project seeks to leverage multiple dataset files to perform intricate data manipulations, such as joins and correlations, to enrich the dataset and extract meaningful insights.

Crucially, the results of the sentiment analysis will be published on ElasticSearch, a highly scalable search and analytics engine, enabling easy access and visualization. Basic views on sentiment will be provided, offering stakeholders a clear understanding of sentiment trends and patterns within the dataset.

In summary, this project embodies the essence of modern data analytics, combining advanced techniques in sentiment analysis with distributed computing frameworks to extract actionable insights from complex datasets. By addressing these objectives, the project aims to showcase the significance of data analytics in today's data-driven landscape, underscoring the importance of Spark and distributed computing environments in enabling scalable and efficient data processing. Additionally, the project aims to contribute to the ongoing discourse surrounding machine learning libraries and their applicability in sentiment analysis tasks.

2. Building Blocks of Data Science

In the realm of data science, while distributed technologies are essential for handling large-scale data, standard data science technologies play a crucial role in exploratory data analysis, data pre-processing, model development and visualization. These technologies aren't environment-dependent and are fundamental in any machine-learning project

2.1 Data science pipeline

The next paragraphs explain what makes up a typical process in data science. Think of it like a road map that helps us turn raw data into useful insights. It's a structured framework that guides us through analyzing data to make better decisions.

2.1.1 Load dataset

The initial step in a typical machine learning approach involves acquiring the dataset from various sources such as databases, files, or APIs.

Depending on the case, this operation might be less or more complex. Such as the case in which the dataset is stored among different sources and in different formats, or the case in which the dataset is stored in a single location and is ready to be analyzed.

2.1.2 Exploratory Data Analysis

Exploratory Data Analysis is the process of analyzing and visualizing data to understand its underlying patterns and relationships. EDA helps in gaining insights into the dataset, identifying outliers, detecting correlations between variables, and selecting relevant features for modeling. It is a critical step that involves examining and visualizing the data to gain a deeper understanding before diving into further analysis.

2.1.3 Data pre-processing

Data Preprocessing involves cleaning, transforming, and preparing raw data for analysis. It includes tasks like handling missing values, encoding categorical variables, scaling numerical features, and splitting the data into training and testing sets. Data Preprocessing ensures that the data is in a suitable format for model development. In this context, we will also perform any operation that we observed during the exploratory data analysis phase. Any problem encountered, such as wrong data formats, data normalization, by first visualizing the distribution of the dataset, or even data-grouping, in case is necessary to group different similar values of a feature in a single one that represents them all.

2.1.4 Model development

Model development stands as a pivotal stage where data transforms into actionable insights. This phase involves selecting the most suitable algorithm, training it on preprocessed data, and fine-tuning its parameters to optimize performance. Through meticulous hyperparameter tuning, cross-validation techniques, and rigorous evaluation metrics, the model evolves into a powerful tool for making predictions or classifications. The iterative process of refining the model ensures its robustness and accuracy, paving the way for impactful applications across diverse domains. Successful model development is not just about building predictive models; it's about crafting intelligent solutions that drive innovation, enhance decision-making processes, and unlock the true potential of data-driven technologies.

2.1.5 Visualization

The last phase is visualization, that involves the visualization of the model in order to understand and evaluate the model. It is very useful to visualize metrics like accuracy, precision, recall, and F1 score aids in evaluating model performance, and serves as a universal language for conveying complex ideas simply and intuitively, facilitating effective communication with stakeholders. The goal is to simplify the comprehension of complex algorithms and data patterns, making them more accessible to both technical and non-technical stakeholders. Visualization bridges the gap between the intricate inner workings of machine learning models and human understanding.

3. Distributed and Non-distributed Data Science Technologies

In the era of big data, the ability to process and analyze large datasets is crucial. This project leverages distributed data science technologies to handle the vast Yelp dataset, which contains millions of reviews from users worldwide. In this chapter, we will also explore the actual implementation and libraries utilized in a data pipeline and analyze the differences between distributed and non-distributed technologies.

3.1 Exploratory Data Analysis

In the realm of exploratory data analysis (EDA), Python libraries such as Seaborn and Matplotlib play a crucial role. They provide a high-level interface for creating visually appealing and informative statistical graphics, aiding in the exploration of patterns, trends, and relationships within datasets. EDA involves studying, exploring, and visualizing data to derive valuable insights, helping in formulating hypotheses, and guiding further investigations[2].

For distributed technologies the go to choice is Spark, thanks to the many libraries present it can cover almost all of the data pipeline. Apache Spark stands out as a powerful open-source unified analytics engine that serves as the backbone of distributed data science environments. Its in-memory computing capabilities enable rapid processing of large-scale datasets, making it particularly well-suited for iterative machine learning algorithms like those used in processing the Yelp dataset efficiently[7][3].

3.2 Data manipulation

As far as non distributed technologies are concerned the best library to handle data is Pandas. Pandas is a powerful open-source data manipulation and analysis library for

Python, essential for data cleaning and pre-processing tasks. It offers functions to work with structured data seamlessly, integrating well with other libraries like NumPy and Matplotlib. Pandas introduces key data structures like Series and DataFrame, enabling efficient manipulation, cleaning, and analysis of datasets[5].

The versatility of Apache Spark extends beyond its in-memory computing prowess. It provides a general programming model that allows developers to compose various operators for tasks like mapping, reducing, joining, grouping, and filtering. This flexibility enables a wide range of computations, including iterative machine learning, streaming analytics, complex queries, and batch processing. Spark's ability to track data produced by operators and store it in memory significantly boosts performance by minimizing disk accesses.

3.3 Model Development

In the realm of non-distributed data science technologies, Scikit-learn, Keras and TensorFlow stands out as widely used machine learning libraries in Python, offering a rich set of classification, regression, and clustering algorithms but also developing and training deep learning models. They seamlessly integrates with essential Python libraries like NumPy and SciPy, enabling efficient data manipulation and analysis[6]. leveraging these libraries allows data scientists to explore a wide range of machine learning and deep learning techniques efficiently.

Distributed machine learning libraries play a pivotal role in enabling distributed data science technologies to efficiently process and analyze large-scale datasets across multiple computing nodes. These libraries provide pre-built algorithms and functions that simplify the development of machine learning models in distributed environments. Popular distributed machine learning libraries include PySpark.ML, TensorFlow and PyTorch, all of them offer built-in support for distributed training allowing training across multiple GPUs with minimal code modifications. These libraries ensure efficient processing of vast amounts of data for advanced analytics and modeling tasks[4][1][8].

3.4 Visualization

Matplotlib is a versatile plotting library for Python. It provides a comprehensive set of tools for creating static, animated, and interactive visualizations in Python. Matplotlib allows users to embed plots into applications using various general-purpose GUI toolkits. Matplotlib is widely used in the scientific computing community and is integrated into libraries like SciPy.

Elasticsearch and Kibana are integral components of the Elastic Stack, a powerful search platform designed to help users search, analyze, and visualize data efficiently. Elasticsearch is a JSON-based search and analytics engine that enables users to store, search, and analyze data with speed at scale. Kibana is a browser-based analytics and search dashboard specifically designed for Elasticsearch, offering users an intuitive interface to explore and visualize data.

3.5 Performance Comparison

Non-distributed data science technologies play a crucial role in laying the groundwork for data science projects by enabling data scientists to understand their data and develop initial models effectively. These non-distributed technologies, including Python, Pandas, Scikit-learn, Matplotlib, Keras, and TensorFlow, are instrumental in a data science pipeline. They provide essential functionalities for cleaning and transforming data, visualizing insights, and building machine learning and deep learning models in a non-distributed environment. While distributed technologies like Apache Spark become more prominent as projects scale up due to their ability to handle large volumes of data efficiently, the significance of non-distributed technologies remains paramount. They form the foundation of data science projects by facilitating the initial stages of data exploration, analysis, and model building. These tools not only enhance data comprehension but also guide decision-making processes by extracting valuable insights from datasets.

The advantages of distributed technologies are varied and mainly linked to performance and cost, both very important to businesses, the main users of distributed technologies. Distributed systems can enhance performance by utilizing the resources of multiple computers to handle workloads efficiently operating at higher levels compared to centralized systems. While distributed systems may have high implementation costs, they are relatively cost-effective in the long run especially when implemented through comodity hardware. Distributed systems are designed to be efficient, with multiple computers working independently to solve problems. Distributed systems are inherently scalable, allowing users to add mo There are no restrictions on the number of machines that can be added, enabling easy handling of high-demand workloads. This is especially important when there is the temporarily need of more computation power but later is not needed anymore. Distributed systems exhibit higher reliability compared to single systems in terms of failures. Even if one node malfunctions, it does not disrupt the functioning of other servers, ensuring continuous operation.

These advantages make distributed technologies a preferred choice for applications requiring improved performance, cost-effectiveness, efficiency, scalability, reliability and security. On the other hand non distributed technologies are less complex and more easy to use especially with smaller datasets.

4. Study Case

In this chapter, we explain the implementation of the project centered around sentiment analysis and user reliability assessment within the expansive Yelp dataset. Through the utilization of advanced technologies and methodologies in Natural Language Processing (NLP) and machine learning, we delve deep into the intricacies of analyzing sentiments within the reviews. Beginning with an overview of the Yelp dataset's significance in academia and data analysis, we navigate through the deployment of a Spark standalone cluster orchestrated via Docker Compose. Delving into the implementation details, we explore the application of PySpark's machine learning library for sentiment analysis, alongside an alternative approach employing Hugging Face Transformers. Furthermore, we discuss the pivotal role of word embedding in augmenting NLP tasks and delve into the assessment of user reliability, enriching our understanding of user-generated content's credibility.

4.1 Yelp Dataset

The Yelp dataset is a rich collection of real-world data encompassing information on businesses, reviews, user interactions, pictures, tips, business attributes, and aggregated check-ins from multiple metropolitan areas. It includes a vast amount of data such as millions of reviews, details on hundreds of thousands of businesses, user-contributed pictures and tips, business attributes like operating hours and amenities, and historical check-in data. This dataset serves as a valuable resource for academic research, teaching, learning, and various data analysis tasks in fields such as natural language processing, machine learning, and database-related projects.

4.2 Docker

In our project the Docker Compose file orchestrates a multi-container environment for deploying a Spark standalone cluster alongside auxiliary services. Specifically, it defines three services for Spark: one master node and two worker nodes. These services are configured to run Spark processes using specific commands and to communicate with each other over the network. Additionally, the file sets up services for JupyterLab, Elasticsearch, and Kibana, each with their respective configurations and dependencies. Docker Compose simplifies the management of this complex setup by allowing developers to define and deploy multiple interconnected containers with a single configuration file. With this setup, users can efficiently develop, test, and scale Spark-based applications while leveraging additional tools for data exploration and analysis. The 4.1 below represent the structure of our distributed architecture with a diagram.

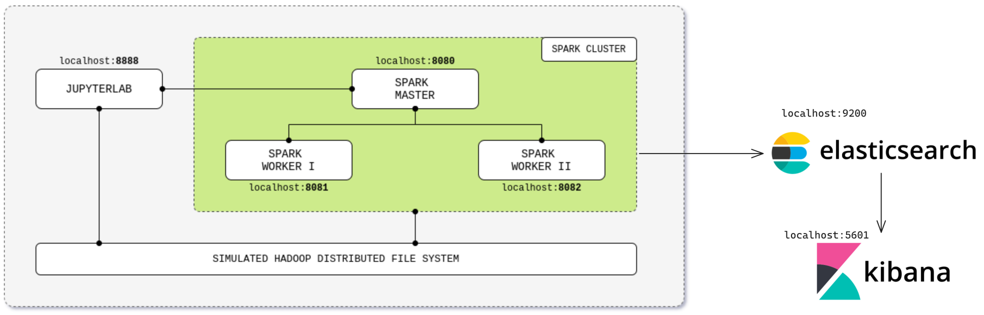


Figure 4.1: Docker-compose structure

4.3 PySpark's machine learning library

In the sentiment analysis conducted using PySpark's machine learning library on a substantial Yelp dataset, the process begins with transforming star ratings from reviews into sentiment categories - positive, negative, or neutral. This transformation allows for a more nuanced understanding of the sentiments expressed in the reviews. The data undergoes pre-processing steps like tokenization, stop words removal, and TF-IDF vectorization to prepare it for classification. A Random Forest classifier is then employed to categorize the reviews based on their sentiments. To address potential class imbalances within the dataset, class weights are calculated to ensure a more balanced training process. The model is trained on the pre-processed data and evaluated using various metrics to assess its performance. Through this approach, valuable insights can be gleaned from the Yelp reviews, shedding light on the sentiments expressed by customers towards various shops and establishments.

The trained sentiment analysis model achieved promising results with an accuracy of 68.85%, indicating that it correctly classified nearly 70% of the reviews. Precision, measuring the proportion of correctly identified positive sentiments among all predicted positive sentiments, stood at 73.45%. This suggests that when the model predicted a positive sentiment, it was correct around 73.45% of the time. Moreover, the model demonstrated a recall rate of 68.85%, meaning that it successfully captured approximately 68.85% of all actual positive sentiments present in the dataset. Finally, the F1 score, which considers both precision and recall, reached 70.70%, highlighting a balanced performance between the two metrics. These results suggest that while the model shows promise in accurately identifying sentiment in reviews, there is still room for improvement, particularly in enhancing recall to capture more positive sentiments. Continued refinement and optimization of the model could lead to even more accurate sentiment analysis outcomes, ultimately providing valuable insights for businesses aiming to understand customer perceptions and sentiments. The 4.2 The **??**

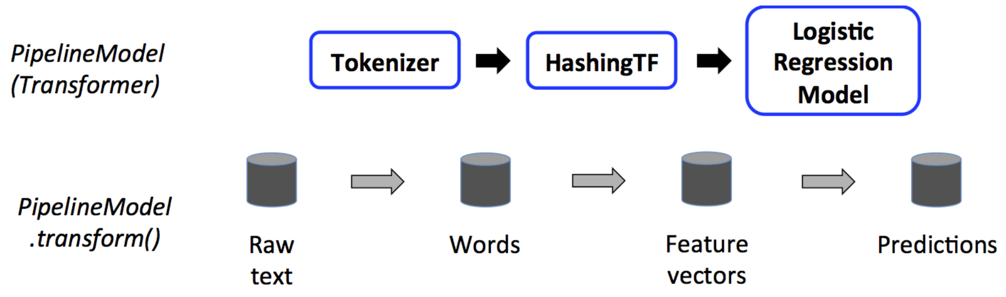

PipelineModel (Transformer)
Tokenizer
HashingTF
Logistic Regression Model
PipelineModel .transform()
Raw text
Words
Feature vectors
Predictions

Figure 4.2: Example of a PipelineModel in Pyspark.ml

4.4 Hugging Face Transformers

In this paragraph, we explore an alternative approach to sentiment analysis using Hugging Face Transformers, a powerful library for natural language processing tasks. Unlike the previous approach, which relied on traditional machine learning techniques and custom feature engineering, Hugging Face Transformers leverages pre-trained models to achieve state-of-the-art performance in various NLP tasks, including sentiment analysis. Hugging Face Transformers provides a user-friendly interface for accessing and fine-tuning pre-trained models for specific NLP tasks. These pre-trained models, such as BERT, RoBERTa, and DistilBERT, have been trained on large corpora of text data and fine-tuned on specific downstream tasks, including sentiment analysis. In our project we use BERT-sentiment because it is the model that provides better results with our dataset. Compared to the previous approach, which involved manual feature engineering and the training of a machine learning model, the Hugging Face Transformers approach offers several advantages:

1. **Ease of Use**: Hugging Face Transformers provides a simple and intuitive interface for performing sentiment analysis, requiring minimal setup and configuration.

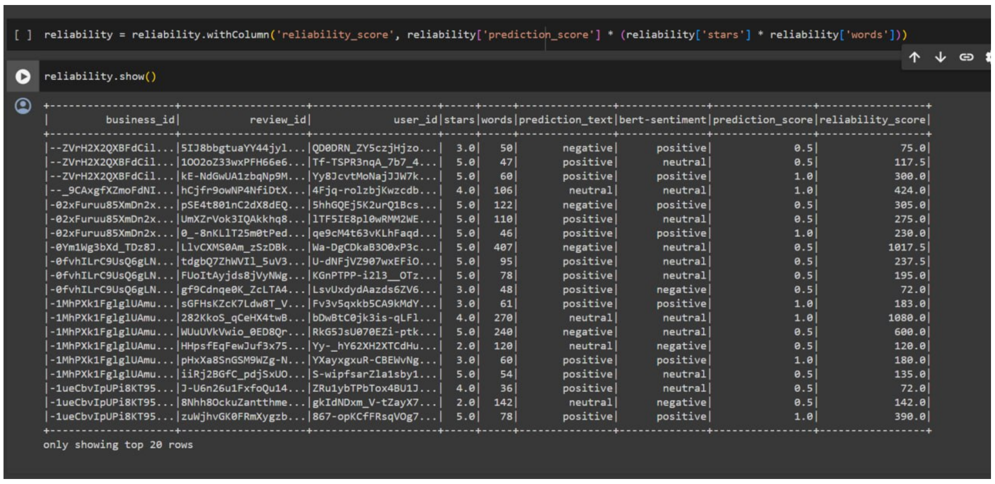
2. **State-of-the-Art Performance**: By leveraging pre-trained models fine-tuned on large datasets, Hugging Face Transformers achieves state-of-the-art performance in sentiment analysis tasks.

3. **Flexibility**: Hugging Face Transformers supports a wide range of pre-trained

models, allowing users to choose the model that best suits their specific use case and requirements.

4.5 User reliability score

The calculation of the user reliability based on their reviews involves a comprehensive evaluation of various factors. In general these factors include the consistency of user reviews over time, the level of detail and completeness in their feedback, the frequency of providing reviews, and the overall experience and expertise of the user in the specific domain or product being reviewed. Given the scope of this project we decided to focus on other parameters related to the sentiment analysis. We focused on user feedback including star ratings and text reviews, both plays a significant role in adjusting this initial rating based on sentiment analysis and text mining techniques. By analyzing user comments and star ratings through Machine Learning algorithms (Chapter3), a holistic view of user reliability can be established to enhance the credibility and trustworthiness of reviews[9]. In particular to calculate the score we exploited the high precision of the Hugging Face (HF) model. We compared the sentiment of the review with the number of stars assigned by the user, specifically coded to be comparable with the HF model result, if the two did not match then the review wasn't fully reliable and affected negatively the reliability score of the user otherwise it affected the score positively.



```
[ ] reliability = reliability.withColumn('reliability_score', reliability['prediction_score'] * (reliability['stars'] * reliability['words']))
```

```
reliability.show()
```

```
+--------------------+--------------------+--------------------+-----+-----+---------------+--------------+----------------+-----------------+
|         business_id|           review_id|             user_id|stars|words|prediction_text|bert-sentiment|prediction_score|reliability_score|
+--------------------+--------------------+--------------------+-----+-----+---------------+--------------+----------------+-----------------+
|--ZVrH2X2QXBFdCil...|5IJ8bbgtuaYY44jyl...|QD0DRN_ZY5czjHjzo...|  3.0|   50|       negative|      positive|             0.5|             75.0|
|--ZVrH2X2QXBFdCil...|1OO2oZ33wxPFH66e6...|Tf-TSPR3nqA_7b7_4...|  5.0|   47|       positive|       neutral|             0.5|            117.5|
|--ZVrH2X2QXBFdCil...|kE-NdGwUA1zbqNp9M...|Yy8JcvtMoNajJJW7k...|  5.0|   60|       positive|      positive|             1.0|            300.0|
|--_9CAxgfXZmoFdNI...|hCjfr9owNP4NfiDtX...|4Fjq-rolzbjKwzcdb...|  4.0|  106|        neutral|       neutral|             1.0|            424.0|
|-02xFuruu85XmDn2x...|pSE4t801nC2dX8dEQ...|5hhGQEj5K2urQ1Bcs...|  5.0|  122|       negative|      positive|             0.5|            305.0|
|-02xFuruu85XmDn2x...|UmXZrVok3IQAkkhq8...|1TF5IE8pl0wRMM2WE...|  5.0|  110|       positive|       neutral|             0.5|            275.0|
|-02xFuruu85XmDn2x...|0_-8nKLlT25m0tPed...|qe9cM4t63vKLhFaqd...|  5.0|   46|       positive|      positive|             1.0|            230.0|
|-0Ym1Wg3bXd_TDz8J...|LlvCXMS0Am_zSzDBk...|Wa-DgCDkaB3O0xP3c...|  5.0|  407|       negative|       neutral|             0.5|           1017.5|
|-0fvhILrC9UsQ6gLN...|tdgbQ7ZhWVIl_5uV3...|U-dNFjVZ907wxEFiO...|  5.0|   95|       positive|       neutral|             0.5|            237.5|
|-0fvhILrC9UsQ6gLN...|FUoItAyjds8jVyNWg...|KGnPTPP-i2l3__OTz...|  5.0|   78|       positive|       neutral|             0.5|            195.0|
|-0fvhILrC9UsQ6gLN...|gf9Cdnqe0K_ZcLTA4...|LsvUxdydAazds6ZV6...|  3.0|   48|       positive|      negative|             0.5|             72.0|
|-1MhPXk1FglglUAmu...|sGFHsKZcK7Ldw8T_V...|Fv3v5qxkb5CA9kMdY...|  3.0|   61|       positive|      positive|             1.0|            183.0|
|-1MhPXk1FglglUAmu...|282KkoS_qCeHX4twB...|bDwBtC0jk3is-qLFl...|  4.0|  270|        neutral|       neutral|             1.0|           1080.0|
|-1MhPXk1FglglUAmu...|WUuUVkVwio_0ED8Qr...|RkG5JsU070EZi-ptk...|  5.0|  240|       negative|       neutral|             0.5|            600.0|
|-1MhPXk1FglglUAmu...|HHpsfEqFewJuf3x75...|Yy-_hY62XH2XTCdHu...|  2.0|  120|        neutral|      negative|             0.5|            120.0|
|-1MhPXk1FglglUAmu...|pHxXa8SnGSM9WZg-N...|YXayxgxuR-CBEWvNg...|  3.0|   60|       positive|      positive|             1.0|            180.0|
|-1MhPXk1FglglUAmu...|iiRj2BGfC_pdjSxUO...|S-wipfsarZla1sby1...|  5.0|   54|       positive|       neutral|             0.5|            135.0|
|-1ueCbvIpUPi8KT95...|J-U6n26u1FxfoQu14...|ZRu1ybTPbTox4BU1J...|  4.0|   36|       positive|       neutral|             0.5|             72.0|
|-1ueCbvIpUPi8KT95...|8Nhh8OckuZantthme...|gkIdNDxm_V-tZayX7...|  2.0|  142|        neutral|      negative|             0.5|            142.0|
|-1ueCbvIpUPi8KT95...|zuWjhvGK0FRmXygzb...|867-opKCfFRsqVOg7...|  5.0|   78|       positive|      positive|             1.0|            390.0|
+--------------------+--------------------+--------------------+-----+-----+---------------+--------------+----------------+-----------------+
only showing top 20 rows
```

Figure 4.3

4.6 Use case results

From the analysis of model reliability in the project, a significant differentiation emerged, with a low reliability impacting the effectiveness of the Random Forest algorithm notably. This discrepancy in model reliability can be attributed to various factors, including data complexity and algorithm sensitivity to variations in the input dataset. Particularly, the Random Forest, extensively used within the project's context, exhibited lower performance than expected due to the poor reliability of the input data.

The analysis of actual performance revealed that a heavyweight architecture like the one developed for the project is challenging to manage and sustain in a local environment. Limited computational resources and storage capabilities may constrain overall performance, especially given the high computational load required to execute machine learning models on a large dataset. Thus, to achieve better performance and effectively manage the project's complexity, it becomes necessary to migrate the solution to a distributed environment. Utilizing a framework like Apache Spark, with its ability to distribute workload across a cluster of nodes, can maximize resource utilization and deliver more reliable and efficient results. Distributing the architecture across a distributed environment can also facilitate system scalability, enabling the handling of increasingly larger datasets without compromising performance.

5. Conclusions and Further Work

In the following sections, we will delve into the conclusions drawn from this project and discuss potential avenues for future work. These insights encapsulate the lessons learned and the opportunities identified, providing a comprehensive overview of the project's outcomes and its implications for future endeavors. Let's explore these in detail.

5.1 Conclusion

In conclusion, this project has provided valuable insights into sentiment analysis methodologies, particularly through the exploration of different machine learning libraries such as PySpark.ML and Hugging Face Transformers. Our findings indicate that Hugging Face Transformers, leveraging pre-trained models, demonstrated superior performance compared to PySpark.ML in sentiment analysis tasks. The utilization of pre-trained models allowed for more accurate and efficient analysis of sentiment within Yelp reviews. Moreover, the contrast between the two libraries facilitated the identification of problematic reviews, which served as the foundation for the development of a user reliability scoring system. By leveraging the differences between these libraries, we were able to assess the credibility of user-generated content, providing businesses with valuable insights for decision-making processes.

Additionally, the implementation of Docker Compose has significantly contributed to the efficiency and scalability of the project. By orchestrating a distributed system with a master and two worker nodes for Spark, Docker Compose has enabled seamless deployment and management of Spark-based applications. This distributed setup has facilitated parallel processing of data, enhancing the speed and performance of sentiment analysis tasks on large datasets like Yelp. Through the integration of Docker Compose, we have achieved a highly efficient and scalable solution for sentiment analysis, paving the way for future enhancements and extensions of the project.

With regard to performance these two approaches have very different outcomes. We expected differences in performance in favour of the distributed libraries especially with larger dataset. The differences between non-distributed and distributed are notable and depend on various factors, in summary:

- Non-distributed: works best with smaller datasets, does not support parallelization, offers a wide range of traditional machine learning algorithms, is focused on machine learning tasks and lacks built-in support for seamless integration with big data technologies.
- Distributed: works best with very large dataset that usually need to be stored across a cluster of machines, supports parallelization, provides basic set of machine learning algorithms including algorithms for classification, regression, clustering, and recommendation systems, integrates well with various big data technologies.

In the figure 5.1 we can see that from dataset of 13GB or bigger scikit-learn(non-distributed) starts to be less efficient than PySpark(distributed) but on the other hand with smaller dataset the latter has worse performance when compared to the first. This condition is known as Overhead and identifies the lesser performance with dataset too small to be handled efficiently by PySpark.

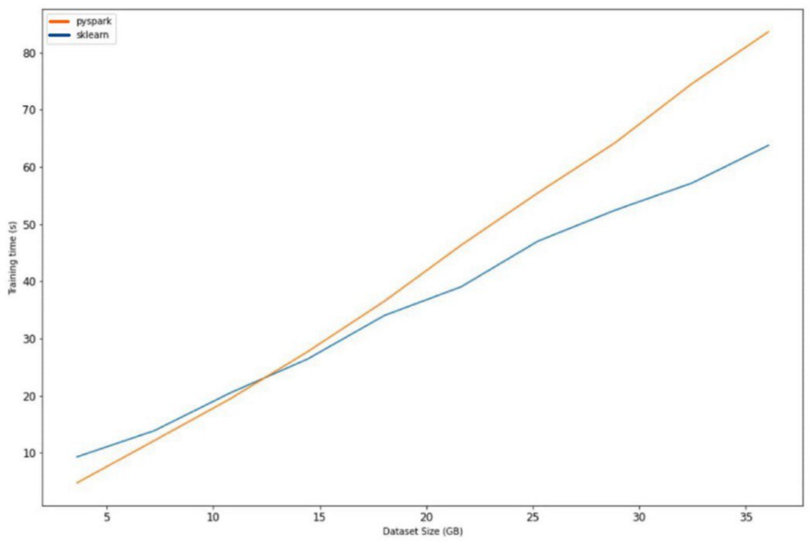


Figure 5.1: Scikit-learn and PySpark results

5.2 Further Works

Moving forward, there are several avenues for expanding upon this project. Firstly, an important work is represented by the addition of algorithms that can calculate the similarity between texts, in order to recognize if two similar texts provide the same result for different stars. One of the most popular approach is represented by word embedding techniques that could enhance the understanding of textual data and improve sentiment analysis outcomes. Implementing algorithms that utilize word embeddings to measure text similarity would allow for more nuanced analysis of reviews. Additionally, the development of a recommendation system based on shop categories could provide users with personalized suggestions, enhancing their overall experience.

In our project we try to develop a simple reccomendation system using the ALS algorithm but it is a simple protatype, in the future it could be it can be improved to ensure a better service for users.

By incorporating these enhancements, we aim to refine our sentiment analysis capabilities and provide businesses with more comprehensive insights into customer sentiments and preferences.

Bibliography

[1] *distributedtools.* URL: https://neptune.ai/blog/distributed-training-frameworks-and-tools.

[2] *Eda.* URL: https://www.geeksforgeeks.org/exploratory-data-analysis-in-python/.

[3] *inmemory.* URL: https://www.cloudduggu.com/spark/in-memory-computation/.

[4] *Machine learning.* URL: https://www.linkedin.com/pulse/machine-learning-libraries-pratibha-kumari-jha.

[5] *Pandas.* URL: https://www.freecodecamp.org/news/data-cleaning-and-preprocessing-with-pandasbdvhj/.

[6] *Scikit-learn.* URL: https://en.wikipedia.org/wiki/Scikit-learn/.

[7] *spark.* URL: https://www.oreilly.com/library/view/data-analytics-with/9781491913734/ch04.html.

[8] *spark3.* URL: https://neptune.ai/blog/distributed-training.

[9] *urs5.* URL: https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0258050.