

**School Of Tech**

**Graduate Diploma in Data Analytics (Level 7)**

**Cover Sheet and Student Declaration**

This sheet must be signed by the student and attached to the submitted assessment.

|  |  |  |  |
| --- | --- | --- | --- |
| **Course Title:** | **Big Data Analytics** | **Course code:** | **GDDA-709** |
| **Student Name:** | Priyank dangwal | **Student ID:** | 764707660 |
| **Assessment No**  **& Type:** | Assessment 2[Portfolio] | **Cohort:** | GDDA7123C |
| **Due Date:** | 01/08/2024 | **Date**  **Submitted:** | 01/08/2024 |
| **Tutor’s Name:** | Sara Zandi | | |
| **Assessment**  **Weighting** | 80% | | |
| **Total Marks** | 100 | | |

***Student Declaration:***

I declare that:

* I have read the New Zealand School of Education Ltd policies and regulations on assessments and understand what plagiarism is.
* I am aware of the penalties for cheating and plagiarism as laid down by the New Zealand School of Education Ltd.
* This is an original assessment and is entirely my own work.
* Where I have quoted or made use of the ideas of other writers, I have acknowledged the source.
* This assessment has been prepared exclusively for this course and has not been or will not be submitted as assessed work in any other course.
* It has been explained to me that this assessment may be used by NZSE Ltd, for internal and/or external moderation.
* If I am late in handing in this assessment without prior approval (see student regulations in handbook), marks will be deducted, to a maximum of 50%.

***Student signature: Priyank Dangwal***

## Date: 01/08/2024

|  |  |  |
| --- | --- | --- |
| **Tutor only to complete** | | |
| **Assessment result:** | **Mark /100** | **Grade** |

Contents

[Date: 01/08/2024 1](#_Toc173425444)

[Task 1: Implementation of Big Data Framework. 3](#_Toc173425445)

[Part A. Dataset Description 3](#_Toc173425446)

[Structure: 3](#_Toc173425447)

[Output and Input Data: 3](#_Toc173425448)

[Part B. Establishing Data Ingestion Pipeline 3](#_Toc173425449)

[Extract Process: 3](#_Toc173425450)

[Transform Process: 4](#_Toc173425451)

[Loading Process: 6](#_Toc173425452)

[Configuration and Cluster Settings: 7](#_Toc173425453)

[Challenges and Mitigation Strategies: 8](#_Toc173425454)

[Task 2: Map Reduce in Big Data 8](#_Toc173425455)

[Part A: Analytical Method Employed 8](#_Toc173425456)

[Temporal Analysis: 8](#_Toc173425457)

[Part B: Explanation of addressing the below aspects 11](#_Toc173425458)

[Input and Output handling: 11](#_Toc173425459)

[Map-Reduce Design: 11](#_Toc173425460)

[Task 3: Implementing Machine Learning Algorithms on Big Data for Predictions and Classification on Time Series Data 12](#_Toc173425461)

[Part A: Developing Predictive Analytics Solution for Time Series dataset using Apache Spark 12](#_Toc173425462)

[Data Preparation and Transformation 12](#_Toc173425463)

[Machine Learning Techniques and Rationale: 16](#_Toc173425464)

[Classification and Prediction Tasks 22](#_Toc173425465)

[Conclusion: 24](#_Toc173425466)

[Part B: Implementation of machine Learning Algorithms using Apache Spark 24](#_Toc173425467)

[Task 4: Discussion and Conclusion 26](#_Toc173425468)

[Analysis and Insights (Map Reduce) 26](#_Toc173425469)

[Analysis and Insights (Machine Learning Implementation) 27](#_Toc173425470)

[Part B: Conclusion 27](#_Toc173425471)

# Task 1: Implementation of Big Data Framework.

## Part A. Dataset Description

**Source:** The dataset was acquired from the UC Irvine Machine Learning Repository (UCI), a repository that the machine learning community uses to store databases, domain theories, and data generators. Financial data is included in it, such as the daily returns of different stock market indices.

**Format:** The data is provided in Excel format.

**Size:** The size of file is around 69 KB. The dataset contains 10 columns and 537 rows, which includes dates and financial indices, with daily records of certain period.

### Structure:

Date: The day the observation was made.  
TL\_BASED\_ISE: Turkish Lira-based index returns for the Istanbul Stock Exchange.  
USD\_BASED\_ISE: Returns on the US dollar-based Istanbul Stock Exchange index.  
SP: The S&P 500 index's returns.  
DAX: The German Stock Index (DAX) returns.  
FTSE: The UK Stock Index's (FTSE) returns.  
NIKKEI: The Japanese Stock Index's (NIkkei) returns.  
BOVESPA: The Brazilian Stock Index's (BOVESPA) returns.  
EU: The generic European index's returns.  
EM: Emerging Markets index returns.

**Content:** The dataset contains daily results for several financial indexes, which are helpful for forecasting stock movements and analysing financial markets. In the dataset, every row represents the results for a specific day over several indices.

### Output and Input Data:

**Target data:** USD\_Based\_ISE: The target variable for modelling and prediction purposes. The numbers represent the daily returns of the Istanbul Stock Exchange index based on the US Dollars.

**Input data:** The input data includes the remaining columns which will be used as a feature for modelling and predictions.

## Part B. Establishing Data Ingestion Pipeline

**Ingesting Process Explanation (ETL) In Spark Data Frame**

### Extract Process:

The extract process involves loading raw dataset from an excel file into Pandas Data Frame.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The data successfully loaded into Pandas Data Frame, Columns are renamed for clarity, the date column is converted to date time type.

### Transform Process:

The transform process involves cleaning the data, handling missing values, identify and removing outliers, removing duplicates and normalizing the data.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A white background with colorful text

Description automatically generated

Missing values handled, outliers identified and removed, duplicate removed, data normalized and cleaned data is saved in a csv format named ‘cleaned\_dataset.csv’

### Loading Process:

The load process involves loading the clean dataset into Spark Data Frame, performing additional transformations and saving it in Parquet format.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Spark session is initialized with custom configurations. Cleaned dataset is loaded into a Pandas Data Frame. Pandas Data Frame is converted to a Spark Data Frame. Missing values are handled. Date column is converted to the appropriate type. Data is partitioned by date. Data is saved to Parquet format. Data is read back from Parquet files and displayed. Spark session is stopped.

### Configuration and Cluster Settings:

Spark's setup for a local 16GB RAM, 8 cores, and i7 CPU is used in the code, which is the local machine specification.

**Executor Memory:** 4 GB per executor to ensure sufficient memory for processing.

**Executor Cores:** 2 cores per executor for parallel processing.

**Driver Memory:** 4 GB for the driver to handle coordination.

**Shuffle Partitions:** 8 partitions to match the number of cores and optimize performance.

**Apache Arrow:** Enabled for efficient conversion between Pandas and Spark Data Frames.

### Challenges and Mitigation Strategies:

1. Challenge: Limited memory and CPU might impact performance while handling large datasets.   
Mitigation: Use data partitioning and optimize Spark configurations. Scale using distributed storage.  
  
2. Challenges in Data Quality: - Missing values, outliers, and duplicates impact data quality.   
Mitigation: Clean data thoroughly during transformation.   
  
3. Resource Management:

Challenge: Inefficient allocation affects performance.   
Mitigation: Resource monitoring, configuration adjustments and use of dynamic allocation if available.   
  
4. Troubleshooting and Debugging   
Challenge: Distributed system debugging is difficult.   
Mitigation: Monitor and log via Spark's web UI. Modularise the pipeline for simpler troubleshooting.

# Task 2: Map Reduce in Big Data

Part A: Analytical Method Employed  
The code is mostly used for time analysis and statistical computations:  
  
Statistical Computations:The average stock value for each date is determined by the MapReduce process. To calculate the average, summing up the stock values for each date, divide that total by the number of stock values for that day.

Temporal Analysis: By arranging the data according to dates, it is possible to examine patterns in stock value across time.

A computer screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A close up of a text

Description automatically generated

## Part B: Explanation of addressing the below aspects

### Input and Output handling:

**Input:** The input data is read from a CSV file named ‘Stock data new.csv’ and loaded into a RDD format (Resilient Distributed Dataset). The data is then split into fields, and the header is filtered out to make sure that only the significant data is processed.

**Output:** The final result which is Average stock value/date is collected and printed to the console table.

### Map-Reduce Design:

**Mapping Step:** The date is the key and the stock value and count of 1 are the values in key-value pairs. Prepares data for aggregation.  
Reducing Step: The reduction step sums stock values and counts by date. Total stock value and count are calculated for each date.  
Final Calculation: The average stock value for each date is the total stock value divided by the count.  
  
**Data Partitioning:** During the map process, data is automatically partitioned by date. Each date is a key, so the reduction step may handle each data separately. This allows parallel processing and efficient aggregation.  
  
  
**Cluster Configuration****:** Spark's setup for a local 16GB RAM, 8 cores, and i7 CPU is used in the code, which is the local machine specification. SparkContext is initialised using "StockDataMapReduce" as the application. Resource allocation optimises hardware consumption in this scenario. This setting can optimise resource allocation and performance based on the cluster's hardware in real life.

**Scalability:** Spark scales jobs horizontally. Spark can effectively process huge datasets by distributing data and calculations over numerous cluster nodes.

**Performance:** Parallel processing boosts performance. Spark reduces data shuffling and maximises computational resources by splitting data by date and processing each partition separately.  
  
In short, the code calculates average stock value per date for statistical computations and temporal analysis. It effectively handles input and output, aggregates data by date in the MapReduce task, splits data organically by date, and uses Spark's cluster setup for scalability and performance. This guarantees the task can efficiently process enormous datasets and provide significant stock value trends over time.

# Task 3: Implementing Machine Learning Algorithms on Big Data for Predictions and Classification on Time Series Data

## Part A: Developing Predictive Analytics Solution for Time Series dataset using Apache Spark

### Data Preparation and Transformation

1. Data Loading and Initial Inspection:

The dataset was loaded from a CSV file. The initial inspection included checking the schema and first few rows to understand the data structure and types.

2. Data Cleaning and Preprocessing:

Date Conversion: The `date` column was converted from string to date type for proper temporal analysis.

Repartitioning: The data was repartitioned to optimize performance during processing.

Handling Missing Values: Rows with null values, especially those arising from lag or rolling calculations, were dropped.

3. Feature Engineering:

Lag Feature: Created a lag feature for the target variable ‘USD BASED ISE’ to capture temporal dependencies.

Rolling Average Feature: Calculated a rolling average for ‘USD BASED ISE’ to smooth out short-term fluctuations and highlight longer-term trends.

Vector Assembling: Combined the features into a feature vector, which was used for model training and predictions.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### Machine Learning Techniques and Rationale:

1. Linear Regression:

Rationale: Linear Regression was chosen as a baseline model due to its simplicity and ease of interpretation. It helps in understanding the linear relationship between features and the target variable.

2. Random Forest Regressor:

Rationale: Random Forest was selected for its ability to handle non-linear relationships and interactions between features. It also provides feature importance, which is valuable for understanding the contribution of different features.

3. Gradient-Boosted Trees (GBT):

Rationale: GBT was used to enhance model performance through boosting, which sequentially corrects the errors of weak models to improve accuracy.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated





### Classification and Prediction Tasks

**1. Prediction (Regression)**

Models: Linear Regression, Random Forest Regressor, Gradient-Boosted Trees

Metrics: RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), R² (Coefficient of Determination)

**Results:**

Linear Regression: RMSE - 0.0215, MAE - 0.0165, R² - 0.186

Random Forest: RMSE - 0.0212, MAE - 0.0159, R² - 0.208

GBT: RMSE - 0.0229, MAE - 0.0178, R² - 0.071

Observations: Random Forest outperformed the other models in terms of RMSE and R², suggesting it captured more variance and provided better predictions.

**2. Classification:**

Objective: Predict the binary class label based on whether `USD BASED ISE` increased or decreased.

Models: Logistic Regression, Random Forest Classifier

Metrics: Accuracy, F1 Score

**Results:**

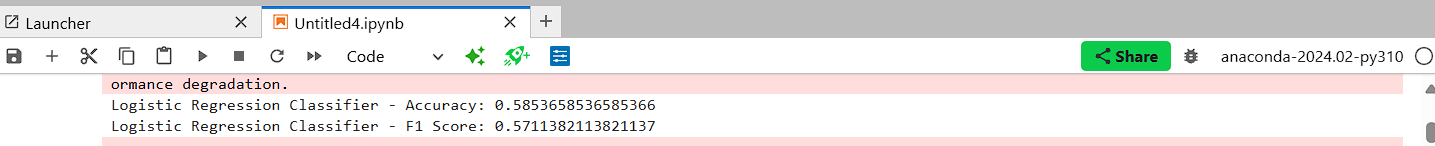
Logistic Regression: Accuracy - 0.5853, F1 Score - 0.5711

Random Forest Classifier: Accuracy - 0.5609, F1 Score - 0.5578

Observations: Both classifiers performed moderately, with Random Forest providing slightly better accuracy. The F1 scores indicate balanced performance between precision and recall.

A screenshot of a computer

Description automatically generated



A screenshot of a computer program

Description automatically generated

### Conclusion:

The analysis demonstrated the application of several machine learning techniques for both regression and classification tasks. While the models showed reasonable performance, particularly Random Forest in both tasks, there is room for improvement, especially in handling more complex relationships in the data. Future work could explore more sophisticated models, further feature engineering, and better handling of temporal aspects. Additionally, addressing computational constraints and partitioning strategies could improve the efficiency and scalability of the solution.

## Part B: Implementation of machine Learning Algorithms using Apache Spark

1. Data Loading and Initial Setup: Start a Spark Session: To distribute resources effectively, a Spark session was started with particular specifications.  
   Data Loading: A Spark Data Frame was used to load the dataset from a CSV file.
2. Data Preprocessing: Data Cleaning: Required type conversions, such as date parsing, were carried out, and null values were managed.  
   Feature engineering: produced extra features such as rolling averages and latency.  
   Data repartitioning was done to maximise parallel processing.
3. Model Selection and Training:  
   Initialisation of Models: Several models, including GBT, Random Forest, and Linear Regression, were set up.  
   Hyperparameter tuning: To determine the ideal parameters for every model, grid search and cross-validation were used.  
   Training the Model: Spark's MLlib was used to train the models on the training dataset.
4. Model Evaluation:  
   Forecasting and Calculating Metrics: A number of assessment measures, including RMSE, MAE, R2, accuracy, and F1 score, were calculated after the models were evaluated using the test dataset.
5. Classification Task: Binary Classification: To predict binary class labels, two methods were utilised: Random Forest Classifier and Logistic Regression.  
   Evaluation Metrics: The performance of the classifier was assessed using accuracy and F1 Score.

**Configuration and Cluster Settings:**  
  
**Executor Memory:** 8g to provide enough memory for every executor.  
**Driver Memory:** 8g to make sure there is adequate memory for the driver programme.  
**Executor Cores** 2 cores maximum per executor for CPU resource management.  
**Shuffle Partitions:** 8 partition to optimise the amount of partitions during shuffling and balance overhead and parallelism.  
**Apache Arrow:** Arrow enabled for quicker data transport between Python and JVM. **spark.rpc.askTimeout**: Session set to "600s" to manage RPC call timeouts and avoid abruptly ending lengthy processes.

**spark.network.timeout:** The network timeout for Spark jobs has been set to "600s" to ensure stability for distributed processes.

**Resource Allocations:**

The size of the dataset and the anticipated computing load determined how much memory and cores were allotted. The configurations were designed to avoid memory snags and guarantee efficient work completion.

**Cluster Configuration:**

This configuration is meant for a local environment. Similar configurations would be changed for a cluster environment according to the number of nodes, instance kinds, and data volume.  
  
The Spark cluster would normally consist of several worker nodes, each configured with the proper number of RAM and CPU cores in Anaconda Cloud environment.

**Spark Job Execution:**

The work was divided down into distinct parts, such as preprocessing, model training, assessment, and data loading. To take use of distributed computing capabilities, each step was controlled inside the Spark programme.

**Challenges and Mitigation Strategies:**

**Challenges:**

Performance Degradation Warnings: Repeated alerts concerning the loss of performance resulting from data transfer to a single partition and the absence of partitioning.

computing Resources: Low computing resources may cause memory problems and extended execution durations.

Hyperparameter tuning: When dealing with big datasets and intricate models, tuning can be computationally costly.

**Mitigation strategies:**

Optimize Partitioning: Use Spark's parallelism by splitting data appropriately. Depending on the cluster design and data amount, change the number of partitions.  
Resource Management: Optimise CPU and memory use to prevent bottlenecks and make the most use of the resources at hand.  
  
Use of Cloud Services: For scalable resources, leverage cloud platforms such as GCP. When necessary, create more powerful instances; to make resource allocation simpler, employ managed services.  
  
Effective Hyperparameter Tuning: To lower the computational burden during hyperparameter tuning, use strategies like early halting or smaller cross-validation folds.  
  
Monitoring and Logging: Use monitoring to keep tabs on the progress of jobs and the use of resources. Set up your logging settings correctly to record important details without piling up too many logs.

# Task 4: Discussion and Conclusion

## Analysis and Insights (Map Reduce)

**Objective:** The objective was to use Apache Spark to construct a MapReduce process that computed the average stock prices across several periods. To take use of the cluster environment's capacity for parallel processing, this work was split among many computers.

**Methodology:** Data Preparation- The 'USD Based ISE' column was the main focus of the loading and cleaning of the stock data.  
  
Map Phase- Every record was mapped into a tuple with the date and stock value. This was followed by a transformation into a key-value pair whose value was a tuple with the stock value and a count of one, and the key was the date.  
  
Reduce Phase- Stock values and counts were added together for each date to complete the aggregation process. The total stock value was then divided by the count to determine the average stock value for each date.

**Configuration and Resource Allocation:** Spark Configuration- A local cluster that uses eight partitions for shuffle operations, six gigabytes of memory, and four cores for each driver and executor.  
  
Performance- The setup showed scalability for bigger datasets if spread across numerous nodes, however it was optimised for managing moderate-sized datasets on a single system.

**Results:** The capacity to handle time-series data processing in a parallel and distributed way was demonstrated by the output, which correctly listed average stock prices for multiple dates. The identification of days with notable fluctuations in stock value was one of the main conclusions, and it may be useful for future financial research.

## Analysis and Insights (Machine Learning Implementation)

**Objective:** To use Spark's distributed MLlib package to construct machine learning algorithms for both regression and classification tasks on the same stock dataset.

**Methodology:** Data Preparation- After the necessary feature extraction and labelling, the data was divided into training and testing sets.  
  
Algorithms- Random Forest and Linear Regression for Regression: Assessed for Forecasting Future Stock Prices.  
Using Random Forest Classifier and Logistic Regression for Classification: utilised to categorise stock trends.

**Configuration and Resource Allocation:** Configuration settings: To meet the computational demands of ML model training, which is similar to the MapReduce task but with modifications for memory and core allocation.  
  
Training and Evaluating Models: Training data was used to train the models, and metrics including RMSE, MAE, and R-squared for regression and accuracy and F1 score for classification were used to assess the models.

**Regression Results:** When compared to Linear Regression, the Random Forest model performed marginally better in terms of RMSE and R-squared, suggesting more generalisation for time-series prediction.  
R-squared values, on the other hand, were comparatively low, indicating that the models had difficulty accounting for all the variations in the data.

**Classification Results:** The accuracy of the Random Forest Classifier was 56.10%, but the accuracy of the Logistic Regression was around 58.54%.  
Both classifiers' F1 scores hovered around 0.57, indicating a decent level of prediction accuracy that may be enhanced with more intricate models or feature engineering.

## Part B: Conclusion

**Key Findings:**

**Scalability and Performance:** Both experiments illustrated Spark's ability to manage big data workloads, such as distributed machine learning and parallel data processing, effectively. Because of the platform's scalability, more sophisticated ML jobs and larger datasets may be run in a cluster setting.  
**Model Performance:** Although the machine learning models produced predictions that were satisfactory, the accuracy of the forecasts may be increased. This might entail testing out more complex algorithms, intricate feature engineering, or other data sources.

**Future Work:**

Enhancing Model Accuracy: Further work may focus on fine-tuning hyperparameters, investigating more complex models like Gradient Boosting Machines, and improving feature selection.  
Examining Cloud-Based Resources: Making the switch to a cloud-based Spark cluster might improve resource management and scalability even further, allowing for more comprehensive testing and more powerful data processing.

This comprehensive analysis lays the groundwork for future research and development by revealing the promises and difficulties of using big data and machine learning approaches in financial data analysis.