االجمهورية التونسية
République Tunisienne
وزارة التعليم العالي و البحث العلمي
Ministry of Higher Education and Scientific Research
كلية الأعمال في تونس
Tunis Business School

End-of-Semester Project

Field of Study: **Information Technology**

Specialization: **Object oriented programming using Java**

Theme

Web-Based Data Processing and Prediction Application

Presented by
Hassen Dabboussi
Houssem Souari
Sinen Frej

Defended on: [**Jan, 2025**]
In front of the jury composed of

Mrs. Ameni Azzouz the Jury

Academic Year: 2024/2025

Acknowledgment

We take this opportunity to express our heartfelt gratitude to everyone who supported us in completing this project.

Firstly, we extend our sincere appreciation to our course instructor, Mrs. Ameni Azouz, for creating an engaging and insightful course. Her dedication and expertise equipped us with the necessary tools and knowledge to approach this project with confidence.

We are also very grateful for the excellent teamwork and collaboration between our group members. The hard work, creativity, and commitment of each member played an integral role in the successful realization of this project.

Finally, we are grateful to our friends and family for their constant encouragement and support throughout this journey.

This project has been an enriching learning experience and we are proud of our collective achievement.

Contents

I. Introduction

Web-Based Data Processing and Prediction Application

This project provides a web interface that allows users to import their datasets and perform automated preprocessing, prediction, and report generation using a Java-based backend. The application is designed to streamline data analysis tasks with minimal user effort.

A. Features

Data Import: Upload datasets through a user-friendly web interface. Automated Preprocessing: Handle missing values, normalize data, and clean the dataset. Prediction: Use machine learning models to predict outcomes based on the processed data. Report Generation: Generate insightful reports and visualizations of the results. Java Backend: Efficient processing and predictions using a robust Java-based engine.

B. How it works

User Uploads Data: The user uploads their dataset through the web interface. Preprocessing: The application automatically cleans and preprocesses the data. Prediction: A tailored prediction model is applied to the dataset. Reporting: Results are displayed on the interface and can be downloaded as a report.

C. Technologies used

Backend: Java: Core logic for preprocessing, predictions, and reporting. Spring Boot (Optional): For API development and backend services. Frontend: HTML/CSS/JavaScript: For the web interface. Bootstrap: Responsive design for better usability. Machine Learning: **Tribuo

D. Class Diagram

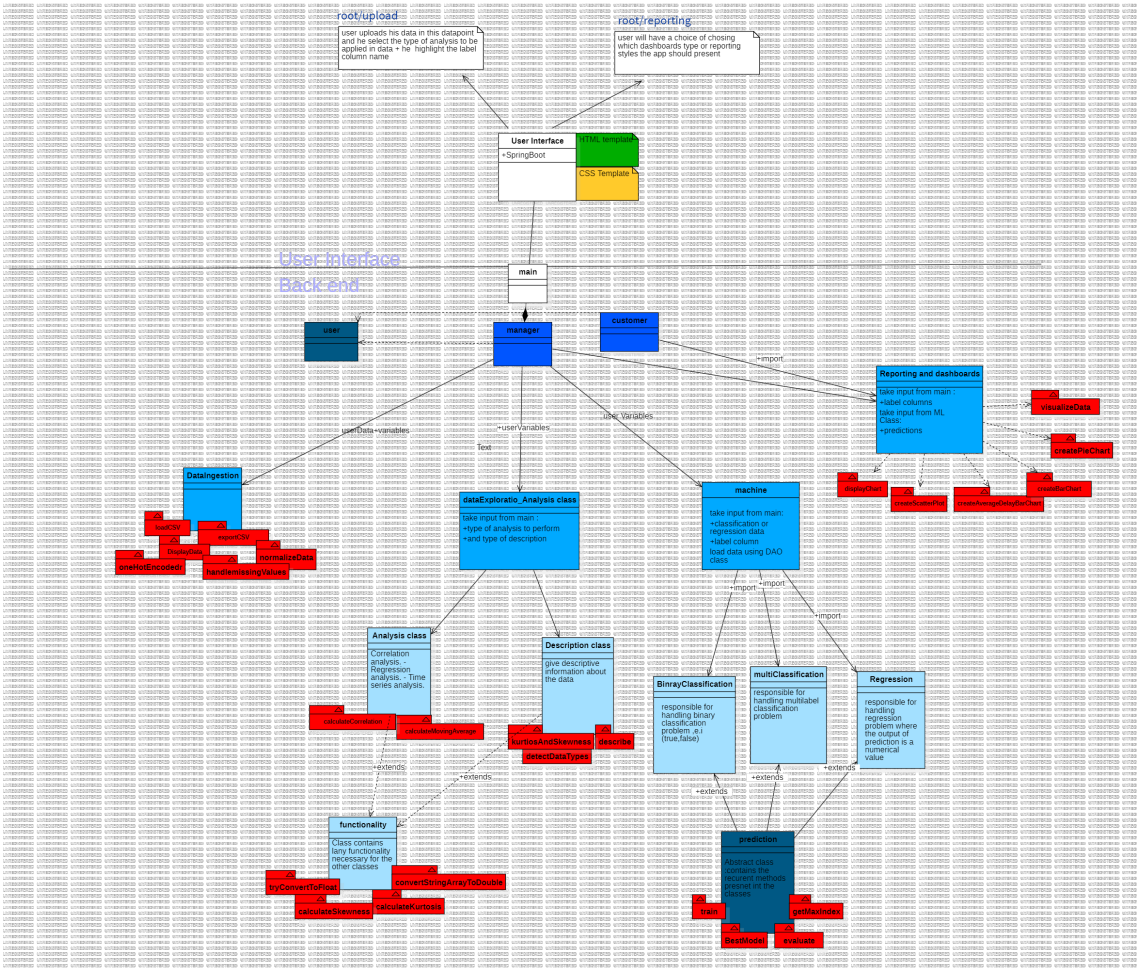

Figure I.1: Class Diagram of the Project

II. Back-End Classes

A. Data Ingestion Class

1. Purpose and Functionality of the CSVPreprocessor Class

The primary goal of the `CSVPreprocessor` class is to simplify the data ingestion process by performing the following tasks:

- Import data from CSV files provided by the user.
- Display the loaded dataset for verification and review.
- Conduct essential data preprocessing steps, including:
  - Managing missing values in the dataset.
  - Normalizing numerical columns for consistency.
  - Applying one-hot encoding to convert categorical data into a machine-readable format.
- Save the cleaned and preprocessed data into a new CSV file.

2. Classes and Implementation

- **CSVPreprocessor Class**: This class handles the entire data ingestion workflow, from importing the raw dataset to preprocessing it and exporting the refined version. It is designed to provide a streamlined and user-friendly experience for handling CSV files.

3. Conclusion

The `CSVPreprocessor` class plays a vital role in the data processing pipeline by automating key tasks like data cleaning, transformation, and export. By addressing common challenges such as missing values and inconsistent data formats, the class ensures that datasets are well-prepared for further analysis and machine learning tasks. This functionality not only saves time but also improves the reliability and accuracy of subsequent data-driven processes.

B. DataExploration_analysis

1. Purpose and Functionality of DataExploration_analysis Class

the main purpose of this class is to perform an EDA(Explaratory Data Analysis) and returns Information concerning our data form and components

- Load data from CSV files imported from the Data Ingestion Class
- perform a type of analysis tailored to our data
- returns descriptive information from our columns

2. Classes and Implementation

The Data Exploration Class contains the following key Classes:

- **DataExpo(Path,userInput)**: this is the main public class in our data exploration phase it will pick the right analysis to apply in our data.

- **Description**: This class provides informative details about the dataset, such as the mean of each column, data types of the columns, and other summary statistics. It helps in understanding the overall structure and characteristics of the data, which are essential for preprocessing and analysis.

- **Analysis**: This class performs specific types of analysis on the dataset. For example, it can apply time series analysis or correlation analysis, depending on the nature of the data. It helps uncover underlying patterns, trends, and relationships within the dataset, enabling better decision-making and modeling.

- **Functionality**: This class provides utility methods that support the other classes in their tasks. It includes methods like 'tryConvertToFloat' to convert columns to float data type, 'calculateKurtosis' and 'calculateSkewness' to compute statistical properties of the data distribution, and 'convertStringArrayToDouble' to handle the conversion of string arrays into numerical values. These methods facilitate data cleaning, transformation, and statistical analysis.

3. Conclusion

The Data Exploration and Analysis Class provides a structured approach to understanding and preparing the dataset, ensuring key insights and data quality for effective analysis and modeling. c

C. Machine Learning Class

1. Purpose and Functionality of Machine Learning Class

The primary purpose of the Machine Learning Class is to:

- Load data from CSV files imported from the Data Ingestion Class

- Perform initial machine learning algorithms tailored to the current data

- evaluate the performance by comparing the predictions by the actual values.

2. Classes and Implementation

The Machine Learning Class contains the following key Classes:

- **machine(Path,ModelType)**: this is the main class in the ML phase .It is responsible for picking the the type of algorithms to run on our Data.

- **BinaryClassification**: this class apply a binary classification on our data (e.g. True or False) .

- **MultiLabelClassification**: Applies a multi-label classification label algorithme on our data in our case our class will use CART algorithme . [1]
- **Regression**: Applies a 3 different type of regression algorithms (XGBoost regressor, CARTRegressor,linear decay SGD) and returns the most suitable model by comparing the R^2 produced by each algorithm.
- **Prediction**: this is an abstract class used by the previous classes. it contains 3 abstract methods:
    1. train:responsible for training our data
    2. evaluate:responsible for evaluating the model on an evaluation dataset.
    3. bestModel: responsible for returning the most accurate results among all available options by selecting the optimal hyperparameters.

3. Conclusion

The Machine Learning Class provides a modular and efficient approach to managing Prediction tasks. Its functionality ensures the quality and consistency of the output, laying the foundation for successful data analysis and processing.

D. Reporting and Dashboards Class

1. Purpose and Functionality of ReportingAndDashboards Class

The Reporting and Dashboards Class is designed to generate insightful visualizations and dashboards that summarize the processed data. The main functions include:

- Data Aggregation: Analyze datasets to compute metrics such as flight delays, route statistics, and flight statuses.
- Visualization: Create a variety of charts, including:
    - Pie Charts for visualizing the distribution of flight statuses.
    - Bar Charts for displaying the most common routes and average delays.
    - Scatter Plots for analyzing relationships between time of day and average delays.
- interactive Dashboards: Render visual outputs in a graphical interface using JFreeChart.

2. Classes and Implementation

- Visualization Functions:
    - createPieChart: Generates pie charts to visualize categorical distributions like flight status counts.
    - createBarChart: Produces bar charts for data such as route frequencies.
    - createAverageDelayBarChart: Displays average delays grouped by aircraft type.
    - createScatterPlot: Plots the relationship between delay averages and time of day.

- Data Aggregation and Analysis:
  - The class reads and processes datasets using org.apache.commons.csv.CSVFormat.
  - Data is aggregated into maps for easy access and manipulation (e.g., Map<String, Integer> for counting occurrences).
- Graphical Interface:
  - Visualization results are presented in a GUI using ChartPanel and JFrame.

3. Conclusion

The Reporting and Dashboards Class provides an essential bridge between raw data and actionable insights. Its ability to create diverse visualizations ensures the end-user can interpret the analysis results effectively. By leveraging libraries like JFreeChart and CSV utilities, this class enhances the accessibility and utility of the application.

E. UserRole Interface

1. Purpose and Functionality of the UserRole Interface

The `UserRole` interface defines methods that control user actions. Depending on the role (Manager or Customer), only the necessary methods are implemented.

- **Purpose**: To enforce role-specific functionality for users
- **Methods used**:
  - `void viewDataset()`
  - `void generateGraphs()`
  - `void preprocessData()`
  - `void triggerModelTraining()`

2. Abstract Class `User`

The `User` abstract class stores common user information like ID, name, and email. It ensures that every user type shares these details.

- **Methods**:
  - `abstract void displayInfo()`

3. Manager Class

The `Manager` class extends the `User` class and implements the `UserRole` interface. Managers have access to all features of the system.

- **Purpose** : Represents users with managerial privileges.
- **Methods Implemented**:
  - `void viewDataset()`
  - `void generateGraphs()`
  - `void preprocessData()`
  - `void triggerModelTraining()`

4. Customer Class

The `Customer` class also extends the `User` class and implements the `UserRole` interface. Customers only have access to viewing datasets and generating graphs.

- **Purpose**: Represents users with restricted access.
- **Methods Implemented**:
  - `void viewDataset()`
  - `void generateGraphs()`

III. Front-End classes

A. web interface

1. Frontend:File Upload Form and Download Button

The HTML form includes:

- An input field of type file to allow users to select a file.
- A POST request sent to the backend to handle the file upload.
- A download button or link that triggers the file download using a GET request.

2. Backend: Java Servlets

File Upload Handling

The FileUploadServlet processes the form submission:

It retrieves the file from the request using the Part API. The file is stored in a designated folder on the server, such as uploads/.

File Download Handling

The FileDownloadServlet serves files for download:

- The file path is retrieved from the request parameter.
- The servlet streams the file as a downloadable attachment.

3. Folder Structure for File Storage :

Uploaded files are stored in a folder named uploads, created on the server:

```
 WebApp/
-- src/
    main/
        java/
            FileUploadServlet.java
-- webapp/
    upload.html
    download.html
-- pom.xml
-- WebApp.iml (IDEA project file)
```

4. Conclusion

This simple file upload and download system demonstrates the integration of Java Servlets with HTML and CSS to handle file management tasks in a web application. The solution can be easily extended or integrated into larger projects requiring file storage and retrieval functionality.

IV. Case Study :Tunis air Flight delays data

A. classes flow from user input to reports output

Will be updated in the next Submission

V. Conclusion

This Report has provided a comprehensive overview of our end-to-end java project from the user interaction with the UI and the submission of the data inputs to the triggering of the back-end classes and the final output which is represented as reports and graphs. Although we have already developed our MVP , many issues remain to be addressed.Here we list some open problems from the perspectives of our developers: for our data ingestion ,the class is still not compatible for various file types(.yaml,.XML..).in addition,the ML class lacks the side of unsupervised learning.