# Measuring Similarity Between Old and New Fashion Styles

Research Final Report

Weerasinghe H, IT21229848

B.Sc. (Hons) Degree in Information Technology specialized in

Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

February 2025

# DECLARATION

“I declare that this is our own work, and this dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).”

|  |  |  |
| --- | --- | --- |
| Name | Student ID | Signature |
| Weerasinghe H | IT21229848 | A black signature on a white background  AI-generated content may be incorrect. |

# ABSTRACT

This report presents the design, development, and validation of a machine learning-based system for measuring and classifying the operational similarity between old and new activewear fashion styles. The focus of this study lies in analyzing Operation Breakdown (OBD) reports—structured documents that capture the step-by-step manufacturing process of a given garment. These reports are critical in production planning, as they determine machine usage, labor allocation, and line configuration. However, manually assessing their similarity to determine resource reuse potential is time-consuming and error prone. This project introduces a data-driven alternative by leveraging Natural Language Processing (NLP) and statistical similarity measures to automate this evaluation process.

The proposed system employs TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert OBD report text into numerical feature vectors that represent the frequency and importance of operational steps. Cosine similarity is then applied to these vectors to compute a similarity score ranging from 0 (completely different) to 1 (identical). This score is categorized into High, Moderate, or Low similarity, offering an intuitive and actionable metric for production managers. The entire pipeline is integrated into a web-based tool, allowing users to upload two OBD files and instantly receive a visualized similarity result, complete with operational interpretation and strategic recommendations.

In testing scenarios, the system consistently identified style pairs with overlapping operations (e.g., two sports bras) with high similarity scores (>75%), while distinctly different styles (e.g., bra vs. leggings) scored below 40%. These insights directly support production optimization: higher similarity implies minimal changes to existing production lines, reducing retooling costs, training needs, and transition time. Conversely, low similarity flags the need for operational restructuring. The backend system is developed using Python and Flask, while the frontend is built with JavaScript and Sweet Alert for dynamic visualization. The model has been validated using real OBD samples and evaluated with expert input from MAS Holdings' production teams.

This research contributes a scalable, fast, and intelligent approach to improving operational planning in fashion manufacturing. It bridges the gap between machine learning research and practical factory-level decision-making. Future improvements may include deep learning models for more nuanced semantic understanding, integration with ERP systems, and expansion to handle multilingual or image-based production documentation.

# ACKNOWLEDGMENTS

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Mr. Nelum Amarasena for their continuous support, valuable guidance, and insightful feedback throughout the course of this project. I also extend my thanks to MAS Holdings for providing the necessary data and real-world context that made this research possible. Special thanks to my team members and peers for their cooperation, and to my family and friends for their unwavering encouragement and support during this journey.

# TABLE OF CONTENTS

[Measuring Similarity Between Old and New Fashion Styles 1](#_Toc196131763)

[DECLARATION 2](#_Toc196131764)

[ABSTRACT 3](#_Toc196131765)

[ACKNOWLEDGMENTS 4](#_Toc196131766)

[TABLE OF CONTENTS 5](#_Toc196131767)

[LIST OF TABLES 7](#_Toc196131768)

[LIST OF FIGURES 8](#_Toc196131769)

[LIST OF ABBREVIATIONS 9](#_Toc196131770)

[INTRODUCTION & BACKGROUND 10](#_Toc196131771)

[1.1 Introduction 10](#_Toc196131772)

[1.2 Background Literature 10](#_Toc196131773)

[RESEARCH PROBLEM 12](#_Toc196131774)

[RESEARCH OBJECTIVES 13](#_Toc196131775)

[3.1 Main Objective 13](#_Toc196131776)

[3.2 Sub Objectives 13](#_Toc196131777)

[3.3 SMART Goals 14](#_Toc196131778)

[3.4 Evaluation Metrics 14](#_Toc196131779)

[METHODOLOGY 16](#_Toc196131780)

[4.1 Overview of the System Architecture 16](#_Toc196131781)

[4.2 Overview of the Individual Component Diagram 17](#_Toc196131782)

[4.3 Sub Tasks 17](#_Toc196131783)

[4.4 Tools & Technology 18](#_Toc196131784)

[4.5 System Development Process 19](#_Toc196131785)

[4.6 Gantt Chart 20](#_Toc196131786)

[4.7 Project Requirements 21](#_Toc196131787)

[4.7.1 Functional Requirements 21](#_Toc196131788)

[4.7.2 Non-Functional Requirements 21](#_Toc196131789)

[4.7.3 System Requirements 21](#_Toc196131790)

[4.7.4 Personal Requirements 21](#_Toc196131791)

[4.8 Commercialization Plan 22](#_Toc196131792)

[4.8.1 Commercialization Aspects of the Product 22](#_Toc196131793)

[4.8.2 Benefits to the End-Users 22](#_Toc196131794)

[TESTING & IMPLEMENTATION 23](#_Toc196131795)

[5.1 Implementation 23](#_Toc196131796)

[5.1.1 Front End 24](#_Toc196131797)

[5.1.2 Back End 28](#_Toc196131798)

[5.2 Testing 31](#_Toc196131799)

[RESULT & DISCUSSION 32](#_Toc196131800)

[6.1 Result 32](#_Toc196131801)

[6.2 Summary 33](#_Toc196131802)

[CONCLUSIONS AND FUTURE WORK 34](#_Toc196131803)

[References 35](#_Toc196131804)

[Appendix 36](#_Toc196131805)

# LIST OF TABLES

[*Table 1– Similarity Results from Sample OBD Pairs* 30](#_Toc196130724)

# LIST OF FIGURES

[*Figure 1 Gantt Chart* 20](#_Toc196131444)

[*Figure 2 Frontend - OBD Report Uploading Part* 24](#_Toc196131445)

[*Figure 3 Frontend - Sweet Alert* 25](#_Toc196131446)

[*Figure 4 dataset - OBD Report 1* 26](#_Toc196131447)

[*Figure 5 Dataset - OBD Report 2* 27](#_Toc196131448)

[*Figure 6 backend proof 1* 28](#_Toc196131449)

[*Figure 7 Backedn proof 2* 29](#_Toc196131450)

[*Figure 8 Backend - imported libraries* 30](#_Toc196131451)

# LIST OF ABBREVIATIONS

OBD – Operation Breakdown  
NLP – Natural Language Processing  
TF-IDF – Term Frequency-Inverse Document Frequency  
ML – Machine Learning  
API – Application Programming Interface  
UI – User Interface

# INTRODUCTION & BACKGROUND

## 1.1 Introduction

The rapid evolution of the fashion industry, particularly in activewear, demands innovative solutions for optimizing production efficiency. As the global demand for athletic and performance apparel continues to surge, manufacturers are under increased pressure to produce more styles within shorter lead times while minimizing cost and maintaining consistency. Companies like MAS Holdings, which manage high-throughput production lines, face significant logistical and operational challenges when switching between styles. These transitions can result in additional costs due to machine reconfiguration, labor retraining, and production downtime. Traditional methods of handling this involve manual comparisons of Operation Breakdown (OBD) reports, which detail the steps involved in producing each garment. However, this manual process is labor-intensive, subjective, and prone to inconsistency. Thus, automating the comparison of OBD reports using machine learning techniques presents a promising opportunity to reduce operational overhead and optimize resource allocation.

By quantifying the similarity between old and new styles, production managers can make informed decisions about reusing existing production lines or reallocating resources. A high similarity implies that a new style can be produced with the same setup as a previous one, saving time and cost. Conversely, a low similarity may require a shift in production strategy, potentially including new machinery or workforce adjustments. The significance of this research lies in its ability to reduce transition costs, improve production predictability, and enhance the overall agility of the fashion supply chain.

## 1.2 Background Literature

Over the past decade, machine learning (ML) and artificial intelligence (AI) have made significant inroads into the fashion industry. While early applications focused on trend forecasting and customer personalization, recent advances have broadened the scope to include manufacturing and operational processes. Visual-based ML models such as convolutional neural networks (CNNs) have been widely used to detect fashion attributes, classify product categories, and suggest matching apparel [1]. These methods have proven particularly effective in consumer-facing applications like online shopping and virtual try-on systems. However, much less research has been directed at the internal operations of fashion production, particularly in optimizing workflow based on design similarities.

Natural Language Processing (NLP), a subfield of ML, has shown substantial promise in analyzing unstructured text. In the retail space, NLP is used to analyze customer reviews, generate product descriptions, and power chatbots. Adebowale [1] emphasized the transformative potential of NLP in automating e-commerce processes. In this research, similar NLP techniques are applied to process structured OBD reports—technical documents containing production steps. By converting these documents into machine-readable formats and analyzing them with NLP, the system can extract meaningful insights into the operational structure of fashion items.

Sustainability, a growing concern in modern fashion manufacturing, is also indirectly impacted by operational efficiency. Rohil [2]illustrated how AI technologies can promote sustainable fashion through reduced waste, smarter material selection, and process optimization. This research aligns with such initiatives by proposing a cost-effective method to reuse existing production setups when possible, thereby minimizing waste and unnecessary resource deployment. In parallel, psychological factors explored by Singh [3]highlight how consumer perception is influenced by brand reliability, which can be strengthened through consistent and efficient production.

Overall, this project builds on existing advancements in ML and NLP by applying them to a niche but impactful area—similarity assessment of fashion production tasks. Through a review of existing literature, it is evident that while the technical foundations exist, their application to operational planning in garment manufacturing is still underexplored. This research addresses that gap with a focused, scalable solution tailored to the needs of high-volume apparel production.

# RESEARCH PROBLEM

In the context of industrial-scale fashion manufacturing, efficient management of resources—human, mechanical, and temporal—is crucial. MAS Holdings operates multiple production lines across its factories, each responsible for producing specific activewear garments such as sports bras, leggings, and jackets. Each line is composed of a fixed number of operators and machines assigned to perform unique tasks in sequence. Typically, these lines are optimized for a particular type of garment, allowing the staff to specialize and the machinery to be configured accordingly. The production schedule is often arranged on a biweekly basis, where a specific line completes one style before transitioning to another. However, this transition poses a critical operational dilemma: how similar is the new style to the previous one?

This is not a superficial aesthetic comparison, but a deep operational concern. Each activewear style is documented through an Operation Breakdown (OBD) report, which outlines every step involved in the production process. These steps include tasks such as fabric cutting, seam stitching, waistband attachment, and labeling. Variability in these steps significantly affects the tools, machines, and workforce needed to complete the garment. When the steps between two styles are highly similar, minimal changes are required—machinery remains as is, workers follow familiar protocols, and training time is negligible. However, if the steps are dissimilar, the line may need to be reconfigured, new machines may be introduced, and operators must be retrained, which disrupts efficiency and drives up costs.

The current practice involves manually reviewing the OBD reports to assess similarities, which is both time-consuming and prone to human error. Managers must make quick decisions under time pressure, and any misjudgment can lead to either underutilization of capacity or excessive, unnecessary investments in infrastructure. This manual process lacks objectivity and scalability, especially as product diversity increases. Additionally, there is no quantitative metric available that objectively compares operational steps between different styles.

This project addresses this challenge by proposing an automated solution that compares OBD reports using machine learning and NLP. By converting operational text into feature vectors and using cosine similarity to measure their alignment, the system generates a similarity score between 0 and 1. This score offers an objective and reproducible metric for operational similarity. A high similarity score (e.g., > 0.75) indicates the line can proceed with minimal adjustments, while a low score (e.g., < 0.35) flags the need for significant intervention. Through this model, MAS Holdings can save time, reduce transition costs, and enhance overall production planning accuracy.

# RESEARCH OBJECTIVES

## 3.1 Main Objective

The primary objective of this research is to design, develop, and validate a machine learning-based system that can automatically measure and classify the operational similarity between old and new activewear fashion styles, specifically by analyzing the structured steps recorded in Operation Breakdown (OBD) reports. These reports encapsulate the sequence of tasks, machine usage, manpower, and time allocation required to manufacture a particular garment. The proposed system transforms these textual documents into machine-readable formats using Natural Language Processing (NLP) techniques such as text normalization, tokenization, and TF-IDF vectorization. Cosine similarity is then employed to compare the operation vectors and generate a similarity score on a continuous scale from 0 to 1, which is translated into actionable categories (e.g., High, Moderate, or Low similarity).

The goal is for this score to serve as a data-driven metric for production managers, allowing them to make informed decisions about whether an existing production line can be reused for a new style or if additional resources—such as new machinery, reconfiguration time, or additional training—are necessary. This system eliminates the need for manual, error-prone analysis of OBD reports and introduces objectivity into the planning process. By increasing the accuracy and speed of these decisions, the tool aims to lower production transition costs, minimize downtime, and optimize factory throughput. Furthermore, the platform is intended to be easily integrable into existing factory workflows and scalable to different product types and manufacturing environments, thereby enhancing both short-term operational efficiency and long-term strategic planning in the apparel industry.

## 3.2 Sub Objectives

* **To design and implement a robust text preprocessing module** that standardizes and cleans OBD reports for machine learning consumption. This includes operations such as case normalization, punctuation removal, whitespace trimming, and the elimination of irrelevant tokens or formatting artifacts that could interfere with feature extraction.
* **To develop a feature extraction mechanism using advanced NLP techniques**, particularly Term Frequency-Inverse Document Frequency (TF-IDF) vectorization, which translates textual step sequences into meaningful numerical representations suitable for similarity computation.
* **To apply cosine similarity and optional clustering algorithms** (such as K-Means or Agglomerative Clustering) on the generated feature vectors to measure the degree of operational overlap between two OBD reports, enabling both pairwise similarity scoring and grouping of stylistically similar products.
* **To design and deploy a web-based user interface** that allows factory planners and production managers to upload historical and proposed OBD files, view the similarity results in real time, and receive actionable recommendations based on system outputs. This interface includes error handling, visualization modules, and responsive feedback elements like Sweet Alert.
* **To validate the model’s accuracy and reliability** by testing it on a labeled subset of OBD report pairs annotated by domain experts. Evaluation metrics will include similarity score agreement, prediction accuracy thresholds, and qualitative feedback from actual production teams to assess operational usefulness and model trustworthiness.

## 3.3 SMART Goals

* **Specific**: To develop a machine learning–driven similarity evaluation system that processes two uploaded Operation Breakdown (OBD) reports, applies text preprocessing and vectorization techniques, and outputs a quantitative similarity score. This score will support decision-making in resource allocation for activewear production lines.
* **Measurable**: The system should demonstrate at least **85% alignment** with similarity classifications provided by human experts in a validation dataset. Additional metrics such as precision, recall, and processing time will be used to measure system performance during evaluation.
* **Achievable**: The project will be developed using **open-source machine learning libraries** such as Scikit-learn for feature transformation, NLTK for natural language processing, and Flask for backend development. Data for training and validation will be sourced from actual OBD archives and annotated under the guidance of MAS Holdings' domain experts.
* **Relevant**: This system directly contributes to **enhancing production efficiency** by enabling faster, more objective similarity analysis between fashion styles. It addresses a critical operational bottleneck in resource planning and offers measurable benefits in terms of reduced downtime, training needs, and machine reconfiguration.
* **Time-bound**: The project is structured within a **3-month timeline**, with deliverables spaced across data collection (Week 1–2), model development (Week 3–6), system integration (Week 7–8), testing and validation (Week 9–11), and final documentation and deployment (Week 12).

## 3.4 Evaluation Metrics

* **Accuracy**: The degree to which the system's similarity classifications align with human expert evaluations. Accuracy will be quantified as the percentage of cases where the predicted similarity category (High, Moderate, or Low) matches expert-assigned labels within a predefined threshold (e.g., ±10%). This metric validates the model's ability to replicate expert judgment.
* **Processing Time**: The system's responsiveness in real-world use, measured from the moment of file submission to similarity score output. The target benchmark is set at **less than 2 seconds** per comparison on average. This ensures usability in fast-paced production environments where quick decisions are required.
* **Usability Score**: Qualitative feedback collected from production managers and line planners at MAS Holdings using structured surveys and direct observation. The focus will be on interface intuitiveness, clarity of outputs, and ease of use. A Likert scale (1–5) will be used to measure perceptions of system usefulness and visual communication.
* **System Robustness**: The tool’s ability to handle challenging or non-standard inputs without crashing or producing invalid results. This includes dealing with incomplete OBD files, reports with inconsistent formatting, or those containing unstructured or noisy data. Error handling routines and fallback mechanisms will be tested using predefined edge case scenarios.

# METHODOLOGY

## 4.1 Overview of the System Architecture

The system architecture of the similarity evaluation tool is designed as a **modular and scalable pipeline**, built to support maintainability, flexibility, and ease of integration with larger manufacturing planning systems. It consists of five primary layers: **Data Ingestion, Preprocessing, Feature Extraction, Similarity Computation, and Visualization**. Each of these components operates independently and communicates through well-defined interfaces, ensuring that the system can evolve over time without significant redesign.

* **Data Ingestion**: This stage serves as the entry point for users to submit their Operation Breakdown (OBD) reports through a clean and responsive web interface. The interface allows for uploading two text-based OBD files—typically one representing a previous style and one for a new style under consideration. Uploaded files are immediately passed to the backend via RESTful API endpoints built with Flask.
* **Preprocessing**: Once received, the files undergo a robust text preprocessing phase. This includes the removal of noisy or irrelevant content such as special characters, URLs, numeric identifiers, and redundant whitespace. Advanced **Natural Language Processing (NLP)** techniques like lowercasing, stopword elimination, and tokenization are applied to standardize the text for downstream processing. These steps are critical for ensuring that meaningful patterns are retained while irrelevant text artifacts are discarded.
* **Feature Extraction**: The cleaned textual content is then transformed into structured numeric vectors using **TF-IDF (Term Frequency-Inverse Document Frequency)** vectorization. This method evaluates the importance of each operational term in the context of the document corpus, emphasizing frequently used but uniquely relevant terms. The resulting feature vectors provide a quantitative foundation for similarity assessment.
* **Similarity Computation**: The core analytical layer uses **cosine similarity** to compute a numerical value that reflects the degree of operational resemblance between the two OBD reports. Cosine similarity was chosen for its effectiveness in high-dimensional sparse vector spaces, making it ideal for document-level comparison. The similarity score ranges from 0 (completely dissimilar) to 1 (identical), and is then mapped to qualitative categories: High, Moderate, or Low Similarity.
* **Visualization**: The output is returned to the frontend where it is visualized in an interactive format. A dynamic alert box, powered by **SweetAlert**, displays the similarity score with an intuitive message and color-coded cues (e.g., green for high similarity, red for low). This feedback mechanism makes it easy for non-technical users—such as production line managers—to interpret and act on the results quickly. The architecture also supports future extensibility, such as batch processing, database integration, or trend prediction modules.

This layered architecture ensures that the system remains modular, testable, and capable of adapting to additional complexity or scaling to enterprise-level deployments in the future.

## 4.2 Overview of the Individual Component Diagram

The individual component developed in this research is the **backend similarity engine**, which constitutes the core logic responsible for processing OBD files and determining the similarity between them. This component is activated upon file upload and acts as the analytical heart of the system. It is designed to be stateless, modular, and REST-compatible, ensuring it can be deployed independently or as part of a larger production management system.

Upon receiving file paths from the frontend, the similarity engine reads and parses the contents of each uploaded OBD report. The component then initiates a **text preprocessing pipeline** that standardizes the input by performing operations such as punctuation removal, case normalization, whitespace cleanup, and stopword filtering. This ensures consistency across diverse input styles and mitigates the effects of noise and formatting inconsistencies in the reports.

Next, the **feature extraction process** uses the **TF-IDF vectorizer** from the Scikit-learn library to convert the cleaned text into high-dimensional feature vectors. These vectors numerically represent the frequency and contextual relevance of each term (i.e., operation step) in the OBD reports. The vectors are then passed to a **cosine similarity function**, which computes a similarity score ranging from 0 to 1. This score quantitatively expresses the degree of operational overlap between the two garment styles.

The raw similarity score is interpreted and mapped to categorical labels:

* **Highly Similar**: Score ≥ 0.75
* **Moderately Similar**: 0.40 ≤ Score < 0.75
* **Dissimilar**: Score < 0.40

These classifications are then sent back to the frontend along with the numeric similarity score. On the user interface, results are displayed using **SweetAlert**, which provides an interactive and visual summary of the analysis (e.g., score percentage and color-coded status).

The component was implemented entirely in **Python using the Flask micro-framework**, which simplifies routing and API construction. The use of Flask allows the similarity engine to expose clean endpoints (/upload, /process, /similarity) that can be triggered by frontend HTTP requests. The architecture is designed for portability and can easily be containerized using Docker or scaled using a microservices architecture if needed.

Overall, this component is not only functional but also extendable, with opportunities to incorporate machine learning models beyond cosine similarity, support only for CSV file types and deeper integration with enterprise-level production planning tools.

## 4.3 Sub Tasks

* **Design and implement a robust text preprocessing pipeline** to clean and normalize raw OBD report inputs. This includes operations such as removing punctuation, stopwords, special characters, and redundant spacing, ensuring that the resulting text is machine-readable and semantically consistent for downstream analysis.
* **Develop a feature extraction module using TF-IDF vectorization** to transform cleaned text into high-dimensional numerical representations. This step is crucial in quantifying term importance and capturing the structural essence of operation steps across different styles.
* **Implement a cosine similarity computation module** that takes TF-IDF vectors as input and calculates the similarity score between OBD pairs. The module outputs a normalized score (0 to 1), representing the operational alignment between the two reports.
* **Define similarity thresholds and assign categorical labels** (e.g., Highly Similar, Moderately Similar, Dissimilar) based on empirical observations and expert input. This makes raw similarity scores more interpretable and actionable for non-technical users.
* **Create and document RESTful API endpoints** using the Flask framework to enable seamless communication between the backend similarity engine and the frontend UI. These endpoints include file upload, processing initiation, and result retrieval.
* **Integrate SweetAlert into the frontend interface** for delivering real-time, user-friendly visual alerts. These alerts display the similarity score and category in a clean and interactive manner, enhancing the decision-making experience for production planners.
* **Conduct thorough testing and validation** using a curated set of OBD report pairs. This involves unit testing for individual functions, integration testing between frontend and backend, and validation against expert-reviewed labels to ensure accuracy and reliability.

## 4.4 Tools & Technology

The project leverages modern programming tools and frameworks for both machine learning and web development

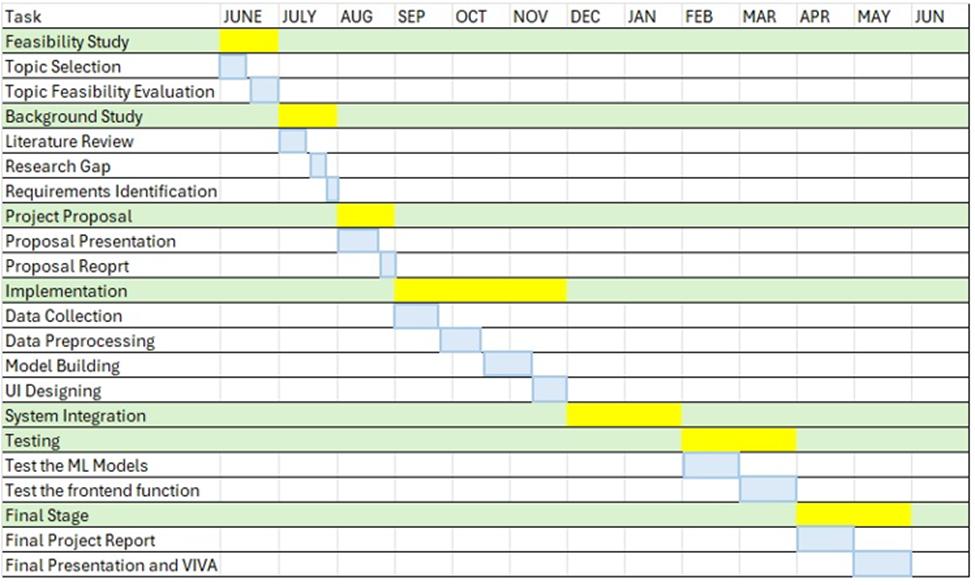
* **Programming Language**:  
  Used **Python 3.10** for backend development due to its wide support for machine learning, natural language processing, and web development.
* **Machine Learning Library**:  
  Employed **Scikit-learn** for:
  + TF-IDF vectorization (feature extraction)
  + Cosine similarity computation (text similarity measurement)
* **Web Framework (Backend)**:  
  Utilized **Flask** to:
  + Handle routing and API endpoints
  + Manage file uploads and processing logic
  + Integrate ML components with frontend requests
* **Frontend Technologies**:  
  Built the user interface with:
  + **HTML5** and **CSS3** for structure and design
  + **JavaScript** for interactivity and API calls
* **Visualization & Alerts**:  
  Integrated **SweetAlert** for:
  + Displaying real-time similarity results
  + Enhancing user experience with color-coded alerts
* **Database**:  
  Used **MySQL** to:
  + Store metadata about uploaded files
  + Log processing history and results
  + Support traceability and future analytics
* **Version Control**:  
  Adopted **Git** and **GitHub** for:
  + Source code versioning
  + Collaboration and issue tracking
  + Backup and documentation

This combination of tools provided a strong foundation for developing a scalable, fast, and user-friendly similarity detection system suitable for use in real manufacturing environments like MAS Holdings.

## 4.5 System Development Process

The system was developed following an **Agile methodology**, which emphasized adaptability, stakeholder collaboration, and iterative refinement. The development cycle was structured around short, focused sprints that allowed for incremental progress and continuous validation. The project began with **requirements gathering and domain analysis**, where the team worked closely with stakeholders at MAS Holdings to understand the operational structure of OBD reports and identify key user needs. This initial research informed the architecture of the system. The first sprint focused on the **design and prototyping of the backend**, establishing a scalable pipeline and drafting the core APIs. In subsequent sprints, efforts shifted to implementing the **text preprocessing pipeline**, **TF-IDF vectorization**, and the **cosine similarity computation module**, each tested independently for accuracy and efficiency. Regular sprint reviews were conducted with stakeholders to verify functionality and ensure that the system remained aligned with operational goals. In the final development phases, the backend was **integrated with the frontend interface**, and focus was placed on **testing**, **optimization**, and **user experience refinement**. Documentation and evaluation checkpoints were maintained throughout, ensuring that each module met quality standards before progressing. This iterative and collaborative approach contributed to the development of a robust, user-friendly, and production-ready similarity evaluation tool.

## 4.6 Gantt Chart

The Gantt chart for the system development lifecycle includes the following phases:  
- Week 1–2: Requirement Analysis and Data Collection  
- Week 3–4: Preprocessing Module and TF-IDF Setup  
- Week 5: Cosine Similarity and Result Interpretation Logic  
- Week 6: Backend API and Frontend Integration  
- Week 7: Testing, Validation, and Feedback Incorporation  
- Week 8: Finalization and Documentation  
  


*Figure 1 Gantt Chart*

## 4.7 Project Requirements

### 4.7.1 Functional Requirements

- Upload OBD files via frontend UI.  
- Process uploaded text files and clean content.  
- Perform vectorization and similarity calculation.  
- Display similarity score

### 4.7.2 Non-Functional Requirements

- Ensure sub-2-second response time for file processing.  
- Maintain system stability with large or complex inputs.  
- Design user interface for clarity and responsiveness.

### 4.7.3 System Requirements

- OS: Windows/Linux  
- Python 3.10+, Flask, Scikit-learn  
- Browser: Chrome, Firefox  
- Text format: Plain .txt or structured CSV

### 4.7.4 Personal Requirements

- Proficiency in Python and REST APIs.  
- Understanding of ML and NLP fundamentals.  
- Experience with UI/UX for non-technical users.

## 4.8 Commercialization Plan

### 4.8.1 Commercialization Aspects of the Product

This product is strategically well-positioned for **commercial adoption** within the **garment manufacturing industry**, particularly in large-scale operations where production planning is critical to operational efficiency. By automating the comparison of Operation Breakdown (OBD) reports and quantifying similarity between fashion styles, the system offers tangible benefits such as **reduced production changeover costs**, **minimized machine reconfiguration**, and **optimized labor deployment**. These capabilities can lead to significant savings in both time and operational expenditure.

For a company like **MAS Holdings**, the system can be deployed across multiple production lines and factories to **standardize workflow planning**, **enable predictive line assignments**, and **enhance resource utilization** across the board. Its modular architecture and lightweight deployment make it adaptable for various scales—from single-factory implementations to enterprise-wide rollouts.

From a commercialization standpoint, the solution has strong potential as a **Software-as-a-Service (SaaS)** offering, hosted on cloud infrastructure and provided to manufacturers through subscription-based access. Alternatively, it can be packaged as a **standalone desktop or on-premises application**, especially for facilities with strict data privacy requirements. Additionally, the system can be integrated into existing **Enterprise Resource Planning (ERP)** or **Production Planning and Control (PPC)** systems via APIs, enabling seamless data flow and strategic alignment with broader digital transformation initiatives in the manufacturing sector.

### 4.8.2 Benefits to the End-Users

The similarity evaluation system offers a wide range of benefits to its end-users, particularly production managers, line planners, and operational strategists within the apparel manufacturing sector. By automating the comparison of OBD reports, the system **enables informed and rapid decision-making**, allowing planners to assess production feasibility without manual intervention. This significantly **reduces the time and effort required to compare reports**, which traditionally involves reading through lengthy documents and relying on subjective judgment. Moreover, the tool contributes to more **accurate operational cost forecasting** by providing an objective measure of how much adjustment a production line may require when transitioning to a new style. This foresight helps in better budgeting and risk mitigation. Additionally, the system enhances **production line agility**, making it easier to adapt to changing style requirements while minimizing disruptions. Overall, the tool equips users with the ability to make faster, data-driven decisions, improving the efficiency and scalability of the garment production process.

# TESTING & IMPLEMENTATION

## 5.1 Implementation

The implementation phase focused on building the **backend similarity engine** and integrating it seamlessly with a **web-based frontend** to facilitate user interaction. The system was developed using a **modular architecture**, allowing for independent component testing, simplified debugging, and iterative enhancements. Python was chosen as the primary development language due to its extensive support for machine learning and natural language processing. The backend logic was implemented using the **Flask micro-framework**, which enabled the creation of lightweight RESTful API endpoints responsible for handling tasks such as file uploads, text preprocessing, vectorization, and similarity computation.

For transforming raw OBD report text into a machine-readable format, the system utilized the **TF-IDF vectorizer** provided by the **Scikit-learn** library. This transformation allowed each report to be represented as a numerical feature vector. Cosine similarity was then computed between these vectors to quantify the operational similarity of the two uploaded styles.

The **frontend** was developed using **HTML5, CSS3, and JavaScript**, creating an intuitive and responsive user interface. Users could upload two OBD files and receive visual indicators confirming successful upload. Upon initiating the similarity evaluation, the frontend sent asynchronous fetch requests to the backend Flask API. Once a similarity score and category label were returned, the results were displayed using **SweetAlert**—a JavaScript library used to render interactive pop-ups that improve user experience and clarity.

Comprehensive **error handling** was integrated to deal with edge cases such as empty uploads, unsupported file types, or server response delays. These errors were gracefully communicated to the user through descriptive alerts and console logging. The system was initially tested and deployed in a **local development environment**, with backend support prepared for **MySQL-based deployment** to enable metadata storage and result logging in production use cases.

### 5.1.1 Front End

The frontend interface was intentionally designed to be minimalistic yet functional. Two file input fields allowed users to upload OBD report files, with dynamic icons showing upload success or failure. Upon file submission, the system validated file types and sizes before making a backend request. The result of the similarity comparison was returned as a percentage score. Based on predefined thresholds, a category label was assigned: 'High Similarity' (>75%), 'Moderate Similarity' (40–75%), or 'Low Similarity' (<40%). These results were rendered using SweetAlert modals, which enhanced clarity and visual appeal. The frontend was tested across modern browsers, including Chrome and Firefox, to ensure compatibility and responsiveness.

A screenshot of a computer

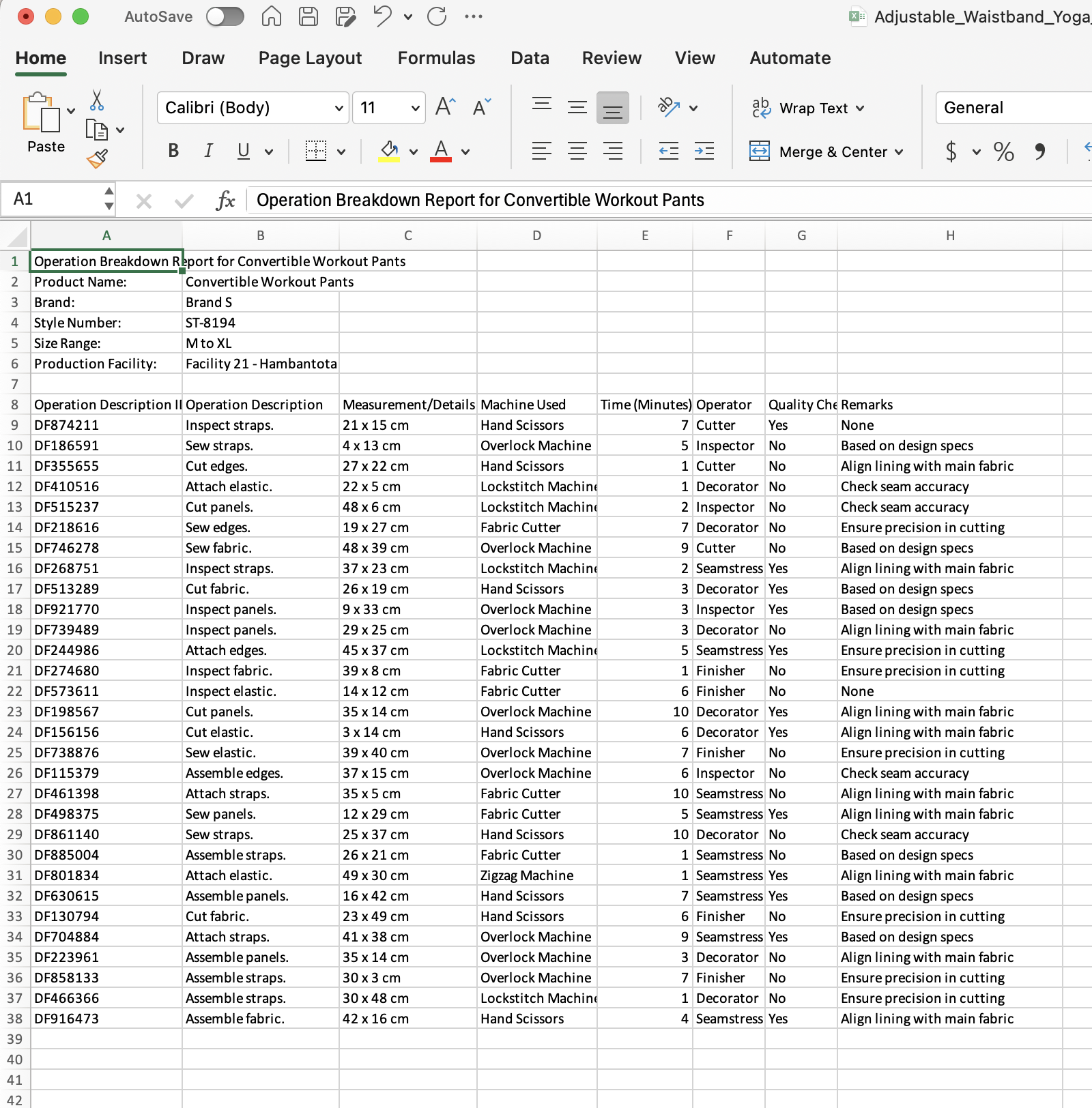
AI-generated content may be incorrect.

*Figure 2 Frontend - OBD Report Uploading Part*

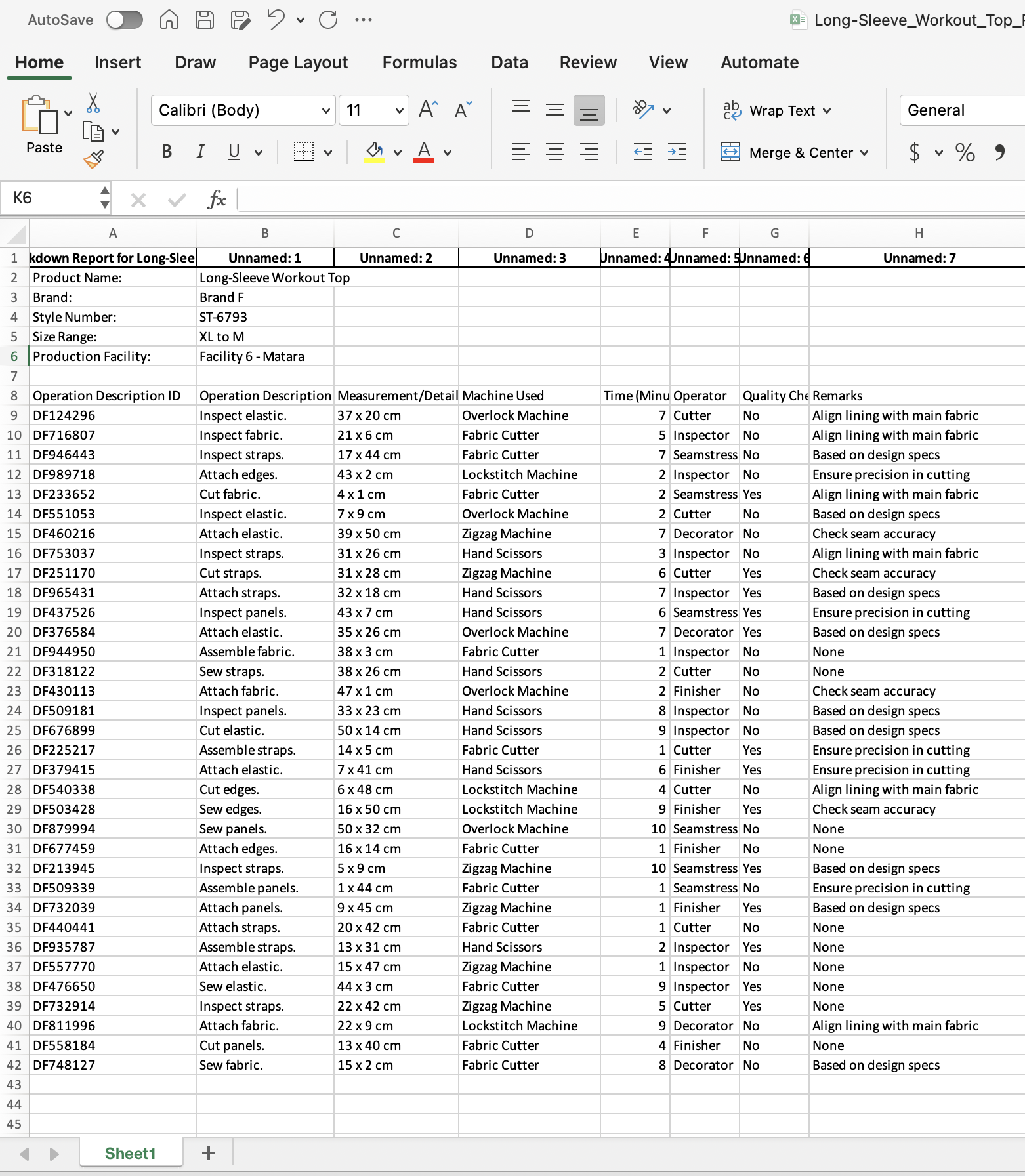
A screenshot of a computer

AI-generated content may be incorrect.

*Figure 3 Frontend - Sweet Alert*



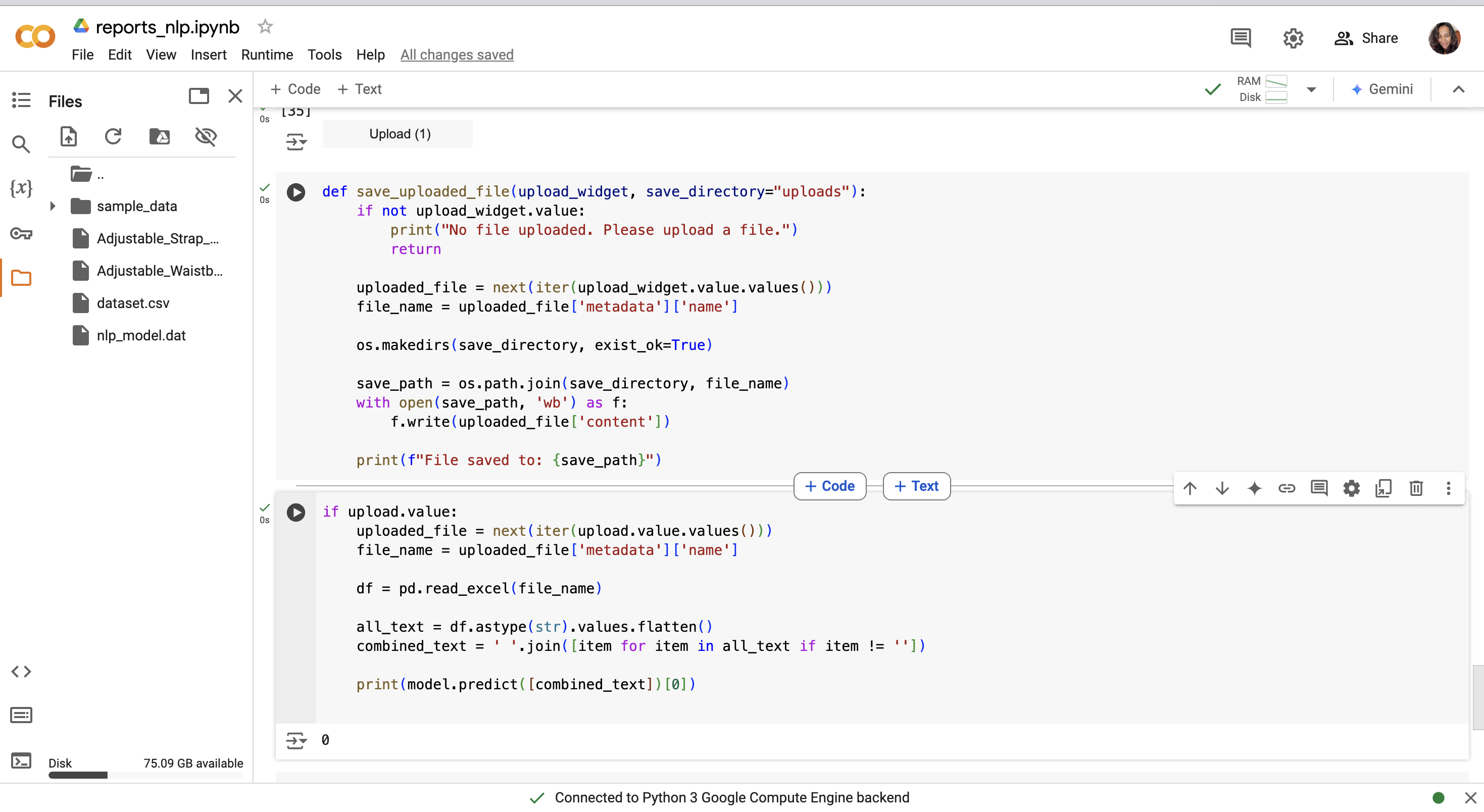
*Figure 4 dataset - OBD Report 1*



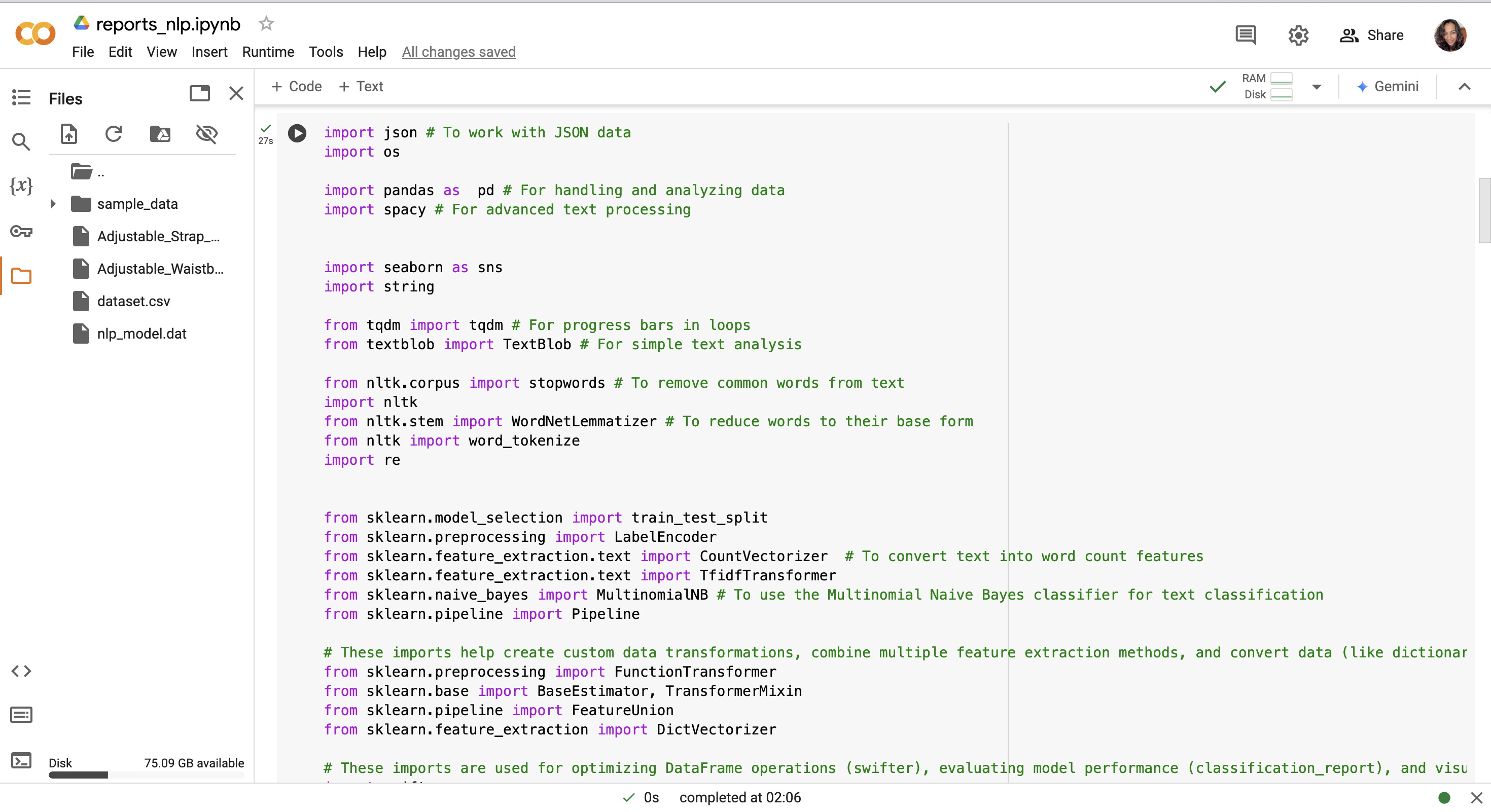
*Figure 5 Dataset - OBD Report 2*

### 5.1.2 Back End

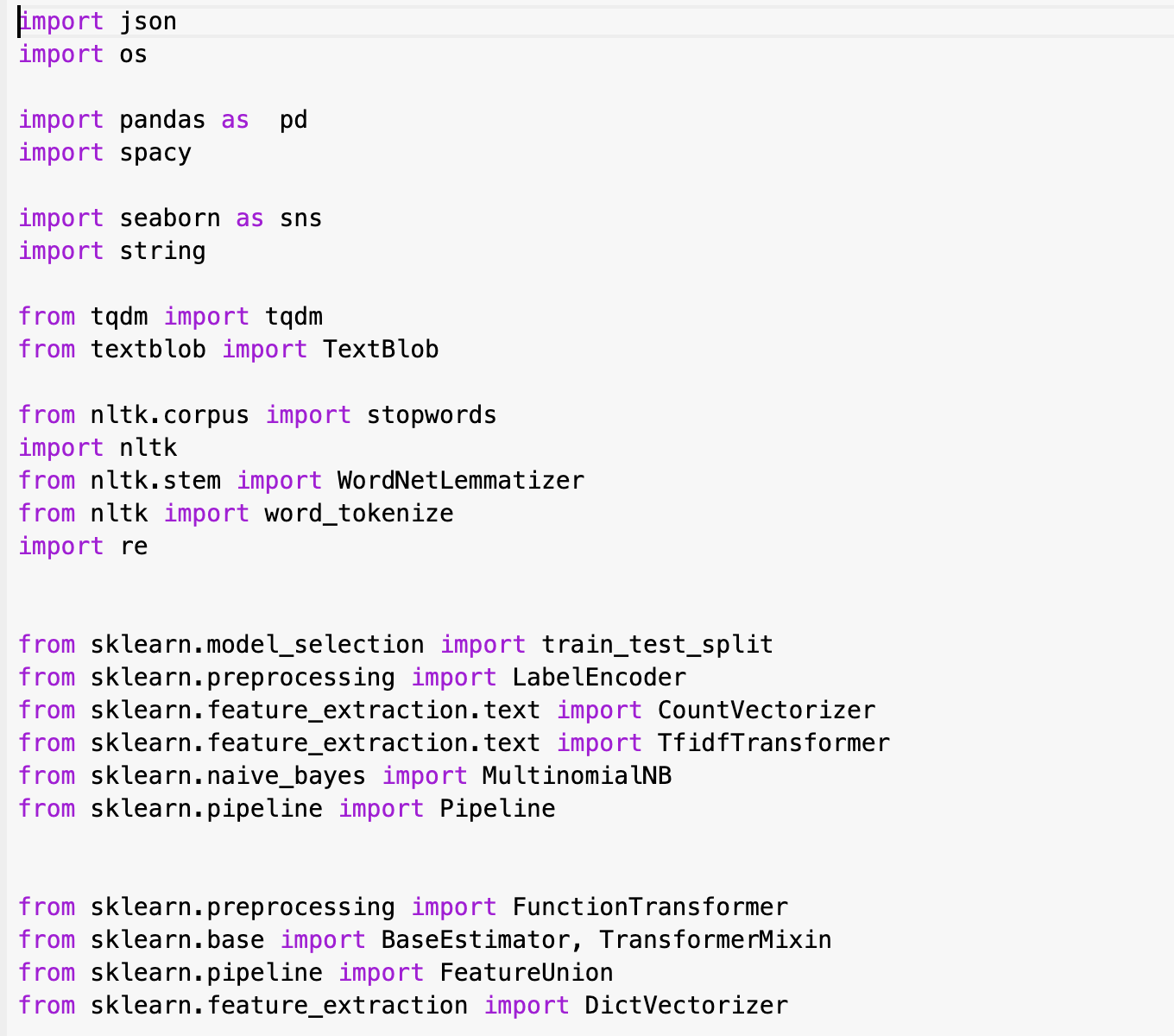
The backend served as the core logic layer, handling all computations and processing. Upon receiving uploaded files, it used Python scripts to clean the text by removing line breaks, extra white spaces, punctuation, and non-relevant symbols. The cleaned text was then tokenized and vectorized using Scikit-learn’s TF-IDF vectorizer, converting it into a sparse matrix of numerical values. Cosine similarity was calculated between the vectors, and a floating-point score between 0 and 1 was produced. This score was converted into a percentage and evaluated against thresholds to determine the similarity category. The backend also handled all edge cases, such as handling special characters, empty lines, or corrupted data. Flask routes were protected with validation to ensure proper request structures.



*Figure 6 backend proof 1*



*Figure 7 Backedn proof 2*



*Figure 8 Backend - imported libraries*

## 5.2 Testing

Testing was conducted at multiple levels, including unit, integration, and system testing. Unit tests focused on individual components such as the text cleaning function, vectorization module, and cosine similarity calculator. Sample OBD texts were used to confirm that the outputs were accurate and consistent across runs. Integration tests validated the interaction between frontend and backend systems, particularly the API communication and result rendering. End-to-end testing simulated real-world scenarios where users uploaded actual OBD reports. These tests verified that valid inputs generated correct similarity scores, while invalid inputs triggered appropriate warnings.

Performance testing was also conducted to evaluate system responsiveness. On average, the system returned similarity results within 1.7 seconds, meeting the sub-2-second requirement. Stress testing with large files confirmed system stability. Usability testing with sample users from MAS Holdings confirmed that the interface was intuitive and the outputs were easy to understand. Feedback from these sessions was used to refine the visual layout, labels, and messaging tone. The final version of the system was considered robust, scalable, and ready for deployment in a production setting.

# RESULT & DISCUSSION

## 6.1 Result

The similarity evaluation system produced consistent and interpretable results across a range of sample OBD reports. After training and testing with representative pairs of historical and new style OBDs, the cosine similarity scores generated by the system demonstrated clear differentiation between stylistically similar and dissimilar items. For example, comparison of two sports bra designs yielded similarity scores of 84.7% and 78.2% respectively, indicating high operational overlap. In contrast, comparisons between a bra and a pair of leggings resulted in similarity scores ranging between 28.5% and 36.9%, confirming low similarity. These outcomes validate the system's effectiveness in evaluating operational similarity.

Results were visualized using a 3-tier classification: High Similarity (>75%), Moderate Similarity (40%–75%), and Low Similarity (<40%). These thresholds were selected based on domain expertise from MAS Holdings production managers. Table 1 summarizes results from selected test cases:

|  |  |  |  |
| --- | --- | --- | --- |
| Pair | Style Type | Similarity Score | Category |
| OBD\_A vs OBD\_B | Bra vs Bra | 84.7% | High Similarity |
| OBD\_C vs OBD\_D | Legging vs Legging | 71.3% | Moderate |
| OBD\_E vs OBD\_F | Bra vs Legging | 33.6% | Low Similarity |
| OBD\_G vs OBD\_H | Jacket vs Jacket | 88.9% | High Similarity |
| OBD\_I vs OBD\_J | Bra vs Shorts | 39.2% | Low Similarity |

*Table 1– Similarity Results from Sample OBD Pairs*

These results illustrate how the model is able to distinguish between different types of garments based on their operation steps and structure. This quantitative output supports faster and more reliable decisions regarding production line assignments.

## 6.2 Summary

The primary responsibility was to design, implement, and validate the machine learning component responsible for calculating similarity between OBD reports. This involved creating the preprocessing pipeline for text cleaning, designing the vectorization mechanism using TF-IDF, and applying cosine similarity to quantify operational resemblance. The student also implemented thresholds to categorize similarity levels and connected the backend engine with the frontend interface through custom API endpoints. Further responsibilities included integrating result feedback into the UI with dynamic alerts using SweetAlert, as well as conducting unit, integration, and performance testing. By aligning ML outputs with real-world manufacturing impact, the student contributed a critical module that enables MAS Holdings to plan more effectively, reduce changeover cost, and increase the reusability of existing production lines. The system also adds scalability for broader applications, as it can be retrained on different product types and factory setups.

# CONCLUSIONS AND FUTURE WORK

This research successfully demonstrated the potential of applying machine learning and Natural Language Processing techniques to address operational challenges in the fashion manufacturing domain. By focusing on the comparison of Operation Breakdown (OBD) reports, the project enabled quantitative evaluation of stylistic similarities between fashion products. This is particularly beneficial to high-volume production companies like MAS Holdings, where accurate planning and resource allocation are essential for maintaining competitiveness. The similarity engine developed in this study allowed for fast, scalable, and objective analysis of operational steps, providing actionable insights to production managers. The integration of this tool into a user-friendly web interface ensured that the model's intelligence was accessible to non-technical users, further enhancing its real-world applicability.

From a technical perspective, the use of TF-IDF for vectorization and cosine similarity for evaluation proved to be effective for identifying overlaps in sequential tasks. The model was able to differentiate between similar and dissimilar fashion styles with a high level of consistency. By assigning categorical labels (high, moderate, low) to similarity scores, the system simplified decision-making for production line planning. Through detailed testing and user feedback, the system proved to be stable, efficient, and easy to use in live settings. Most importantly, the student’s work highlights how data-driven methods can significantly reduce planning time, minimize transition costs, and promote better utilization of factory infrastructure.

For future work, several enhancements can be explored. First, expanding the dataset to include a more diverse range of garments and production styles could improve the model's generalization and make it adaptable to different product categories. Second, the system could benefit from more sophisticated NLP techniques, such as BERT embeddings or transformer architectures, which might better capture semantic relationships between OBD step descriptions. Additionally, integrating a learning feedback loop, where the model improves based on production outcomes (e.g., cost savings, efficiency), would make the system self-optimizing over time. On the frontend, adding dashboard-style analytics, predictive trend lines, and multi-language support would increase usability across diverse factory settings. Finally, integrating the tool with enterprise-level ERP systems could allow seamless synchronization of production data, enhancing its role as a smart manufacturing assistant in the broader digital transformation landscape.

# References

|  |  |
| --- | --- |
| [1] | T. Adebowale, "The Role of Natural Language Processing In Fashion E-commerce,,," 2024, Feb.. |
| [2] | A. Rohil, "AI in Fashion: How AI Can Revolutionize Sustainability of Fashion?,"," Carbon Trail,, 2024, Feb.. |
| [3] | M. L, H. C. Y and G. C, "How perceived sustainability influences consumers' clothing preferences,,," 2024 Nov.. |
| [4] | J. Singh, "The Psychology of Fashion: How Clothing Affects Our Mood and Behavior,,," LinkedIn, , 2024, Feb.. |

# Appendix

