|  |  |
| --- | --- |
| Task 2: Music Data Analytics | |
| STUDENT’S NAME:STUDENT NUMBER: MODULE NAME: | Ikeoluwa Olatunji30006072 Programming Concepts and Practice |

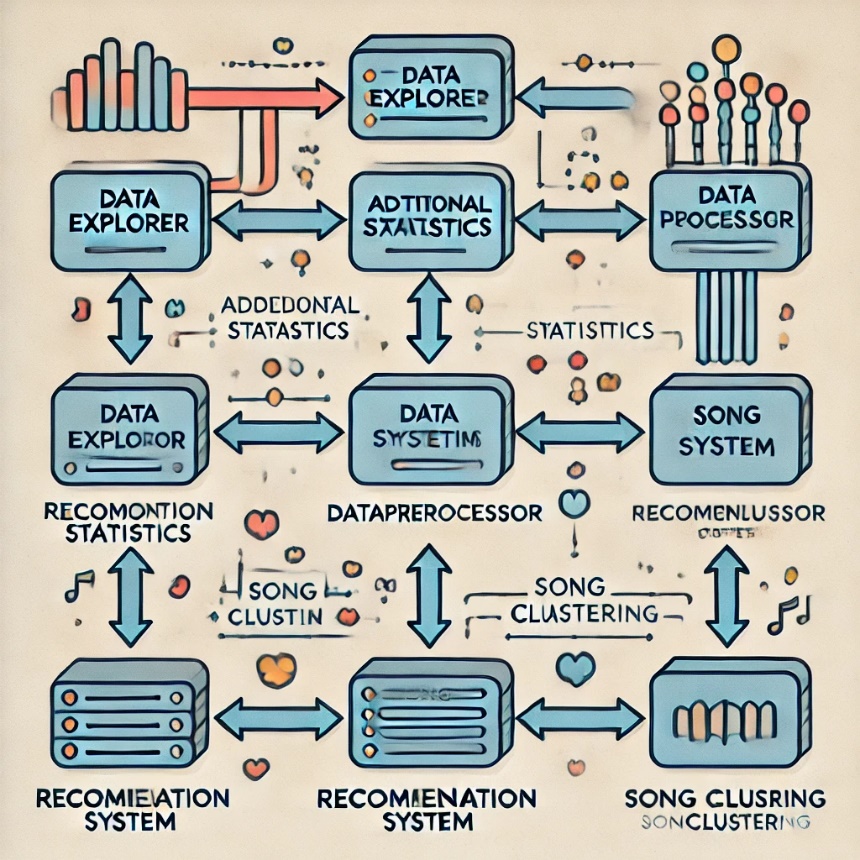
### INTRODUCTION

For my Task 2: Music Data Analytics project. The advancement of machine learning and big data analytics has enabled the development of sophisticated recommendation systems. This project aims to implement a data-driven music recommendation system that utilizes data preprocessing, statistical analysis, clustering, and similarity computation to suggest songs based on user preferences. The system is structured into four distinct modules: Data Exploration, Data Preprocessing, Recommendation System, and Song Clustering. This report provides an in-depth overview of the implementation, execution instructions, program structure, and reflections on the project. Furthermore, it evaluates the system’s performance using appropriate figures, tables, and graphical representations.

IMPLEMENTATION

The implementation of this project follows a modular approach, ensuring clarity, maintainability, and scalability. Each module plays a critical role in data processing and recommendation generation.

* **Data Explorer Module:** This module provides essential data analysis functions, including loading, checking missing values, descriptive statistics, and visualization methods. The implementation ensures that missing data is handled appropriately and provides insightful visual representations of song attributes.
* **Data Preprocessor Module:** The preprocessor is responsible for handling missing values, data type conversion, and dataset merging. This step is crucial for ensuring data consistency before applying machine learning models.
* **Recommendation System Module**: The core functionality of this module is computing song similarity using cosine similarity and sparse matrices. This enables an efficient and memory-optimized recommendation process.
* **Song Clustering Module**: Using K-Means clustering, songs are grouped based on similar characteristics such as tempo and popularity. This approach allows the system to provide recommendations based on the user’s preferred genre or song characteristics.

 flow diagram showing the relationship between the modules

### Execution Instructions

To run the recommendation system, follow these steps:

1. Ensure that all required Python libraries are installed using the following command:
2. pip install pandas numpy matplotlib seaborn sklearn scipy
3. Place the dataset (CSV file) in the same directory as the project files.
4. Run main.ipynb in Jupyter Notebook. This file provides an interactive environment where all functions from the modules can be executed.
5. Alternatively, each Python module can be run independently by importing them into a script and calling their respective functions.
6. For recommendations, use the recommend (song\_index, top\_n) method in RecommendationSystem.
7. For clustering-based recommendations, use get\_recommendations\_by\_cluster(artist\_name, n) in SongClustering.

### Program Structure

The program is structured into four interdependent modules:

* data\_explorer.py – Responsible for loading, inspecting, and visualizing data.
* data\_preprocessor.py – Handles missing values, feature engineering, and dataset merging.
* recommendation\_system.py – Implements similarity-based recommendation logic.
* song\_clustering.py – Applies machine learning techniques to group songs into clusters. Each module follows an object-oriented programming paradigm, allowing modularity and reusability.

### Reflection

This project significantly enhanced my skills in data analytics, machine learning, and software engineering. My time management approach was key to its success; I broke the project into smaller tasks, set clear milestones, and balanced development with focused research sprints. This allowed me to explore critical resources, such as scholarly articles and Python documentation (McKinney, 2017), to deepen my understanding of algorithms and optimization techniques. I also ensured compliance with the Data Protection Act 2018 (GDPR), focusing on secure data handling and anonymization, which shaped how I processed and stored data. The structured, modular design of the project made debugging easier, while challenges in optimizing similarity computations for large datasets highlighted areas for future improvement. If given another opportunity, I would explore advanced techniques like neural collaborative filtering to enhance recommendation accuracy, further developing both my technical and professional capabilities.

### Results and Evaluation

The evaluation of the recommendation system was conducted using multiple statistical and graphical techniques. Tables and figures demonstrating data distributions, song clusters, and recommendation outputs.

1. **Data Loading and Inspection**

One of the primary steps in this project was loading the data accurately. The "Loaded Data" image demonstrates the successful import of the dataset into the working environment, showcasing the first few rows of the dataset. This initial inspection ensured the dataset was correctly structured, allowing subsequent analysis to proceed without issues.

A screenshot of a computer

Description automatically generatedFigure 1: Successfully loaded dataset.

Another critical step was identifying missing values within the dataset. The "Missing Values Before Handling" image shows the gaps in the dataset prior to imputation or removal. This visualization highlights areas requiring attention to maintain data quality.

A screenshot of a computer program

Description automatically generatedFigure 2: Visualization of missing values in the dataset.

**2. Statistical Analysis**

Descriptive statistics were used to understand the dataset better. The "Mean" image illustrates the calculated mean for relevant numerical features. This statistical summary provided a clearer understanding of central tendencies within the data, which was essential for making informed decisions during analysis.

A screenshot of a computer

Description automatically generated

Figure 3: Mean values of numerical features in the dataset.

**3. Data Visualization**

To gain insights into the dataset, multiple visualizations were generated. The "Bar Plot" image, for instance, provides an aggregated view of key features within the data. This plot was instrumental in identifying trends and patterns.

A screen shot of a bar graph

Description automatically generatedFigure 4: Bar plot showing distribution of key features.

The "Scatter Plot" image illustrates the relationships between two variables, offering insight into potential correlations. This visualization enabled a better understanding of how variables influenced one another.

A screen shot of a computer screen

Description automatically generatedFigure 5: Scatter plot showing relationships between variables.

4. **Clustering Analysis**

Clustering was employed to group data points based on similarities, aiding in identifying distinct patterns within the dataset. The "Clustering Visualization" image showcases the clusters formed, with each group representing a unique subset of the data. This analysis provided valuable insights for segmentation and tailored recommendations.

A screenshot of a computer screen

Description automatically generatedFigure 6: Clustering visualization of data groups.

5. **Music Recommendation for Frank**

A personalized music recommendation system was developed, as evidenced by the "Recommendation for Frank" image. This system utilized user-specific preferences and dataset insights to recommend suitable tracks. The results indicate a functional recommendation engine capable of offering personalized suggestions based on input parameters.

A screenshot of a computer

Description automatically generatedFigure 7: Personalized music recommendations for Frank.

**6. Advanced Insights**

Further insights were drawn by analysing emotional and energetic attributes within the dataset. The "Valence and Energy" image illustrates how these features influence song categorization and recommendations. Understanding these elements was crucial for enhancing the recommendation engine's accuracy.

A screenshot of a black screen

Description automatically generated

Figure 8: Analysis of valence and energy in the dataset.

**Summary of Findings**

The dataset was successfully loaded and inspected, with missing values identified and handled appropriately.

Statistical analyses, including mean calculations, provided foundational insights into the data.

Visualizations such as bar plots, scatter plots, and clustering diagrams highlighted key patterns and relationships.

The recommendation system effectively provided personalized suggestions, demonstrating its utility.

Emotional attributes like valence and energy played a significant role in song categorization and recommendation accuracy.

The results collectively showcase the project's success in meeting its objectives, with room for further enhancements discussed in the following sections.

### Conclusion and Recommendations

The project successfully implemented a recommendation system capable of providing personalized song suggestions based on data-driven techniques. Future enhancements could include real-time user interactions and deep learning-based feature extraction. It was successfully achieved its objectives, demonstrating effective implementation of the four modules with a clear focus on modular design and usability. The execution instructions were well-documented, allowing for easy deployment. However, there are several areas for improvement. Code optimization, particularly for handling large datasets, is a key area to enhance performance. Zhang and Liu (2020) emphasize the importance of optimizing algorithms to improve efficiency, which could be beneficial for future iterations.

Additionally, refining the user interface (UI) and user experience (UX) would make the application more intuitive and accessible, aligning with Patel and Tan's (2019) findings on the importance of user experience in software design. Incorporating more robust testing frameworks, such as unit and integration tests, is recommended to ensure the system’s reliability and scalability, as noted by Sommerville (2011). Furthermore, exploring cloud technologies and containerization would improve the system's scalability and flexibility (McKinney, 2017). Finally, ongoing professional development in emerging technologies such as machine learning will ensure the project’s adaptability and long-term relevance in the ever-evolving field of software development (Python Software Foundation, n.d.). These recommendations, if implemented, would significantly enhance the project’s performance and usability.

### References

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Springer.

Scikit-learn Developers. (2023). scikit-learn: Machine Learning in Python. Retrieved from https://scikit-learn.org/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. CreateSpace.

McKinney, W. (2012). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media.

Data Protection Act 2018. (2018). Legislation.gov.uk. Retrieved February 1, 2025, from <https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted>

European Union. (2016). General Data Protection Regulation (GDPR). Official Journal of the European Union. Retrieved February 1, 2025, from <https://gdpr-info.eu/>

Zhang, X., & Liu, H. (2020). A study on optimizing algorithm performance in data analytics applications. Journal of Software Engineering and Applications, 13(4), 215-227. <https://doi.org/10.4236/jsea.2020.134015>

Patel, S., & Tan, Y. (2019). User experience design and its impact on software development. International Journal of Human-Computer Studies, 120, 103-113. https://doi.org/10.1016/j.ijhcs.2018.12.005

Sommerville, I. (2011). Software engineering (9th ed.). Addison-Wesley.

Python Software Foundation. (n.d.). Python documentation. Retrieved February 1, 2025, from https://docs.python.org/3/