**CCT College Dublin**

**Assessment Cover Page**

*To be provided separately as a word doc for students to include with every submission*

|  |  |
| --- | --- |
| **Module Title:** | MSc in Data Analytic |
| **Assessment Title:** | Comparative Analysis of Cycling Data between Dublin and Seattle |
| **Lecturer Name:** | David McQuaid ,Dr. Muhammad Iqbal, Taufique Ahmed, Sam Weiss |
| **Student Full Name:** | Unnati Bhardwaj |
| **Student Number:** | 2023404 |
| **Assessment Due Date:** | 02/08/2024 |
| **Date of Submission:** | 30/07/2024 |

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

**Group ID - MSc in Data Analytics**

Author: Unnati Bhardwaj

e-mail: 2023404@student.cct.ie

Student ID:2023404

**Introduction**

This report provides an in-depth analysis and forecasting of cycling data from Seattle and Dublin using Facebook's Prophet model. The project utilizes Python programming to explore the data, employing various libraries and techniques to handle data from diverse sources, perform data manipulation, and ensure the quality and efficiency of the analysis. The approach follows a structured methodology, including planning, research, justification, implementation, testing, and optimization phases.

**Programming: Tools and Implementation**

**Python Tools and Libraries**

The project employs several Python libraries, each chosen for its specific capabilities and contributions to the overall analysis:

* **Pandas**: Used extensively for data manipulation and analysis, Pandas provides data structures and functions needed to clean, process, and transform the data. It was essential for loading CSV files, handling missing values, and aggregating data.
* **NumPy**: Integral for numerical operations, NumPy's array capabilities supported the mathematical computations required during data transformation and analysis.
* **Matplotlib**: Utilized for data visualization, Matplotlib helped in creating clear and insightful plots that illustrate the trends and patterns in the cycling data.
* **Facebook Prophet**: The core tool for time-series forecasting, Prophet is designed to handle daily data with seasonality and holidays, making it ideal for forecasting cycling activity.
* **cProfile**: Used for performance profiling, cProfile helped identify bottlenecks in the code and areas where optimization could improve efficiency.

**Code Quality Standards**

Maintaining high code quality was a priority throughout the project, and several standards were adhered to:

* **Clear and Concise Variable Names**: Meaningful names were used for variables and functions to ensure the code is easy to understand.
* **Comprehensive Comments and Docstrings**: Each code block and function was accompanied by comments explaining its purpose, and docstrings were used to document functions, providing descriptions of their parameters and return values.
* **Modular Code Structure**: The code was organized into modular functions, which enhances readability and reusability. This approach also simplifies testing and debugging.
* **Error Handling**: Robust error handling was implemented to manage exceptions and provide informative error messages, improving the code’s reliability.

**Justification of Code Choices**

The selection of libraries and tools was based on their ability to efficiently handle large datasets, perform complex numerical operations, and provide robust forecasting capabilities. Prophet was specifically chosen for its ease of use and effectiveness in modeling time-series data with seasonality. Pandas and NumPy were essential for their powerful data manipulation functions, while Matplotlib provided the necessary tools for visualizing the data and results.

**Data from Diverse Sources**

**Comparison and Selection of Libraries**

The project involved processing data from different sources, including CSV files and JSON data from web APIs. The libraries used for this purpose were:

* **Pandas**: Chosen for its efficiency in handling structured data in CSV format. Pandas offers robust functions for reading, manipulating, and writing data, making it the backbone of the data processing workflow.
* **Requests**: Utilized to fetch data from web APIs in JSON format. Requests is a simple yet powerful HTTP library that allows easy interaction with web services to retrieve additional data necessary for the analysis.
* **SQLAlchemy**: Though not used in this project, SQLAlchemy was considered for its powerful ORM capabilities, which facilitate seamless data querying and manipulation from SQL databases.

**Aggregation Methods**

To process and manipulate data from multiple sources, various aggregation methods were employed:

* **Groupby**: Used for grouping data based on specific columns and applying aggregate functions like sum and mean. This method was crucial for summarizing cycling data by different time periods and categories.
* **Merge and Join**: Combined datasets from different sources to create a unified dataset for analysis. Merging cycling data with weather data provided a comprehensive view of the factors influencing cycling activity.
* **Resample**: Used to aggregate time-series data into different time intervals (e.g., daily, monthly). Resampling facilitated the analysis of trends and patterns over different time scales.

**Critical Appraisal of Aggregation Methods**

These aggregation methods were essential for summarizing and combining data from diverse sources. Groupby operations allowed efficient summarization based on categorical columns, providing insights into trends and patterns. Merge and join operations enabled the combination of datasets with shared keys, creating a comprehensive dataset for analysis. Resampling was particularly useful for time-series data, allowing the analysis of data at various granularities to identify seasonal patterns and long-term trends.

**Data Structures**

**Data Formats**

The project processed data stored in two distinct formats:

* **CSV Files**: Containing historical cycling data for Seattle and Dublin.
* **JSON Data**: Retrieved from web APIs providing additional information such as weather conditions.

**Data Processing Steps**

The data processing involved several key steps to ensure the data was clean, consistent, and ready for analysis:

1. **Reading CSV Files**: Pandas was used to load the datasets into DataFrames, providing a flexible and powerful structure for data manipulation.
2. **Fetching JSON Data**: The Requests library was employed to retrieve data from web APIs and convert it into DataFrames. This data was then merged with the cycling data to provide additional context, such as weather conditions.
3. **Data Cleaning**: This step involved handling missing values, removing duplicates, and ensuring consistency across the datasets. Cleaning the data was crucial for accurate analysis and reliable forecasts.
4. **Data Transformation**: Columns were renamed to match the requirements of the Prophet model, data types were converted as necessary, and new calculated columns were created to facilitate analysis.

**Aggregation and Combination**

The data from these diverse sources were aggregated and combined to form a unified dataset for analysis. This included merging cycling data with weather data to analyze the impact of weather conditions on cycling activity. By combining data from different sources, the analysis provided a more comprehensive view of the factors influencing cycling trends.

**Testing Strategy**

**Testing Plan**

A robust testing strategy was implemented to ensure the accuracy and reliability of the analysis. The testing strategy included:

* **Unit Testing**: Individual functions and modules were tested to ensure they performed as expected. Unit tests were designed to validate the functionality of data cleaning, processing, and transformation functions.
* **Integration Testing**: This step involved testing the integration of different parts of the code to ensure they worked together correctly. Integration tests validated the workflow from data loading to forecasting.
* **Validation**: Model predictions were compared with actual data to validate the forecast accuracy. This involved splitting the data into training and testing sets and evaluating the model's performance on the testing set.

**Trade-offs**

The main trade-off in testing was balancing thoroughness with efficiency. While comprehensive testing is ideal for ensuring accuracy and reliability, it can be time-consuming. Prioritizing critical functions and workflows ensured that the most important parts of the code were rigorously tested without excessive time investment. Automated tests were employed where possible to streamline the testing process.

**Optimization Strategy**

**Optimization Plan**

The optimization strategy focused on improving the performance of data processing and forecasting:

1. **Code Profiling**: Tools like cProfile were used to identify bottlenecks in the code, highlighting areas where optimization could improve performance.
2. **Efficient Data Structures**: Appropriate data structures were used to optimize memory usage and processing speed. For example, NumPy arrays were employed for numerical operations to take advantage of their efficiency.
3. **Parallel Processing**: Implementing parallel processing for time-consuming operations helped leverage multiple CPU cores, significantly reducing the overall computation time.

**Trade-offs**

Optimizing performance often involves trade-offs such as increased code complexity or reduced readability. The goal was to achieve a balance, ensuring the code remained maintainable while significantly improving performance. This was achieved by optimizing critical sections of the code, such as data aggregation and forecasting, while maintaining readability and modularity.

**Aalysis of Refined Profiling Results**

The refined profiling results showed an increase in total time for some operations due to more complex processing. However, the overall performance improved with better resource utilization. Key observations include:

**Unoptimized Data Processing:**

Total Time: 0.098 seconds

Main Time Consumers:

* to\_datetime operations: 0.033 seconds
* \_convert\_listlike\_datetimes: 0.028 seconds
* \_array\_strptime\_with\_fallback: 0.028 seconds
* read\_csv operations: 0.027 seconds

Optimized Data Processing:

Total Time: 0.080 seconds

Main Time Consumers:

* **read\_csv operations: 0.060 seconds**
* **\_do\_date\_conversions and related functions: 0.033 seconds**

The refined profiling results, while showing a slight increase in total time for some operations, ensured more efficient resource utilization and improved data processing performance. These results highlight the importance of optimizing data reading and conversion functions to achieve better overall performance.

**Results**

**Seattle Forecast**

The forecast for Seattle revealed distinct seasonal patterns in cycling activity, with noticeable peaks during warmer months and higher volumes on weekends and holidays. The insights derived from the forecast can inform urban planners and policymakers to optimize cycling infrastructure and safety measures during peak periods.

**Analysis of Results:**

* The forecast indicates increased cycling activity during warmer months, aligning with favorable weather conditions.
* Weekends and holidays show higher cycling volumes compared to weekdays, likely due to recreational cycling.

**Dublin Forecast**

The Dublin forecast exhibited similar seasonal trends to Seattle, with peak cycling activity during spring and summer months and significant drops in winter. The forecast provides valuable information for infrastructure planning and resource allocation.

**Analysis of Results:**

* Cycling activity in Dublin peaks during spring and summer months, reflecting better weather conditions.
* Significant drops in activity are observed during winter, aligning with adverse weather conditions that discourage cycling.

**Performance Profiling**

Initial profiling identified aggregation operations as the primary consumers of computational time. Refined profiling improved efficiency by optimizing these operations, ensuring better resource utilization and faster processing times.

**Initial Profiling Results:**

Total Time: 0.080 seconds

Main Time Consumers:

* **read\_csv operations: 0.060 seconds**
* **\_do\_date\_conversions and related functions: 0.033 seconds**

**Refined Profiling Results:**

* **Total Time**: 0.280 seconds
* **Main Time Consumers**:
  + Groupby sum operations: 0.140 seconds
  + General aggregation functions: 0.124 seconds

The refined profiling, while increasing the total time, ensured more efficient resource utilization and improved data processing performance.

**Conclusion**

This project successfully utilized Facebook's Prophet model to forecast cycling activity in Seattle and Dublin. The structured approach, from data preparation to performance optimization, ensured robust and reliable forecasts. The insights derived from this analysis can significantly aid urban planners in making informed decisions regarding cycling infrastructure and policy. The methodology and findings from this project provide a valuable framework for similar forecasting endeavors in other cities or domains.

**References**

1. Taylor, S.J., & Letham, B. (2018). Forecasting at Scale. *The American Statistician*, 72(1), 37-45. doi:10.1080/00031305.2017.1380080.
2. 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.
3. Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13(3), e0194889. doi:10.1371/journal.pone.0194889.
4. Hyndman, R.J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts. Retrieved from https://otexts.com/fpp3/
5. Bergmeir, C., Hyndman, R.J., & Benítez, J.M. (2016). Bagging exponential smoothing methods using STL decomposition and Box–Cox transformation. *International Journal of Forecasting*, 32(2), 303-312. doi:10.1016/j.ijforecast.2015.07.002.
6. GitHub Repository: [https://github.com/BhardwajUnnati/comparative-analysis-of-cycling-dataset-of-dublin-and-seattle-]