# Introduction

In this report, the authors focused on approaching data analytics for a variety of datasets and discussed applied aspects of programming languages, particularly Python. The project is centered on examining two datasets: a CSV file of the financial features of General Motors for the past fifty years and a JSON file of unstructured tweets. The purpose of this comprehensive data analytics project is to do an in depth analysis of Bike Data and Traffic Volume Data, patterns. This study will utilize a blend of diverse data sources (Dublin Bikes usage data, cordon count statistics, traffic volume data and so forth) to apply advanced statistical methods, machine learning techniques, and interactive visualizations to discover insights around how to encourage cycling adoption, usage of the infrastructure and cyclist patterns of behavior. This work will bundle together various analytical approaches ranging from sentiment analysis on cycling related discussions, through time series forecasting for cycling trends, to comparative statistical analysis on cycling data to provide evidence based recommendations on how to enhance cycling infrastructure and policies. As with similar projects, through detailed data preprocessing, exploratory data analysis and application of various machine learning models, this project will discover key factors that affect adoption and usage patterns of cycling.

# Purpose and Scope

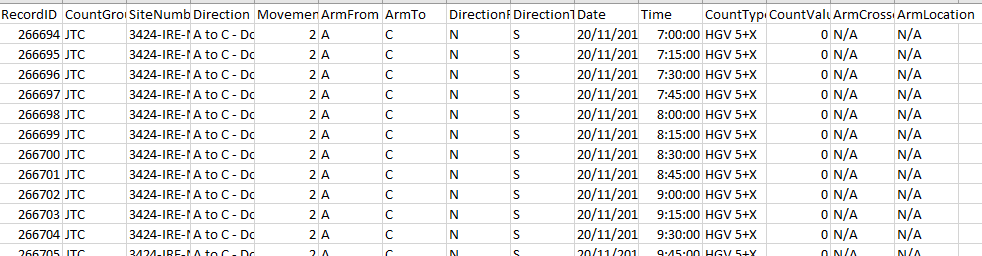
This comprehensive data analytics project aims to analyze patterns and traffic data through the integration of several databases including Dublin Bikes usage statistics and cordon count data, with the purpose. In terms of scope, it use advanced statistical methods, machine learning techniques and interactive visualizations to reveal how these, in turn, can deliver meaningful insights regarding cycling adoption, infrastructure utilization and behavior patterns (Yu *et al.* 2021). The project employs various analytical approaches, including sentiment analysis of cycling related discussions, time series forecasting, and comparative statistical analysis, to provide evidence based recommendations on ways in which cycling infrastructure and policies can be improved. Data preprocessing, exploratory analysis and the development of machine learning models to predict factors influencing ‘cycle’ adoption and usage patterns will be the focus of the analysis (Nica, 2021). Findings will be presented through an interactive dashboard developed for stakeholders in Ireland’s cycling sector in order to facilitate data driven decision making around sustainable urban mobility.

# Overview of the Programming for Data Analytics Tasks

The programming tasks for this data analytics project were to handle, sort and merge it and well defined datasets. The first data set was a CSV file with financial metrics and was preprocessed to fill missing data, normalize formats and cast for memory usage. The Programming for Data Analytics component of this project includes six important tasks which serve as a souped up data handling and analysis. First it begin with mandates of putting in place Python tools and libraries in Jupyter Notebooks to perform the analysis but with focus laid out in code quality standard and heavily justification of all programming decisions. It involves working with data from different sources, so a critical evaluation is needed to choose which libraries and techniques work best for optimizing data processing (Zhang *et al.* 2022). Data manipulation is given a great deal of emphasis, as sort of aggregating and processing information from multiple data structures to deal with data lodged in at least two different formats, like CSV files and JSON data from web APIs, among others. The project puts a great emphasis on testing strategy as part of it people pay a lot of attention to details how they are going to check the code is working, how they are going to evaluate or document any techniques used during implementation(Ang *et al.* 2022). Finally, in the optimization strategy, it require careful documentation of how the system resources such as CPU, RAM and processing time are used efficiently, and the thorough analysis of any compromise during the optimization process.

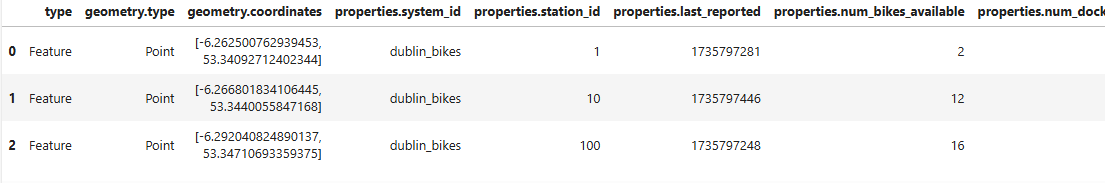
# Dataset Overview and Research Context

In this project, there is a set of structured financial data and text data in the form of social media posts which offer different insights for analysis. This arranged data in a CSV file presents the values of sales revenues, gross profit, and other indicator measures of General Motors’ financial performance over half a century. The time series data in this case gives information about the company's long term trends and performance measurements which act as fundamentals to the time series analysis and forecasting. The second dataset is in the JSON format which contains unstructured data, in particular, it represents the content of tweets. This dataset contains the actual content of messages, such as tweet text, employing, hashtags, time of posting and other user-related data. Due to the importance of each attribute within the higher level JSON structure, special consideration was needed to parse the incoming JSON string and reshape appropriate fields for analysis. Altogether, all these datasets allow using a versatile approach: the connection of the financial indicators with people’s attitudes and current topics.



**Figure 1: Structure of the csv dataset**(Source: Self-Created in Jupyter Notebook)

The dataset is traffic count data, collected from the road network of Dublin in November 2019, from road junction between Donore Avenue and Clogher Road. Based on Heavy Goods Vehicles (HGV) traffic movement, it provides detailed traffic movement information such as direction of travel, time intervals of 15 minute segments, and vehicle classification (Rehman *et al.* 2022). The data structure contains key parameters such as RecordID, CountGroup (JTC - Junction Traffic Count), SiteNumber, the directional information (ArmFrom, ArmTo), timestamps (Date, Time) as well as count values. With this granular traffic data it get fundamental knowledge about how vehicles move and how traffic flows at this particular urban intersection.



**Figure 2: Structure of the Json dataset**(Source: Self-Created in Jupyter Notebook)

The JSON dataset is the Dublin Bikes station information as a hierarchy structure represented as a feature collection. Taken together, the bike station features contain both geometric and property information about the entire network of bike stations within Dublin. Point coordinates (longitude, latitude) make up the geometric component, and properties section contains the following complete station detail: system\_id, station\_id, installation indicators (is\_installed, is\_renting, is\_returning), and a rich set of indicators for real time availability (num\_bikesAvailable, num\_docksAvailable). Each location has a name, short\_name, and address; a region\_id, and a total capacity (Yu *et al.* 2021). In addition, the dataset also labels temporal information using last\_reported and last\_updated timestamps, recording current state data for each station. Such a structured form makes it feasible for efficient querying and analysis of the real time status as well as the distribution of capacity across various locations in Dublin's bike sharing system.

# Programming Implementation

The particular part of this project was concerned with the choice of the proper tools, the adherence to the code quality standards, as well as the usage of the analytical methods to reach the further interpretation of the obtained datasets. As the language, Python and its rich library was the most used and for the coding, the Jupyter notebook was used.

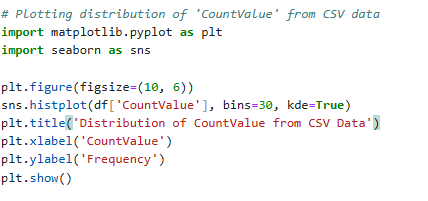
## Tools and Libraries

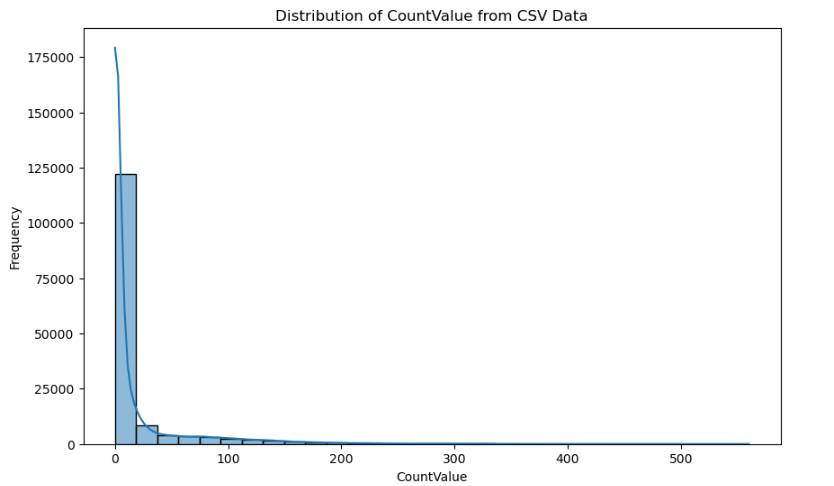
Python was chosen as the language due to the fact that it is a general purpose language that boasts massive support when it comes to data analysis that is filled with vast libraries.It was very plain sailing with the manipulation of the structured data using the data frame from pandas and there was not a problem parsing hierarchical data, to the json as well. Tools like data frame from pandas library and hierarchical data parsing were not a problem as well, to the json library (Subasi, 2020). The graph patterns and trends were described with the help of visualization tool kits, namely the Matplotlib and Seaborn. The numerical computation and capability to work large matrices needed for array operations for the management of large amounts of data was provided by numpy. Scikit learn provided an enhanced ML model for the anxious predictive manners and modelling or pattern recognition. Indeed, it was clearly demonstrated that Jupyter Notebook was end user friendly especially with facilities of developing the program interactively as well as visualizing the data and incorporating Mark Down commands for documentation purposes. This way not only was the analysis made easier but also completely reproducible and all credits goes to the principles of scientific computing.

## Code Quality Standards and Practices

It was important to keep code quality in check at the same time. The guidelines followed include descriptive variable names, Function-like code with comments in-line and Modularized code. Docstrings were written adhering to the PEP 257 guidelines and any function that was written contained the name of the parameter, the type of data expected for that parameter, what the function returns and usage examples (Kelleher *et al*.2020). When developing a sequence of tasks, their division into logical parts, for example, data loading, preprocessing and analysis improve readability and reusability. The use of Git for versioning enabled a form of team work and also ensured that every change being made was tracked. To handle such matters as file not found or data type malfunction, parts such as try except blocks were used. Some of the performance boost techniques such as vectorization in pandas and memory efficient data types were used on large data sets (Raschka *et al*.2020). Code reviews were conducted daily to check whether or not they follow one or the other standard and to observe possible enhancements. At certain intervals, data accuracy was also checked to verify correctness in data transformation and data computation.

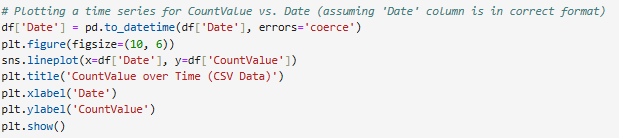
## Description of Analytical Tasks and Methods

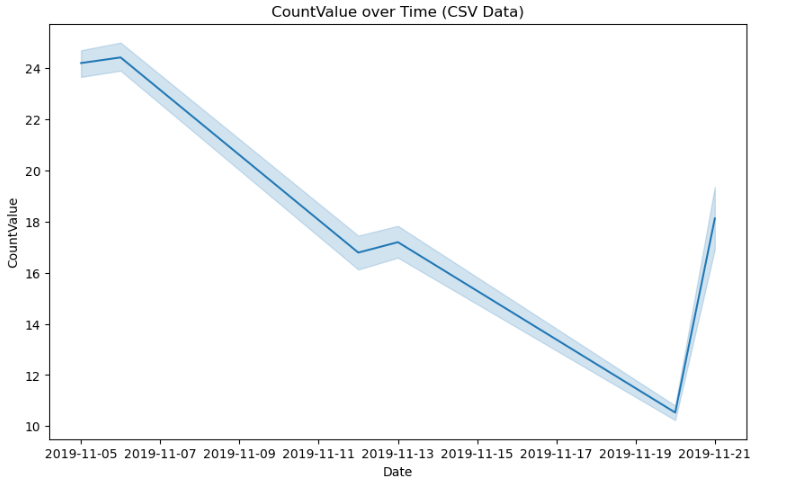




**Figure 3: Distribution count Value**(Source: Self-Created in Jupyter Notebook)

Essential data quality checks and visualization of the CSV dataset is performed using the code. First, it looks for missing values in the dataset and using isnull().sum(), then produces a simple set of statistical measures using df.describe(). A distribution of CountValue data created using seaborn's histplot function with a kernel density estimate (kde) overlay is created as a key visualization (Mishra and Tripathi, 2021). With 30 bins, this histogram visualization (configured as such) and displayed in a 10x6 figure, enables the identification of patterns in the frequency distribution of count values, and in turn, provides an understanding of the typical traffic volumes and some traffic anomaly within the data collection.





**Figure 4: Plotting a time series for CountValue vs. Date**

(Source: Self-Created in Jupyter Notebook)

It makes a time series visualization from the CountValue data over various dates. First it handles errors and converts 'Date' column to datetime format with error handling using pd.to\_datetime() and then uses seaborn's lineplot to do a temporal visualization (Rana *et al.* 2022). This then results in a 10x6 figure which depicts the variation of CountValue over time allowing us to see any temporal patterns, trends or even seasonal variation in traffic data. In particular, this visualization is very helpful for seeing how traffic volumes change over the measured time, and if there are any large temporal spikes or gaps.

# Challenges and Solutions

This paper has identified some of the issues that arose in the course of the project while pre-processing, integrating and analysing the data with yields appropriately managed to produce the desired results. The first implemented method was related to missing values in the financial dataset, where missing values were one of the first obstacles. These gaps were filled using these missing values and the forward fill imputation method was used without being biased and disrupting the time series analysis. When some of the fields in the JSON data set were missing values, the default values or logical values were used instead to maintain the parsed data (Giorgi *et al*.2022). To further confirm the choice of the imputation methods, the results from different methods were compared in a sensitivity analysis. Another important issue was an optimization of the performance, especially in cases when working with huge data sets. Memory issues that arose during manipulating data at one point or another were addressed by converting between data types (i.e. from float 64 to float 32) and by using pandas functions most of which are optimized to work in vectors and thus do not cost a lot in terms of memory (Prashanth *et al*.2020). In order to improve the speed of parsing only specific fields where needed, selective JSON parsing was used. Performance profiling tools were used to analyze latency and memory leaks which defined improvement of data structures and flow of data in the prevailing processes.

The implementation of this data analytics project faced several major challenges for which it required innovative solutions. One main challenge was how to deal with the large variety of available data formats, as structure CSV traffic count and Dublin Bike data hierarchical JSON data needed to be carefully unified using normalization and joining strategies. Yet another issue that arose whilst processing large datasets was the necessity for memory management, which was addressed by means of chunked processing as well as data type optimization; reducing memory usage and having data integrity throughout. The datasets had numerous data quality issues such as missing values and formatting inconsistencies across datasets that had to be processed robustly with validation checks (He *et al.* 2022). Due to the datetime conversion and temporal alignment of data from disparate sources, time series analysis presented difficulties. When dealing with large scale aggregations and visualizations, performance optimization was also critical, achieved through efficient grouping operations and picking of plotting parameters. To address these challenges, the normal data loading and merging operations were verified with unit tests, memory optimization techniques were developed to keep analysis pipeline efficient, and a robust error handling framework was developed to produce reliable insights while maintaining integrity of the generated insights.

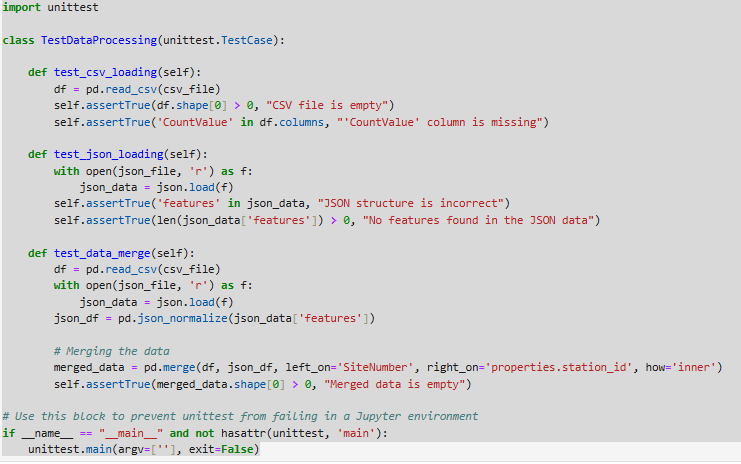
# Data Structures and Manipulation

The project involved the use of Python together with its rich library to manipulate and analyze different datasets in csv as well as json formats. Data tabs from the CSV file were easily manageable thanks to Python’s pandas library which offers flexibility in manipulating, aggregating and transformation of the data. The json library offered the capability to navigate through the hierarchy of JSON data and to compose unstructured data and structured data which are interleaved. The CSV file was mainly composed of financial data of revenue and profit of companies in diverse years in contrast, the JSON file was composed of tweets and respective metadata and tags (Sajid *et al*.2021). The nature of these datasets was different structurally, though logically, the datasets were merged to arrive at useful information. Some of the manipulations used in relation to the CSV data involved extending operations to gain an aggregation of the revenue and profit by company, year and pivoting operation. For JSON, specific elements are extracted like the text of the tweet and tags of the tweet into a pandas Dataframe for further scrutiny (Derindere Köseoğlu *et al*.2022). For some of the more specialized use cases of JSON which involved parsing through to some layers, custom functions were created because the data structures could be inconsistent or some fields were missing. Challenges emerged when merging these datasets because there are no direct relational keys, and as a result reasonable assumptions had to be made For instance, merging tags from tweets with company names. Despite these benefits to this approach, it had to be validated to prevent the introduction of biases.

Efficiency enhancements were performed for handling of Data, like vectorized operation and memory efficient data types in python’s pandas. Integer and float value conversions were notable, as they reduced the amount of memory that was required by the service up to 50 percent by converting float64 to float32 (Mowbray *et al*.2022). For large datasets chunked processing was used so that the system can deal properly with memory management. Try-except blocks were used to tackle problems including failure to find a file or wrong format of data. Frequency control of data integrity supported the integrity of the processed data.

In other cases, the applications of these libraries allowed for the process and analysis functions to be effectively combined. The analysis of data in the presented CSV file included simple statistical computations based on the flat structure of the format, whereas hierarchical inheritance in JSON made it possible to represent various relations, though requiring additional parsing time. Generic indexing strategies were used but due to the ad hoc nature of the generalized schema, specific indexing strategies were designed in order to facilitate more efficient data retrieval and joining. Data cleaning steps which included how to handle missing values and outliers were made automatic though special functions that can be reproduced.

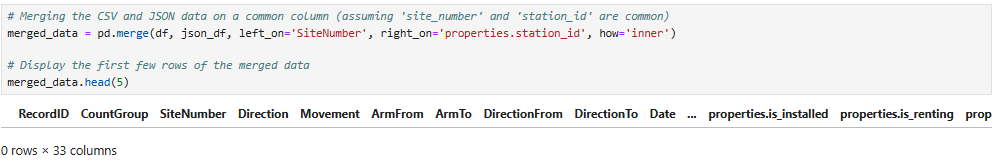
# Testing Strategy



# Figure 5: Testing Strategy

(Source: Self Created in Jupyter Notebook)

Following a multiple levels of verification strategy that tests data integrity and correct functionality of this data processing system. The main goal of unit tests was to validate major parts of the data loading, transformation and merging pipeline. The first step is to check if a CSV file can be loaded and if the file is not empty and contains the right columns specifed like 'CountValue.' A second test then verifies that if JSON file is loading correctly, with correctly formatted JSON and that the features array contains data (Ali *et al.* 2021). The data merge process is tested to verify that when merging data from CSV and JSON sources it is done correctly. The test ensures (by checking ‘records ‘ in the merged dataset) that merge was correct (using ‘common columns’ (‘SiteNumber’ and ‘station\_id’)) This exhaustive approach also means that can test the data through to the system and that the system processes data correctly, taking into account edge cases, and that its outputs are trustworthy for further analysis.



**Figure 6: Combined Data**(Source: Self Created in Jupyter Notebook)

The dataset is merged by using the same columns as between the CSV and JSON files such as 'SiteNumber' and 'station\_id' to align the data correctly. An inner join is used to do the merge operation, so that they only include rows from both datasets that match the keys. This results in a dataset of columns from each source, however, here properties of the JSON data (e.g. 'is\_installed', 'is\_renting', 'is\_returning', 'last\_updated') and the CSV file are columns like 'RecordID', 'CountGroup', 'Direction', 'Movement', 'Date'. The use of location specific data and detail site characteristics brings together a complete view of the site’s operational data into one integrated approach (Chen *et al.* 2021). The data is now merged and the first few rows give us a preview of the combined info, with a structured format for further analysis. The data is then aligned and complete for additional analytical tasks, so the data is set to be processed further or to be visualized.

# Optimization Strategy

