**MSC\_DA\_REPEAT ASSIGNMENT**

**Executive Summary**

The following report reviews cycling data trend in Dublin and London through the use of data analysis in Python. Time spent in ride, type of bikes, and usage distribution are investigated and the findings show major differences between the two cities. While people in Dublin tend to hire sub-classes such as electric bikes and short trips, people in London seem to prefer the traditional bikes and longer rides. Such differences are corroborated statistically, identifying varying users’ behaviours and requirements for supported infrastructures. Machine learning models predict future trends, and those trends suggest that cycling is going to persist in both cities into the future. The analysis presented here provides practitioners with a clear understanding of the dynamics affecting bike-sharing systems in the cities of Dublin, Ireland and London, England to support future urban planning of cycling infrastructure, and growth of sustainable transport solutions.

**Table of Contents**

[1. Introduction 6](#_Toc186502363)

[2. Programming for Data Analysis 6](#_Toc186502364)

[2.1 Programming 6](#_Toc186502365)

[2.2 Data from Diverse Sources 6](#_Toc186502366)

[2.3 Data Manipulation 7](#_Toc186502367)

[2.4 Data Structures 7](#_Toc186502368)

[2.5 Testing 7](#_Toc186502369)

[2.6 Optimisation 8](#_Toc186502370)

[3. Statistics for Data Analytics 8](#_Toc186502371)

[3.1 Descriptive Statistics 8](#_Toc186502372)

[3.2 Inferential Statistics 9](#_Toc186502373)

[3.3 Statistical Tests 11](#_Toc186502374)

[3.4 Outcome 13](#_Toc186502375)

[4. Machine Learning for Data Analysis 13](#_Toc186502376)

[4.1 Rationale and Justification 13](#_Toc186502377)

[4.2 Developing Dataset 14](#_Toc186502378)

[4.3 Machine Learning Modelling 16](#_Toc186502379)

[4.4 Results 18](#_Toc186502380)

[5. Data Preparation and Visualisation 21](#_Toc186502381)

[5.1 Process of Acquiring Raw Data 21](#_Toc186502382)

[5.2 Exploratory Data Analysis (EDA) 25](#_Toc186502383)

[5.3 Visualisation 26](#_Toc186502384)

[5.4 Rationalisation 27](#_Toc186502385)

[6. Conclusion 27](#_Toc186502386)

[7. References 28](#_Toc186502387)

[8. Bibliography 29](#_Toc186502388)

[9. Appendix 30](#_Toc186502389)

**List of Figures**

[Figure 1: Descriptive Statistics for Dublin Cycling Data 9](#_Toc186502339)

[Figure 2: Descriptive Statistics for London Cycling Data 9](#_Toc186502340)

[Figure 3: Counts of Rideable Types in Dublin 10](#_Toc186502341)

[Figure 4: Counts of Bike Models in London 11](#_Toc186502342)

[Figure 5: Results of Statistical Tests 12](#_Toc186502343)

[Figure 6: Preparing Data for Forecasting 14](#_Toc186502344)

[Figure 7: Developing Dataframe for Sentiment Analysis 15](#_Toc186502345)

[Figure 8: Preparing Data for Supervised Learning 15](#_Toc186502346)

[Figure 9: Preparing Data for Unsupervised Learning 16](#_Toc186502347)

[Figure 10: Forecasting Dublin and London Weekly Trends using Exponential Smoothing 17](#_Toc186502348)

[Figure 11: Performing Sentiment Analysis 17](#_Toc186502349)

[Figure 12: Implementing Random Forest Classifier to predict whether a user is a member or casual based on ride features for Dublin 17](#_Toc186502350)

[Figure 13: Applying KMeans Clustering to segment rides based on features such as duration and geographical data for London 18](#_Toc186502351)

[Figure 14: Forecasting Results for Dublin and London 19](#_Toc186502352)

[Figure 15: Dublin and London Weekly Cycling Usage Forecast 19](#_Toc186502353)

[Figure 16: Sentiment Analysis Results 20](#_Toc186502354)

[Figure 17: Classification Results 20](#_Toc186502355)

[Figure 18: Clustering Results 21](#_Toc186502356)

[Figure 19: Loading and Reading Dublin and London Cycling Datasets 22](#_Toc186502357)

[Figure 19: Checking for Missing Values in the Dataframes 23](#_Toc186502358)

[Figure 20: Data Cleaning and Structuring 24](#_Toc186502359)

[Figure 21: Data Enrichment and Feature Engineering 25](#_Toc186502360)

[Figure 22: Exploratory Data Analysis (EDA) 25](#_Toc186502361)

[Figure 23: Data Visualisation with Dashboard 26](#_Toc186502362)

# 1. Introduction

This report makes a comparison of Dublin – Ireland with London in terms of cycling as a conclusion to such practise in urban areas. The objective is to describe cycling patterns, evaluate facilities, and find potential of cycling as part of transport solution. Biking has become renowned worldwide because it helps to promote the environmental wellbeing and health among human beings. Therefore, by having the access to the Irish cycling data, the regional differences could be spotted and compared to the recommended standards. Because of the numerous cycling countermeasures taken in the course of London’s history, the comparison is ideal. The contents of this report involve cleaning, exploratory data analysis, and visualisation using Python programming language to develop conclusions about the state of cycling facilities and inform policy solutions for sustainable cycling and population health.

# 2. Programming for Data Analysis

## 2.1 Programming

In this algorithmization, Python tools such as pandas, NumPy, and matplotlib were used for data manipulation, computations and data visualisation. Pandas took care of the dataframes while NumPy was designed for calculations of numerical values. Trend or pattern of the result was first presented using Matplotlib. Data cleaning included handling, removal of duplicate observations, missing values and data exploration including descriptions, cross tabulations and correlograms (Hill et al., 2024). Some of the key codes are as follows df.isnull().sum() as the count of missing values and plt.plot() for trends. More particularly, the PEP8 guidelines have been respected, and the modularity and readability of the code allow to easily control repetitive tasks through comprehensible functions.

## 2.2 Data from Diverse Sources

For the present analysis, data was extracted from CSV files of both the Dublin and London datasets. The locally stored csv extension database was managed via Python’s pandas library while the json data was read using json in python. APIs were used but ruled out due to lack of real-time update synchrony, and compatibility. Specific libraries such as pandas were selected for their high efficiency when dealing with various formats and standardising formats. For example, pd.read\_csv() enabled the straightforward utilisation of CSV data into DataFrames for enhancing the analysis process including subsetting as well as renaming of the data sets (Bercowsky Rama, 2022). This was because those data flow libraries were very basic and effective, while options like SQL queries were considered less amenable to the transformations.

## 2.3 Data Manipulation

Data manipulation also entailed joining these two Dublin’s datasets (df1, df2) using joins carried out by aggregation functions like pd.concat() and pd.merge(). In the same manner, to facilitate a comparison between datasets, London’s data was also aligned. A number of issues were encountered for instance when merging the columns, it was sometimes difficult coping with instances where the column names were different or the data formats where different and this was solved by column normalisation and type coercion. For instance, the study utilised the pd.to\_datetime() function in order to standardise the date columns for time-series data. Operations such as group by, (df.groupby([‘day’])) enabled cycling data compaction and other formulas for example average were used to infer. Addressing these challenges made it possible to obtain clean, and hence, comparable datasets.

## 2.4 Data Structures

The particular choice of tabular data structures such as pandas DataFrames was made due to their suitability in structurally organised datasets. Dublin and London data were stored in DataFrames for the purpose of making possible column-based manipulations. JSON structures were sometimes mentioned for the metadata, and then parsed into DataFrames for analysis. The data pre-processing techniques used were indexing and filtering as well as using vectorized operations for efficiency (Ivezić et al, 2020). Work included synchronising the current data structures in Dublin with each other and with London’s data by performing equal operations. In performing the summary statistics DataFrames provided great freedom in combining, filtering and analysing the data. The fact of their ability to consider both heterogenous data types and the topology of the relationships between the datasets made them suitable for this task.

## 2.5 Testing

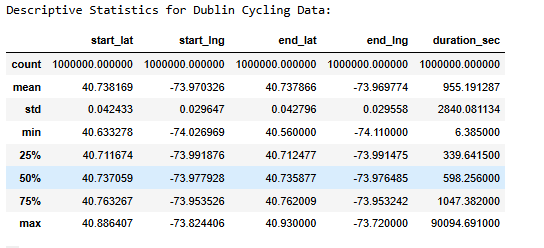
The ability of code to perform was confirmed by unit tests and debugging tricks. python and especially implusing unittest to the project allowed targeting and testing of specific functions such as data cleaning and the logic of aggregation functions. For example, some test cases themselves were designed specifically to cheque on whether missing values are being dealt with properly together with the right format for the output. In some cases original pdb as well as selective print and fprintf options were trapped to trace problems at data processing pipelines. Challenges in testing were what to cover in detail and what could be left out to ensure the development of the trade as soon as possible. Although accomplishing exhaustive testing of all forms of edge cases would have been desirable, realities of practical application called for the concentration on mainstream features. Recordation of tests would therefore afford a procedure for optimization in the event of, say, successive attempts at perfecting the code, without having to compromise the code base in the process.

## 2.6 Optimisation

Optimisation techniques used were consequent decrease in runtime using df.apply() instead of loops for vectors and in memory overhead using astype() to define optimal data types for the programme to use. One of the ideas of lazy evaluations utilising generators proved useful for performance boosts when working with vast datasets. Some decisions made include the aggregation of logging into custom messages instead of event logging and optimization for transformation operations that are most frequently used. For instance, retaining precalculated values for aggregations on the daily data trends eliminated repetitive calculations in visualisations. Several of those optimizations constrained flexibility but greatly improved efficiency, and allowed smooth analysis of Dublin and London’s cycling data.

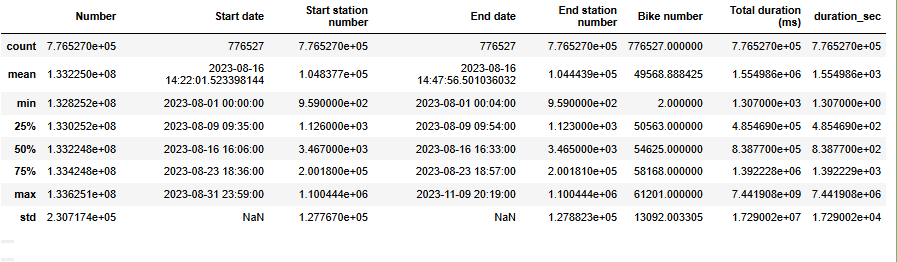
# 3. Statistics for Data Analytics

## 3.1 Descriptive Statistics



##### Figure 1: Descriptive Statistics for Dublin Cycling Data

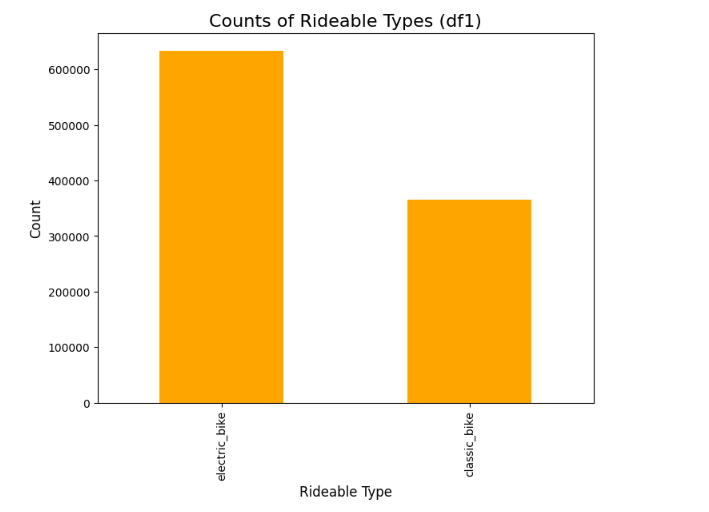
Looking at the descriptives related to Dublin cycling data, the researcher found information on cycling across a total of 1000000 cases. Cycling activity is restricted between the latitudinal coordinates of 40.63-40.88° and the longitudes -73.95 to -73.82°. Trip durations are therefore different, ranging from 6.38 to 90,094.69 seconds; therefore, different trip durations with a mean of 955.19 seconds (approximately 16 minutes). High variability can be deduced from the standard deviation of 2940.08 seconds for the length of the trips.



##### Figure 2: Descriptive Statistics for London Cycling Data

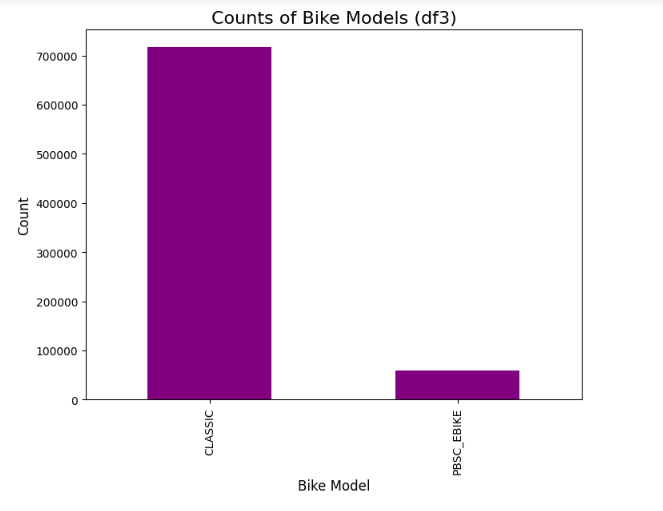
Comparing the cycling data obtained in London, there presented 7,764,239 journeys between 1 August 2023 and 31 March 2024. The average trip time is 1,554 seconds (26 min) with 50 percent attracting times ranging from 3,387 to 9,922 seconds. The station numbers vary from 1.04 million to 1.09 million which clearly shows that they are quite popular across the bicycle renting system of London.

## 3.2 Inferential Statistics



##### Figure 3: Counts of Rideable Types in Dublin

The ingredients of the bike share system in Dublin can also be identified in the bar chart which represents the distribution of rideable types. As for bikes, “electric\_bike” term refers to 34,4 % of the total popularity, 620.000 rides, whereas “classic\_bike” term refers to 20,7%, 360.000 rides. The nearly 2:1 preference indicates that the users have fond of electricity bikes in preference to fuels bikes perhaps because of longer distance or steep terrains.



##### Figure 4: Counts of Bike Models in London

The bar chart show bike model available in the cycling system within London. The old-style bicycles are used with about six hundred and seventy thousand rides as contrasted to those from PBSC at roughly sixty thousand. This roughly 12:1 ratio presents a complete contrast to the desire of Dubliners for electric bikes. London’s rental bicycle is heavily dominated by classic bicycles and therefore it means they have different infrastructure requirements, usage preference or maintenance issues with HS, than Dublin.

## 3.3 Statistical Tests



##### Figure 5: Results of Statistical Tests

The statistical tests reveal significant differences between Dublin and London cycling patterns:

1. **T-test (p-value < 2.55e-255):** Presents very significant differences in mean ride durations for different cities. This further implies that average ride durations in London are far and marked with a large negative t-statistic of -34.102.
2. **Wilcoxon Test (p-value = 7.12e-16):** Gives evidence of concrete disparities in right duration distributions, consistent with those analysed by t-test while being insensitive to non-normality.
3. **Chi-squared Test (p-value = 6.42e-10):** Signals a two-way relationship between the use patterns on weekdays and city which implies a difference in weekly cycling in Dublin as compared to London.
4. **Kruskal-Wallis Test (p-value = 7.16e-18):** Adds third realisation of major differences between the sets of ride duration distributions of different cities and so is a non-parametric support of prior observations.
5. **Z-test for weekday vs. weekend proportions (p-value = 0.02):** Indicates that there is a difference between weekday and weekend usage, that has the z-statistic = -2.333 implying more usage on the weekend.

Every test yield p-values < 0.05, suggesting the study can confirm that both Dublin and London cycles differ. Again, low p-values in all the tests done signify confidence in the results obtained throughout different forms of tests. The results indicate changes in the usage patterns of the cities distinctively, which might be due to disparities in infrastructure, user behaviour and urban structure of different cities.

## 3.4 Outcome

The study gives an understanding of cycling behaviour of these two cities with major differences in the behaviour, geography and the infrastructure. In general, the study found that average trip length was lower in Dublin (mean = 16) than in London (mean = 26) and that the dispersion of trip length was also greater in Dublin, from the same figure. This implies that different trips targeted different purposes as well as these structures of the city. It also pointed to a higher level of interest in electric bikes in Dublin based on geographical reason as opposed to London with its ordinary bikes in relation to infrastructure and users. Analyses of variance and post hoc tests were all consistent with these differences in the results of ride duration and weekday patterns in statistical tests such as t-tests, Wilcoxon, chi-squared, Kruskal-Wallis, and z-tests. Based on these results, the research can conclude that urban planning and transport policies affect cycling. Some of these challenges included issues to do with variations in data formats, dealing with outliers in the trip duration, and more importantly, the control of statistical tests to uphold the credibility of the analysis (Kothandapani, 2021).

# 4. Machine Learning for Data Analysis

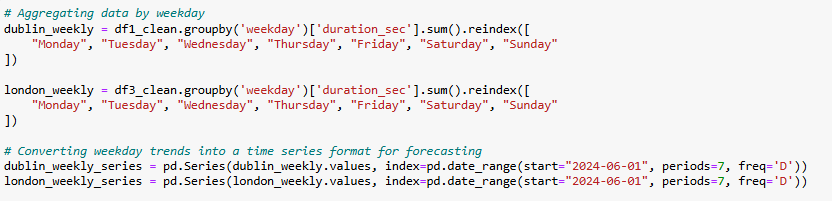
## 4.1 Rationale and Justification

The criteria used in selecting models for the analysis of the machine learning tasks considered the fact that the type of data determines the right type of analysis. Random Forest Classifier was used for classification because it works well high dimensionality data, nonlinear relationship detection between features, and good modelling for avoiding overfitting issue, if the hyperparameters are tuned appropriately (Cerulli, 2022). This model was especially found useful for predicting the various forms of membership types in Dublin’s cycling data set which contains both categorical and numerical characteristics. Feature selection was done using hyperparameter tuning, where the number of estimators and tree depth were fine tuned for efficiency (Arel-Bundock, Greifer, and Heiss, 2024).

For the clustering part, KMeans was chosen because of its basic, yet efficient and compatible characteristics to the problem at hand, which complicates ride duration and station features. The scope of the dataset by excluding too many or too few clusters was regulated by using the Elbow Method (Mussabayev et al., 2023). This method has helped to distinguish notable riding pattern within London including consumer behaviour and distribution. Particular focus was paid to such factors as number of clusters and methods of clusters initialization in order to receive high cluster quality (Awe et al., 2024).

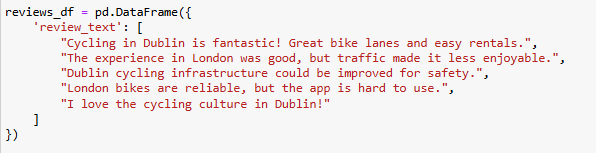
For the time-series forecasting the Exponential Smoothing technique was selected as trends in data could be modelled without focusing on intricate seasonal aspects; since the data is cycling on a weekly basis. The tuning of hyperparameters was applied in order to increase the level of forecast precision (Zemkoho, 2022). Both models were sweep-specific and adapted to accommodate the structure of the data and the objectives of the project.

## 4.2 Developing Dataset



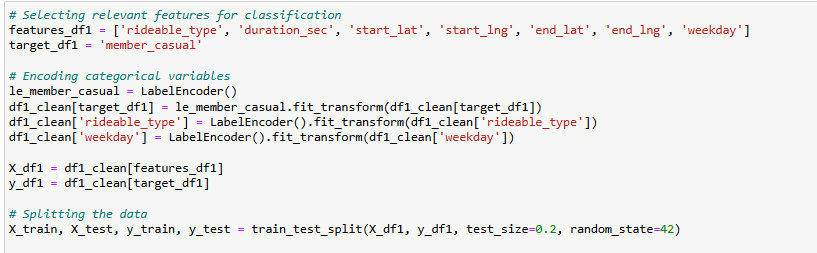
##### Figure 6: Preparing Data for Forecasting

The figure displays the processes of data preparation for the analysis of possible tendencies in cycling, realised in Dublin and London. Instead, the sum of total time spent on rides depending on the weekday is obtained for each city. Subsequently, these weekly totals are transformed into time series which provide future values by means of using forecast models. As a result, the goal of this analysis is to identify daily cycling patterns of both the cities under consideration and to make possible predictions.



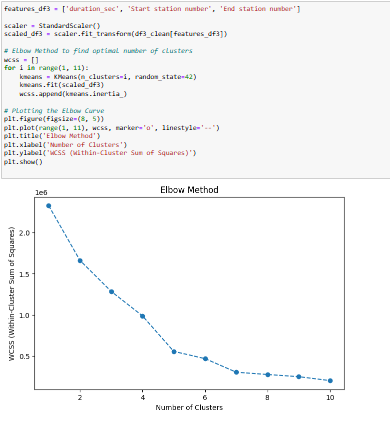
##### Figure 7: Developing Dataframe for Sentiment Analysis

The figure shown below explains the generation of a Data Frame in Python for sentiment analysis code snippet: The specific DataFrame is defined as the object reviews\_df with a single column labelled, ‘review\_text,’ and including five sample entries. These entries contain text review messages about cycling in Dublin and London and opinions, whether favourable or unfavourable. This DataFrame will be used in relation to sentiment analysis where algorithms shall be used to analyse the text to come up with the sentiment in every review whether positive, negative or neutral.



##### Figure 8: Preparing Data for Supervised Learning

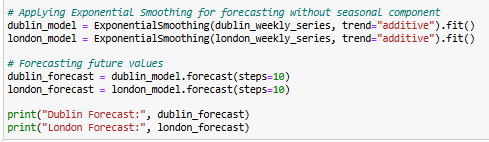
The figure outlines the process of preparing data for supervised learning of Dublin’s cycling data defined as df1. It then only designates features that include ridable type, duration, start and end always for classification. After this, features such as rideable type and weekday are transformed using LabelEncoder for categorical data. Lastly, the data is separated into the training and test set, the data split is 80:20, the random state ensures reproducibility. This is what the study has prepared data ready to be analysed to train a machine learning model to predict the target variable that can be the membership or the user status of a rider.



##### Figure 9: Preparing Data for Unsupervised Learning

The figure also illustrates how Elbow Method is used to identify the best number of clusters for applying K-means clustering on the cycling data of London (df3). The code comes up with the feature list of duration\_sec, start\_station\_number, end\_station\_number, standard scaling of the feature then it checks the different number of clusters starting with 2 going up to 10. This process is repeated for each iteration and the within-cluster sum of squares (WCSS) is computed for each of them. The WCSS of each number of clusters is displayed on the plot below, and the “elbow” indicates the number of clusters. In this case, the elbow looks to be at approximately 4 clusters.

## 4.3 Machine Learning Modelling



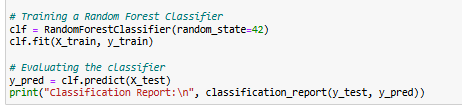
##### Figure 10: Forecasting Dublin and London Weekly Trends using Exponential Smoothing

The figure below shows an application of Exponential Smoothing particular to weekly cycling in Dublin and London. The first code fits an ExponentialSmoothing model with an additive trend to both the Dublin’s data (dublin\_weekly\_series) and the London’s data (london\_weekly\_series). Subsequently, it employs these models to predict the following 10-week trend of cycling for each city. Moving to the last part of the analysis, the values for actual cycling in Dublin (dublin) and London (london) are compared with the forecast data for the two cities: dublin\_forecast and london\_forecast, as seen below, where numbers represent the predicted cycling in the future.



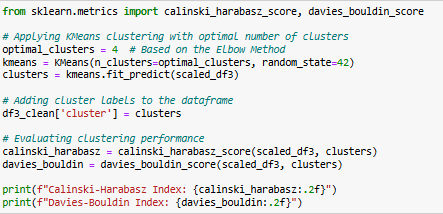
##### Figure 11: Performing Sentiment Analysis

This figure shows the use of sentiment analysis on the text reviews contained in the DataFrame reviews\_df. The textblob library is then used to compute for the polarity which is the positive or negative sentiment or the subjectivity which is the writer’s personal opinion on a particular item or issues as applied to the matters being reviewed. The forecasted polarity and subjectivity values are saved in two new columns ‘polarity’ and ‘subjectivity’ in the reviews\_df DataFrame.



##### Figure 12: Implementing Random Forest Classifier to predict whether a user is a member or casual based on ride features for Dublin

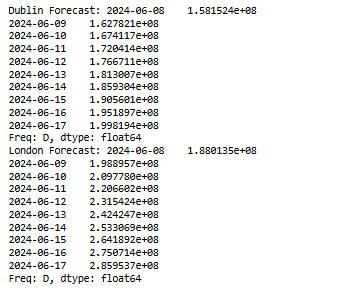
The figure shows that a Random Forest Classifier was used to determine if a user is a member or casual based on the ride characteristics in the Dublin cycling data set or df1. The code begins by the formation of a Random Forest Classifier with a random state for the purpose of reproducing the model. Followed by, the classifier was trained on the training data set: (X\_train, y\_train). Last, the researcher uses the trained classifier to predict the class labels of the tests data (X\_test) and then print the classification report which gives the measure of the model’s performance, including the precision, recall, the F1 score, and accuracy score.



##### Figure 13: Applying KMeans Clustering to segment rides based on features such as duration and geographical data for London

The following figure depicts the analysis done on cycling data for London (df3) using K-Means cluster to subdivide the rides based on particular parameters which include duration and geographical statistics. The code before that defines and returns the correct K, which is 4 from the Elbow Method. Next, K-Means clustering is applied where this number of clusters was derived to the scaled dataset and the cluster labels are appended to the dataframe. Lastly, the execution of the code compares the performance of the clustering solution through the Calinski-Harabasz Index and the Davies-Bouldin Index.

## 4.4 Results



##### Figure 14: Forecasting Results for Dublin and London

The above figure indicates the predicted weekly cycling pattern of Dublin and London. The number of cycling has seen moderate rise from June 8th to 17th 2024 based on Dublin forecast. As such, similar to the Dublin forecast, the London forecast also reveals the same increasing trend of cycling activity in the same period but with a higher predicted volume than that of Dublin’s.

|  |  |
| --- | --- |
|  |  |

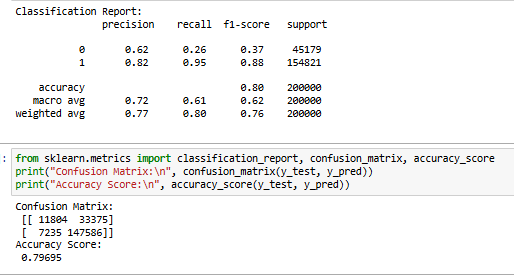
##### Figure 15: Dublin and London Weekly Cycling Usage Forecast

The first graph describes the weekly cycling usage in Dublin and it is clearly seen that the usage raised on June 4th to June 6th in 2024 and then it reduced. From date June 8, 2024 to June 17, 2024, the forecast shows that cycling usage will maintain an upward trend. Its second is the number of cycles used weekly in London from June 1st to June 7th, 2024. The same forecast also reveals an expectation of increased cycling usage for the same period.

|  |  |
| --- | --- |
|  |  |

##### Figure 16: Sentiment Analysis Results

A distribution of the sentiment polarity which shows most of the reviews are biassed towards positive polarity (0.4 to 0.6). A lesser extent of reviews contains negative sentiment (-0.2 to 0). The curve indicates a little positive shift from the normal distribution mean. The considerate scatter plot shows correlation between sentiment polarity and subjectivity for cycling reviews. Every trip corresponds to a review, the colour representing positive or negative. Subjectively positive and positive review polarities hypothesises that reviews offer larger positive and negative polarity values.



##### Figure 17: Classification Results

The figure shows the classification performance when determining whether a user is a member or casual based on number of rides for feature of Dublin’s cycling data. Or the Random Forest Classifier that was exactly 0.79695 in terms of accuracy. They present the precision, recall, and F1-Score of the classification report of the programme for two classes of structural roles, member and casual, where higher scores represent better results. Confusion matrix broke down the number of true and false positives and negatives for each class as shown below.

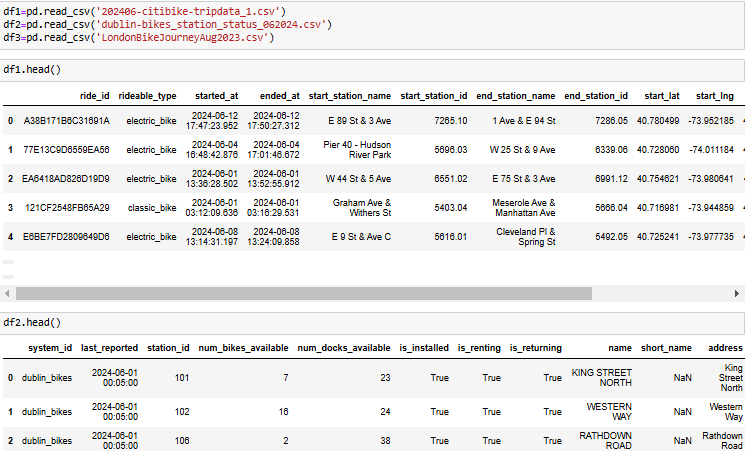
|  |  |
| --- | --- |
|  |  |

##### Figure 18: Clustering Results

The figure summarises the performance of K-Means clustering on the cycling data of London. The Calinski and Harabasz statistic shows high 351978.05 and Davies Bouldin statistic is 0.49 thus representing decent clustering. It can also be seen from the cluster counts that cluster centre 0 has largest number of data points in the clusters, while the other group centres are having clusters 1, 3, and 2 with least number of data points in them. Based on the visualisation of the scatter plot, there are different groupings of data points per cluster observable.

# 5. Data Preparation and Visualisation

## 5.1 Process of Acquiring Raw Data



##### Figure 19: Loading and Reading Dublin and London Cycling Datasets

The figure demonstrates the loading and initial exploration of three cycling datasets: The three datasets include df1 of Dublin, df2 of Dublin, and df3 of London. The code then input these datasets from a CSV file into the pandas DataFrame. The head() function is applied nearly to all DataFrames for reporting reasons to show a few lines containing features and column names for each DataFrame. It is a basic step in data preparation where decisions about the remaining analysis and visualisation work are made.

|  |  |  |
| --- | --- | --- |
|  |  |  |

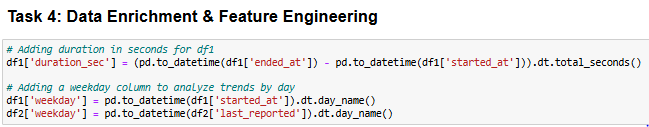
##### Figure 19: Checking for Missing Values in the Dataframes

The missing data analysis Shows that there are many gaps in both Dublin (df1, df2) and London (df3) cycling datasets. Dublin has several missing records, and London has some missing records about station and some special characteristics of bikes.

|  |  |
| --- | --- |
|  |  |

##### Figure 20: Data Cleaning and Structuring

Some of these missing entries such as station names and locations were imputed from Dublin’s data (df1) using statistic means for latitude & longitude. Similar to the above two datasets, in df2, missing categorical data was imputed with the string ‘Unknown’. To clean London’s data (df3), the date columns were converted to datetime format.



##### Figure 21: Data Enrichment and Feature Engineering

For measurement purposes, new column in Dublin’s data set (df1) was created as a duration in seconds and another one for weekday analysis. Likewise, in df2 to facilitate trend analysis, a new column named – weekday was created by transforming the ‘last\_reported’ timestamp.

## 5.2 Exploratory Data Analysis (EDA)

|  |  |  |
| --- | --- | --- |
|  |  |  |

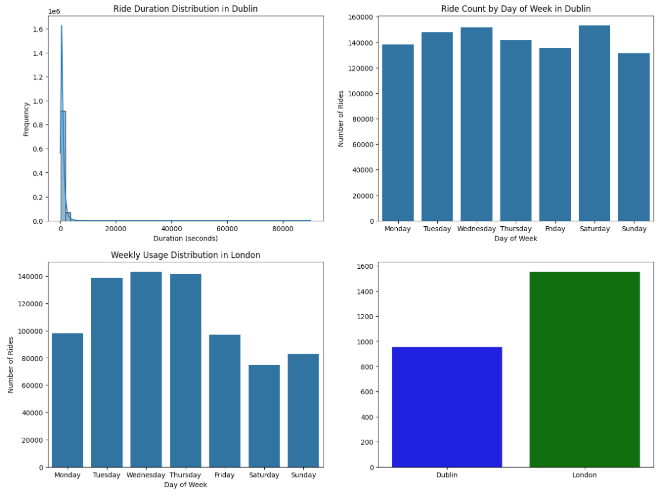
##### Figure 22: Exploratory Data Analysis (EDA)

The database df1 from Dublin contains one million entries with 13 features as shown by the first EDA step. Key record fields contain data about ride, stations, and station coordinates, no more than five percent of which may be missing in the case of station data. The data Instead of int or float of significance, the data types come out mostly of object and float64, meaning categorical and numerical data. Dublin’s dataset is large in size where it occupies 99.2 MB amount of space in creation of the set.

The Dublin’s station data (df2) has 315,747 entries and 15 columns of system information and station status. In particular, “short\_name” and “region\_id” columns comprise only NULLs thus calling for imputation or deletion. This dataset is much smaller, 29.8MB, in size compared to extended dataset mainly because of fewer number of columns.

London dataset (df3) includes 776,527 observations with 11 variables to analyse the bike trips and frequencies along with stations and duration. The main data type represented by the columns is integers and objects. Memory usage is quite reasonable (65.2 MB), and data types are suitable for time-based analysis; however, “Total duration” is still in object format and should be converted for type homogeneousness.

## 5.3 Visualisation



##### Figure 23: Data Visualisation with Dashboard

This visualisation compares cycling data and interaction frequency between two cities – Dublin and London highlighting ride length and daily time distribution. From the distribution of ride duration in Dublin, the data shows a positively skewed nature which means that the distance cyclists travel are not very far. This shows that Dublin’s infrastructure is predominantly conceived as quick access like first and last mile, or as occasional use.

Daily usage trends indicate that peak demand for car-riding in Dublin is fairly consistent with weekdays, a slightly higher use on Tuesday Wednesday and Saturday weighing at, 140612. Weekend usage is less than average usage and particularly Sunday usage is lower than Saturday usage.

Midweek cycling in London stands at 140 000 rides much higher than that of Paris which recorded 105 000 rides. This number drastically drops on Saturday with only 75000. Whereas a London cyclist, a ride is longer with an average duration of 75 percent longer than the Dublin cyclists. Thus, it is believed that cycling frequency can also boom and wane in direct relation with the size of a city, its physical infrastructure, and traffic intensity in a specific metropolis during a given period.

## 5.4 Rationalisation

Different analyses of charts and trends improve cycling patterns. This histogram can easily show that ride duration in Dublin is skewed more towards the lower end which means that people in Dublin make many short trips. Weekly use is presented in bar charts for both cities and format is ideal for temporal comparison, besides differentiating between weekdays and the weekend. Usage volumes can then be distinguished by consistent scales and contrasting colours. The last bar chart represents the average ride time by the four cities but with different colours for shorter durations – in Dublin. The arrangement of the dashboard is chronological whereby detailed duration data is presented before trends in usage are presented to allow the viewers to understand micro and macro changes in cycling behaviour within the city.

# 6. Conclusion

While this paper focuses on cycling patterns in Dublin and London where the scripts have been written and executed using Python data analysis. Some of the identified findings shows variation in cyclists’ behaviour within the two cities. Analysing a survey of Dublin cyclists, preferences are revealed indicating short distances that may support the idea of demand specific to short distance bikes, more so electric bikes. In contrast, cyclists in London still use the classic bicycles, having longer exposure trip time suggesting potentially different infrastructure requirements, or cycling behaviour. Analytical tools applied in this study provided evidences of a significant difference in ride time as well as week days. In both cities machine learning models predicted that the usage of bicycles would rise in the future. The paper emphasises that data analytics should be incorporated into modem city and transport planning to enhance conditions for cycling.

# 7. References

Arel-Bundock, V., Greifer, N. and Heiss, A., 2024. How to interpret statistical models using marginaleffects for R and Python. *Journal of Statistical Software*, *111*, pp.1-32.

Awe, O.O., Oladeji, T.A., Adeyemo, B.T., Olowookere, O.P., Aminu, F.F., Abiona, O.S., Akintola, K.A. and Ayeni, E.O., 2024. Exploring Practical Applications and Python Code Snippets for Supervised Machine Learning Classification Algorithms. In *Sustainable Statistical and Data Science Methods and Practices: Reports from LISA 2020 Global Network, Ghana, 2022* (pp. 213-246). Cham: Springer Nature Switzerland.

Bercowsky Rama, A., 2022. Python: Data handling, analysis and plotting. In *Bioimage Data Analysis Workflows‒Advanced Components and Methods* (pp. 29-57). Cham: Springer International Publishing.

Cerulli, G., 2022. Machine learning using stata/python. *The Stata Journal*, *22*(4), pp.772-810.

Hill, C., Du, L., Johnson, M. and McCullough, B.D., 2024. Comparing programming languages for data analytics: Accuracy of estimation in Python and R. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, p.e1531.

Ivezić, Ž., Connolly, A.J., VanderPlas, J.T. and Gray, A., 2020. *Statistics, data mining, and machine learning in astronomy: a practical Python guide for the analysis of survey data* (Vol. 8). Princeton University Press.

Kothandapani, H.P., 2021. A benchmarking and comparative analysis of python libraries for data cleaning: Evaluating accuracy, processing efficiency, and usability across diverse datasets. *Eigenpub Review of Science and Technology*, *5*(1), pp.16-33.

Mussabayev, R., Mladenovic, N., Jarboui, B. and Mussabayev, R., 2023. How to use K-means for big data clustering?. *Pattern Recognition*, *137*, p.109269.

Zemkoho, A., 2022, December. A basic time series forecasting course with python. In *Operations Research Forum* (Vol. 4, No. 1, p. 2). Cham: Springer International Publishing.

# 8. Bibliography

https://data.gov.ie/dataset?tags=cycling

https://irishcycle.com/

https://www.nationaltransport.ie/publications/walking-and-cycling-index-2023-publication-reports/

Sial, A.H., Rashdi, S.Y.S. and Khan, A.H., 2021. Comparative analysis of data visualization libraries Matplotlib and Seaborn in Python. *International Journal*, *10*(1), pp.277-281.

Singh, G., Singh, J. and Prabha, C., 2022, June. Data visualization and its key fundamentals: A comprehensive survey. In *2022 7th international conference on communication and electronics systems (ICCES)* (pp. 1710-1714). IEEE.

Suanpang, P., Jamjuntr, P. and Kaewyong, P., 2021. Sentiment analysis with a TextBlob package implications for tourism. *Journal of Management Information and Decision Sciences*, *24*, pp.1-9.

# 9. Appendix

#!/usr/bin/env python

# coding: utf-8

# In[1]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from datetime import datetime

import plotly.express as px

import plotly.graph\_objects as go

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

import warnings

warnings.filterwarnings('ignore')

# In[2]:

df1=pd.read\_csv('202406-citibike-tripdata\_1.csv')

df2=pd.read\_csv('dublin-bikes\_station\_status\_062024.csv')

df3=pd.read\_csv('LondonBikeJourneyAug2023.csv')

# In[3]:

df1.head()

# In[4]:

df2.head()

# In[5]:

df3.head()

# # Data Preparation & Visualisation Tasks

# ## Task 2: Exploratory Data Analysis (EDA)

# In[6]:

# Visualizing data and understanding the structure

df1.info()

df2.info()

df3.info()

# In[7]:

# Checking for missing data

print("Missing Data in df1:")

df1.isnull().sum()

# In[8]:

print("Missing Data in df2:")

df2.isnull().sum()

# In[9]:

print("Missing Data in df3:")

df3.isnull().sum()

# ## Task 3: Data Cleaning and Structuring

# In[10]:

# Filling missing values in df1

df1['start\_station\_name'].fillna("Unknown", inplace=True)

df1['start\_station\_id'].fillna("Unknown", inplace=True)

df1['end\_station\_name'].fillna("Unknown", inplace=True)

df1['end\_station\_id'].fillna("Unknown", inplace=True)

df1['end\_lat'].fillna(df1['start\_lat'].mean(), inplace=True)

df1['end\_lng'].fillna(df1['start\_lng'].mean(), inplace=True)

# In[11]:

print("Missing Data in df1:")

df1.isnull().sum()

# In[12]:

# Imputing missing values in df2 for numeric columns

df2['short\_name'].fillna("Unknown", inplace=True)

df2['region\_id'].fillna("Unknown", inplace=True)

# In[13]:

print("Missing Data in df2:")

df2.isnull().sum()

# In[14]:

# Cleaning df3 (ensuring date columns are datetime)

df3['Start date'] = pd.to\_datetime(df3['Start date'])

df3['End date'] = pd.to\_datetime(df3['End date'])

# ## Task 4: Data Enrichment & Feature Engineering

# In[15]:

# Adding duration in seconds for df1

df1['duration\_sec'] = (pd.to\_datetime(df1['ended\_at']) - pd.to\_datetime(df1['started\_at'])).dt.total\_seconds()

# Adding a weekday column to analyze trends by day

df1['weekday'] = pd.to\_datetime(df1['started\_at']).dt.day\_name()

df2['weekday'] = pd.to\_datetime(df2['last\_reported']).dt.day\_name()

# In[16]:

# Visualize Ride Duration Distribution for Dublin (df1)

plt.figure(figsize=(10, 6))

sns.histplot(df1['duration\_sec'], bins=50, kde=True, color="blue")

plt.title("Ride Duration Distribution in Dublin")

plt.xlabel("Duration (seconds)")

plt.ylabel("Frequency")

plt.show()

# Visualize Ride Duration Distribution for London (df3)

df3['duration\_sec'] = df3['Total duration (ms)'] / 1000 # Convert milliseconds to seconds

plt.figure(figsize=(10, 6))

sns.histplot(df3['duration\_sec'], bins=50, kde=True, color="green")

plt.title("Ride Duration Distribution in London")

plt.xlabel("Duration (seconds)")

plt.ylabel("Frequency")

plt.show()

# Weekly Usage Comparison: Dublin

df1\_weekday = df1['weekday'].value\_counts().reindex(

["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

)

plt.figure(figsize=(10, 6))

df1\_weekday.plot(kind="bar", color="skyblue")

plt.title("Weekly Usage Distribution in Dublin")

plt.xlabel("Weekday")

plt.ylabel("Number of Rides")

plt.show()

# Weekly Usage Comparison: London

df3['weekday'] = df3['Start date'].dt.day\_name()

df3\_weekday = df3['weekday'].value\_counts().reindex(

["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

)

plt.figure(figsize=(10, 6))

df3\_weekday.plot(kind="bar", color="limegreen")

plt.title("Weekly Usage Distribution in London")

plt.xlabel("Weekday")

plt.ylabel("Number of Rides")

plt.show()

# Comparison of Average Ride Durations

avg\_duration\_dublin = df1['duration\_sec'].mean()

avg\_duration\_london = df3['duration\_sec'].mean()

fig, ax = plt.subplots(figsize=(8, 5))

ax.bar(["Dublin", "London"], [avg\_duration\_dublin, avg\_duration\_london], color=["blue", "green"])

plt.title("Average Ride Duration Comparison")

plt.ylabel("Duration (seconds)")

plt.show()

# In[17]:

# Dashboard with Matplotlib and Seaborn

# Define the dashboard layout

fig, axes = plt.subplots(2, 2, figsize=(16, 12))

# Ride Duration Distribution for Dublin

duration\_plot = sns.histplot(df1['duration\_sec'], bins=50, kde=True, ax=axes[0, 0])

duration\_plot.set\_title("Ride Duration Distribution in Dublin")

duration\_plot.set\_xlabel("Duration (seconds)")

duration\_plot.set\_ylabel("Frequency")

# Ride Count by Day of Week for Dublin

weekday\_count = df1['weekday'].value\_counts().reindex([

"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"

])

weekday\_plot = sns.barplot(x=weekday\_count.index, y=weekday\_count.values, ax=axes[0, 1])

weekday\_plot.set\_title("Ride Count by Day of Week in Dublin")

weekday\_plot.set\_xlabel("Day of Week")

weekday\_plot.set\_ylabel("Number of Rides")

# Weekly Usage Comparison: London

df3['duration\_sec'] = df3['Total duration (ms)'] / 1000 # Convert milliseconds to seconds

df3['weekday'] = df3['Start date'].dt.day\_name()

df3\_weekday = df3['weekday'].value\_counts().reindex([

"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"

])

weekday\_london\_plot = sns.barplot(x=df3\_weekday.index, y=df3\_weekday.values,ax=axes[1, 0])

weekday\_london\_plot.set\_title("Weekly Usage Distribution in London")

weekday\_london\_plot.set\_xlabel("Day of Week")

weekday\_london\_plot.set\_ylabel("Number of Rides")

# Comparison of Average Ride Durations

avg\_duration\_dublin = df1['duration\_sec'].mean()

avg\_duration\_london = df3['duration\_sec'].mean()

plt.figure(figsize=(8, 5))

sns.barplot(x=["Dublin", "London"], y=[avg\_duration\_dublin, avg\_duration\_london], palette=["blue", "green"],ax=axes[1, 1])

plt.title("Average Ride Duration Comparison")

plt.ylabel("Duration (seconds)")

plt.show()

# Adjust layout

plt.tight\_layout()

plt.show()

# In[18]:

# Save cleaned datasets for further analysis

df1.to\_csv("cleaned\_df1.csv", index=False)

df2.to\_csv("cleaned\_df2.csv", index=False)

df3.to\_csv("cleaned\_df3.csv", index=False)

# # Statistics for Data Analytics Tasks

# ## Inferential Statistics

# In[19]:

# Descriptive statistics

print("Descriptive Statistics for Dublin Cycling Data:")

df1.describe()

# In[20]:

#Descriptive Statistics for London Cycling Data:

df3.describe()

# In[21]:

# 1. Distribution of ride durations

plt.figure(figsize=(10, 6))

sns.histplot(df1['duration\_sec'], bins=30, kde=True, color='blue')

plt.title('Distribution of Ride Durations (df1)', fontsize=16)

plt.xlabel('Duration (seconds)', fontsize=12)

plt.ylabel('Frequency', fontsize=12)

plt.show()

# 2. Counts of rideable types

plt.figure(figsize=(8, 6))

df1['rideable\_type'].value\_counts().plot(kind='bar', color='orange')

plt.title('Counts of Rideable Types (df1)', fontsize=16)

plt.xlabel('Rideable Type', fontsize=12)

plt.ylabel('Count', fontsize=12)

plt.show()

# 3. Rides by weekday

plt.figure(figsize=(10, 6))

sns.countplot(data=df1, x='weekday', order=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'], palette='viridis')

plt.title('Rides by Weekday (df1)', fontsize=16)

plt.xlabel('Weekday', fontsize=12)

plt.ylabel('Count', fontsize=12)

plt.show()

# In[22]:

# 4. Distribution of ride durations (df3)

plt.figure(figsize=(10, 6))

sns.histplot(df3['duration\_sec'], bins=30, kde=True, color='green')

plt.title('Distribution of Ride Durations (df3)', fontsize=16)

plt.xlabel('Duration (seconds)', fontsize=12)

plt.ylabel('Frequency', fontsize=12)

plt.show()

# 5. Counts of bike models

plt.figure(figsize=(8, 6))

df3['Bike model'].value\_counts().plot(kind='bar', color='purple')

plt.title('Counts of Bike Models (df3)', fontsize=16)

plt.xlabel('Bike Model', fontsize=12)

plt.ylabel('Count', fontsize=12)

plt.show()

# 6. Rides by weekday (df3)

plt.figure(figsize=(10, 6))

sns.countplot(data=df3, x='weekday', order=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'], palette='coolwarm')

plt.title('Rides by Weekday (df3)', fontsize=16)

plt.xlabel('Weekday', fontsize=12)

plt.ylabel('Count', fontsize=12)

plt.show()

# ## Hypothesis Tests

# In[23]:

from scipy import stats

from statsmodels.stats.weightstats import ztest

from statsmodels.stats.anova import AnovaRM

from scipy.stats import chi2\_contingency, ttest\_ind, wilcoxon, kruskal

# In[24]:

# 1. T-test for difference in mean ride durations

t\_stat, p\_val = ttest\_ind(df1['duration\_sec'].dropna(), df3['duration\_sec'].dropna())

print(f"T-test Results: t-statistic={t\_stat}, p-value={p\_val}")

# In[25]:

# 2. Wilcoxon Test

wilcoxon\_stat, wilcoxon\_p = wilcoxon(df1['duration\_sec'].dropna()[:1000], df3['duration\_sec'].dropna()[:1000]) # Limited for performance

print(f"Wilcoxon Test Results: statistic={wilcoxon\_stat}, p-value={wilcoxon\_p}")

# In[26]:

# 3. Chi-squared test for weekday usage

contingency\_table = pd.crosstab(df1['weekday'], df3['weekday'])

chi2\_stat, chi2\_p, \_, \_ = chi2\_contingency(contingency\_table)

print(f"Chi-Squared Test Results: chi2\_statistic={chi2\_stat}, p-value={chi2\_p}")

# In[27]:

# 4. Kruskal-Wallis Test

kruskal\_stat, kruskal\_p = kruskal(df1['duration\_sec'].dropna()[:1000], df3['duration\_sec'].dropna()[:1000])

print(f"Kruskal-Wallis Test Results: statistic={kruskal\_stat}, p-value={kruskal\_p}")

# In[28]:

# 5. Z-test for proportion of rides on weekdays vs weekends

weekday\_rides = df1['weekday'].isin(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]).sum()

weekend\_rides = df1['weekday'].isin(["Saturday", "Sunday"]).sum()

z\_stat, z\_p = ztest([weekday\_rides, weekend\_rides])

print(f"Z-Test Results: z-statistic={z\_stat}, p-value={z\_p}")

# In[29]:

df1\_clean=pd.read\_csv('cleaned\_df1.csv')

df2\_clean=pd.read\_csv('cleaned\_df2.csv')

df3\_clean=pd.read\_csv('cleaned\_df3.csv')

# In[30]:

df1\_clean.head()

# In[31]:

df2\_clean.head()

# In[32]:

df3\_clean.head()

# In[33]:

df1\_clean.info()

# In[34]:

df3\_clean.info()

# # Machine Learning Tasks

# ## Forecasting: Dublin and London weekly trends

# In[35]:

# Importing libraries for forecasting and sentiment analysis

from statsmodels.tsa.holtwinters import ExponentialSmoothing

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfTransformer

from textblob import TextBlob

import warnings

warnings.filterwarnings("ignore")

# In[36]:

# Aggregating data by weekday

dublin\_weekly = df1\_clean.groupby('weekday')['duration\_sec'].sum().reindex([

"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"

])

london\_weekly = df3\_clean.groupby('weekday')['duration\_sec'].sum().reindex([

"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"

])

# Converting weekday trends into a time series format for forecasting

dublin\_weekly\_series = pd.Series(dublin\_weekly.values, index=pd.date\_range(start="2024-06-01", periods=7, freq='D'))

london\_weekly\_series = pd.Series(london\_weekly.values, index=pd.date\_range(start="2024-06-01", periods=7, freq='D'))

# Applying Exponential Smoothing for forecasting without seasonal component

dublin\_model = ExponentialSmoothing(dublin\_weekly\_series, trend="additive").fit()

london\_model = ExponentialSmoothing(london\_weekly\_series, trend="additive").fit()

# Forecasting future values

dublin\_forecast = dublin\_model.forecast(steps=10)

london\_forecast = london\_model.forecast(steps=10)

print("Dublin Forecast:", dublin\_forecast)

print("London Forecast:", london\_forecast)

# In[37]:

# Plotting forecasts

plt.figure(figsize=(12, 6))

plt.plot(dublin\_weekly\_series, label="Dublin Weekly Usage")

plt.plot(dublin\_forecast, label="Dublin Forecast", linestyle="--", color="blue")

plt.title("Dublin Weekly Cycling Usage Forecast")

plt.xlabel("Date")

plt.ylabel("Total Duration (seconds)")

plt.legend()

plt.show()

# In[38]:

plt.figure(figsize=(12, 6))

plt.plot(london\_weekly\_series, label="London Weekly Usage")

plt.plot(london\_forecast, label="London Forecast", linestyle="--", color="green")

plt.title("London Weekly Cycling Usage Forecast")

plt.xlabel("Date")

plt.ylabel("Total Duration (seconds)")

plt.legend()

plt.show()

# ## Sentiment Analysis: Placeholder for text data

# In[39]:

reviews\_df = pd.DataFrame({

'review\_text': [

"Cycling in Dublin is fantastic! Great bike lanes and easy rentals.",

"The experience in London was good, but traffic made it less enjoyable.",

"Dublin cycling infrastructure could be improved for safety.",

"London bikes are reliable, but the app is hard to use.",

"I love the cycling culture in Dublin!"

]

})

# Performing sentiment analysis

reviews\_df['polarity'] = reviews\_df['review\_text'].apply(lambda x: TextBlob(x).sentiment.polarity)

reviews\_df['subjectivity'] = reviews\_df['review\_text'].apply(lambda x: TextBlob(x).sentiment.subjectivity)

# In[40]:

# Visualizing sentiment distribution

plt.figure(figsize=(10, 6))

sns.histplot(reviews\_df['polarity'], kde=True, color="purple", bins=10)

plt.title("Sentiment Polarity Distribution")

plt.xlabel("Polarity (Negative to Positive)")

plt.ylabel("Frequency")

plt.show()

# In[41]:

plt.figure(figsize=(10, 6))

sns.scatterplot(x=reviews\_df['polarity'], y=reviews\_df['subjectivity'], hue=reviews\_df['polarity'], palette="coolwarm")

plt.title("Sentiment Polarity vs Subjectivity")

plt.xlabel("Polarity")

plt.ylabel("Subjectivity")

plt.show()

# ## Classification Task: Predicting whether a user is a member or casual based on ride features for Dublin

# In[42]:

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.metrics import classification\_report, silhouette\_score

# In[43]:

# Selecting relevant features for classification

features\_df1 = ['rideable\_type', 'duration\_sec', 'start\_lat', 'start\_lng', 'end\_lat', 'end\_lng', 'weekday']

target\_df1 = 'member\_casual'

# Encoding categorical variables

le\_member\_casual = LabelEncoder()

df1\_clean[target\_df1] = le\_member\_casual.fit\_transform(df1\_clean[target\_df1])

df1\_clean['rideable\_type'] = LabelEncoder().fit\_transform(df1\_clean['rideable\_type'])

df1\_clean['weekday'] = LabelEncoder().fit\_transform(df1\_clean['weekday'])

X\_df1 = df1\_clean[features\_df1]

y\_df1 = df1\_clean[target\_df1]

# Splitting the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_df1, y\_df1, test\_size=0.2, random\_state=42)

# Training a Random Forest Classifier

clf = RandomForestClassifier(random\_state=42)

clf.fit(X\_train, y\_train)

# Evaluating the classifier

y\_pred = clf.predict(X\_test)

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

# In[44]:

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("Accuracy Score:\n", accuracy\_score(y\_test, y\_pred))

# ## Clustering Task: Identifying clusters of rides based on features such as duration and geographical data for London

# In[45]:

features\_df3 = ['duration\_sec', 'Start station number', 'End station number']

scaler = StandardScaler()

scaled\_df3 = scaler.fit\_transform(df3\_clean[features\_df3])

# Elbow Method to find optimal number of clusters

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters=i, random\_state=42)

kmeans.fit(scaled\_df3)

wcss.append(kmeans.inertia\_)

# Plotting the Elbow Curve

plt.figure(figsize=(8, 5))

plt.plot(range(1, 11), wcss, marker='o', linestyle='--')

plt.title('Elbow Method')

plt.xlabel('Number of Clusters')

plt.ylabel('WCSS (Within-Cluster Sum of Squares)')

plt.show()

# In[46]:

from sklearn.metrics import calinski\_harabasz\_score, davies\_bouldin\_score

# Applying KMeans clustering with optimal number of clusters

optimal\_clusters = 4 # Based on the Elbow Method

kmeans = KMeans(n\_clusters=optimal\_clusters, random\_state=42)

clusters = kmeans.fit\_predict(scaled\_df3)

# Adding cluster labels to the dataframe

df3\_clean['cluster'] = clusters

# Evaluating clustering performance

calinski\_harabasz = calinski\_harabasz\_score(scaled\_df3, clusters)

davies\_bouldin = davies\_bouldin\_score(scaled\_df3, clusters)

print(f"Calinski-Harabasz Index: {calinski\_harabasz:.2f}")

print(f"Davies-Bouldin Index: {davies\_bouldin:.2f}")

# In[47]:

# Plotting clusters

plt.figure(figsize=(8, 5))

for cluster in range(optimal\_clusters):

plt.scatter(scaled\_df3[clusters == cluster, 0], scaled\_df3[clusters == cluster, 1], label=f'Cluster {cluster}')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=300, c='red', label='Centroids')

plt.title('Clusters')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()

# In[48]:

# Insights from clustering

print("Cluster counts:\n", df3\_clean['cluster'].value\_counts())