Prediction of Dublin's Traffic Congestion Using Machine Learning

Vanshika Sharma
School of computing
National College of Ireland
Dublin, Ireland
x23198389@student.ncirl.ie

***Abstract*—This study develops a machine learning model that predicts Dublin's traffic congestion as Low, Medium, or High from spatial and temporal data. A comparison of Logistic Regression, Random Forest, and XGBoost performance shows that XGBoost had the best accuracy (Macro F1=0.80), proving it effective for real time smart city traffic management**

I. Introduction

How much time does a city like Dublin lose every day to traffic? How much fuel is wasted everyday while waiting for the red signal to turn green? Traffic congestion is not just an inconvenience, it is direct relation of a city's productivity too its sustainability. A report[1] published by The Irish Independent, revealed that Dublin is the third most traffic congested city in Europe with a driver losing an average of 81 hours annually just in delays caused by traffic. The primary goal of this study is to develop a data-driven classification model which could help in prediction of congestion level, in classes as Low, Medium, and High, based on temporal, spatial, and other parameters. The aim is streamlining traffics, enabling better routes, dynamic signal control, and improving commuter management.

The research focuses on traffic patterns within Dublin, specifically focusing on North City (NCITY) and South City (SCITY) regions of Dublin, to generate insights that reflect realistic traffic conditions while maintaining commute feasible. By applying machine learning methodology , the project seeks answers to questions such as: when does congestion peak? Which part has higher traffic rate? What factors contribute to the generation of traffic? The outcomes are expected to provide actionable insights that can guide better the urban traffic management strategies, including the planning of new orad networks, optimized highway routes, the introduction of bypass roads or tunnels, and the formulation of improved traffic regulations. Furthermore, these findigns can also help in development of other growing metropolitan cities facing similar congestion challenges, supporting smarter infrastructure planning and sustainable development.

II. Data Selection

The data was selected from the links provided in the description of the assessment by the professor. It was combined using two sites, one from the Irish Government's open data portal[2], [3], which lists public sector datasets, and the other one from Smart Dublin [4], a platform which lists datasets related to Dublin's 'smart city' initiative.

Since the goal of the research is to develop models to predict the congestion levels based on various predictors (like location, season, time of the day, whether, the traffic is higher when people leave for work or return, or differences across weekdays). The focus was on getting recent years datasets for the study, but the aim was too use the dataset which covered all twelve months. Since 2025 data is incomplete and the 2024 datasets had missing months, the year 2023 was the safest and the most suitable option.

Originally, twelve month datasets (January to December 2023) were downloaded in CSV format. Each month's dataset had the same columns or features, like, 'End_Time' , 'Site', 'Region', 'Detector', etc. Another one had site descriptions, like 'SiteID', 'Region', 'Lat' (which defines the latitude), 'Long' (which defines the longitude), etc. All these datasets had nearly 1 million records, hence only last 5000 data rows were chosen with a filter to only include regions 'NICTY' (North Dublin) and 'SCITY' (South Dublin). These both datasets were combined as the main goal of research was to get the congestion level understanding based on factors which were based on both, location and time. The merging process was done using a Python script. This part is discussed more deeply in data preprocessing step and data transformation step.

Since the target value (Congestion Level) was not readily available in the dataset, it had to be created. A small component of unsupervised learning was applied to get the congestion categories using a classification model. All the steps are discussed in detail in further sections.

III. Data Preprocessing

As discussed in the previous section, the congestion level was not provided and had to be constructed using the combined dataset. After running the Python script too merge the data, the goal was to generate congestion levels too serve as the target variable. Various techniques were initially considered (such as NLP and other advanced methods), but these were not selected due to dataset size limitations and device constraints. Therefore, a more practical approach was used.

The combined dataset contained traffic volume and time-based fatures. The features selected for clustering were 'Sum_Volume' (total traffic count, indicating traffic intensity), 'Avg_Volume' (normalised per time interval too capture, and 'Hour' (too provide temporal context). Categorical features like 'Region' and 'DayOfWeek' were excluded during clustering too avoid distortion, but they were later analysed too ensure the congestion levels were not biased across different regions or days.

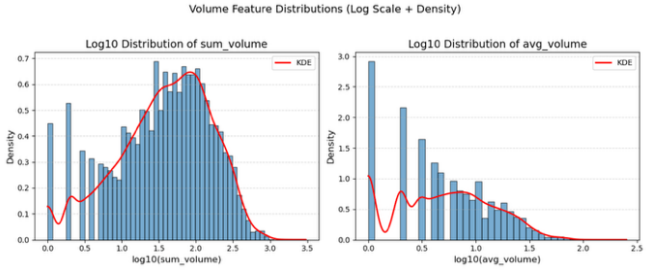


Fig 2.1 Log10 Distribution of Traffic Volume Features

A simple Scikit-learn Pipeline was used for preprocessing before clustering. It applied, SimpleImputer (median) too handle missing values, PowerTransformer (Yeo-Johnson) too reduce skewness, which was confirmed through visual representation as seen below, and also StandardScaler was used to standardise the features so that they contribute equally too distance calculations. This ensured clean, normalised input for clustering.

K-means clustering was chosen because it is easy too understand as a beginner, also, very efficient and works well with grouping of numeric features based on similarity. After clustering, the resulting groups were interpreted and assigned as 'Low', 'Medium', and 'High' congestion levels.

Too determine an appropriate number of congestion groups, K-means clustering was evaluated using various k ranging from 2 too 6. The silhouette score plot showed that while the silhouette score showed high value of k, reaching its peak at k = 6, he resulting clusters became increasingly granular and difficult too interpret in terms of real-world congestion states. Since the goal of this research was too classify traffic into practical categories, therefore, a value of k = 3 was chosen.

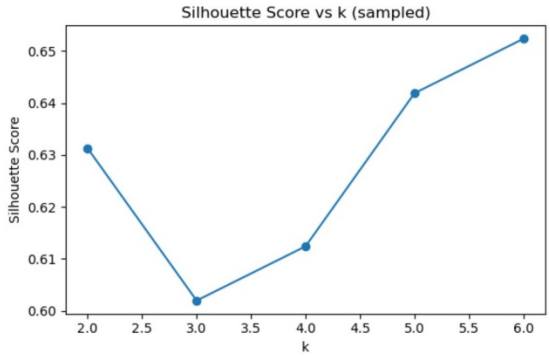


Fig 2.2 Silhouette Score Plot for Cluster Evaluation

After clustering, the resulting groups were evaluated by comparing the mean and distribution of traffic volume values with each cluster too confirm that the cluster meaningfully represented different congestion intensities. Based on the comparison, the clusters were assigned into levels, namely, 'Low', 'Medium', and 'High' congestion levels. This new label was stored in a new column named 'Congestion_Level', and the final dataset was saved as 'congestion_with_labels.csv'.

IV. Data Transformation

Before clustering and model development, the monthly dataset went through transformation steps, too ensure consistency. The first the relevant columns were selected, since the original combined dataset, only had 'End_Time' as the temporal feature, and was in a raw form, many other features were extracted from it, including 'Hour', 'Day', 'Year', 'Minute', etc were extracted. Making the features go from six features to thirteen and this dataset was saved as a file named as 'file_one.csv' which had the temporal features.

Next, the dataset was enhanced by merging site level metadata which was obtained from Smart City platform, and had information like street description, latitude, longitude, providing information about the spatial context was used. This was done as these factors would help congestion to be analysed not only across time but also across physical location within the city. This expanded the number of features from fourteen to nineteen. This new file was saved as 'working_file_with_street_names.csv'.

V. Exploratory Data Analysis (EDA)

In this step, univariate analysis was done to explore the distribution of features individually. Both numerical and categorical features were examined separately using the training dataset only. For numeric features, basic summary table was generated (shown in Jupyter Notebook). Visualisations like histograms and boxplots were plotted to check skewness and identify the presence of outliers. It was observed that the dataset had 14 numeric and 5 categorical columns. The extra numerical columns was generated as a 'Cluster_Id' during the process of generated 'Congestion_Level'. The boxplots showed that there were a lot of outliers, especially in columns like, 'Avg_Volume', 'Sum_Volume', and 'Detector', and many features showed skewed distributions.

An analysis was conducted to check how features behave with the dependant or target variable. Again, the numeric variables were analysed using the summary statistics, histograms, and boxplots to examine the distribution, central tendency, and skewness. For categorical variables, columns with highly imbalanced category distributions were noted for grouping or encoding in later steps. This step made sure that the data quality issues and imbalances were identified early. To check how features relate to the target, a bivariate analysis was conducted. For numeric variables, class comparisons were examined.

Property	File Type	Missing Values	No. of Records	No. of Attributes	Input Feature types	Size
Value	Csv	No	60,025	14	Numerical + categorical	

Table1: Dataset Overview and Characteristics

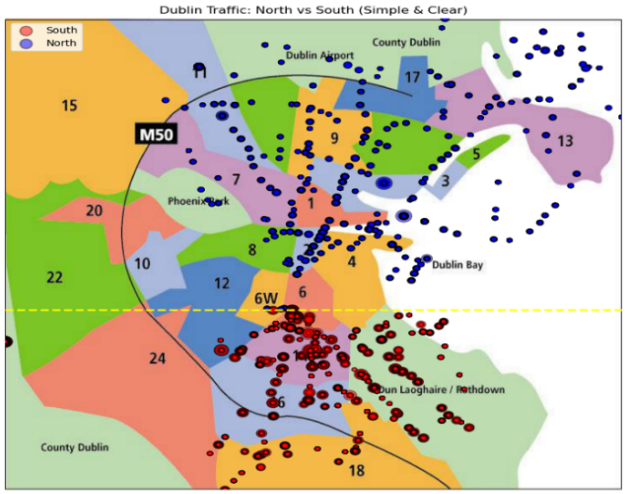
The dataset used in this paper was derived from various large traffic datasets, each containing over 1,000,000 records. Due to incompatibility constraints with device, data reduction techniques were applied during dataset preparation under the guidance of the supervising professor. In the working dataset, major filters were applied, like, focusing only of north and south regions of Dublin, and, the last 5,000 entries for each month's dataset, as discussed previously. This sampling helped to make too ensure that the dataset was computationally manageable while reflecting realistic data patterns.

The dataset was loaded using pandas, and the first step was to standardise the features into a lowercase value for easy usage in future analysis. Then, basic investigation was done for the features, like getting the shape, head values, information about the types of variables. All these are computed in the table below for easy understanding. One thing which was noticed that the expected number of rows for the dataset was 60,001 including the header line, but during the investigation, 60,025 rows were presented, which showed that the extra 24 rows occurred due to overlapping records across monthly files, causing duplicates due to shared timestamps at month boundaries. Then, it was observed that the data types in the dataset were 'object', 'int64', and 'float64'. There were no null values, although some cells in the dataset did contain blanks entries representing true zeros rather than missing data. A summary table of categorical columns was generated to identify the most frequent categories for each feature using

value_counts() for the top 10 occurrences'. This helped to get insights into the distribution and potential data imbalances. It was observed that SCITY which is the south part of Dublin, had more entries than North part, 'NCITY', also, the entries for the days, was more for the day 'Friday', followed by, 'Tuesday', and then, 'Thursday', and the least amount of entries were for the day, 'Sunday'. One more major observation, was that the class imbalance was significant determined, with 'Medium' being the most dominant one, followed by 'High', and then, 'Low' category. A plot was plotted as well which confirmed the distribution of congestion level.

Then, an exploratory data analysis was conducted, the dataset was examined, through quantitative and visual techniques to identify distribution patterns, outliers, correlations, and regional variations. The anlaysis was done using, libraires, such as, Numpy, Matplotlib, Seaborn, Scikit-learn, and Statsmodels. Numerical categories and their distributions were plotted, using Freedman Diagnosis binning rule, which allowed to get accurate assessment of skewness and variability. While plotting the log transformed distribution plots for features like 'Sum_Volume' and 'Avg_Volume' it was observed that both were positively skewed. The Kernel Density Estimation (KDE) curve was applied which higlighted the density of data points, helping in understanding the traffic flow.

Then, Interquartile Range (IQR) method was used to detect outliers, to identify extreme values in numeric features. The output showed that 19.8% values in 'Avg_Volume' and 17.1% of values in 'Sum_Volume' were outliers.These outliers detected weren't some random errors, rather, they represented the extreme traffic conditions like peak time, or



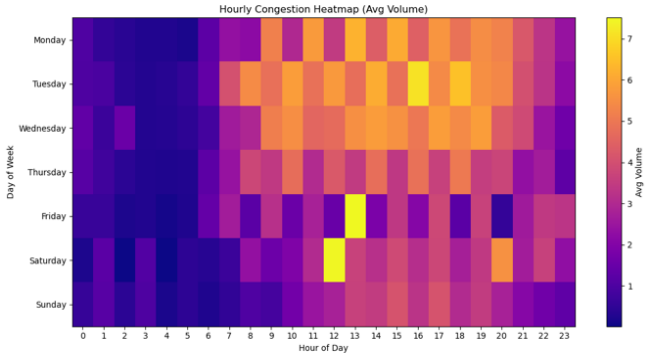
low congestion level. A region-wise analysis was performed using Python (Pandas) to compare traffic volumes across North and South City. The results showed that Noth City exhibited higher mean congestion volumes (41.75 for 'Sum_Volume') compared to South City ('Sum_Volume' being 19.38) indicating higher traffic density in the northern region.

To further get insights of the dataset, various measures were conducted to understand the features, like applying Variance Inflation Factor (VIF) method through Statsmodels, with RobustScaler applied to stabilize class imbalance which is used to detect the multicollinearity, ANOVA tests were performed to determine whether the numeric features differ significantly across congestion levels. These tests were performed as a part of exploratory data analysis to assess feature relationships and prevent data leakage in future modelling and also to prevent overfitting as without these measures the data model will perform perfectly with 100% accuracy which will not be good. The results showed very high VIF values for 'sum_volume', 'avg_volume', 'month', and 'iso_week', indicating strong linear dependencies among these features. To reason these findings, a Pearson correlation hetamap was generated to visually determine the correlation between these features, and this confirmed that 'sum_volume' and 'avg_volume' (r = 1.00) along with 'month' and 'iso_week' (r = 0.99) were highly correlated. This step helped to identify redundant predictors which could impact the results of the model to an overfitting or a biased behaviour. Then, the one-way ANOVA test (for numerical features) and Chi-sqaure tests (for categorical features) were performed using SciPy library. Both these analysis produced p-value < 0.05, confirming that these features have a significant relationship with the target value i.e., the 'congestion_level'.

The results of these exploratory data analysis provided a clear udnerstanding of the dataset's structure and statistical relationships, ensuring data is well structured before proceeding to data modelling step. The further sections discusses about the visualisations, model development, feature selection, evaluation and interpretation.

VI. Data Visualisation

The visualisation step was performed separately to get an understanding of the structure of the data, and get any information which might give some insights about the dataset. Visualising the data set prior to modelling, allowed for better



understanding of the relationships of variables such as the time of day, day of week, location and traffic volume, to the target, congestion_level.

Fig 6.1 Hourly Congestion Heatmap

The very first plot created was too get an insight of time and the congestion level out of curiosity and what is better than a heatmap to predict it. The plot shows how the average traffic volume varies at different times of the day (and days of the week) in Dublin. The dark shades of purple represent a free congestion level, i.e. low congestion level whereas the yellow spots are the high congestion level, the pink too orange ones are the medium. As seen from the plot, Monday to Thursday the traffic congestion seems to be a lot especially during the office hours, from 9 am too 5pm. Whereas, Friday to Sunday is easy, with maximum, traffic during the afternoon time, like 12pm on Saturdays and 1pm on Fridays.

Fig 6.2 Geospatial Scatter plot

The next plot was a Geospatial Scatter Plot, for this, a background image of Dublin's map was used and the scatter plots were plotted using the latitude and longitude values from the dataset to get the site. Each point on the map represented a traffic site location, with blue points being in the north part of the city and red ones in the south part. This visualisation helped in identifying which regions are mostly congested more, and it was observed that the north showed a wider spatial spread while the south was densely clustered. Most of the areas of the city centre are usually the hot spots for high traffic level as seen through the plot. This plot indicates that while traffic sites in the north capture more regional and highway flow, the southern points represent the urban traffic, where congestion is more intense due to the population density and shorter commuting routes.

But, the only limitation for this plot was that the plotted points were restricted by the accuracy of the background image. Since the base map was a static image and not a true representation of map of Dublin. Therefore the latitude and longitude points plotted might have not aligned truly too the map. Hence, this plot might be a good interpretation of approximation of the pots but should not be interpreted as an exact representation.

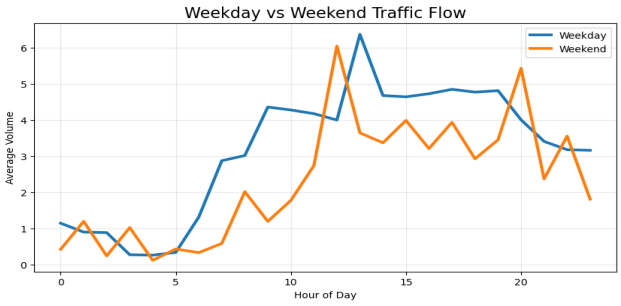


Fig 6.3 Weekday vs Weekend Traffic Flow Line Graph

The line graph illustrates the average hour traffic levels that occur on weekdays (blue) and on weekends (orange) in Dublin. Weekday traffic has clear peaks in both the early morning (7-9 AM) and evening (4-6 PM), which show the metered commuter traffic to and from the city centre. Traffic on weekends remains low in the early hours and peaks later in the day, near midday to early afternoon, which is indicative of hit and miss shopping and leisure activity. This difference highlights the true commuter based nature of weekday jams, especially on the major routing lines into the city as Drumcondra Road, N11 and M50, while there is a more evenly spaced distribution of traffic on weekends. Overall, the visuals shows that the time of the day and the type of day (weekday/weekend) is an important characteristic for accurately modelling Dublin's traffic correlates.

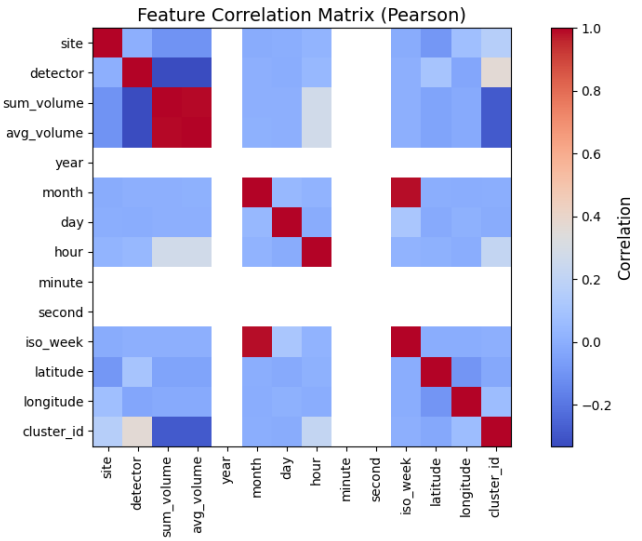


Fig 6.4 Pearson Feature Correlation Matrix

The Correlation matrix in figure above indicates the degree of linear association between the quantitative features of the data. There is extremely high correlation (r=1.0) between 'sum_volume' and 'avg_volume', indicating that both features do capture the same pattern of traffic flow. Similarly month and iso_week show a very high correlation coefficient (r=0.99), indicating the natural reliance of these measures of our temporal variable x. A moderately high correlation coefficient between day, hour and traffic volume confirms that congestion is a function of both the time of day and the relevant calendar cycle patterns which are consistent with the habitual commuting behaviour of the citizens of Dublin. There was no statistically significant degree of multicollinearity evident between the other relevant features (latitude, longitude, hour), clearly indicating that each produced a different type of relevant information for the model. This insight gave rise to the decision to dispense with redundant measures (such as avg_volume, iso_week etc) altogether during the training of the model, leading to a more robust and interpretable machine learning pipeline.

All the visualisations created for this section has its own limitations since the dataset was created after the application of a lot of filers, which included having last 5,000 records per month of north and south parts of Dublin. Also, the visualisations aggregates across months, which doesn't clearly tell information about how these traffic levels might change with different seasons or weather conditions as they are always subjective.

Together, these visualisations not only determined which features were pertinent to the model, but also offered real-world implications for congestion management. They suggest that city planners can introduce time-based initiatives like traffic light timing adjustments or altering routes within peak weekday hours. Although some of the graphs were limited by plotting static map images and aggregated monthly data, they nonetheless gave a clear suggestion of how city structure and human activity affect traffic. Overall, these findings confirm that time is the most influential parameter for Dublin's congestion, with location data adding to the accuracy of forecasts.

VII. Model Creation

After the completion of exploratory data analysis and identifying multicollinear and redundant features, the modelling phase was initiated. The very first step was to make sure that the EDA dataframe was copied to create a clean modelling dataframe and this was done using Python's pandas library. As a best practice, all the duplicate records were removed to maintain data integrity and prevent bias in model learning. Then, all the redundant and low-variance features were dropped so that the model does not show an overfitting behavior and give the best accuracy rate. These dropped columns were, namely, 'region_pretty', 'end_time', 'date', 'description', 'avg_volume' and 'iso_week'. This decision was made due to columns like 'avg_volume' and 'iso_week' showing a highly correlated value with other features like 'sum_volume' and 'month', while 'region_pretty' was created just to easily understand the abbreviations, 'NCITY' and 'SCITY' which later turned out to be pointless and turned out as a duplicate for 'region'. The removal of 'end_time' and 'date' prevented redundancy with already engineered temporal variables like 'hour,' 'day', and 'month'. Lastly, the 'description' column was excluded as it had too many unique text entries, which could complicate encoding and increase the model complexity without improving predictive performance.

To further refine the dataset, all the constant value fields like 'minute', 'second, and 'year' were excluded. Additionally, identifiers such as 'site', 'detector', and 'cluster_id' were reclassified as categorical rather than numeric variables. A data leakage preventions step was conducted to make sure that the model did not indirectly access the information about the target variable ('congestion level'). Features, such as, 'sum_volume', 'avg_volume', 'site', 'detector', and 'cluster_id' were removed since they could indirectly act as proxies for the target and could cause the model to 'memorize' rather than learn, causing overfitting.

After all the filtering and data cleaning steps, the set of features which became the predictors of the model were, 'month', 'day', 'hour', 'latitude', 'longitude', 'region', and 'day_of_week'. These variables were considered highly relevant to the research objective as the study aimed to predict traffic congestion levels on both temporal and spatial factors. Finally, a preprocessing pipeline was build using Scikit-learn to prepare the cleaned dataset for training. All the numeric features were scaled using RobustScaler, which applied median and interquartile range scaling to minimize the impact of outliers. While the categorical features were encoded using OneHotEncoder.

The entire preprocessing workflow was implemented using a ColumnTransformer within a Scikit-learn Pipeline as it provides a structured way to perform preprocessing and model training together. It makes sure that the exact same preprocessing steps are applied throughout the process on both train and test sets, helping to avoid mistakes and guarantees a reproducible and a stable model performance.

Before building the baseline model[5], the dataset was divided into feature set (X) and the target variable (y). The target variable was encoded into numerical labels using LabelEncoder, ensuring that the model could interpret the congestion levels as distinct classes without imposing ordinal bias. All these transformations were confirmed by applying the preprocessing pipeline and running the smoke test which confirmed that these data transformation was consistent. This preprocessing increased the number of features from seven to fourteen.

A 10-Fold Stratified Cross-Validation method (used using cross_val_score) was used to split the data, as it made sure that model's performance is fair and reliable. Instead of training the model using train/test split method which splits the dataset once and provide biased results which can be lucky or unlucky, the dataset was split into ten parts, where the model trains on nine parts, and tests on the remaining parts. Stratification was used as the dataset was imbalance, and it was done using StratifiedKFold[6]. Cross-validation averages performances across 10 different train/test splits, giving a more reliable score. Using 10 folds offered the right balance between bias and variance, fewer folds could lead to unstable results, and it is best practice to choose ten when dealing with a moderate size dataset.

A baseline classifier was trained using Scikit-learn, and the chosen baseline model was Logistic Regression[7] instead of Naïve Bayes which is usually the most obvious choice for a classification model as the Logistic Regression provides a solid starting point for multiclass problems and operates on the principle of estimating the probability that an input belongs to a particular class using sigmoid function. The model assumes a linear relationship between independent variables and works by combining the input features in a linear way and then getting the result between 0 and 1. Unlike Naïve bayes, which assumes all the features are independent of each other, logistic regression can handle if feature are not entirely independent of each other, which was need in this study as the dataset showed correlation between variables during analysis. In this study, the model used the LBFGS solver, which is fast and stable optimization method especially for multi class problems. The maximum number of iterations (max_iter = 1000) was increased to make the model fully converged due to the size and complexity of the data. All the results and further implications are discussed later in valuation section.

This baseline model achieved a Macro F1-score of 0.60 across the 10-fold cross-validation (using cross_val_score), with a negligible standard deviation, showing stable performance across the dataset. This allowed to get a reliable classification report, and a confusion matrix to be produced. Even though the model showed an accuracy of 0.70, accuracy wasn't chosen to evaluate the model as it is not the right metric in this case because classes are imbalance. For evaluation, Macro F1 was preferred because it treats all three classes equally and does not let any class dominate the results. It measures both precision and recall, whereas accuracy can be misleading if some classes appears more frequently, leading to biased results. The report showed that even though the model achieved a reasonable overall accuracy, the performance was highly uneven across classes. The 'Medium' class was predicted well, the 'Low' class moderately, and the 'High' class performed poorly, with very low score of recall and F1-score, indicating frequent misclassification. This showed the need to develop a better model which can distinguish classes well.

After observing the results of the baseline model with Logistic Regression, a more advanced model, called as, XGBoost (Extreme Gradient Boosting)[8] was used. It is a tree absed model that can learn complex datasets and has a

build in regularization to prevent overfitting. the main intention behind using this algorithm was that it is highly efficient at capturing complex and non-linear patterns in the dataset, which is important for a topic like traffic congestions where conditions change based on both time and location. Unlike simple models, it can handle correlated features and changing patterns over time and location, while also making sure that we don't overfit the model, by regularization. These factors made it stable and accurate choice for predicting real-world traffic levels.

After running the code for XGBoost model, the result improved significantly, the model achieved a Macro F1-score of approximately 0.80 with a very low standard deviation, indicating a very stable model regardless of how the data is split. Like the baseline model, a pipeline was created and after running the classifier using the Stratified 10-Fold Cross-Validation, a report was generated which showed an improvement across all evaluation metrics compared to the baseline model. The 'High' class, which previously had a score of 0.22, improved significantly to 0.65, which meant that the model was able to predict High class more effectively. The 'Low' and 'Medium' class also improved, with F1- scores increasing from 0.74 to 0.86 in case of Low class, and 0.85 to 0.89 in case of 'Medium' class. The Macro F1-score improved by 20%, as the result went from 0.60 to 0.799, confirming that the performance is more balanced across the classes. These improvements showed that XGBoost is better at handling class imbalance more effectively.

Few plots were plotted as well for evaluating model's performance, which includes Q-Q plots for predicted probabilities, ROC curve, and Calibration curves for each class to check and visualize model's performance, these were all plotted in Jupyter Notebooks, but one plot which gave the best insights is below:

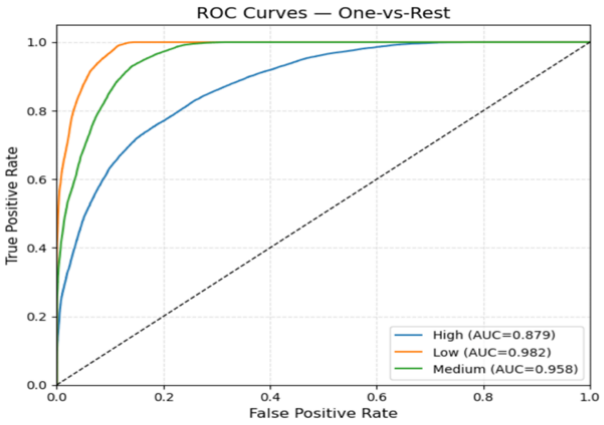


Fig 7.1 ROC Curves- One vs Rest

The plot shows that all the three curves are above the diagonal, which means the model performs significantly better than just random guessing. The 'Low' congestion has the highest separability, with AUC value of 0.982 meaning, the model can distinguish 'Low' class extremely well, followed by 'Medium', with AUC value, 0.958, and then, the lowest AUC score of 0.879, for 'High' class which is still a strong score, but model find it the hardest to predict.

The next method which was used for model making models Random Forest[9] as it is an ensemble learning method that combines many decision trees to generate more reliable and accurate predictions. Random Forest can be easily applied for regression as well as classification problems, particularly when data have complex, non-linear relationships among them. It copes well with high dimensional data, noise, and class imbalance, making it an ideal algorithm for problems like traffic congestion prediction where variables of time and location interact arbitrarily. It uses stratified version to make sure that consider the class imbalance and it was observed that the result change significantly compared to the baseline model and went from 0.60 to 0.79 in terms of macro F1 score and for 'High' the result went from 0.22 to 0.63 but it was still less than the extreme XGBoost and a similar pattern was observed for other class 'Low', while for 'Medium' class the result was same. The next step involved was to get prominent features and get the best outcome after doing feature selection.

In summary, it may be said that both methods based upon ensemble methodologies (Random Forest and XGBoost) performed better than the baseline by a considerable margin, but that XGBoost delivered the most uniform and generalisable results and hence was the model which was used for predicting the traffic congestion levels in the present study. Overall performance of the XGBoost model, was the most accurately calibrated and balanced, therefore demonstrating its powerful feasibility in the modelling of the successful complexity of transport traffic activity present in Dublin city

The outputs of the models which were run have been described below for easy interpretation:

Model	Accuracy	Macro F1-score	High (F1)	Low (F1)	Medium (F1)
Logistic Regression	0.70	0.60	0.22	0.74	0.85
XGBoost	0.82	0.80	0.65	0.86	0.89
Random Forest	0.81	0.79	0.63	0.85	0.89

Table2: Comparative Performance of Classification Models

Among all the different models ssesed XGBoost, performed better than both Logistic Regression and Random Forest Classifier, obtaining the highest Macro F1-score (0.80) with low variance across folds. Random Forest achieved an accuracy of 0.81 and a macro F1-score of 0.79, firmly reflecting on the baseline Logistic Regression. Performance was strong at entry level of congestion, with large gains for the High congestion class (F1=0.63), indicating that the model effectively captured non-linear temporal and spatial relationships within the traffic data. The Low (F1=0.85) and medium (F1=0.89) classes also predicted strong accuracy, showing balanced performance and sound generalization, the results above show the ability of Random Forest to detect complex interactions between time, place, and congestion patterns in the dataset. As foreseen, Logistic Regression performed as an adequate linear baseline but struggled to accurately learn relationships between temporal and spatial variables that were non-linear properties. Nevertheless, the limitation of XGBoost was evident in that this model was limited in its interpretability and complexity of implementation in real time situations in traffic systems. Nevertheless, the significant recall value on the High Congestion class makes application of XGBoost Model particularly suitable for applications where urban traffic prediction is required, due to convergence of concern and reasons being more towards opportunity of adequately identifying peaks of congestion with accuracy as opposed to minimum false positives. All results indicate in summation that congestion in Dublin is simply a combination of both

temporal and spatial issues and ensemble methods such as XGBoost can effectively classify and therefore models these multi-dimensional parameter interdependencies.

The results of this study present several important implications. From a practical standpoint, the produced model can be incorporated into Dublin smart city infrastructure to produce real time transformation forecasts of congestion to facilitate dynamic signal, control and introduce rules to improve traffic management for a commuter. It can be interpreted from the research that the traffic majorly is affected by temporal and spatial factors like day of the week and the time of the day. In addition, these studies do demonstrate that advanced machine learning methods such as XGBoost are superior to classic modeling methods for handling imbalance and nonlinear open data while providing a methodology. The major limitation for this study is that the datasets combined were limited to two areas and selected records from them were the last 5000 entries. But, apart from the limitations, this research does form a strong foundation for building a scalable and easily extendable data-driven scheme for transport prediction to encapsulate sustainable oven mobility.

VIII. Feature Selection

Feature selection is the process of identifying and retaining the most relevant features in the dataset which helps to improve model performance. While VIF-based filtering in EDA addressed redundancy by removing multicollinear variables, it did not evaluate predictive usefulness, therefore, actual feature selection using proper techniques, was performed, to determine the features that truly contribute to model performance. Before applying any of the techniques, the input dataset was transformed again (defined again for clarity purpose), this step, expanded the features from 7 to 14 because of the encoding part through scaling and one-hot encoding as well. The applied techniques, included filter based methods, as well as, ensemble methods, the former relies o statistical measures to determine how strongly each feature is associated with the target. On the other hand, the wrapper method tests different combinations of features by actually training a model and checking how well it performs. It wraps the feature selection process around the model itself.

As observed from the Model Creation section, the XGBoost model already had strong and stable performance even in the absence of feature selection. After feature selection with both Mutual Information and SelectFromModel, the results were still largely identical, with very minimal differences in Macro F1-score. It confirmed that the initial preprocessing and feature engineering steps had already captured the most significant predictors of congestion and the removed features carried very minimal extra predictive information to contribute. The process thereby enhanced interpretability with no loss in model accuracy.

The first technique which was applied was Mutual Information (MI), using SelectKBest, which is a filter base feature selection method which measures the extend of importance of each feature to the target variable, congestion level, in this case. This technique was chosen as it can easily work for both numeric and categorical features, is reliable and fast, also, it works well even when the relationships are non-linear. A higher MI score means that features is important and helps in predicting the class, while a low MI score means the feature contributes very little to predict the target variable. After applying this, the features, were reduced from 14 encoded features to just 5 as seen in the visualisation below, mostly the features which contribute are numeric, and these 5 were the ones, which were selected as there contribution was significant, and all those, below 10% were dropped as those features didn't add any value to the model as such. The new predictors, were 'month', 'day', 'hour', 'latitude' and 'longitude'. The 'num' in front of these features is just to segregate between a numeric variable and categorical variable. All the other features like 'region' and 'day_of_week' were dropped as their individual contribution was less than 10% which is not very significant. The maximum is from time factors, like month, day, hour while locational factors are mdoerately important.

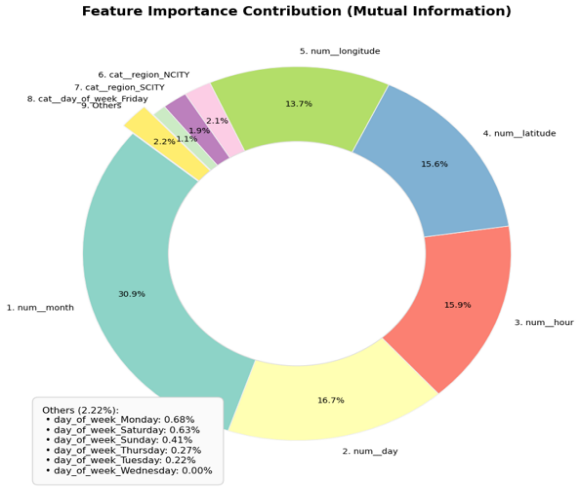


Fig 7.3 Feature Importance Contribution Chart[10] based

Feature selection was carried out Mutual Information (MI) and SelectFromModel (SFM) to identify the most important features to predict traffic congestion. General model performance was almost the same XGBoost was still at Macro F1-score 0.80, and Random Forest improved slightly from 0.79 to 0.80. While accuracy was not increased significantly, feature selection have simplified the model, made it faster, and easier to interpret by removing less informative features like weak categorical variables. Both methods have their flaws and merits. Mutual Information is quick and effective at picking up non-linear relationships between single features and the target, but only looks at features one at a time and might fail to capture how features interact with each other. SelecrFromModel, on the other hand, uses XGBoost to pick features that have an impact on prediction and obtains complex interactions, but might overestimate continuous features and depends a lot on the underlying model. Overall, feature selection simplified the model without harming its performance and made it more efficient.

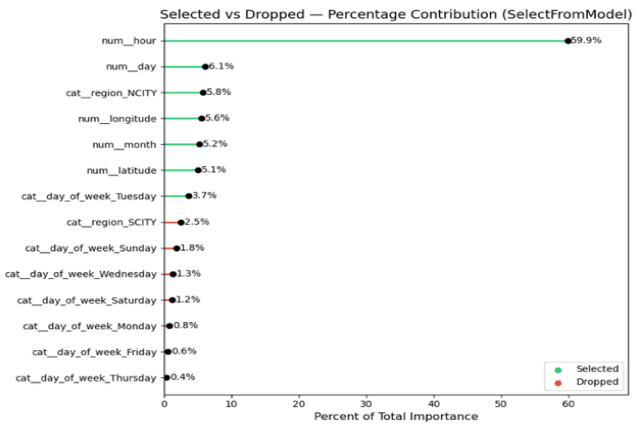


Fig 7.4 Feature Importance for SelectFromModel

After applying Mutual Information feature selection, the XGBoost model retained only the new five features, interestingly, the performance improved marginally, indicating that removing weak feature helps to reduce noise. meaning that the removed features contributed minimal predictive value. The XGBoost model performance was uniform, signifying that features removes contributed nothing to minimal predictive ability.

Following the feature selection, hyperparameter tuning [11] was performed using RandomSearchCV with a 10-fold stratified cross-valdiation keeping outleirs in mind. The search explored combinations of learning rate, maximum tree depth, subsampling ratios, and the number of estimators too find the best performing configuration of the XGBoost classifier. The tuning objective was to maximise the macro F1-score, chosen due to class imbalance. The outcome achieved by tuned XGBoost model was 0.80 which was similar to the one after feature selection, so nothing changed. The classifier achieved te highest accuracy and generalization compared too other baseline method like random Forest.

IX. Result

The research successfully developed a machine learing based congestion classification model using the traffic data. The combined preprocessing, feature selection, and XGBoost classification pipeline showed that it is possible to predict congestion levels with high reliability, but these are not just without any limitations, as the dataset was based on a lot of filters and traffic is quite unpredictable, depending on various factors, like construction, accident, bank holiday, etc.

One interesting conclusion has been those temporal features, particularly, ‘Hour’, and ‘Day’ have a lot of importance in determining the congestion, while the spatial indicators act as secondary modifiers. The outcome suggested that traffic congestion in Dublin is primality time driven.

References

[1] “Dublin is third most traffic-choked city in Europe, new report finds,” Irish Independent. Accessed: Oct. 28, 2025. [Online]. Available: https://www.independent.ie/irish-news/dublin-is-third-most-traffic-choked-city-in-europe-new-report-finds/a1894781976.html

[2] “Traffic Volumes from SCATS Traffic Management System Jan-Jun 2023 DCC - data.gov.ie.” Accessed: Oct. 28, 2025. [Online]. Available: https://data.gov.ie/dataset/dcc-scats-detector-volume-jan-jun-2023

[3] “Traffic Volumes from SCATS Traffic Management System Jul-Dec 2023 DCC - data.gov.ie.” Accessed: Oct. 28, 2025. [Online]. Available: https://data.gov.ie/dataset/dcc-scats-detector-volume-jul-dec-2023

[4] “Traffic signals and SCATS sites locations DCC - data.smartdublin.ie.” Accessed: Oct. 28, 2025. [Online]. Available: https://data.smartdublin.ie/dataset/traffic-signals-and-scats-sites-locations-dcc

[5] “What is Baseline Models,” Iguazio. Accessed: Oct. 28, 2025. [Online]. Available: https://www.iguazio.com/glossary/baseline-models/

[6] “StratifiedKFold,” scikit-learn. Accessed: Oct. 28, 2025. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

[7] “LogisticRegression,” scikit-learn. Accessed: Oct. 28, 2025. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[8] “XGBoost Documentation — xgboost 3.1.1 documentation.” Accessed: Oct. 28, 2025. [Online]. Available: https://xgboost.readthedocs.io/en/stable/

[9] “RandomForestClassifier,” scikit-learn. Accessed: Oct. 28, 2025. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[10] “Donut Chart using Matplotlib in Python,” GeeksforGeeks. Accessed: Oct. 28, 2025. [Online]. Available: https://www.geeksforgeeks.org/python/donut-chart-using-matplotlib-in-python/

[11] “RandomizedSearchCV,” scikit-learn. Accessed: Oct. 28, 2025. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html