**Traffic Prediction**

**BY:-Nupriya Saxena(STB03)**

**MENTOR:-UROOJ KHAN**

# I. Introduction

**Problem Statement**:- Traffic prediction means forecasting the volume and density of traffic flow, usually for the purpose of managing vehicle movement, reducing congestion, and generating the optimal (least time- or energy-consuming) route. The task of detecting traffic for the next day, week etc.

**Dataset**:- The dataset contains 4 components (DateTime, Junction, Vehicles, id) across 48120 rows. The DateTime column shows the timestamp along with the respective date.

* DateTime - It shows the date and timestamp
* Junction - It tells the junction number for which vehicles are present
* Vehicles - It tells the number of vehicles
* id - It tells the id number of a timestamp

Dataset Link - [Kaggle Dataset](https://www.kaggle.com/datasets/fedesoriano/traffic-prediction-dataset)

# II. Methodology

**Overview:-** We have implemented the following algorithms in our code: -

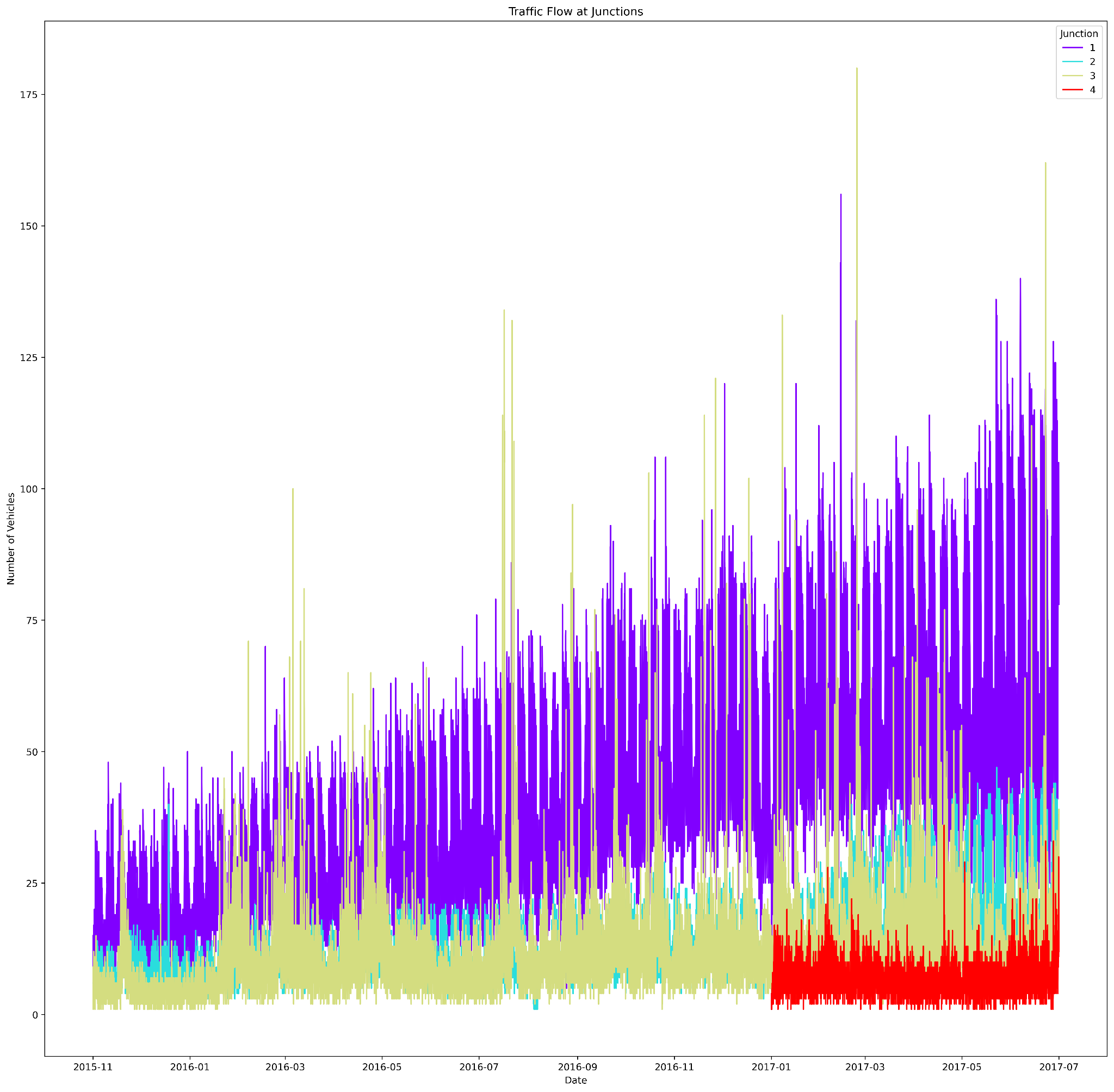
* *Decision Tree Regressor*
* *Linear Regression*
* *Random Forest Regression*
* *XGB Regression*
* *Grid Search CV*

**Preprocessing:-** It is important to clean the data before applying any model. The preprocessing steps involved are:-

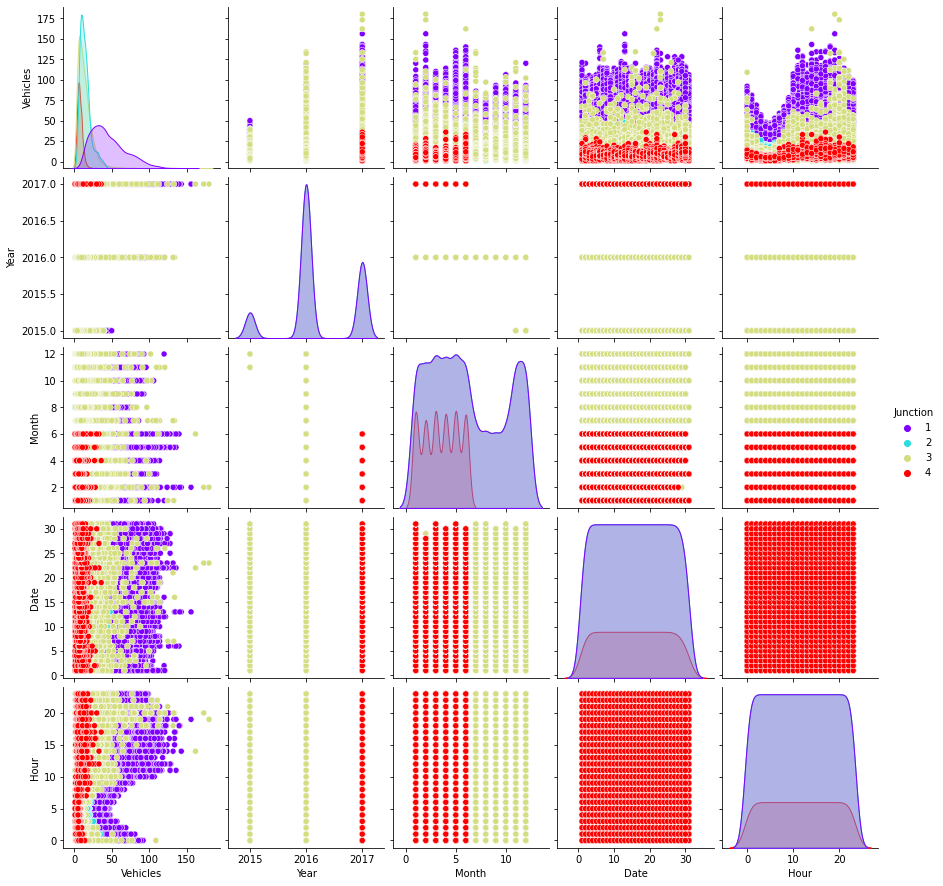
* *Dropping unnecessary columns ( such as id)*
* *Conversion of date and time to DateTime format*
* *Adding more columns for the day of the week, date, month, year and hour*
* *Encoding the day of the week data column*
* *Creating different data frames for different junctions*
* *Splitting the data into train and test for different junctions and adding them to a list*

**Visualization:-** It is important to understand different aspects of every feature at a minute level before applying any model. The types of visualizations applied are:-

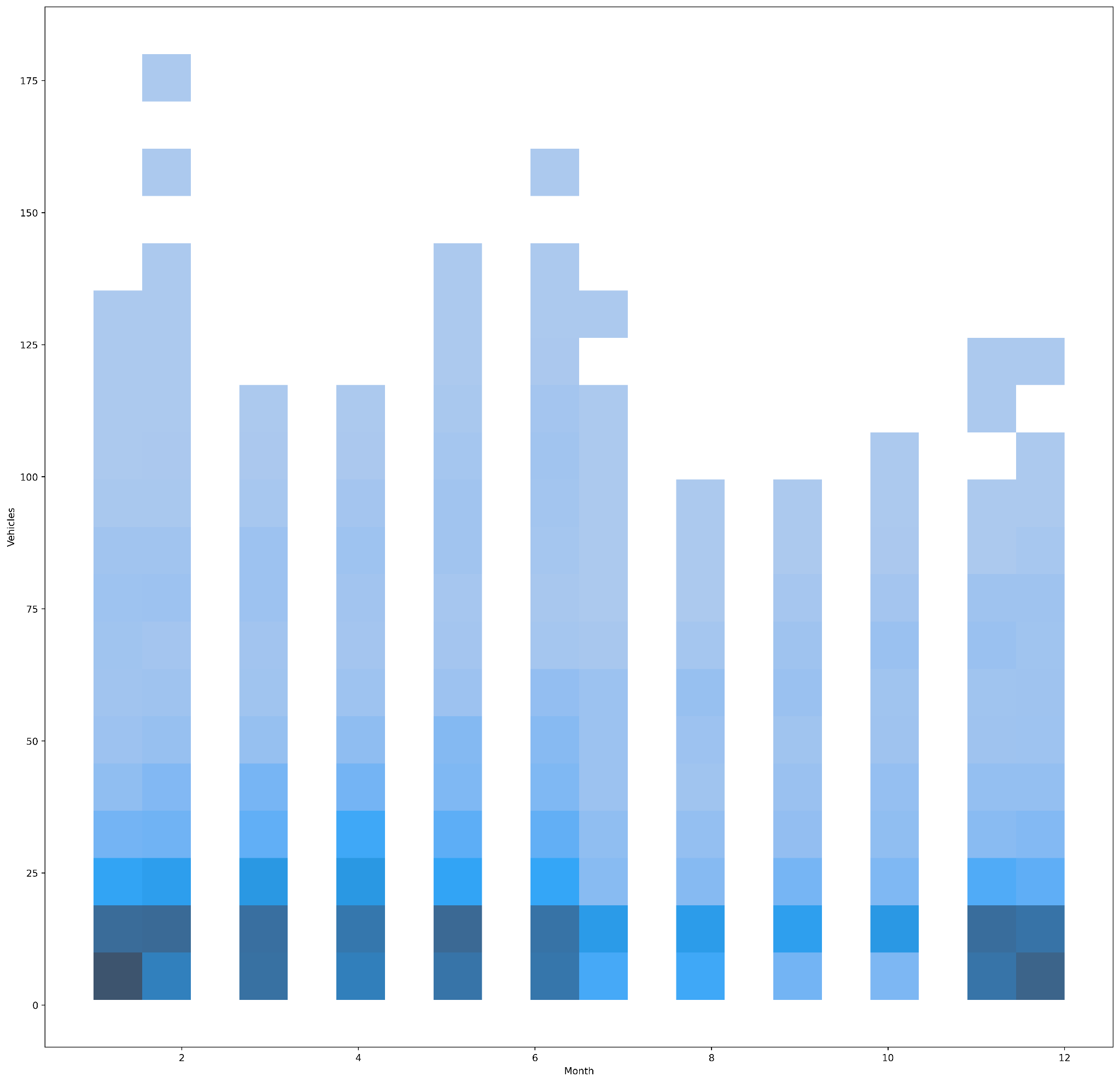
* *Traffic flow across different junctions across time-period*



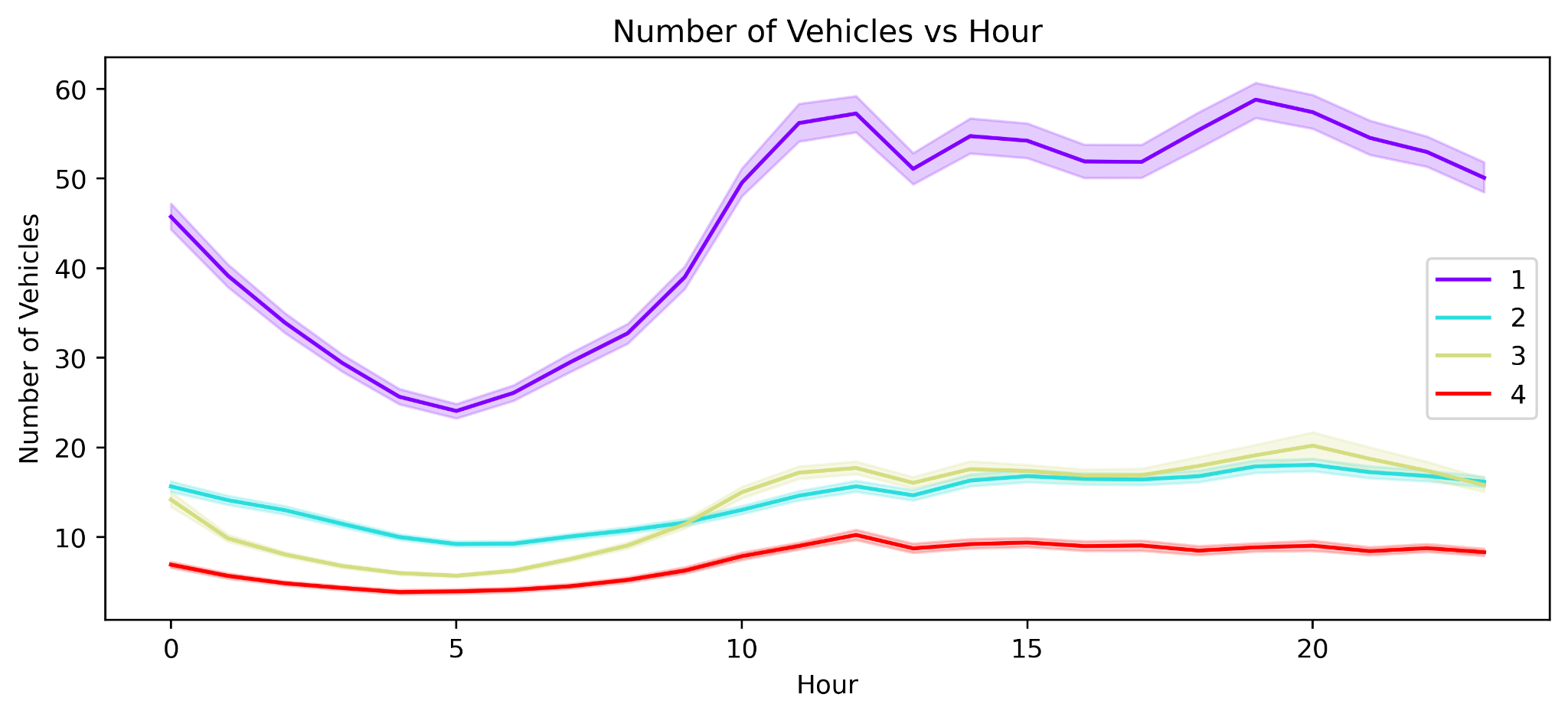
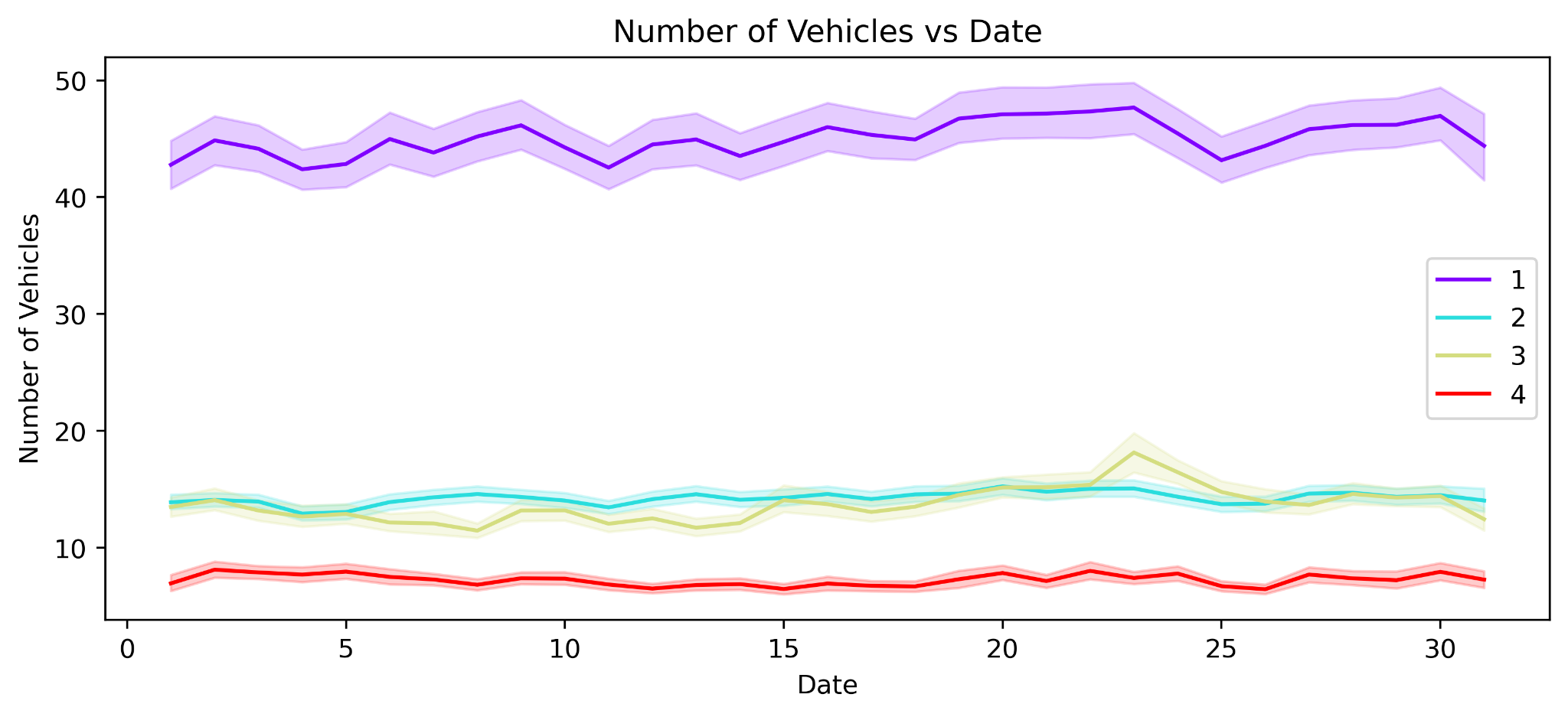
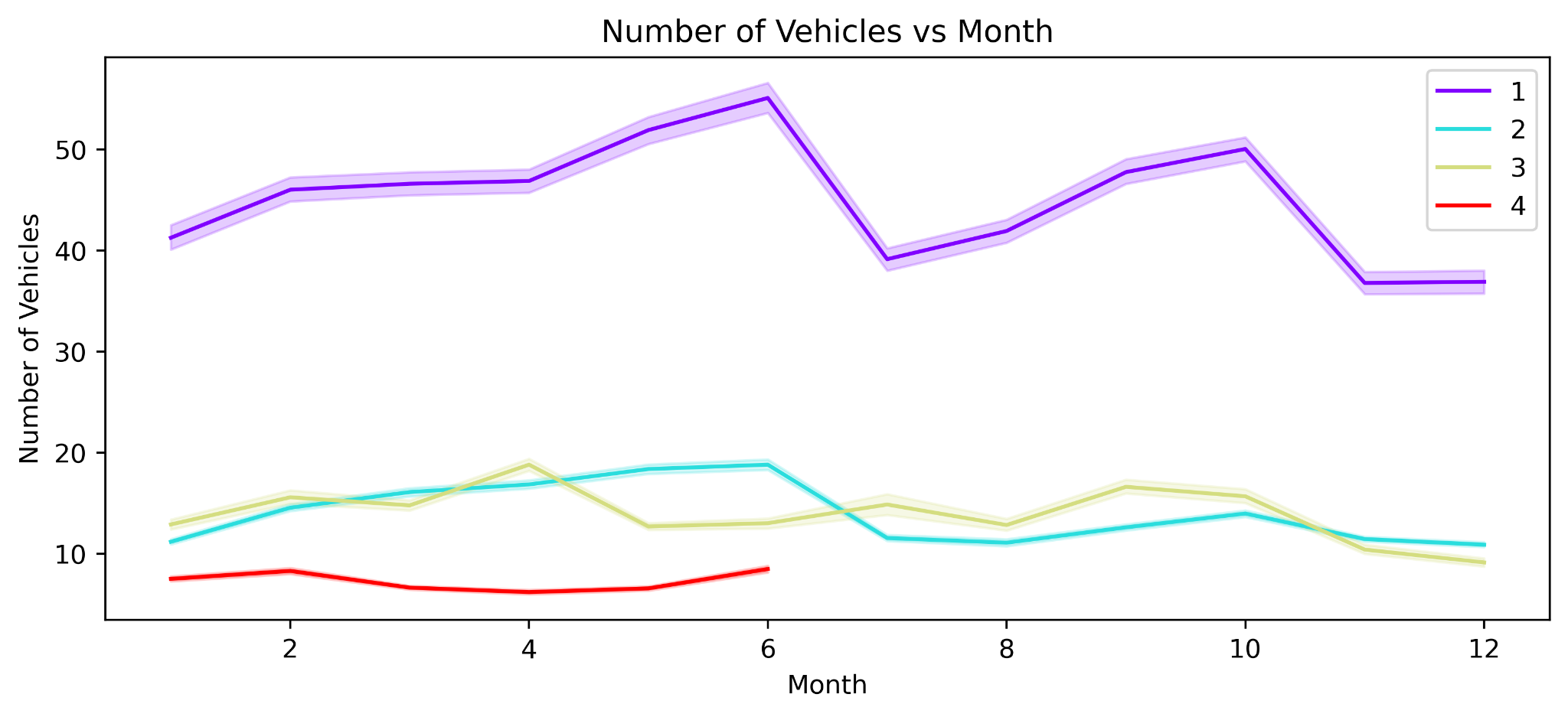
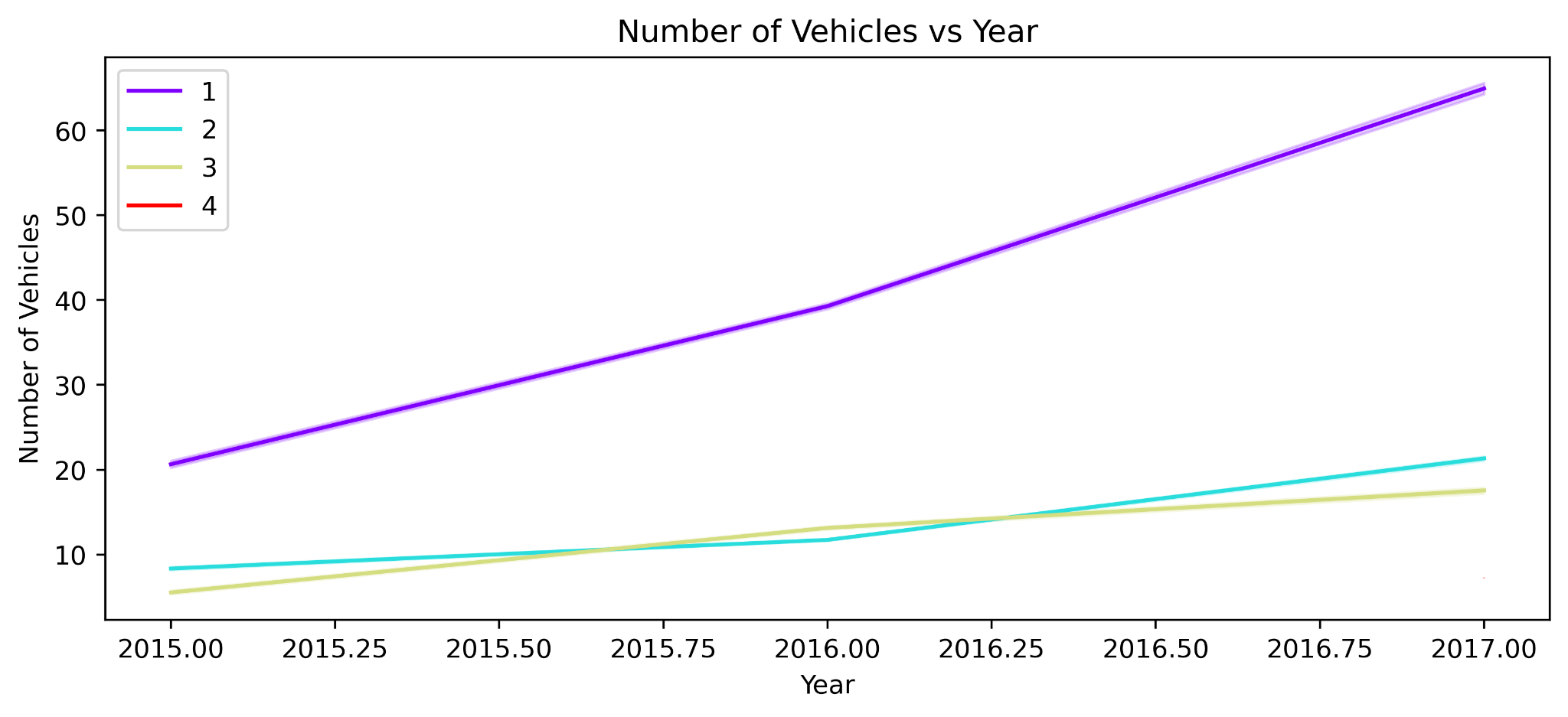
*Pairplot of different features*



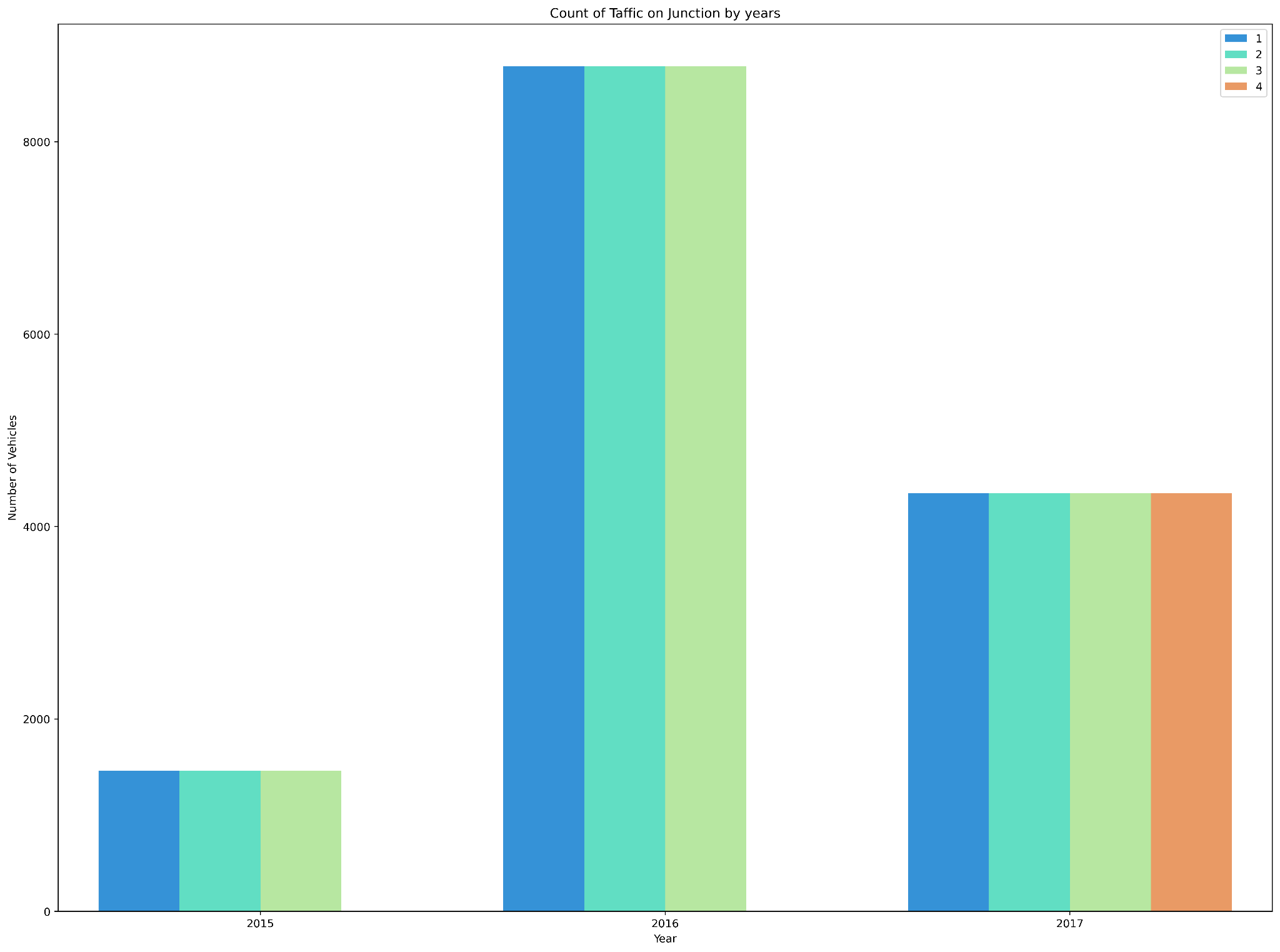
* *Histogram of month-wise traffic flow*

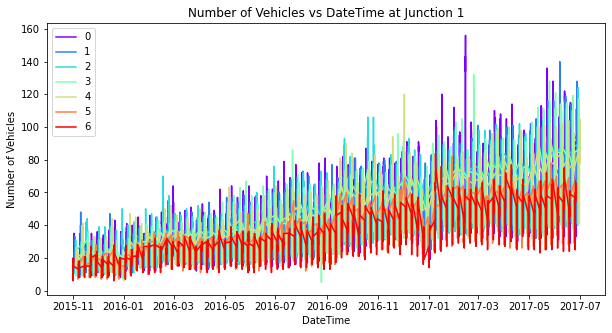


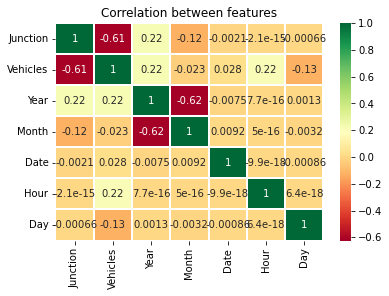
*Lineplots for number of vehicles across the year, month, day and hour across different junctions.*



*Countplot of traffic across different years*



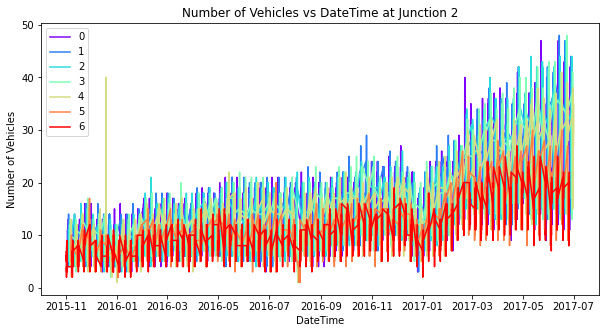
* *Correlation Heatmap for different features*



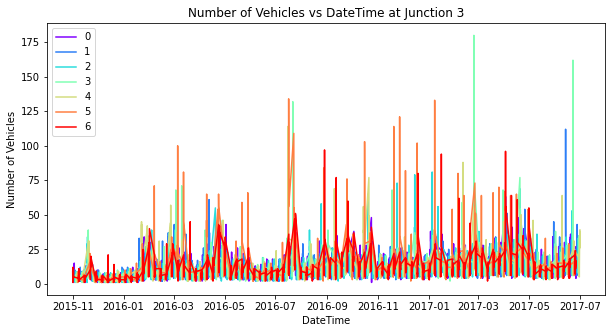
* *Plotting the number of vehicles across every day of the week for each junction.*

○ Junction 1

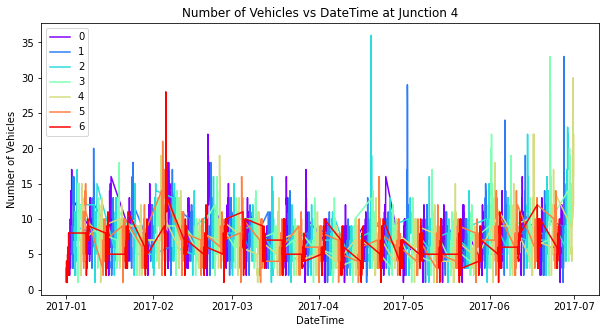
○ Junction 2



○ Junction 3



○ Junction 4



## Implementation:-

For Feature Selection, We see that our data has only date-time as a feature for every junction, which we then separate into 5 features namely, Year, Month, Date, Hour and Day. Since there were no redundant features, we do not employ a Feature Selection Algorithm.

Now, The Models used:

* ***Decision Tree Regressor:*** *Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.*
* ***Linear Regression:*** *Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output)*
* ***Random Forest Regression:*** *A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.*
* ***XGB Regression:*** *Extreme Gradient Boosting, or XGBoost for short, is an efficient ensemble learning algorithm via gradient boosting. It provides a parallel implementation of decision trees that are created in a sequential form, where weights play an important role.*

*For tuning the hyperparameters, We have used* ***Grid Search CV****, which uses a different combination of all the specified hyperparameters and their values and calculates the performance for each combination and selects the best value for the hyperparameters.*

# III. Evaluation of Models

We have applied the above-mentioned algorithms for different junctions individually and recorded the test, train and RMSE scores.

* *For Junction 1:-*

|  |  |  |  |
| --- | --- | --- | --- |
| Name of the Model Applied | Train Score | Test Score | RMSE  Score |
| Decision Tree Regressor | 0.84795 | 0.83595 | 0.06118 |
| Linear Regressor | 0.63901 | 0.63250 | 0.09157 |
| Random Forest Regressor | 0.84706 | 0.84275 | 0.05990 |
| Grid Search CV (Random Forest Regressor) | 0.99545 | 0.96399 | 0.02769 |
| XGB Regressor | 0.97506 | 0.96968 | 0.02630 |
| Grid Search CV (XG Boost) | 0.99829 | 0.97411 | 0.02430 |

* *For Junction 2:-*

|  |  |  |  |
| --- | --- | --- | --- |
| Name of the Model Applied | Train Score | Test Score | RMSE  Score |
| Decision Tree Regressor | 0.80671 | 0.78502 | 0.07196 |
| Linear Regressor | 0.54524 | 0.51561 | 0.10801 |
| Random Forest Regressor | 0.82137 | 0.80292 | 0.06889 |
| Grid Search CV (Random Forest Regressor) | 0.97937 | 0.90306 | 0.04832 |
| XGB Regressor | 0.93635 | 0.90692 | 0.04734 |
| Grid Search CV (XG Boost) | 0.93635 | 0.90692 | 0.04734 |

* *For Junction 3:-*

|  |  |  |  |
| --- | --- | --- | --- |
| Name of the Model Applied | Train Score | Test Score | RMSE  Score |
| Decision Tree Regressor | 0.38390 | 0.35075 | 0.04633 |
| Linear Regressor | 0.24722 | 0.24254 | 0.05005 |
| Random Forest Regressor | 0.40431 | 0.37868 | 0.04533 |
| Grid Search CV (Random Forest Regressor) | 0.96402 | 0.70860 | 0.03104 |
| XGB Regressor | 0.76980 | 0.63556 | 0.03471 |
| Grid Search CV (XG Boost) | 0.99937 | 0.73741 | 0.02946 |

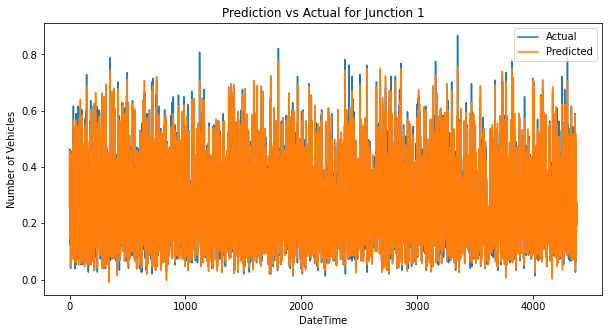
* *For Junction 4:-*

|  |  |  |  |
| --- | --- | --- | --- |
| Name of the Model Applied | Train Score | Test Score | RMSE  Score |
| Decision Tree Regressor | 0.49650 | 0.42116 | 0.07954 |
| Linear Regressor | 0.19975 | 0.20613 | 0.09315 |
| Random Forest Regressor | 0.52271 | 0.43272 | 0.07874 |
| Grid Search CV (Random Forest Regressor) | 0.80133 | 0.49095 | 0.07459 |
| XGB Regressor | 0.79586 | 0.48834 | 0.07478 |
| Grid Search CV (XG Boost) | 0.79586 | 0.48834 | 0.07478 |

**IV. Result & Analysis**

After analyzing through various models and junctions, a conclusion can be made to know which is the best model for a junction. The conclusion is :-

* **Junction 1** - XGBRegressor *after hyperparameter tuning with GridSearchCV*



*Accuracy score for the model GridSearchCV(*XGBRegressor*) on the train set:*

***0.9846079330767183***

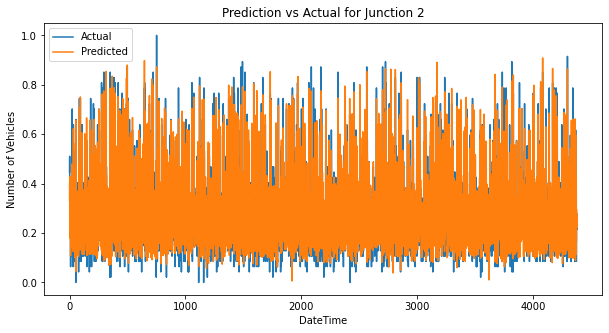
*Accuracy score for the model GridSearchCV(*XGBRegressor*) on the test set:*

***0.9753823289109363***

*RMSE score for the model GridSearchCV(*XGBRegressor*) on the test set:*

## 0.023702447113830984

● **Junction 2** - XGBRegressor *after hyperparameter tuning with GridSearchCV*



*Accuracy score for the model GridSearchCV(*XGBRegressor*) on the train set:*

***0.9418674199141464***

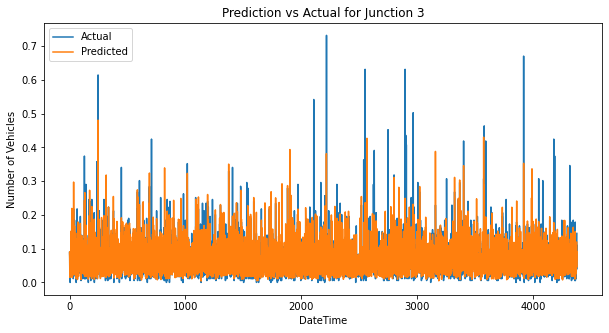
*Accuracy score for the model GridSearchCV(*XGBRegressor*) on the test set:*

***0.9117378990695377***

*RMSE score for the model GridSearchCV(*XGBRegressor*) on the test set:*

## 0.04610918678100267

● **Junction 3** - xgboost *after hyperparameter tuning with GridSearchCV*



*Accuracy score for the model GridSearchCV(*xgboost*) on the train set:*

***0.9640296641946572***

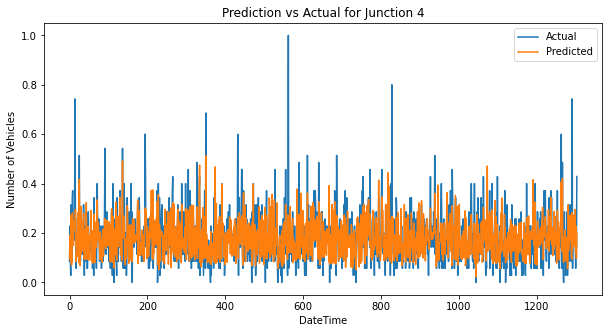
*Accuracy score for the model GridSearchCV(*xgboost*) on the test set:*

***0.7086006232467421***

*RMSE score for the model GridSearchCV(*xgboost*) on the test set:*

## 0.031044055027950327

● **Junction 4** - XGBRegressor *after hyperparameter tuning with GridSearchCV*



*Accuracy score for the model GridSearchCV(*XGBRegressor*) on the train set:*

***0.7009171982881327***

*Accuracy score for the model GridSearchCV(*XGBRegressor*) on the test set:*

***0.5237397514165021***

*RMSE score for the model GridSearchCV(*XGBRegressor*) on the test set:*

***0.07215436594321141***

# V. Web Application and Pipeline

We implemented an end-to-end pipeline using the python pickle library. The four best models for four different junctions were saved along with the preprocessing done on the data using the pickle library and the four “.pkl” model files were created.

Then, we converted our directory into a pipenv environment and installed all the necessary libraries.

We created a python file where we used the Python StreamLit library to render our website server which included a home page (predictions page) which takes the input features from the user in the form of radio buttons. Features input by the user:

* *Year*
* *Month*
* *Date*
* *Hour*
* *Junction*

We import all the four saved models in our py file and select which model to apply according to the Junction information provided by the user and finally return the rounded off predictions (Number of Vehicles) to the user.

# VI. References

* [Decision Tree Regressor](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html) | [RandomForestRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html) | [XGBoost](https://xgboost.readthedocs.io/en/stable/) |[LinearRegression](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)
* Vehicle Dataset | [Kaggle](https://www.kaggle.com/datasets/fedesoriano/traffic-prediction-dataset)
* Seaborn Plots | [Seaborn Docs](https://seaborn.pydata.org/examples/index.html)

# 

# VII. Acknowledgement

We would like to express our sincere gratitude towards **Mentor Urooj Khan** for guiding us through this training and helping us in grasping all the aspects of the various concepts covered. Furthermore, We are grateful to our Teaching Assistants for helping us in clearing all our doubts and guiding us all the way through the training. Finally, we would like to thank our company Scifor Technology ,Banglore, for providing us with the opportunity to work on this project.