Mini-Project – 2B Web based on ML (ITM 601)

Smart Traffic Controller

T. E. Information Technology

By

Advay Gujar 19
Aditya Kareer 25
Srujana Makarande 29
Dhruuv Naik 34

Under Guidance of
Prof. Sunantha Krishnan

Department of Information Technology
Don Bosco Institute of Technology
Vidyavihar Station Road, Mumbai - 400070

2023-2024

The Salesian Society's

Don Bosco Institute of Technology

Vidyavihar Station Road, Mumbai - 400070

CERTIFICATE

This is to certify that the project entitled **"Smart Traffic Controller"** is a bonafide work of **"Advay Gujar, Aditya Kareer, Srujana Makarande and Dhruuv Naik" Roll Nos. 19, 25, 29 and 34** submitted to the University of Mumbai towards completion of mini project work for the subject of **Mini Project – 2B Web Based on ML (ITM 601).**

Prof. Sunantha Krishnan
Mini Project Guide

Prof. Prasad Padalkar
H.O.D-I.T

Examiners

1. _____

2. _____

Date: 06-May-2024

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Advay Gujar

Aditya Kareer

Srujana Makarande

Dhruuv Naik

Date: 06-May-2024

ABSTRACT

Traffic congestion in urban areas has become a significant challenge, leading to various negative impacts such as road rage, accidents, air pollution, fuel wastage, and time delays. One of the primary causes of this congestion is the inefficiency of existing traffic management systems. To address this issue, this project proposes the design and implementation of an Adaptive Traffic Signal Control System (TSCS) that leverages real-time traffic density classification on individual lanes to dynamically allocate signal timings.

The core objective of the TSCS is to optimize traffic flow by adjusting signal cycles based on the current traffic density, thereby reducing congestion, and enhancing overall transportation efficiency. The system classifies traffic density into four categories: low, medium, high, and heavy using data collected from sensors or cameras installed at traffic signals. This real-time data is continuously analyzed by an adaptive control algorithm that determines the optimal signal cycle and timings for each lane based on the current traffic conditions.

Furthermore, the TSCS includes a historical data storage capability, allowing for the archiving of traffic data over time. This stored data can be utilized by analytical tools to identify traffic patterns, trends, and areas for improvement, enabling informed decision-making and proactive traffic management strategies.

The proposed system also considers future enhancements, such as integration with YOLO v8 for improved real-time data collection accuracy and automation of traffic density classification. A user-friendly interface will be developed for traffic management personnel to monitor traffic conditions, configure system parameters, access historical data analytics, and generate reports.

Through thorough testing and validation in simulated and real-world traffic scenarios, the effectiveness, reliability, and efficiency of the TSCS will be evaluated. The successful implementation of this project is expected to contribute significantly to reducing traffic congestion, improving air quality, saving fuel, enhancing productivity, and providing commuters with a more enjoyable travel experience in urban areas.

INDEX

List of Abbreviations

Sr. No.	Abbreviation	Full Form
1	TSCS	Traffic Signal Control System
2	AQI	Air Quality Index
3	RFID	Radio Frequency Identification
4	MPC	Model Predictive Control
5	RL	Reinforcement learning
6	ATSC	Adaptive Traffic Signal Control
7	SMOTE	Synthetic Minority Oversampling Technique
8	KNN	K Nearest Neighbors
9	SVM	Support Vector Machine
10	MARL	Multi Agent Reinforcement Learning

List of Figures

List of Tables

Chapter 1 - Introduction

1.1 Background

The rapid urbanization and increasing vehicular population have led to a surge in traffic congestion, posing substantial challenges for urban transportation systems worldwide. This congestion not only results in wasted time and fuel but also contributes significantly to air pollution and road safety concerns. Traditional traffic management systems often struggle to adapt to dynamic traffic conditions, leading to inefficiencies in signal timings and traffic flow optimization. To address these issues, there is a critical need for innovative solutions that can intelligently manage traffic in real-time based on the current traffic density on individual lanes. This project aims to develop an Adaptive Traffic Signal Control System (TSCS) that utilizes advanced technologies such as real-time data collection, traffic density classification, adaptive control algorithms, and historical data analysis to dynamically allocate signal timings and optimize traffic flow, ultimately enhancing urban mobility, reducing environmental impact, and improving overall transportation efficiency.

1.2 Need and Scope of the project

The proposed Adaptive Traffic Signal Control System (TSCS) addresses the shortcomings of current traffic management systems by dynamically adjusting signal timings based on real-time traffic density classification. This approach aims to reduce idle traffic, subsequently leading to decreased Air Quality Index (AQI) levels and fuel savings. Additionally, the system incorporates storage capabilities for historical traffic data, enabling analytical tools to identify patterns, trends, and areas for improvement in urban transportation. By optimizing traffic flow and reducing time spent in congestion, the TSCS promotes improved productivity, lower stress levels, and a more enjoyable commute for urban commuters.

1.3 Objectives and Problem Statement

Design an adaptive Traffic Signal Control System that utilizes real-time traffic density classification (low, medium, high, heavy) on individual lanes to dynamically allocate signal timings. The system aims to optimize traffic flow by adjusting signal cycles based on the current density, reducing congestion, and enhancing overall transportation efficiency in urban areas.

Chapter 2 - Literature Review

2.1 Summary of the investigation in the published papers

1. "RFID Technology Adoption": The system focuses on implementing magnetic inductive loop technology to address traffic congestion. The successful implementation of the inductive loop is expected to lead to a measurable reduction in traffic congestion in cities like India. The evaluation of inductive loop demonstrates effectiveness, with results aligning within a set threshold. However, there are gaps in explaining the threshold determination and providing a detailed comparison with an existing solution. Implementing a system like this may prove to be expensive.

2. "Reducing Congestion": The primary objectives of the system include reducing congestion, detecting bottlenecks early, and ultimately saving time and money for drivers. Demonstration of tangible time and cost savings for drivers. By efficiently managing traffic in real-time and detecting bottlenecks early, the system aims to contribute to a smoother flow of traffic, resulting in reduced travel time and associated costs for commuters.

3. "Revolutionizing Urban Mobility:": A Cost-Effective RFID-Based Smart Traffic Management System for Congestion Mitigation. Detrimental impact of traffic congestion in Indian cities, proposes Radio Frequency Identification (RFID) technology as a cost-effective solution to overcome the limitations of existing traffic management methods, detailing its components and applications, and concludes by emphasizing the system's potential to efficiently manage traffic while suggesting avenues for future improvements. It lacks a comprehensive discussion on the potential challenges and ethical considerations associated with the implementation using RFID.

4. "A Hierarchical Model Predictive Control Approach for Signal Splits Optimization in Large-Scale Urban Road Networks": They have proposed a hierarchical model predictive control (MPC) approach for signal split optimization in large-scale urban road networks. To reduce the computational complexity, a large-scale urban road network is first divided into several subnetworks using a network decomposition method. Second, the MPC optimization problem of the large-scale urban road network is presented, in which the interactions between neighboring subnetworks are described with interconnecting constraints. To coordinate the subnetworks, Lagrange multipliers are introduced to deal with interconnecting constraints among subnetworks, and an augmented Lagrange function is constructed. Then, based on dual optimization theory and a decomposition strategy, the dual optimization problem of the original MPC problem is divided into several new subproblems. In addition, they have developed a coordination algorithm based on an interaction prediction approach to coordinate the resulting subproblems with a two-level hierarchical structure. Finally, experimental results by means of simulation on a benchmark road network are presented, which illustrate the performance of the proposed approach.

5. “Large-scale traffic grid signal control with regional Reinforcement Learning”: Reinforcement learning (RL) based traffic signal control for large-scale traffic grids is challenging due to the curse of dimensionality. Most particularly, searching for an optimal policy in a huge action space is impractical, even with approximate Q-functions. On the other hand, heuristic self-organizing algorithms could achieve efficient decentralized control, but most of them have few efforts on optimizing the real-time traffic. This paper proposes a new regional RL algorithm that could form local cooperation regions adaptively, and then learn the optimal control policy for each region separately. They maintain a set of learning parameters to capture the control patterns in regions at different scales. At each step, they first decompose the large-scale traffic grid into disjoint sub-regions, depending on the real-time traffic condition. Next, they apply approximate Q-learning to learn the centralized control policy within each sub-region, by updating the corresponding learning parameters upon traffic observations. The numerical experiments demonstrate that their regional RL algorithm is computationally efficient and functionally adaptive, and it outperforms typical heuristic decentralized algorithms.

6. “Multi-Agent Deep Reinforcement Learning for Large-scale Traffic Signal Control”: Reinforcement learning (RL) is a promising data driven approach for adaptive traffic signal control (ATSC) in complex urban traffic networks, and deep neural networks further enhance its learning power. However, centralized RL is infeasible for large-scale ATSC due to the extremely high dimension of the joint action space. Multi-agent RL (MARL) overcomes the scalability issue by distributing the global control to each local RL agent, but it introduces new challenges: now the environment becomes partially observable from the viewpoint of each local agent due to limited communication among agents. This paper presents, for the first time, a fully scalable and decentralized MARL algorithm for the state-of-the-art deep RL agent: advantage actor critic (A2C), within the context of ATSC.

7. “Traffic Prediction Using Machine Learning”: The paper deals with traffic prediction that can be done in intelligent transportation systems which involve the prediction between the previous year’s dataset and the recent year’s data which ultimately provides the accuracy and mean square error. This prediction will be helpful for the people who are in need to check the immediate traffic state. The traffic data is predicated on a basis of 1 h time gap. Live statistics of the traffic is analyzed from this prediction. So, this will be easier to analyze when the user is on driving too. The system compares the data of all roads and determines the most populated roads of the city. I propose the regression model to predict the traffic using machine learning by importing Sklearn, Keras, and TensorFlow libraries. Keywords: Traffic, Regression, Intelligent transport system (ITS), and Machine learning prediction.

8. “Traffic flow prediction using machine learning and deep learning”: Based on our findings, the common and frequent machine learning techniques that have been applied for traffic flow prediction are Convolutional Neural Network and Long-Short Term Memory. The performance of their proposed techniques was compared with existing baseline models to determine their effectiveness. This paper is limited to certain

literature pertaining to common databases. Through this limitation, the discussion is more focused on (and limited to) the techniques found on the list of reviewed articles. The aim of this paper is to provide a comprehensive understanding of the application of ML and DL techniques for improving traffic flow prediction, contributing to the betterment of ITS in smart cities. For future endeavors, experimental studies that apply the most used techniques in the articles reviewed in this study (such as CNN, LSTM or a combination of both techniques) can be accomplished to enhance traffic flow prediction. The results can be compared with baseline studies to determine the accuracy of these techniques.

2.2 Comparison between the tools / methods / algorithms

Table 2.2.1 Comparison of Tools available in the market

Cisco Systems Inc	Utopia by Swarco
This solution supports Transit Signal Prioritization (TSP), Computer Aided Dispatch and Automated Vehicle Location (CAD/AVL), Dedicated Short Range Communications (DSRC), network management, data analytics, and Wireless Bulk Data Transfer (WBDT) for transforming urban roadway systems.	UTOPIA is an adaptive control system that enables traffic management authorities to optimize road traffic flow. Under the interurban traffic management segment, the company offers highway and tunnel systems, variable speed control systems, vehicle classification and detection systems, and dynamic and static signage.
Dedicated Short Range Communications (DSRC) may pose a limitation as it may not be universally adopted or compatible with emerging technologies, potentially restricting interoperability.	Uncertainties about adaptability to diverse infrastructures and regions, and the integration challenges with existing traffic management systems are not addressed.

Chapter 3 - Proposed Work

3.1 Architectural Details

The architecture diagram employed in the project illustrates a comprehensive process flow for the implementation of a machine learning-based traffic density classification system. Divided into two distinct sections, the diagram outlines both the training and testing processes essential for the development and validation of the classification model. The training process begins with the acquisition of a training sample from the dataset, followed by feature extraction to identify pertinent characteristics from the input data. Subsequently, pattern analysis is conducted to discern underlying patterns and relationships within the extracted features, facilitating the generation of corresponding labels. These features and labels are then utilized in the machine learning process to train the classification model, enabling it to accurately classify traffic density into predefined categories such as low, medium, and high.

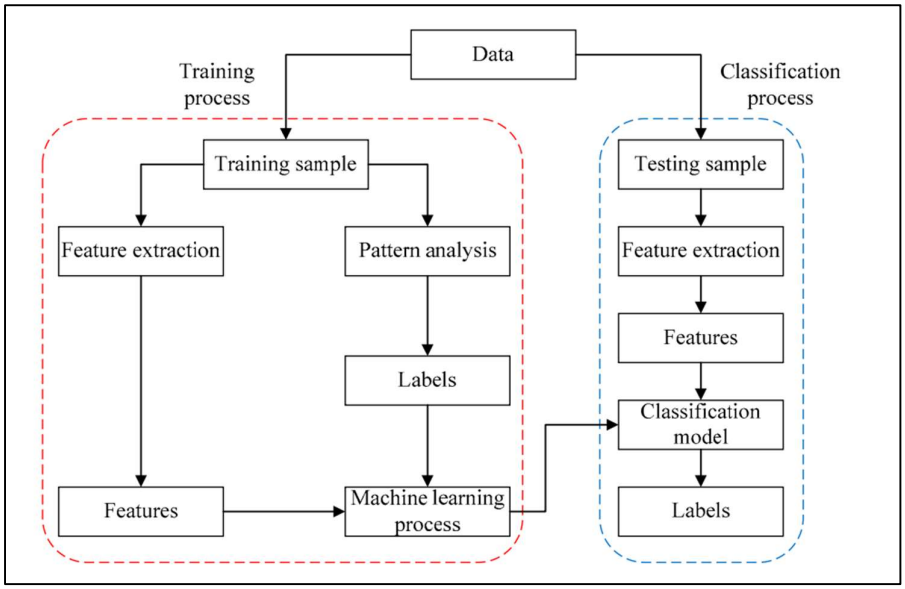

Figure 3.1.1 Architecture of proposed system

Conversely, the testing process depicted in the architecture diagram mirrors the training process, albeit with a focus on assessing the model's performance on unseen data. Commencing with the selection of a testing sample from the dataset, feature extraction is performed to extract relevant features from the input data. These features are then inputted into the previously trained classification model, which predicts labels based on the learned patterns and relationships. The predicted labels are subsequently compared against the ground truth labels to evaluate the model's accuracy and effectiveness in classifying traffic density. Together, these

processes exemplify the intricate yet systematic approach employed in the project to develop and validate an adaptive Traffic Signal Control System capable of dynamically adjusting signal timings based on real-time traffic density classification.

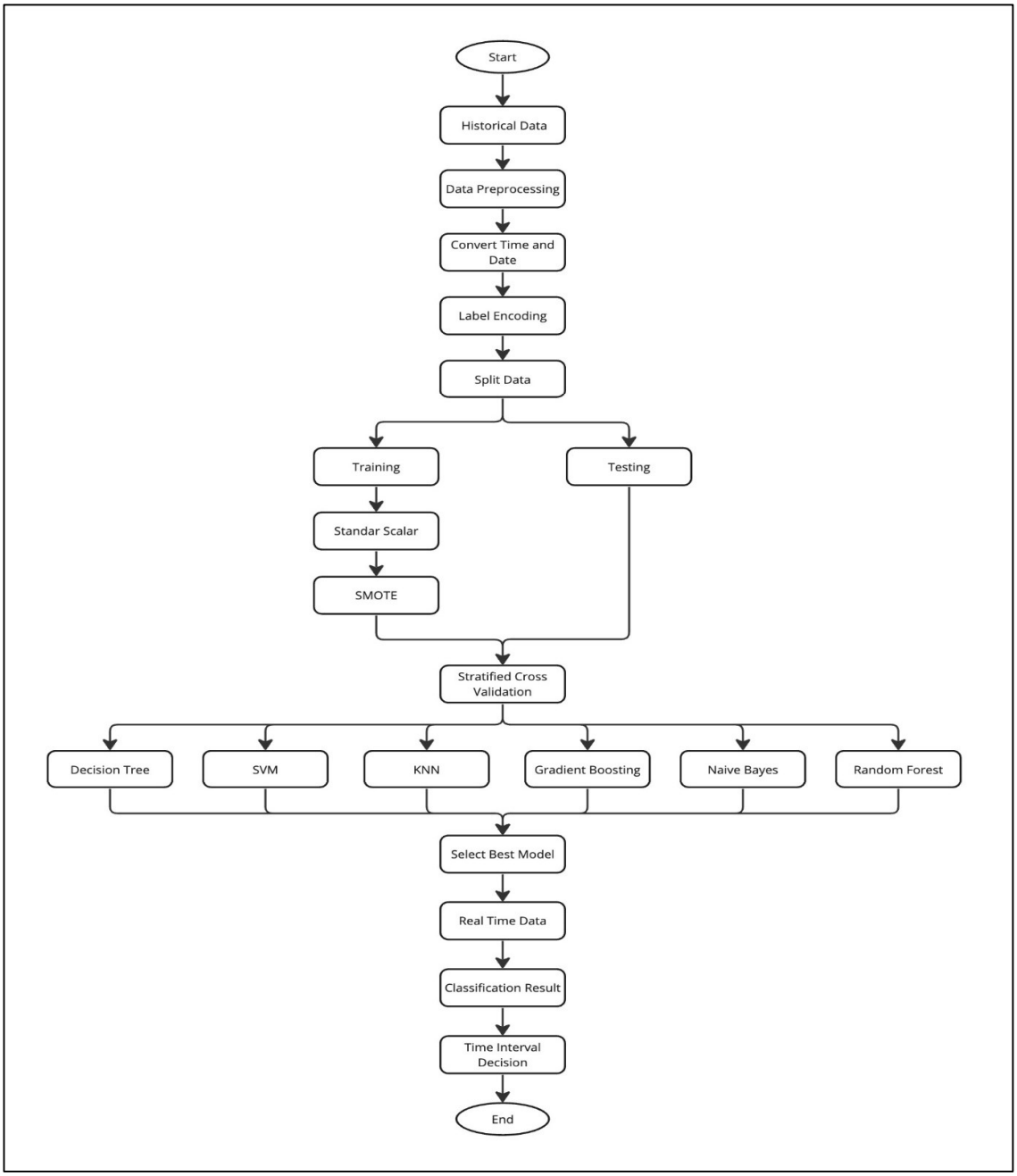

Figure 3.1.2 Flow diagram

In our project, data preprocessing serves as our initial crucial step. Here, we meticulously handle outliers and missing values within the dataset, ensuring that our subsequent analyses are based on reliable and accurate information. By addressing these anomalies, we pave the way for robust model training and decision-making processes.

Once the data is cleansed, we proceed to convert the time and date information into a format compatible with Python. This transformation is pivotal as it enables us to conduct in-depth temporal analyses, allowing us to uncover temporal patterns and trends in traffic density across different time periods, such as peak hours or seasonal variations.

Subsequently, we employ label encoding to convert categorical labels into numerical representations. By doing so, we enable our machine learning algorithms to interpret and learn from the categorical features present in our dataset effectively. This step is particularly important in our project as it ensures that our models can make informed decisions based on various categorical factors such as weather conditions, road types, and traffic regulations.

Following label encoding, we split our dataset into training and testing sets. This division is essential as it allows us to train our models on a subset of the data while reserving another portion for evaluating their performance. This ensures that our models are not only capable of learning from the data but also of generalizing well to unseen instances—a crucial requirement for real-world deployment.

To further enhance the performance of our models, we apply standard scaling to standardize the features and address class imbalances using the Synthetic Minority Over-sampling Technique (SMOTE). These techniques ensure that our models are not biased towards certain features or classes, thus improving their overall accuracy and reliability in predicting traffic density levels.

Stratified cross-validation further strengthens our model evaluation process by ensuring that our models are tested on representative samples of each class, accounting for the inherent variability in traffic density across different lanes and road conditions.

We then train multiple machine learning models, including Decision Trees, Support Vector Machines (SVM), k-Nearest Neighbors (KNN), Gradient Boosting, Naive Bayes, and Random Forest. Each model captures unique patterns and relationships within the data, allowing us to explore various avenues for predicting traffic density accurately.

Following model training, we select the best-performing model based on metrics such as accuracy and F1 score. This ensures that our chosen model not only accurately predicts traffic density levels but also maintains a balance between precision and recall—a crucial consideration in real-world traffic management scenarios.

With our model training phase completed, we transition to real-time data analysis. Here, our trained model analyzes incoming data from traffic lanes, predicting their respective density levels in real-time. These predictions serve as valuable inputs for making informed decisions regarding traffic signal timings and interval adjustments.

Based on the classification results provided by our model, traffic management authorities can make timely adjustments to the time intervals between traffic signals. For instance, if our model predicts high traffic density levels in a particular lane, authorities may opt to shorten the time interval between signals to alleviate congestion and improve traffic flow, thereby enhancing overall road safety and efficiency.

3.1.1 Data Collection

Data collection for the project was facilitated through the acquisition of relevant datasets from Kaggle, a popular platform for open datasets. Due to logistical constraints and the unavailability of access to real-time traffic cameras, manual data collection was not feasible. Therefore, leveraging Kaggle provided an accessible and efficient alternative, offering pre-existing datasets that encompassed a diverse range of traffic scenarios and conditions. While real-time data from traffic cameras would have been ideal for capturing dynamic traffic patterns, the utilization of Kaggle datasets enabled the project team to overcome limitations and proceed with the development and testing of the Traffic Signal Control System. Through meticulous selection and evaluation of Kaggle datasets, the project ensured the availability of comprehensive and representative data crucial for training and validating the machine learning-based classification model.

3.1.2 Data Preprocessing

Data preprocessing plays a pivotal role in the development of machine learning models, including the Adaptive Traffic Signal Control System. It involves a series of steps aimed at cleaning, transforming, and preparing the raw data to make it suitable for training and testing. In the context of this project, data preprocessing encompasses several key operations. The provided summary statistics describe the traffic dataset, including counts of cars, bikes, buses, and trucks, as well as the total count of all vehicle types detected within a 15-minute duration, facilitating an understanding of traffic patterns and volumes for analysis in the Traffic Signal Control System project.

```
print(data.shape)
data.describe()
```

```
(5952, 9)
```

	Date	CarCount	BikeCount	BusCount	TruckCount	Total
count	5952.000000	5952.000000	5952.000000	5952.000000	5952.000000	5952.000000
mean	16.000000	65.440692	12.161458	12.912970	18.646337	109.161458
std	8.945023	44.749335	11.537944	12.497736	10.973139	55.996312
min	1.000000	5.000000	0.000000	0.000000	0.000000	21.000000
25%	8.000000	18.750000	3.000000	2.000000	10.000000	54.000000
50%	16.000000	62.000000	9.000000	10.000000	18.000000	104.000000
75%	24.000000	103.000000	19.000000	20.000000	27.000000	153.000000
max	31.000000	180.000000	70.000000	50.000000	60.000000	279.000000

Figure 3.1.3 Summary statistics of dataset

The boxplot visualization offers insights into the temporal variations of traffic situations, highlighting the prevalence of "normal" traffic conditions for extended periods compared to other situations. Occasional shifts between "low" and "heavy" traffic indicate fluctuations in traffic volume, while sporadic spikes in "high" traffic hint at congestion during specific time intervals. Notably, the absence of visible outliers in the boxplot suggests that the dataset does not contain extreme or anomalous traffic events. This analysis underscores the dynamic nature of traffic patterns over time, providing valuable insights for optimizing the Traffic Signal Control System.

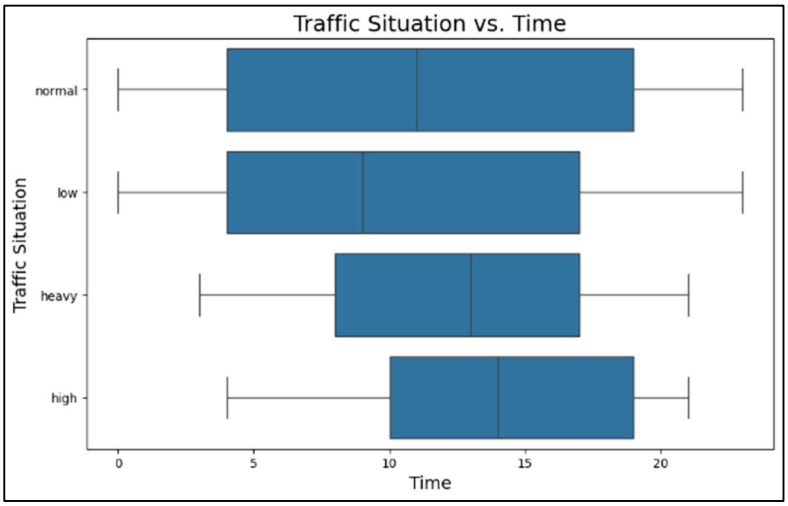

Figure 3.1.4 Boxplot of Traffic Situation Distribution Across Time Intervals

The distribution of traffic situations is further analyzed based on the day of the week. It shows variations in traffic counts across different days, with certain days exhibiting higher or lower traffic volumes in specific categories. For example, Fridays and Mondays tend to have higher counts of "normal" and "low" traffic situations compared to other days, while Sundays and Wednesdays experience relatively lower counts across all traffic situations. This analysis provides valuable insights into the temporal patterns of traffic conditions throughout the week,

which can inform strategies for optimizing traffic management and resource allocation in the Traffic Signal Control System.

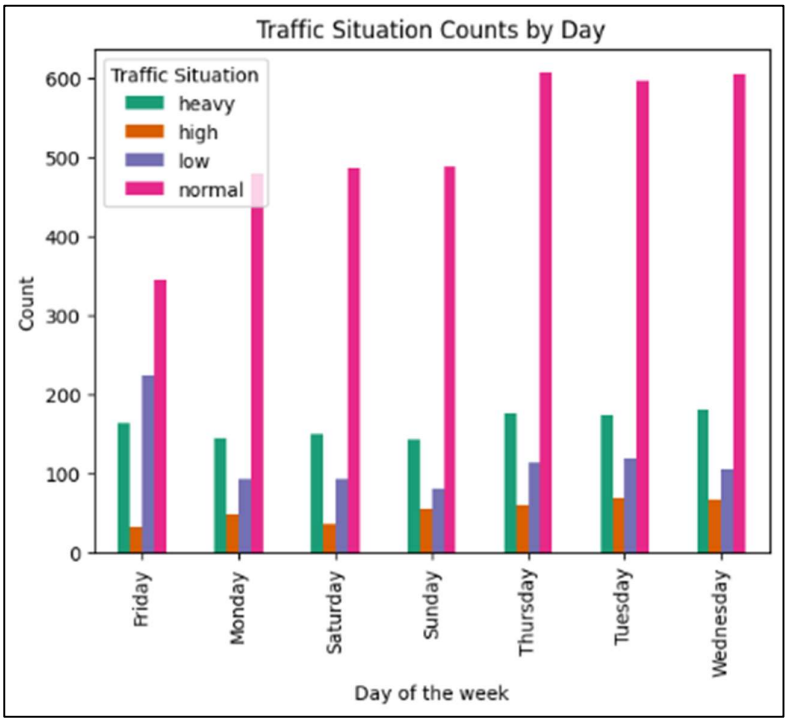

Figure 3.1.5 Traffic Situation Counts by Day of the Week

The 'Time' column is converted to datetime format, with only the hour component retained, providing temporal information relevant to traffic patterns. Redundant features such as the 'Date' column are then removed to streamline the dataset.

```
data['Time'] = pd.to_datetime(data['Time']).dt.hour
data.drop(['Date'], axis=1, inplace=True)
data.head()
```

Figure 3.1.6 Snapshot of step for formatting time

	Time	Day of the week	CarCount	BikeCount	BusCount	TruckCount	Total	Traffic Situation
0	0	Tuesday	13	2	2	24	41	normal
1	0	Tuesday	14	1	1	36	52	normal
2	0	Tuesday	10	2	2	32	46	normal
3	0	Tuesday	10	2	2	36	50	normal
4	1	Tuesday	11	2	1	34	48	normal

Figure 3.1.7 Screenshot of dataset after datetime formatting

Categorical variables like 'Traffic Situation' and 'Day of the week' are encoded using LabelEncoder to convert them into numerical representations, ensuring compatibility with machine learning algorithms. The mappings between original categorical values and their numerical equivalents are recorded for reference.

```
label_encoder = LabelEncoder()
# Encoding the days of week to convert categorical values into numerical representations
data['Day of the week'] = label_encoder.fit_transform(data['Day of the week'])
print("Mapping:", dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_))))

data['Traffic Situation'] = label_encoder.fit_transform(data['Traffic Situation'])
print("Mapping:", dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_))))
```

Figure 3.1.8 Snapshot of code for Label Encoding

```
Mapping: {'heavy': 0, 'high': 1, 'low': 2, 'normal': 3}
Mapping: {'Friday': 0, 'Monday': 1, 'Saturday': 2, 'Sunday': 3, 'Thursday': 4, 'Tuesday': 5, 'Wednesday': 6}
```

	Time	Day of the week	CarCount	BikeCount	BusCount	TruckCount	Total	Traffic Situation
0	0	5	13	2	2	24	41	3
1	0	5	14	1	1	36	52	3
2	0	5	10	2	2	32	46	3
3	0	5	10	2	2	36	50	3
4	1	5	11	2	1	34	48	3

Figure 3.1.9 Screenshot of dataset after Label Encoding

Additionally, to maintain consistency and improve model convergence, the dataset is standardized using StandardScaler, resulting in features with a mean of 0 and a standard deviation of 1. These preprocessing steps are essential for enhancing the quality, usability, and performance of the dataset, laying a solid foundation for the subsequent training and evaluation of the Traffic Signal Control System.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print(pd.DataFrame(X_train_scaled))
```

Figure 3.1.10 Snapshot of code for Standard Scaling

	0	1	2	3	4	5	6
0	0.941037	0.917901	0.793277	1.525799	0.824629	-0.050123	1.123471
1	0.508994	0.917901	0.704063	-0.876661	-0.626780	-0.050123	0.232208
2	-0.355093	-0.077926	1.395466	0.324569	1.711601	-1.689297	1.230423
3	-0.643121	0.917901	1.551589	1.525799	0.340826	-0.778645	1.479976
4	0.364979	1.415814	0.681760	0.839382	1.872868	-1.233971	0.891743
...	...	...	...	...	...	...	...
4161	-0.643121	1.415814	1.685408	0.238767	0.663361	-1.233971	1.301724
4162	-1.507207	0.419987	-1.214020	-1.048265	-1.029949	-0.232253	-1.461193
4163	-0.211078	0.419987	0.257998	0.925184	0.098924	0.132007	0.446111
4164	-1.219179	-0.575839	-1.035593	-0.619254	-0.949315	1.771182	-0.819483
4165	1.661108	1.415814	-1.303233	-1.048265	-0.868681	1.497986	-1.158163

Figure 3.1.11 Screenshot of dataset after Standard Scaling

The initial distribution of classes in our data, as depicted in the first image below, reveals a significant skew towards the "normal" traffic situation category, while the counts for "high," "low," and "heavy" situations are comparatively lower. This imbalance in class distribution can lead to biased model predictions, particularly if the minority classes are underrepresented during model training. To address this issue, we applied Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples for the minority classes, as shown in the second image below. Through SMOTE, we augmented the training data by creating synthetic instances of the minority classes, thereby balancing the distribution across all traffic situation categories. This approach ensures that the machine learning models are trained on a more representative dataset, improving their ability to generalize and make accurate predictions for all traffic situations.

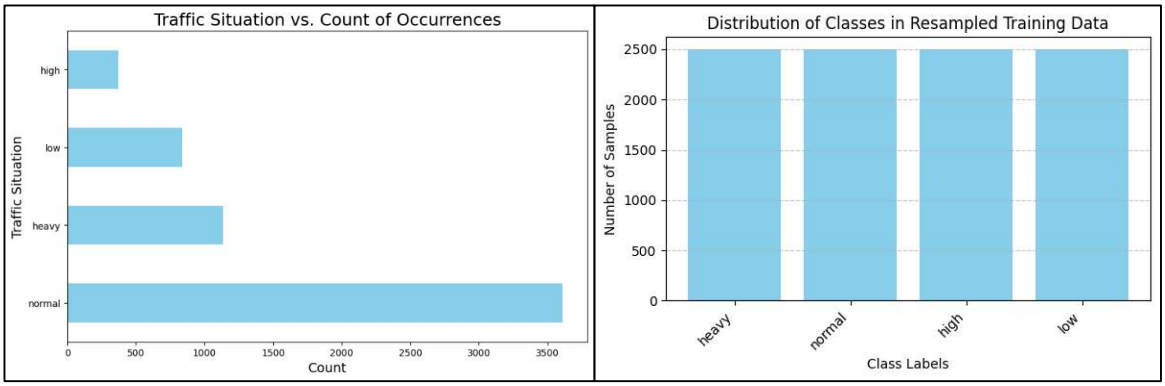

Figure 3.1.12 Class Distribution Analysis and SMOTE Resampling for Traffic Situation

3.1.3 Feature Selection

Despite the limited availability of multiple features in the dataset, feature selection remains a critical aspect of the model development process. Although traditional feature selection techniques may not be applicable due to the dataset's constraints, it is imperative to carefully consider the relevance and significance of each feature in the context of the Traffic Signal Control System. While the dataset primarily comprises vehicle counts and traffic situation classifications, feature selection efforts can focus on identifying and prioritizing the most informative features that contribute to the accurate prediction of traffic density and congestion levels. Moreover, feature engineering techniques such as aggregation, transformation, and creation of new features based on domain knowledge can be explored to enhance the dataset's richness and utility for model training. By adopting a thoughtful and creative approach to feature selection, even with limited feature availability, the Traffic Signal Control System can be developed with improved efficiency and effectiveness in optimizing traffic flow and enhancing urban mobility.

3.1.4 Train Test Split

The utilization of a 70-30 train-test split offers a balanced approach that strikes a harmonious balance between model training and evaluation. This split allocates 70% of the dataset for training the classification algorithms, allowing ample data to learn and generalize patterns effectively. Simultaneously, reserving 30% of the dataset for testing provides a robust evaluation framework, enabling thorough assessment of the model's performance on unseen data. By employing this split, the Traffic Signal Control System benefits from sufficient training data to build robust classification models while ensuring an extensive evaluation process that accurately gauges the model's efficacy in real-world scenarios. This approach subtly emphasizes the importance of comprehensive model evaluation while maximizing the utilization of available data, ultimately enhancing the system's ability to optimize traffic flow and improve urban mobility.

3.1.5 Performance Evaluation

Performance evaluation of the Traffic Signal Control System is paramount to ensuring its effectiveness in optimizing traffic flow and enhancing urban mobility. Leveraging classification metrics such as F1 score, accuracy, precision, and recall, the system's performance is rigorously assessed against the test dataset. By analyzing these metrics, the model's ability to accurately classify traffic density and predict congestion levels can be gauged, providing valuable insights into its reliability and robustness. Additionally, techniques such as cross-validation and hyperparameter tuning are employed to fine-tune model parameters and improve performance further. Through meticulous performance evaluation, the Traffic Signal Control System can be iteratively refined, enhancing its capability to alleviate congestion, reduce travel time, and enhance overall transportation efficiency in urban areas.

Chapter 4 - Implementation

4.1 Algorithm Details

A. Decision Tree

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes, and leaf nodes.

The Decision Tree algorithm operates by first starting at the root node, which signifies the entire dataset. It then proceeds to identify the most crucial feature or question that can effectively divide the data into distinct groups, akin to choosing a path at a fork in the tree. Subsequently, based on the answer to this question, the algorithm creates new branches by dividing the data into smaller subsets, each branch representing a potential route through the tree. This process of asking questions and splitting the data at each branch is repeated iteratively until the algorithm reaches the leaf nodes, which symbolize the final predicted outcomes or classifications. In essence, the Decision Tree algorithm systematically navigates through the data, employing a series of questions and divisions to arrive at accurate predictions or classifications.

Pseudocode:

GenDecTree(Sample S, Features F)
Steps:

1. ***If*** *stopping_condition(S, F) = true* ***then***
   a. *Leaf = createNode()*
   b. *leafLabel = classify(s)*
   c. ***return*** *leaf*
2. *root = createNode()*
3. *root.test_condition =findBestSpilt(S,F)*
4. *V = {v | v a possible outcomecfroot.test_condition}*
5. ***For each*** *value v ∈ V:*
   a. S_v = {*s | root.test_condition(s) = v and s ∈ S*};
   b. *Child = TreeGrowth* (S_v,*F*);
   c. *Add child as descent of root and label the edge {root → child} as v*
6. ***return*** *root*

Figure 4.1.1 Pseudo code of decision tree [10]

Code Snippet:

```
decision_tree = DecisionTreeClassifier(max_depth=10)
decision_tree.fit(X_train_scaled, y_train)
y_pred = decision_tree.predict(X_test_scaled)
```

Figure 4.1.2 Code snippet of decision tree

Mathematical Formulas:

$$\text{Information Gain} = E(Y) - E(Y|X) \quad (1)$$

$$\text{Entropy (S)} = -p\, log_2\, p - q\, log_2\, q \quad (2)$$

where, p is the probability of positive class
q is the probability of negative class
S is the subset of training example

B. Support Vector Machine

Support Vector Machine (SVM) is a robust classification and regression technique that maximizes the predictive accuracy of a model without overfitting the training data. SVM is particularly suited to analyzing data with very large numbers (for example, thousands) of predictor fields.

Pseudocode:

Data : Dataset with p^* variables and binary outcome.
Output: Ranked list of variables according to their relevance.

Find the optimal values for the tuning parameters of the SVM model;
Train the SVM model;
$p \leftarrow p^*$;
while $p \geq 2$ **do**
  $SVM_p \leftarrow$ SVM with the optimized tuning parameters for the p variables and observations in **Data**;
  $w_p \leftarrow$ calculate weight vector of the SVM_p $(w_{p1}, \ldots, w_{pp})$;
  $rank.criteria \leftarrow (w_{p1}^2, \ldots, w_{pp}^2)$;
  $min.rank.criteria \leftarrow$ variable with lowest value in $rank.criteria$ vector;
  Remove $min.rank.criteria$ from **Data**;
  $Rank_p \leftarrow min.rank.criteria$;
  p ← p - 1 ;
end
$Rank_1 \leftarrow$ variable in **Data** $\notin (Rank_2, \ldots, Rank_{p^*})$;
return $(Rank_1, \ldots, Rank_{p^*})$

Figure 4.1.3 Pseudo code of SVM [11]

Code Snippet:

```
svm = SVC()
svm.fit(X_train_scaled, y_train)
y_pred = svm.predict(X_test_scaled)
```

Figure 4.1.4 Code snippet of SVM

C. K-NN

The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today.

Pseudocode:

Algorithm 4: *kNN kernel Algorithm*

Input : D, a chunk of the original distance matrix;
$n_{chunksize}$, dimension of the chunk;
split, index of split;
chunk, index of chunk;
Maxk, an array to hold the farthest neighbors for each row index in the chunks;

Output: none, an (intermediate) *k*NN graph stored in Gk′;

```
row′ ← blockIdx.x × blockDim.x + threadIdx.x;
if row′ < n_chunksize then
    row ← split × n_chunkSize + row′ // absolute row in the distance matrix;
    for column′ ← 1 to n_chunksize do
        column ← chunk × n_chunkSize + column′ //absolute column in the distance matrix;
        if row = column or row > n_row or column > n_col then
            continue /* exclude diagonal and pad regions */;
        if D[row′ × n_chunksize + column′] < Gk′[Maxk[row′]].weight then
            Gk′[Maxk[row′]].source ← row;
            Gk′[Maxk[row′]].target ← column;
            Gk′[Maxk[row′]].weight ← D[row′ × n_chunksize + column′];
            Search the new maximum element in row′(D) and store the index in Maxk[row′];
```

Figure 4.1.5 Pseudo code of KNN [12]

Code Snippet:

```
knn = KNeighborsClassifier()
knn.fit(X_train_scaled, y_train)
y_pred = knn.predict(X_test_scaled)
```

Figure 4.1.6 Code snippet of KNN

Mathematical Formulas:

Euclidean Distance: $\mathbf{d(x,y)} = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$ (3)

Manhattan Distance: $\mathbf{d(x,y)} = \sum_{i=1}^{m}|x_i - y_i|$ (4)

$$\underline{\text{Centroid}}: (\mathbf{G}) = \left(\frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n}\right) \quad (5)$$

D. Gradient Boost Classifier

Boosting is an ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors. Boosting algorithms can improve the predictive power of your data mining initiatives. Ensemble learning

Ensemble learning gives credence to the idea of the "wisdom of crowds," which suggests that the decision-making of a larger group of people is typically better than that of an individual expert. Similarly, ensemble learning refers to a group (or ensemble) of base learners, or models, which work collectively to achieve a better final prediction. A single model, also known as a base or weak learner, may not perform well individually due to high variance or high bias. However, when weak learners are aggregated, they can form a strong learner, as their combination reduces bias or variance, yielding better model performance.

Gradient boosting works by sequentially adding predictors to an ensemble with each one correcting for the errors of its predecessor. However, instead of changing weights of data points like AdaBoost, the gradient boosting trains on the residual errors of the previous predictor. The name, gradient boosting, is used since it combines the gradient descent algorithm and boosting method.

Pseudocode:

1. Initialize model with a constant value:

$$F_0(x) = \underset{\gamma}{argmin} \sum_{i=1}^{n} L(y_i, \gamma)$$

2. for $m = 1$ *to* M:

2-1. Compute residuals $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ $for\ i = 1,\ldots, n$

2-2. Train regression tree with features x against r and create terminal node reasions R_{jm} for $j = 1,\ldots, J_m$

2-3. Compute $\gamma_{jm} = \underset{\gamma}{argmin} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$ $for\ j = 1,\ldots, J_m$

2-4. Update the model:

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} 1(x \in R_{jm})$$

Figure 4.1.7 Pseudo code of Gradient Boost Classifier [13]

Code Snippet:

```
gradient_boosting = GradientBoostingClassifier()
gradient_boosting.fit(X_train_scaled, y_train)
y_pred = gradient_boosting.predict(X_test_scaled)
```

Figure 4.1.8 Code snippet of Gradient Boost Classifier

E. Naïve Bayes Classifier

The Naïve Bayes classifier is a supervised machine learning algorithm that is used for classification tasks such as text classification. They use principles of probability to perform classification tasks.

Pseudocode:

Input:

Training dataset T,

F= (f_1, f_2, f_3,.., f_n) // value of the predictor variable in testing dataset.

Output:

A class of testing dataset.

Step:

1. Read the training dataset T;
2. Calculate the mean and standard deviation of the predictor variables in each class;
3. Repeat

   Calculate the probability of f_i using the gauss density equation in each class;

   Until the probability of all predictor variables (f_1, f_2, f_3,.., f_n) has been calculated.
4. Calculate the likelihood for each class;
5. Get the greatest likelihood;

Figure 4.1.9 Pseudo code of Naïve Bayes Classifier [14]

Code Snippet:

```
naive_bayes = GaussianNB()
naive_bayes.fit(X_train_scaled, y_train)
y_pred = naive_bayes.predict(X_test_scaled)
```

Figure 4.1.10 Code snippet of Naïve Bayes Classifier

Mathematical Formulas:

$$P(c|x) = \frac{P(x|c) * P(c)}{P(x)} \quad (6)$$

P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
P(c) is the prior probability of class.
P(x|c) is the likelihood which is the probability of the predictor given class.
P(x) is the prior probability of the predictor.

Markov Chain, for multiple independent variables:

$$\mathbf{P\ (c|X)} = P(x_1|c) \times P(x_2|c) \times \ldots \times P(x_n|c)\ \times\ P(c) \quad (7)$$

F. Random Forest

Random forest is a commonly-used machine learning algorithm, that combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

Pseudocode:

To generate c classifiers:
for i = 1 to c **do**
  Randomly sample the training data D with replacement to produce D_i
  Create a root node, N_i containing D_i
  Call BuildTree(N_i)
end for

BuildTree(N):
if N contains instances of only one class **then**
  return
else
  Randomly select x% of the possible splitting features in N
  Select the feature F with the highest information gain to split on
  Create f child nodes of N, N_1,..., N_f, where F has f possible values (F_1, ... , F_f)
  for i = 1 to f **do**
    Set the contents of N_i to D_i, where D_i is all instances in N that match
    F_i
    Call BuildTree(N_i)
  end for
end if

Figure 4.1.11 Pseudo code of Random Forest [15]

Code Snippet:

```
random_forest = RandomForestClassifier(n_estimators=100, max_depth=13, min_samples_split= 5, min_samples_leaf=1, random_state=42)
random_forest.fit(X_train_scaled, y_train)
y_pred = random_forest.predict(X_test_scaled)
```

Figure 4.1.12 Code snippet of Random Forest

4.2 Dataset Details

The dataset presented contains a comprehensive record of traffic observations, capturing crucial details essential for studying traffic patterns and conditions. Each entry in the dataset corresponds to a 15-minute interval and includes information such as the time of observation, date, and day of the week, facilitating temporal analysis of traffic trends. Additionally, the

dataset provides counts for various types of vehicles, including cars, bikes, buses, and trucks, as detected by a computer vision model. These vehicle counts offer valuable insights into the composition and volume of traffic during different time periods, enabling a granular examination of traffic flow dynamics.

Time	Date	Day of the week	CarCount	BikeCount	BusCount	TruckCount	Total	Traffic Situation
12:00:00 AM	10	Tuesday	13	2	2	24	41	normal
12:15:00 AM	10	Tuesday	14	1	1	36	52	normal
12:30:00 AM	10	Tuesday	10	2	2	32	46	normal
12:45:00 AM	10	Tuesday	10	2	2	36	50	normal
1:00:00 AM	10	Tuesday	11	2	1	34	48	normal

Figure 4.2.1 Screenshot of dataset

Furthermore, the dataset encompasses a pivotal feature—the "Traffic Situation" column, categorizing traffic conditions into four classes: heavy, high, normal, and low. This classification scheme allows for the assessment of congestion severity and facilitates the monitoring of traffic conditions at different times and days of the week. By providing a holistic view of traffic situations, this dataset equips traffic management authorities and urban planners with the necessary information to devise effective strategies for mitigating congestion, optimizing traffic flow, and improving overall transportation efficiency. In essence, the dataset serves as a valuable resource for studying and understanding urban traffic dynamics, offering insights that can inform evidence-based decision-making and contribute to the development of smarter, more efficient transportation systems.

4.3 Performance Metrics Details

1. Precision:

Precision measures the proportion of true positive predictions among all positive predictions made by a classification model. It focuses on the accuracy of positive predictions and is calculated as:

$$\textbf{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (8)$$

2. Recall:

Recall (also known as sensitivity or true positive rate) measures the proportion of true positive predictions among all actual positive instances in the dataset. It focuses on capturing all positive instances and is calculated as:

$$\textbf{Recall} = \frac{True\ Positive}{True\ Positive + Fals\ \ Negatives} \quad (9)$$

3. F1 Score:
The F1 score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance that considers both false positives and false negatives. It is calculated as:

$$\textbf{Precision} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (10)$$

The F1 score is useful when there is an imbalance between classes or when both false positives and false negatives need to be minimized.

4. Accuracy:
Accuracy measures the overall correctness of predictions made by a classification model and is calculated as the ratio of correctly classified instances to the total instances in the dataset:

$$\textbf{Accuracy} = \frac{True\ Positive + True\ Negative}{Total\ Instances} \quad (11)$$

Accuracy is a general measure of model performance but may not be suitable for imbalanced datasets where one class dominates the others.

Table 4.3.1 Performance metrics table

	Precision	**Recall**	**F1 Score**	**Accuracy**
Decision Tree	1.00	1.00	1.00	0.99
SVM	0.96	0.96	0.96	0.95
KNN	0.93	0.93	0.93	0.93
Gradient Boosting Classifier	1.00	1.00	1.00	0.99
Naïve Bayes	0.85	0.81	0.82	0.80
Random Forest	1.00	1.00	1.00	0.99

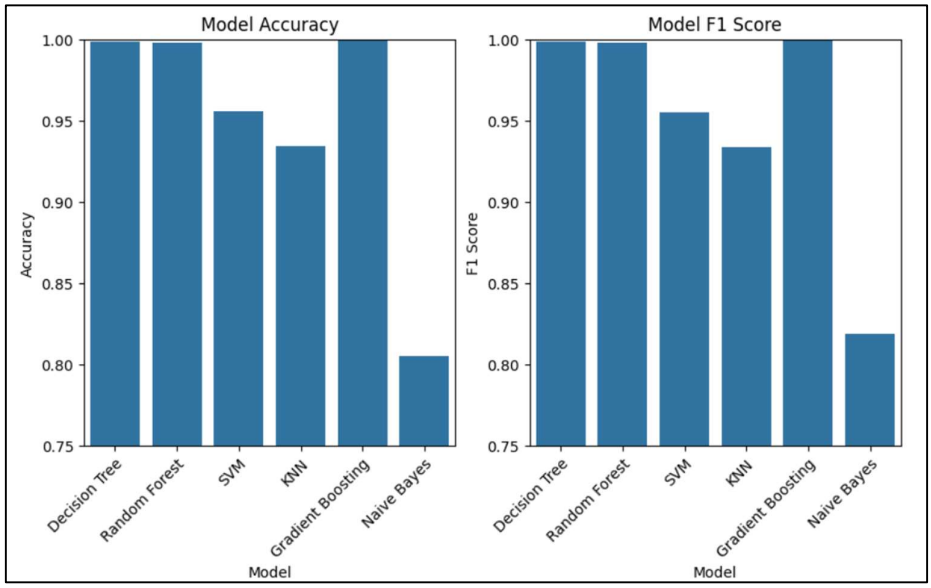

Figure 4.3.1 Comparison Between Models

In the first visualization, depicting model accuracy, Gradient Boosting emerges as the top performer, boasting the highest accuracy among the considered models. Conversely, Naive Bayes demonstrates relatively lower accuracy. The second visualization showcases the F1 scores, with Gradient Boosting again excelling and Naive Bayes trailing behind. These insights serve as invaluable guidance for model selection, facilitating informed decisions in the pursuit of optimal predictive analytics solutions.

4.4 Implementation Plan

Table 4.4.1 Work Plan table

Activity	Start Date	Duration (in days)
Literature review	25-01-2024	6
Refinement of problem formulation	27-01-2024	1
Identifying or creating data	18-01-2024	11
Acquaintance with the tool to implement	18-01-2024	7
Implementation and Presentation 1	01-02-2024	5
Coding/Discovering the properties of table	08-02-2024	2
Generating/reporting visualizations	10-02-2024	5
Importing necessary libraries/apis for implementation	15-02-2024	1
Code ML algorithm	16-02-2024	30
Performance evaluation of ML algorithm	17-03-2024	5
Refine algorithm again and Presentation 2	22-03-2024	10
Documentation	01-04-2024	5

Chapter 5 - Results and Discussions

The evaluation of different classification algorithms for the Traffic Signal Control System offers valuable insights into their performance and suitability for real-world deployment. Among the algorithms assessed, the Decision Tree, Gradient Boosting Classifier, and Random Forest stand out as top performers, achieving perfect precision, recall, and F1 score values of 1.00. Moreover, these algorithms demonstrate high accuracy levels ranging from 0.99 to 0.95, highlighting their exceptional ability to accurately predict traffic situations. The SVM algorithm also delivers commendable results, with precision, recall, and F1 score values of 0.96 and an accuracy of 0.95, affirming its efficacy in traffic classification tasks.

On the other hand, while the KNN algorithm shows slightly lower performance compared to the top performers, it still maintains respectable precision, recall, and F1 score values of 0.93, alongside an accuracy of 0.93. Despite its slightly lower accuracy, the KNN algorithm remains competitive, suggesting its potential suitability for integration into the Traffic Signal Control System. However, the Naïve Bayes algorithm exhibits comparatively lower performance across all metrics, with precision, recall, F1 score, and accuracy values of 0.85, 0.81, 0.82, and 0.80, respectively. Although falling short of the top-performing algorithms, the Naïve Bayes results provide valuable insights into potential trade-offs between performance and computational efficiency, which may be relevant in certain deployment scenarios.

Overall, the findings from this evaluation shed light on the strengths and weaknesses of different classification algorithms for traffic situation prediction. By leveraging the top-performing algorithms identified in this analysis, the Traffic Signal Control System can be effectively optimized to accurately classify traffic situations, ultimately contributing to improved traffic management, reduced congestion, and enhanced urban mobility. The comprehensive analysis of algorithm performance underscores the importance of informed decision-making in selecting the most suitable algorithms for real-world deployment, considering factors such as accuracy, computational efficiency, and scalability to ensure the system's effectiveness and reliability in dynamic traffic environments.

Chapter 6 - Conclusion and Future Scope

6.1 Conclusion

In conclusion, the evaluation of various classification algorithms for the Traffic Signal Control System has provided valuable insights into their performance and applicability in real-world scenarios. The Decision Tree, Gradient Boosting Classifier, and Random Forest algorithms emerged as top performers, showcasing exceptional precision, recall, F1 score, and accuracy values. These algorithms demonstrate robust capabilities in accurately predicting traffic situations, laying a solid foundation for the development of an effective traffic management system. Additionally, the SVM and KNN algorithms exhibit commendable performance, further expanding the range of potential algorithmic choices for the Traffic Signal Control System. However, the lower performance of the Naïve Bayes algorithm highlights the importance of carefully selecting algorithms based on specific requirements and objectives.

Moving forward, the findings from this evaluation will inform the development and deployment of the Traffic Signal Control System, facilitating the optimization of traffic flow, reduction of congestion, and enhancement of urban mobility. By leveraging the insights gained from algorithm performance evaluation, stakeholders can make informed decisions regarding algorithm selection, ensuring the system's effectiveness and reliability in dynamically changing traffic environments.

6.2 Future Work

Future work in the development of the Traffic Signal Control System could focus on several key areas to further enhance its effectiveness and adaptability. Firstly, integrating real-time data collection capabilities using advanced technologies such as IoT sensors and cameras at traffic signals could provide a continuous stream of data for more accurate and dynamic traffic situation analysis. Additionally, exploring the use of advanced machine learning techniques such as deep learning models could improve the system's ability to capture complex traffic patterns and make more precise predictions. Furthermore, expanding the scope of the system to incorporate additional factors such as weather conditions, special events, and road infrastructure could enhance its capability to respond to a wider range of traffic scenarios and optimize traffic flow more effectively. Finally, collaboration with city planners, and transportation authorities could facilitate the integration of the Traffic Signal Control System into broader smart city initiatives, fostering more sustainable and efficient urban transportation systems.

References

[1] I J. E. Naranjo, M. Humayun, et al, "Smart Traffic Management System for Metropolitan Cities of Kingdom Using Cutting Edge Technologies," Journal of Advanced Transportation, vol. 2022, pp. 4687319, Sep. 29, 2022. DOI: 10.1155/2022/4687319.

[2] N. Lanke and S. Koul, "Smart Traffic Management System," International Journal of Computer Applications, vol. 75, pp. 19-22, 2013. DOI: 10.5120/13123-0473.

[3] B.-L. Ye, W. Wu, L. Li, and W. Mao, "A Hierarchical Model Predictive Control Approach for Signal Splits Optimization in Large-Scale Urban Road Networks," in IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 8, pp. 2182-2192, Aug. 2016, doi: 10.1109/TITS.2016.2517079.

[4] T. Chu, S. Qu, and J. Wang, "Large-scale traffic grid signal control with regional Reinforcement Learning," in 2016 American Control Conference (ACC), Boston, MA, USA, 2016, pp. 815-820, doi: 10.1109/ACC.2016.7525014.

[5] T. Chu, J. Wang, L. Codecà, et al., "Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control," in IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 3, pp. 1086-1095, March 2020, doi: 10.1109/TITS.2019.2901791.

[6] H. R. Deekshetha, A. Madhav, and Amit Tyagi, "Traffic Prediction Using Machine Learning," 2022. DOI: 10.1007/978-981-16-9605-3_68.

[7] N. A. M. Razali et al., "Gap, techniques and evaluation: traffic flow prediction using machine learning and deep learning," Journal of Big Data, vol. 8, no. 1, p. 152, Dec. 04, 2021. DOI: 10.1186/s40537-021-00542-7.

[8] Cisco. "Internet of Things for Roadways and Intersections," Cisco. Available: https://www.cisco.com/c/en/us/solutions/internet-of-things/roadways-intersections.html. Accessed: Feb. 1, 2024.

[9] SWARCO. "Urban Traffic Management - UTOPIA," SWARCO. Available: https://www.swarco.com/products/software/urban-traffic-management/utopia. Accessed: Feb. 1, 2024.

[10] "Pseudocode of Decision Tree Algorithm." ResearchGate. [Online]. Available: https://www.researchgate.net/figure/Pseudocode-of-Decision-Tree-Algorithm_fig1_338528758. [Accessed: Apr. 3, 2024].

[11] "Pseudo-code of the SVM-RFE algorithm using the linear kernel in a model for binary classification." ResearchGate. [Online]. Available: https://www.researchgate.net/figure/Pseudo-code-of-the-SVM-RFE-algorithm-using-the-linear-kernel-in-a-model-for-binary_fig1_329055871. [Accessed: Apr. 1, 2024].

[12] "Pseudocode for the k-NN Kernel: The kNN kernel algorithm utilizes one-dimensional block." ResearchGate. [Online]. Available: https://www.researchgate.net/figure/Pseudocode-for-the-k-NN-Kernel-The-kNN-kernel-algorithm-utilizes-one-dimensional-block_fig8_230769875. [Accessed: Apr. 1, 2024].

[13] "Pseudocode of the Gradient Tree Boosting algorithm. Source: Friedman, 2002." ResearchGate. [Online]. Available: https://www.researchgate.net/figure/Pseudocode-of-the-Gradient-Tree-Boosting-algorithm-Source-Friedman-2002_fig17_338195250. [Accessed: Apr. 1, 2024].

[14] "Pseudocode of naive bayes algorithm." ResearchGate. [Online]. Available: https://www.researchgate.net/figure/Pseudocode-of-naive-bayes-algorithm_fig2_325937073. [Accessed: Apr. 1, 2024].

[15] "Pseudo-code of the RF algorithm." ResearchGate. [Online]. Available: https://www.researchgate.net/figure/Pseudo-code-of-the-RF-algorithm-4_fig2_335375894. [Accessed: Apr. 3, 2024].

[16] "Decision Tree Algorithm." Analytics Vidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/. [Accessed: Apr. 1, 2024].

[17] "Naive Bayes Explained." Analytics Vidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/. [Accessed: Apr. 1, 2024].

[18] "Metrics to Evaluate Your Classification Model to Take the Right Decisions." Analytics Vidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/. [Accessed: Apr. 1, 2024].

[19] Google Colab Notebook. [Online]. Available: https://colab.research.google.com/drive/18kyPc6n41F_ACfGL0Sy0oE5eE6_Y7H95?usp=sharing. [Accessed: Apr. 3, 2024].

Acknowledgment

We are thankful to our college **Don Bosco Institute of Technology** for giving us this chance to gain exposure in solving real world problems and acquire practical knowledge and skill sets that will prove to be very crucial to our long-term career prospects. We would take this opportunity to express our sincerest gratitude to our panel members Prof. Prasad Padalkar (H.O.D) and Prof. Shiv Negi (Dean R&D) for their encouragement and guidance, that they gave during our term presentations.

This project, and the research that we undertook, could not have been realized without the utmost support of our Project Guide Prof. Sunantha Krishnan, who guided us every step of the way, starting from the conception of the project, right up to the execution of the finished solution.