A Novel Approach to Proactively Optimize Urban Traffic Systems through Machine Learning

Sage Wang

Received December 02, 2023
Accepted February 07, 2024
Electronic access February 29, 2024

The inefficiency of traffic-light systems causes the average American to spend 122 days of their life waiting at stoplights. New York, one of the most densely populated cities in the United States, currently uses a timer-based traffic light system. In order to adjust the signal timing schedule, NYC conducts a twelve-week study to adapt to the new traffic patterns. This resource intensive analysis can be optimized through machine learning. Previous efforts in the domain of smart traffic control systems propose a real-time traffic signal control system which prioritized lanes with the maximum number of vehicles. However, this system is reactive which prevents further traffic build-up once it has occurred, but does not mitigate the issue. Alternatively, the proposed end-to-end solution takes a proactive approach by modeling on past traffic data. The solution accumulates the dataset from NYC DOT and trains several machine learning models on the metadata extracted from livestream video cameras. The dataset consisted of 4796 samples collected on a Tuesday. Of the seven algorithms tested, Random Forest achieved the lowest test mean squared error of 6.53. This approach to predicting an influx of vehicles is a step toward eliminating the need for manually conducted studies and enables smarter traffic management, emergency response readiness, and environmental benefits.

Introduction

Over the course of their lifetime, the average American dedicates one-third of a year to waiting at traffic lights [1]. This demonstrates a significant inefficiency and introduces an optimization task ripe for machine learning. The growing problem of urban congestion has become a significant concern worldwide, impacting city life by increasing travel times, contributing to environmental pollution, and reducing the overall quality of urban living. Conventionally, there are two categories of traffic light systems: timer-based, which follow fixed intervals, and sensor-based, which use binary weight sensors. Binary weight sensors are embedded in road surfaces to register vehicle presence, but they are limited in their ability to prioritize lanes, because they cannot differentiate between the weight of one or many waiting cars in a lane. This global issue is exemplified in the case of New York City, since it struggles with a high volume of traffic as one of the most densely populated cities in the United States (U.S. Census Bureau) [2]. New York's Department of Transportation (DOT) [3] currently implements a timer-based signal timing schedule between 45 and 120 seconds and mandates a manual twelve-week traffic timer study for any adaptation to the timer lengths (NYC Department of Transportation) [3]. By focusing on New York City, this study aims to address a microcosm of a larger, global challenge, providing insights that can be applicable to urban centers around the world. Rather than sticking to traditional methodologies, this paper proposes moving towards smart-city solutions. A smart city leverages digital technology in its infrastructure to enhance urban life quality, efficiency, and sustainability. This study's aim aligns with the smart city concept, focusing on optimizing traffic light control to improve urban traffic flow. This optimization is crucial for smart cities, as it directly impacts traffic congestion and urban efficiency. Along this vein, the proposed solution employs neural network-based object detection on existing NYC DOT livestream video data to extract metadata and enables machine learning models to predict an estimated number of vehicles at a given time. Prior research in this field used real-time data to determine the traffic light activity. This can be considered a greedy approach, meaning that traffic signals were controlled solely based on the current traffic conditions at any given moment. However, greedy approaches are sub-optimal in a setting which already has traffic. Signals might be changed based on the number of vehicles currently present at an intersection without considering a potential increase in traffic that might be expected shortly. Contrarily, predicting the estimated number of vehicles can mitigate these scenarios by facilitating designated response activities when an influx in transportation is anticipated. This can reduce waiting times, emissions, fuel costs, and potentially prevent accidents.

The ubiquity of camera systems in urban environments presents a unique opportunity for traffic management on a global scale. These cameras, installed all over the world for monitoring speed limits and regulating traffic lights, form an extensive network that can be repurposed for traffic flow optimization. By tapping into this existing infrastructure, our data curation pipeline, which relies on object detection methods, demonstrates

a high degree of replicability and scalability. Contrary to existing research, GPU capabilities are not required on the edge at all. This approach not only enhances the practicality of implementation but also underscores the generalizability of our machine learning model. It can be adapted to various urban settings around the world, leveraging the widespread presence of these camera systems to gather essential data. This adaptation would allow for a more dynamic and responsive approach to traffic management, tailored to the unique patterns and needs of different cities, while avoiding the necessity of installing new, costly hardware at each site.

Literature Review

Vehicle counts have been demonstrated to be an important factor to consider when routing emergency vehicles (Nellore)[4]. This is especially relevant for the approach of scheduling emergency vehicles ahead of time, because our system could greatly contribute to planning out emergency vehicle routes proactively based on predicted volumes of traffic in specific intersections at a specific time of day. Furthermore, in New York City, 29% of citywide emissions come from on-road vehicles. In 2016, vehicle emissions produced 14.88 million metric tons of carbon dioxide (New York City Mayor's Office of Sustainability)[5]. By optimizing traffic signal lengths, we aim to reduce idling time and reduce the carbon footprint of urban transportation. If applied effectively, our model has the potential to make a measurable difference in overall emission levels, aligning with broader environmental goals and contributing to a more sustainable urban future. The integration of our traffic light control system holds significant policy implications. Governments and city planners can better understand traffic patterns, congestion points, and commuter behaviors, allowing urban developers to make data-driven decisions regarding infrastructure development, public transportation networks, and urban layout. This approach not only has the potential to enhance the quality of life for residents but also provides a dynamic framework for urban development that can adapt to changing transportation trends and needs.

Chen and Huang's paper discusses a method for traffic light detection that combines computer vision and machine learning techniques using PCANet (Chen)[6]. They combine color extraction, blob detection, multi-class classification using a pre-trained PCA network, online multi-object tracking, and false positive filtering, resulting in a precision of 95.7% in traffic light detection. However, this approach may not reach such high levels of efficacy in adverse weather conditions, prompting Verma et al. to design an innovative solution to address this problem.

Verma et al. extend this work by incorporating methods designed for traffic light detection in adverse weather conditions[7]. Monocular video cameras enable the authors to address challenging conditions such as low illumination, haze, and fog. Their system incorporates a color pre-processing module to improve red and green region discrimination while mitigating the "blooming effect" often encountered in these scenarios. The utilization of fast radial symmetry transform for traffic light candidate detection and spatiotemporal persistency verification for false positive reduction helped them achieve a precision of 84.8%. However, while enhancing traffic light detection under various weather conditions is crucial for accurate traffic signal recognition, this only addresses the operational reliability of traffic systems. It does not extend to the optimization of traffic flow or congestion management, which is the central aim of our study.

Faraj and Boskany take this a step further by incorporating object detection to traffic light control, using microcontroller circuitry, YOLOv3, and OpenCV to make decisions based on the presence of emergency vehicles and the number of cars on the road[8]. If the system detects lane congestion or an emergency vehicle, the proposed controller overrides the normal timer of the traffic light to expedite the flow of traffic. Their reported metric was that of accuracy in detecting cars and emergency vehicles, which they accomplished to 100% accuracy. However, this system requires the use of a RAM 16GB, GPU Nvidia Geforce 1060 6GB. This will be costly and difficult to implement, given that each traffic light will need its own separate hardware. In contrast, the approach in this study can model an entire system of traffic lights on a CPU.

Natafgi et al. proposed a reinforcement-learning approach based on a simulated 2-weeks worth of data to traffic light simulation using a novel cycle algorithm and a proposed implementation of ultrasonic sensors to detect the entry and exit of vehicles[9]. The authors reported a decrease in average queue length by 62.82%. However, while simulations can provide valuable information, they may not fully capture the complexities of real-world traffic conditions. Additionally, implementing ultrasonic sensors at every traffic light would both be costly and take a significant amount of time to implement. Depending on local labor costs, the cost of installing ultrasonic parking space availability sensors varies between $300 and $500 per space (United States Department of Transportation)[10]. New York City has around 13,543 intersections with traffic signals. (NYC Department of Transportation)[3]. Installing 4 ultrasonic sensors at every intersection would cost upward of $27M. The period of time taken to install all these sensors would also completely disrupt existing traffic patterns. In contrast, this paper offers a solution to avoid the implementation of external hardware, only requiring an image feed from the existing cameras at each street.

This paper builds upon the foundations laid by these prior works in traffic light control and detection. First, a robust dataset is constructed using YOLOv3 for object detection, which enables the analysis of traffic density patterns from real-world traffic camera feeds in an urban environment. This method distinguishes itself from Natafgi et al.'s reliance on simulated data,

as it ensures practical adaptability to the complexities of real-world traffic conditions, thus enhancing traffic operation and road safety. Following the dataset creation, we implement a binning process to categorize the data, making it more manageable and conducive to pattern recognition. Then, various machine learning models are developed to interpret these patterns and predict traffic volumes. Finally, leveraging these predictions, our end-to-end solution proposes adjustments to traffic light signal timings. This proactive strategy, as opposed to the reactive nature of existing systems, not only overcomes limitations seen in previous methods but also significantly contributes to more efficient traffic management, improved emergency response readiness, and environmental sustainability.

Results

The assembled dataset yielded a distribution reflecting the number of vehicles per street, which contained 404 outliers. Each outlier exceeded three-times the standard deviation. Additionally, half the samples which contained no presence of vehicles were determined to be outliers to avoid an overly right-skewed distribution. These 404 outliers were removed from the dataset to create a more accurate and representative analysis. Examining these extreme cases provides insights into the robustness of the models and their ability to handle atypical traffic scenarios. The resulting distribution can be visualized as follows in Figure 1. It is notable that the two streets that contained a majority of the data points with 0 vehicles were East Houston Street at Ave D, and Allan St (1 Ave) and Houston St (B). Additionally, East Houston St at Ave B contained all of the outliers that exceeded three-times the standard deviation. This is likely because there is a montessori located by East Houston St at Ave B where students are being transported to and from school every day, resulting in a higher average traffic volume.

Each data point was categorized based on the time of day when the images were captured. This categorization included four distinct time periods: morning, afternoon, evening, and night. This grouping is illustrated in Figure 2, which shows the traffic flow at a specific intersection (6 Ave and West Houston St) by representing the number of vehicles during these four time periods. This plot represents the mean as the box, and the error bar represents the standard deviation. In the case of 6 Ave and West Houston St, traffic volume appears to peak in the morning and steadily decrease until night. This was the most frequent trend between all 13 examined streets, and it is logical due to rush hours during morning commutes (Bartuska)[11]. As people disperse from work, school, and social activities, traffic gradually diminishes, resulting in fewer cars on the roads as the day progresses.

It can be seen that the standard deviations of the data are non-negligible. Additionally, note that each street has its own mean and standard deviation depending on the volume of traffic

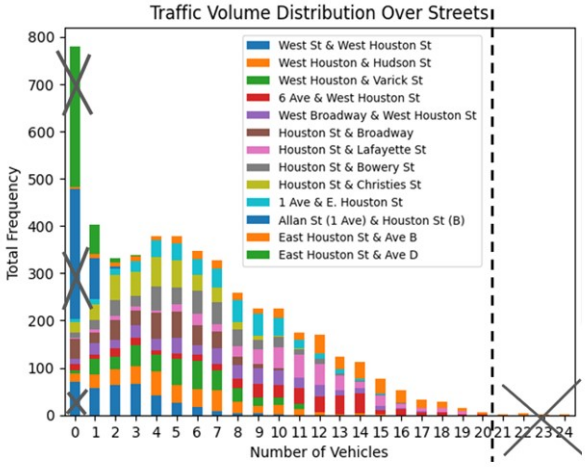

Fig. 1. Traffic Volume Distribution Over Streets

Fig. 1 Traffic Volume Distribution Over Streets.

in that area. This real-world traffic flow provides for a difficult machine learning problem.

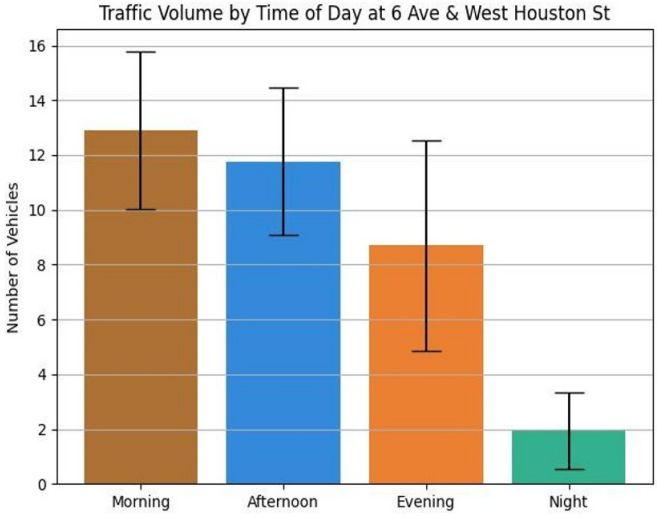

Fig. 2 Traffic Volume by Time of Day at 6 Ave and West Houston St.

The proposed system of making a real-time inference is illustrated by Figure 3. By predicting ahead of time when a particular street will be congested, traffic congestion can be alleviated by adjusting signal lengths accordingly.

Table I presents the mean squared error (MSE), root mean squared error (RMSE), and R squared values of seven different models which have been trained on the train set and evaluated on the test set. The MSE is the average squared difference between $\hat{y}$, the predicted number of vehicles, and y, the ground truth across the dataset. Because the task is a regression problem,

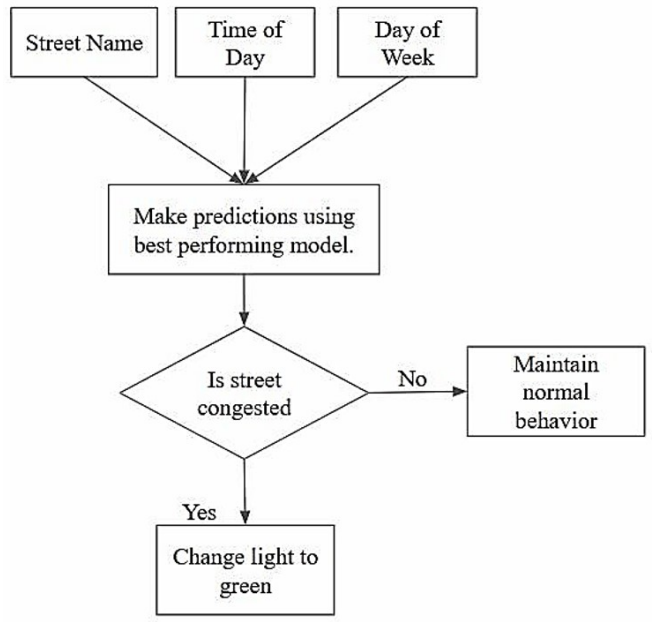

Fig. 3 Inference Diagram

MSE is the best evaluation metric. Not only does it help to evaluate the performance of the models, but also serves as the loss function for the multi-layer perceptron; this incentivizes the network to minimize the difference between its predictions and the target values.

RMSE is the square root of the MSE. It provides a measure of the average magnitude of errors in the same units as the original data. Like MSE, lower RMSE values indicate better model performance. R-squared, also known as the coefficient of determination, measures the proportion of the variance in the dependent variable (y) that is explained by the independent variables in the model. It ranges from 0 to 1: 0 indicates that the model does not explain any of the variance, indicating a poor fit, while 1 indicates that the model perfectly explains all the variance, indicating an excellent fit.

$$MSE = \left(\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right)$$

Random Forest achieved the lowest MSE among the seven algorithms with an MSE of 6.53. Given that traffic flow data is often influenced by many interacting factors that do not have straightforward linear relationships, Random Forest's ability to model these complex interactions likely contributes to its superior performance. Random Forest trains these decision trees in parallel, and each tree decides which features are most critical to making accurate predictions at each node. For instance, a tree might find that certain streets have consistently high traffic at specific times, forming a rule that prioritizes these streets during peak hours. Collectively, these rules form a comprehensive set of guidelines that the Random Forest uses to predict traffic volume. The visual representation in Figure 4 illustrates a decision tree being split by the street number. Despite Random Forest's high efficacy, the algorithm inherently inhibits model interpretability due to the complex combination of trees. Hence, in situations which require a non-black-box approach, it is suboptimal.

As complex gradient boosting methods, XGBoost and Adaboost are more sensitive to individual samples, meaning that they are more prone to having results influenced by outliers. As such, their ability to perform on the small dataset is weaker compared to Random Forest. Boosting algorithms like XGBoost and Adaboost may be more likely to excel in scenarios with very large datasets because the individual samples share more common patterns.

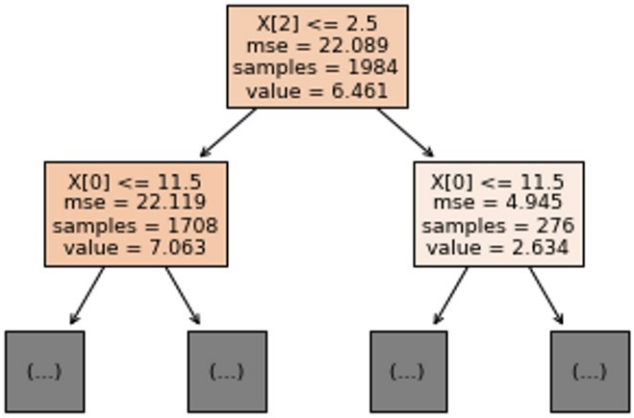

Fig. 4 Random Forest decision tree visualization

Additionally, it can be seen that the ensemble models capture the relationship between the data much better than linear models such as Lasso and Ridge Regression. This is evident from the large difference between their R^2 values, considering that all non-ensemble models had an R^2 value of less than 0.5. Lasso and Ridge assume linear relationships between variables, which limits their capability to capture complexities of the dataset used in this study, which likely contains nonlinear patterns. Ensemble models, on the other hand, do not make such linear assumptions, and are thus better suited to model the relationships present in the data.

Table 2 provides valuable insights into the computational demands of the tested algorithms. The computational efficiency, measured in execution time on the test dataset, varies significantly among the algorithms. Lasso Regression and Ridge Regression demonstrate the highest efficiency with execution times of 0.09 and 0.11 milliseconds, respectively. In contrast,

Table 1 Evaluation Metrics Across Algorithms

Algorithm	Train MSE	Train RMSE	Train R^2	Test MSE	Test RMSE	Test R^2
Random Forest	6.47	2.54	0.70	6.53	2.56	0.70
XGBoost	8.01	2.83	0.64	7.88	2.81	0.64
Adaboost	9.76	3.12	0.55	9.50	3.08	0.57
Polynomial Regression	14.36	3.79	0.34	13.91	3.73	0.35
MLP	17.41	4.17	0.20	16.55	4.07	0.23
Lasso	19.57	4.42	0.10	19.41	4.41	0.10
Ridge	19.57	4.42	0.10	19.42	4.41	0.10

Algorithm	**Execution Time on 1,317 Samples (ms)**
Lasso	0.09
Ridge	0.11
MLP	0.86
Polynomial Regression	0.95
XGBoost	9.51
Random Forest	14.98
Adaboost	116.62

Table 2 Computational efficiency across algorithms

more complex models such as Random Forest and Adaboost take considerably longer, with execution times of 14.98 and 116.62 milliseconds. While being more computationally demanding, these complex models offer much greater accuracy in predicting traffic volumes.

The resources used to train these models were based on a Microsoft Azure CPU, demonstrating the feasibility of the approach without the need for specialized GPU hardware. The absence of a requirement for GPU hardware at runtime not only makes the system more accessible but also more practical to implement in a variety of urban settings. The ability to run these models efficiently on standard CPU hardware ensures that the system can be deployed widely without the need for significant infrastructure changes or investments in high-end computing resources. Out of the 240 data points that Random Forest predicted to have less than 0.5 cars, 72 of these data points had at least one car. This is a false positive rate of 30%. Other than this false positive rate, however, Random Forest's test RMSE of 2.56 cars will have a negligible impact on traffic management. Predictions with an error of only 1 to 2 cars will not skew the overall perception of traffic volume for a particular data point.

Moreover, our methodology presents a substantial advancement over existing traffic management solutions. The use of real-time traffic camera feeds for data collection and the application of sophisticated machine learning models offer a more adaptable and responsive approach to traffic management than traditional systems. By forecasting traffic flow, our solution not only mitigates the immediate impacts of congestion but also contributes to long-term urban planning strategies. This has far-reaching implications, including enhanced emergency response by allowing quicker route optimization for emergency vehicles, reduced environmental impact through lower vehicle emissions, and overall better utilization of urban infrastructure. The scalability and adaptability of this model make it a viable solution for cities worldwide, aiming to tackle the ever-growing challenge of urban traffic management.

Discussion

This paper addresses the prevalent issue of traffic congestion in major urban centers, focusing on New York City. A machine learning-based approach was developed, which tested out seven different regression models to predict traffic volumes at certain times of day. This method marks a significant shift from conventional traffic management techniques, leveraging real-time data from NYC DOT video cameras for predictive modeling. The Random Forest algorithm emerged as the most effective model, demonstrating the best test MSE in predicting traffic volumes.

The potential applications of this research are vast, ranging from improving emergency response routes to contributing to environmental sustainability through reduced vehicle emissions. The adaptability of this methodology to other urban contexts promises a scalable solution to the global challenge of traffic management. Future research could increase the scale of the dataset and expand the samples over various days and geographic locations to allow for more generalizable predictions. Based on the model's current one day dataset, on a Tuesday night at East Houston St and Ave B, 5.22 cars will be predicted with a standard deviation of around 2.5 cars. While the model performs adequately under these specific conditions, applying the same model to different days of the week, such as Wednesday or Saturday, without adjusting for the variability in traffic patterns, could lead to inaccuracies because of its missing understanding of daily variations in traffic flow. Traffic patterns can also vary from one area to another, influenced by local infrastructure and weather conditions.

However, this study does introduce an innovative approach

to traffic flow estimation that leverages existing infrastructure, specifically the network of cameras already installed in traffic intersections worldwide. By integrating data from these cameras, the model could easily be significantly enhanced to provide more accurate traffic pattern analyses. This would not only improve the model's reliability but also its applicability in a variety of settings, thereby making it a more versatile tool for urban planning and traffic management.

Methods

Dataset Creation

The dataset used in this research paper was collected from the NYC DOT Real Time Traffic Information (NYC DOT)[12]. A random sample of 13 streets in NYC were chosen for this study, all of which are connected to East Houston Street. These streets are West St at West Houston St, West Houston at Hudson St, West Houston at Varick St, 6 Ave at West Houston St, West Broadway at West Houston St, Houston St at Broadway, Houston St at Lafayette St, Houston St at Bowery St, Houston St at Christie St, 1 Ave at E. Houston St, Allen St (1 Ave) at Houston St (B), East Houston St at Ave B, and East Houston St at Ave D. A map of these streets is displayed in Figure 5, showing the streets to lie between the East Village and SoHo District, which are both central locations in New York City. Consisting of various shopping centers and eateries, this area has a high traffic density. These locations were selected to represent a diverse range of traffic conditions in New York City, providing a comprehensive perspective on urban traffic dynamics. Researching the traffic patterns and conditions on these streets can provide valuable insights into the transportation dynamics of one of the busiest cities in the world. New York City is known for its high volume of traffic, making it an ideal location to predict traffic patterns. The findings from this research can have implications for other cities around the world facing similar traffic density. The credibility of this data source is notable as the NYC DOT is a governmental entity responsible for monitoring and managing the city's transportation infrastructure and is recognized for maintaining high standards in data collection and accuracy.

The data collection process involved 13 cameras, each capturing an image every 2 seconds for a duration of 5 minutes every other hour. The 5-minute capture period was randomly selected from within the 60 minutes. Randomly selecting this capture period ensures that the collected data is representative of various traffic conditions throughout the hour, reducing the risk of bias that might result from always capturing data during one specific time interval. This continuous capture schedule was maintained for 12 hours of the day, resulting in a total of 4,796 images. These frequent data collection intervals provide a comprehensive representation of the traffic conditions at different hours of the day across the selected locations. YOLOv3 was applied to each image sample in the dataset, in which the number of vehicles present was extracted and the corresponding street name, and the date and time of collection was recorded[13]. This information facilitates the construction of a time series dataset, enabling the analysis of traffic patterns, vehicle density, and other relevant factors over time. This data curation process is represented by Figure 4.

An important aspect of the features of this time series dataset that required careful consideration was the metadata associated with each traffic image. Originally, the exact hour, minute, and second of each image was captured. However, these features proved to be uninformative due to the limited sample size of image data available for each unique timestamp. Consequently, it posed difficulties for the machine learning models to generalize patterns effectively, as it required a vast amount of data for every precise timestamp, which was difficult to collect. Capturing images on a minute-by-minute basis could improve authorities' responsiveness to dynamic conditions like accidents or severe weather by providing precise timestamps detailing how traffic was impacted throughout each minute. However, this approach lacked the necessary practicality for building a robust machine learning model for traffic density prediction.

To solve this, the dataset was transformed to incorporate binned features. Binning is a data preprocessing technique to categorize continuous or numerical data into discrete intervals or bins. It involves dividing the range of data values into fixed or variable-width bins and assigning each data point to the appropriate bin based on its value. Binning is often used to simplify complex datasets, reduce noise, and make data more manageable for analysis or visualization. In this study, images were categorized based on whether they were captured on weekends or weekdays and the general time of day - morning, afternoon, evening, or night. This adjustment enabled the machine learning model to better capture and generalize the underlying traffic dynamics.

The dataset was split into a 70 percent train set, and a 30 percent test set. Figure 7 illustrates how this dataset was trained and evaluated on a set of seven algorithms: Random Forest, XGBoost, Adaboost, Polynomial Regression, Ridge Regression, Lasso Regression, and Multi-Layer Perceptron. GridSearchCV was used to identify the optimal hyperparameters for these algorithms before evaluating their performance on the test dataset. Table 3 provides the details of the chosen hyperparameters and their ranges for each model.

Proposed Method

1. ***Polynomial Regression:*** In contrast to linear regression, which uses a straight line (a first-degree polynomial) to model a linear relationship between the data features x and the target y, polynomial regression fits a nonlinear relationship. It accomplishes this by using an nth degree

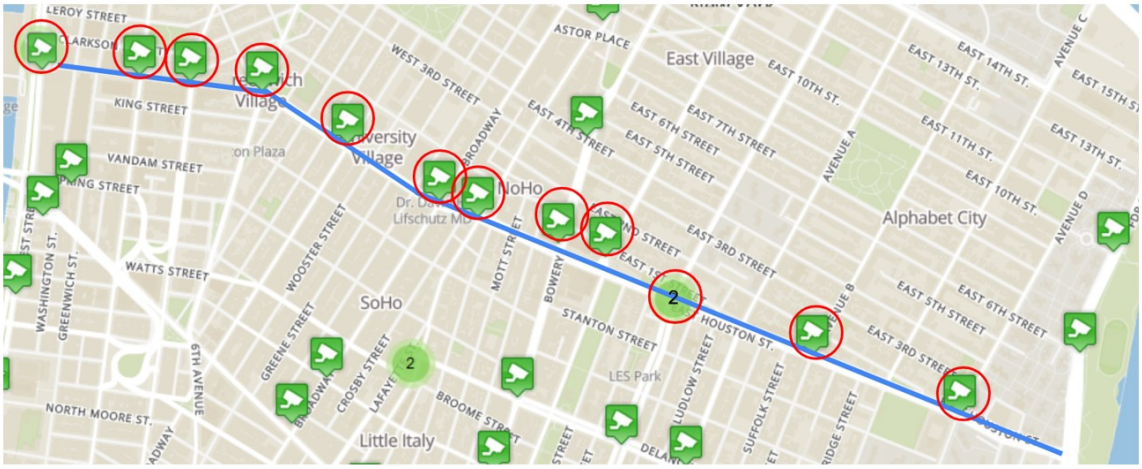

Fig. 5 The 13 streets examined in the dataset lie between the East Village and SoHo District, starting at Pier 40 and running to the East River.

Algorithm	Chosen Hyperparameters	Ranges
Random Forest	max_depth: None, min_samples_split: 2, n_estimators: 100	n_estimators: [10, 20, 50, 100], max_depth: [None, 10, 20, 30], min_samples_split: [2, 10, 20, 60]
XGBoost	n_estimators: 100, max_depth: 3, learning_rate: 0.01	n_estimators: [50, 100, 150], max_depth: [3, 4, 5], learning_rate: [0.01, 0.1, 0.2]
Adaboost	n_estimators: 100, base_estimator__max_depth: 3	n_estimators: [50, 100, 150], base_estimator__max_depth: [1, 2, 3]
Polynomial Regression	polynomialfeatures__degree: 5, polynomialfeatures__include_bias: True, polynomialfeatures__interaction_only: False	polynomialfeatures__degree: [2, 3, 4, 5, 6], polynomialfeatures__include_bias: [True, False], polynomialfeatures__interaction_only: [True, False]
MLP	activation: relu, alpha: 0.0001, early_stopping: True, hidden_layer_sizes: (100,), learning_rate: constant, max_iter: 200	hidden_layer_sizes: [(100,), (50, 50)], activation: [relu, tanh], alpha: [0.0001, 0.001], learning_rate: [constant], max_iter: [100, 200], early_stopping: [True, False]
Lasso	alpha: 0.05, max_iter: 2000, positive: False, selection: cyclic	alpha: [0.1, 0.5, 0.05, 1], max_iter: [1000, 2000, 5000], positive: [True, False], selection: [random, cyclic]
Ridge	alpha: 1, solver: svd	alpha: [0.1, 0.5, 0.05, 1], solver: [svd, cholesky, lsqr, sparse_cg, sag, saga, auto]

Table 3 GridSearchCV chosen hyperparameters and ranges for each algorithm

polynomial. This polynomial function can be curved, allowing for a better fit to data that exhibits nonlinearity. However, polynomial regression may not be suitable for all types of datasets because it relies on certain assumptions, particularly the normality of the error term. Hence, it will not be generalizable to a dataset which deviates from the original. Polynomial Regression was chosen to be tested because it is useful for modeling the non-linear relationship that likely exists in traffic data. It can provide a more nuanced fit compared to simple linear regression.

2. ***Random Forest:*** In Random Forest Regression, a decision tree recursively splits data into subsets based on the values of input features, to make sequential decisions leading to a

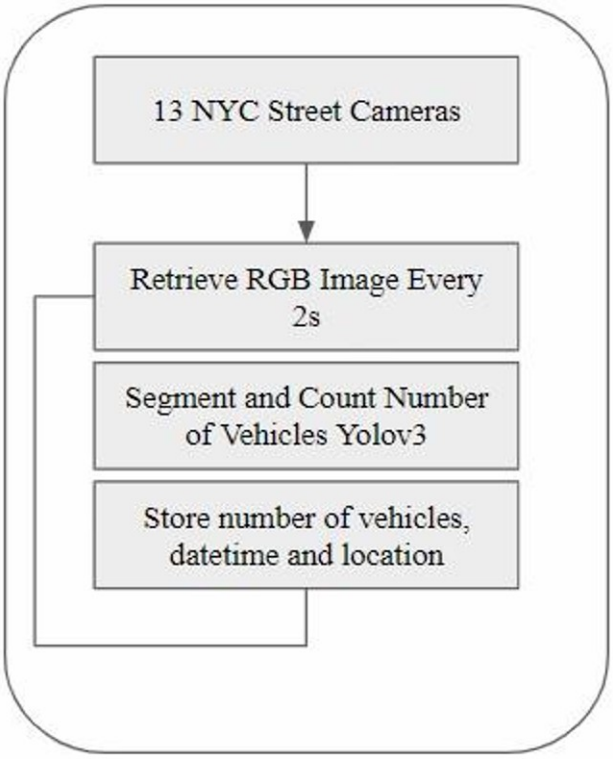

Fig. 6 Dataset Curation Pipeline

final decision. This rule-based learning approach involves creating decision trees that learn from the data and create decision rules to make predictions. Random Forest constructs multiple decision trees during the training phase and then combines their predictions to get a more accurate result. The final prediction in a Random Forest is determined through a process called ”voting.” In a classification task, each tree ”votes” for a class, and the class with the most votes becomes the predicted class. To maximize information gain, each decision tree in the Random Forest selects the features that are most informative at each split, ensuring that the trees collectively provide a diverse set of rules to improve predictive performance. In regression tasks, the individual tree predictions are averaged to produce the final prediction. By aggregating the outputs of multiple decision trees, Random Forest leverages the wisdom of the crowd, reducing the risk of overfitting and increasing the overall predictive power of the model. Known for its robustness and ability to handle complex data relationships, Random Forest was tested because it is well suited for regression problems, which aligns with the task of predicting traffic volumes.

The next two regression models tested, Lasso Regression and Ridge Regression, are both examples of regularized regression. Lasso and Ridge Regression combine feature selection and regularization techniques. Feature selection involves choosing a subset of the most relevant input features from the available set of independent variables, aiming to exclude less informative or redundant attributes. This process helps simplify models, reduce dimensionality, and improve model generalization. Regularization, on the other hand, is a method that adds penalty terms to the model's coefficients during the training process. It encourages the model to find a balance between fitting the training data and preventing overfitting by controlling the complexity of the model.

Regularized regression works by finding the coefficients for a linear model that minimize the sum of squared differences between the observed target values and the predicted values generated by the linear model. In other words, it aims to fit a linear equation to the data in a way that best explains the relationship between the independent variables and the dependent variable. Lasso and Ridge Regression were tested for this study because they both include regularization parameters that help prevent overfitting. Overfitting is a common concern in machine learning, particularly in complex datasets such as a traffic congestion dataset where a model might perform well on training data but poorly on unseen data. The regularization in Lasso and Ridge could help to build a model that generalizes better to new, unseen data.

Generalizability is one classical goal of machine learning. Efforts towards goal have been introduced through regularized regression which enforces a constraint on the coefficients of the model. This constraint drives the weights of the model towards zero and can potentially remove insignificant feature columns which do not aid the model's learning.

3. ***Ridge (L2) Regression:*** Ridge Regression is a linear regression that addresses overfitting issues by using L2 regularization. Ridge Regression adds a penalty term to the linear regression cost function that is proportional to the square of the magnitude of the regression coefficients. This penalty encourages the model to reduce the magnitude of all coefficients, effectively shrinking them toward zero, but not exactly to zero as in Lasso. Encouraging the reduction of the magnitude of regression coefficients towards zero because this regularization technique seeks to strike a balance between fitting the training data well and maintaining model generalizability. By shrinking the coefficients, Ridge Regression helps prevent overfitting, making the model less sensitive to the noise in the training data and enhancing its ability to generalize to new, unseen data.

   Ridge Regression is particularly useful when dealing with

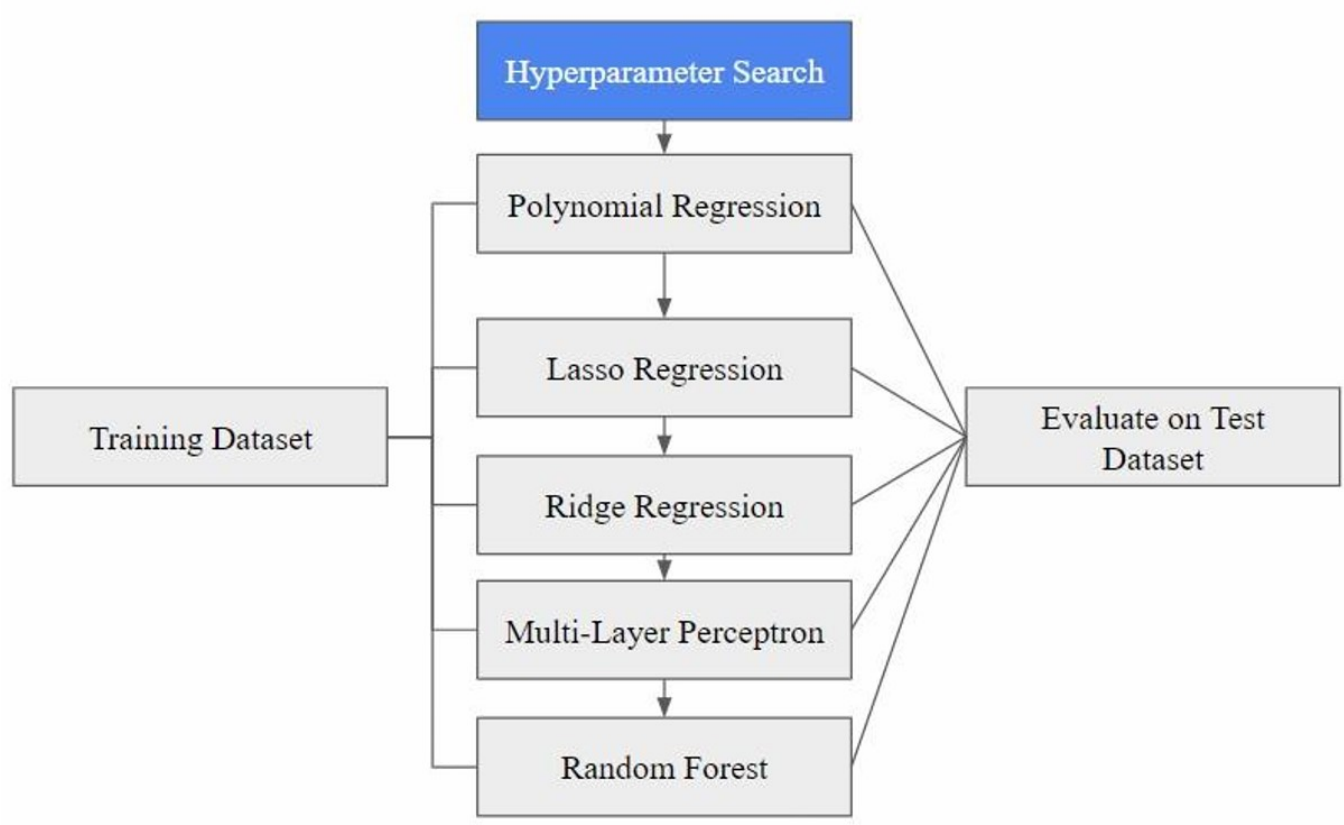

Fig. 7 Machine Learning Pipeline

datasets where multicollinearity exists, which means that some predictor variables are highly correlated. By penalizing the sum of squared coefficients, Ridge Regression mitigates multicollinearity by redistributing the influence among correlated features, leading to more stable and interpretable models. Similar to Lasso, the strength of the penalty in Ridge Regression is controlled by a hyperparameter called lambda, which determines the amount of regularization applied.

$$\hat{\beta}_{\text{ridge}} = \underset{\beta}{\text{argmin}} \left[\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

4. ***Lasso (L1) Regression:*** Another linear regression technique used for feature selection and regularization is Lasso Regression. Lasso Regression adds a penalty term to the linear regression cost function, which is based on the absolute values of the regression coefficients. This penalty encourages the model to shrink or eliminate certain feature coefficients, effectively performing feature selection by setting some coefficients to zero. Lasso helps prevent overfitting by reducing the complexity of the model, making it particularly useful when dealing with high-dimensional datasets with many potentially irrelevant features. The strength of the penalty is controlled by a hyperparameter called lambda, which determines the balance between the model's fit to the data and the degree of regularization.

$$\hat{\beta}_{\text{lasso}} = \underset{\beta}{\text{argmin}} \left[\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right]$$

5. ***Multi-layer Perceptrons:*** The next algorithm tested was Multi-layer Perceptrons. A Multi-layer Perceptron consists of multiple layers of interconnected nodes, including an input layer, one or more hidden layers, and an output layer. Each node applies an activation function to its input, allowing the model to learn complex non-linear mappings. The model learns through backpropagation, where it adjusts its weights to minimize the difference between its predictions and the correct values. MLPs can handle large-scale and high-dimensional data but require sufficient training data to avoid overfitting. MLPs were chosen to be tested be-

cause they are known for their ability to generalize from the training data to unseen data. This is critical in traffic prediction scenarios, where the model needs to perform well not just on historical data but also on future, unseen traffic conditions.

The final two algorithms tested were AdaBoost and XGBoost, which are both boosting algorithms. Boosting algorithms are machine learning techniques that improve predictive performance by combining multiple simple models into a highly accurate and robust model.

6. ***XGBoost:*** XGBoost stands for Extreme Gradient Boosting, an advanced implementation of gradient boosting algorithm. It is renowned for its efficiency, flexibility, and portability. XGBoost builds an ensemble of decision trees in a sequential manner, where each new tree attempts to correct the errors made by the previous ones. It uses a gradient descent algorithm to minimize the loss when adding new models. This process involves not only fitting the new model to the data but also optimizing the ensemble's performance as a whole. XGBoost includes several features that enhance its performance and accuracy, such as handling missing data, providing several regularization techniques to prevent overfitting, and tree pruning. XGBoost was tested for its high performance and speed, especially in large datasets. Additionally, XGBoost's regularization features help in enhancing the model's generalization capability, crucial for predicting traffic flows accurately.

7. ***Adaboost:*** Adaboost, short for Adaptive Boosting, is a popular ensemble machine learning algorithm, primarily used for classification problems, but can also be adapted for regression tasks. It works by combining multiple weak learners, typically simple decision trees, to create a strong, robust model. Each weak learner contributes to the final model with a weighted vote, where the weights are adjusted iteratively. In each iteration, Adaboost changes the weights of training samples based on the previous learner's accuracy; it gives more weight to misclassified instances, encouraging the algorithm to focus on the harder cases in subsequent iterations. This process continues until either the specified number of weak learners are added or no further improvements can be made. Adaboost was tested due to its strength in reducing bias and variance, leading to a generally well-performing model. The adaptive nature of Adaboost makes it suitable for traffic data, which often involves complex, variable patterns that a single model might struggle to capture.

Acknowledgements

I would like to thank my mentor, Samuel Lefcourt, for guiding me through this project, helping me with the research process, and providing his invaluable feedback on the paper.

References

1 S. Crow, *You'll Spend This Much of Your Life Waiting at Red Lights*, //bestlifeonline.com/red-lights, Best Life Online. https.

2 U. Bureau, *Explore Census Data*, https://www.census.gov/quickfacts/fact/table/newyorkcitynewyork.

3 N. Transportation, *Traffic Signals*, https://www.nyc.gov/html/dot/html/infrastructure/signals.shtml.

4 K. Nellore and G. Hancke, *Sensors*, **16**, 1892.

5 N. Y. C. M. O. Sustainability, *Inventory of New York City Greenhouse Gas Emissions in 2016 the City of New York Mayor Bill de Blasio*, https://www.nyc.gov/assets/sustainability/downloads/pdf/publications/GHG%20Inventory%20Report%20Emission%20Year%202016.pdf.

6 Z. Chen and X. Huang, *IEEE Intelligent Transportation Systems Magazine*, **8**, 28–42.

7 P. Verma and V. Singh, *International Journal of Science, Technology Engineering*, **3**, 99–102.

8 M. Faraj and N. Boskany, *UHD Journal of Science and Technology*, **4**, 123–131.

9 M. Natafgi, M. Osman, A. Haidar and L. Hamandi, 2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET, p. 1–6.

10 *United States Department of Transportation*, https://www.itskrs.its.dot.gov/node/209110.

11 L. Bartuska, V. Biba and R. Kampf, International Conference on Traffic and Transport Engineering (ICTTE.

12 D. NYC, *Real Time Traffic Information*, https://webcams.nyctmc.org/map.

13 J. Redmon and A. Farhadi, *YOLOv3: An incremental improvement*, arXiv preprint arXiv:1804.02767.