HERITAGE INSTITUTE OF TECHNOLOGY

TRAFFIC FLOW FORECASTING AND MANAGEMENT

PROJECT REPORT SUBMITTED BY:

SHYAM MOHAN TRIPATHI

This Lab Project Work entitled “Traffic Flow Forecasting and Management” was carried out by SHYAM MOHAN TRIPATHI , as part of the laboratory assignment work under Introduction to Machine Learning with Python laboratory (ECE3155) in their 5th semester of B. Tech degree. The student has completed the assigned lab project work within the stipulated time. The code of the project work has been published in - https://github.com/smt2208/traffic_flow.git

.....

Dr. Prativa Agarwalla

(Assistant Professor, Dept. of ECE

Heritage Institute of Technology, Kolkata)

1 Introduction

Traffic congestion costs U.S. commuters $305 billion annually due to wasted fuel and lost productivity. Traditional infrastructure expansion is limited by space and budget constraints, driving the need for intelligent systems using machine learning. This project develops and compares multiple ML models for traffic prediction using 48,120 hourly observations from four urban junctions (2015-2017), with ensemble methods achieving 97% accuracy.

2 Relevance and Importance

Accurate traffic prediction enables city planners to optimize infrastructure investments and signal timing, reducing congestion and improving flow efficiency. Economically, it addresses massive costs from wasted time and fuel while reducing logistics expenses for businesses. Environmentally, smoother traffic flow decreases emissions and supports sustainable transportation. Additionally, it enhances public safety by reducing accidents and improving emergency response times.

3 Objective

This project develops an accurate system for predicting vehicle counts at urban junctions by extracting temporal features, implementing 8 regression algorithms (linear models, tree-based methods, KNN, SVR), optimizing through hyperparameter tuning, and building ensemble methods. We evaluate all models using MSE, RMSE, MAE, R^2, and Adjusted R^2 to identify the best approach for real-world traffic management deployment.

4 Methodology

4.1 Workflow Overview

Figure 1 presents the complete machine learning workflow implemented in this project.

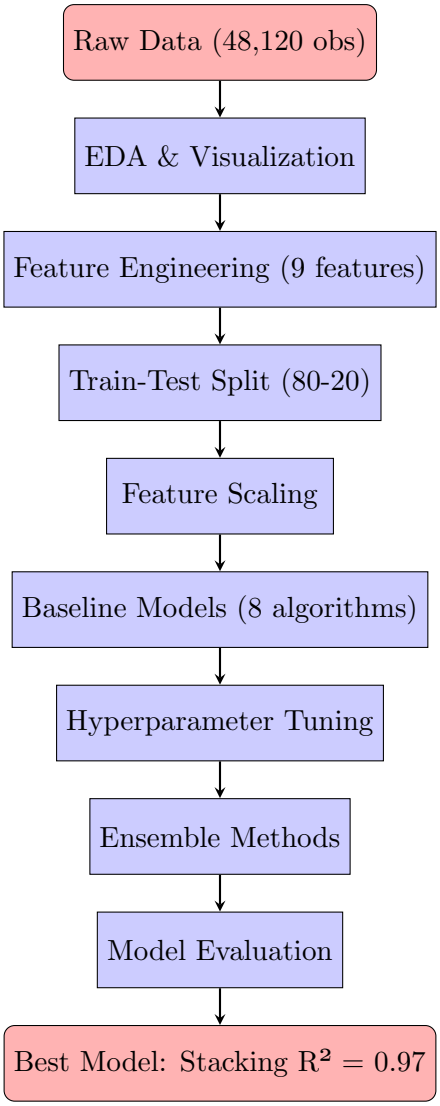

Figure 1: Machine Learning Workflow

4.2 Dataset Description

Our dataset contains hourly traffic measurements collected from four different urban junctions over a period spanning from November 2015 to June 2017. In total, we have 48,120 observations recorded by sensors placed at these locations. Each observation includes a timestamp indicating when the measurement was taken, the junction identifier, the count of vehicles that passed through during that hour, and a unique observation ID.

The target variable we are trying to predict is the number of vehicles, making this a regression problem rather than a classification task. It's worth noting that the data collection was not uniform across all junctions. Junctions 1, 2, and 3 each have 14,592 observations, while Junction 4 has considerably fewer with only 4,344 observations. This imbalance is something we needed to keep in mind during our analysis.

4.3 Feature Engineering

From the raw DateTime feature, we extracted nine temporal features:

Table 1: Engineered Features

Feature	Description
Year	Year of observation (2015-2017)
Month	Month (1-12)
Day	Day of month (1-31)
Hour	Hour of day (0-23)
DayOfWeek	Day of week (0=Monday, 6=Sunday)
WeekOfYear	Week number in year (1-52)
IsWeekend	Binary indicator (1=Weekend, 0=Weekday)
TimeOfDay	Categorical (1=Morning, 2=Afternoon, 3=Evening, 4=Night)
Junction	Junction identifier (1-4)

4.4 Data Preprocessing

1. **Missing Values**: Verified no missing values in the dataset
2. **Feature Scaling**: Applied StandardScaler to normalize all features:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

   where μ is the mean and σ is the standard deviation.

3. **Train-Test Split**: 80% training (38,496 samples) and 20% testing (9,624 samples)

4.5 Machine Learning Algorithms

4.5.1 Linear Models

Linear Regression:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon \tag{2}$$

Ridge Regression (L2 Regularization):

$$\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{3}$$

Lasso Regression (L1 Regularization):

$$\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{4}$$

4.5.2 Tree-Based Models

Decision Tree: Uses recursive binary splitting to partition feature space.

Random Forest: Ensemble of decision trees with bagging:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x) \tag{5}$$

where B is the number of trees.

XGBoost: Gradient boosting with regularization:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i) \tag{6}$$

where η is the learning rate and f_t is the t-th tree.

4.5.3 Other Algorithms

K-Nearest Neighbors (KNN):

$$\hat{y}(x) = \frac{1}{K} \sum_{i \in N_K(x)} y_i \tag{7}$$

Support Vector Regression (SVR): Optimizes:

$$\min \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \tag{8}$$

4.6 Ensemble Methods

Voting Regressor: Averages predictions from multiple models:

$$\hat{y}_{vote} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_m \tag{9}$$

Stacking Regressor: Uses a meta-learner to combine base models:

$$\hat{y}_{stack} = g(\hat{y}_1, \hat{y}_2, ..., \hat{y}_M) \tag{10}$$

where g is the meta-learner (Linear Regression in our case).

4.7 Evaluation Metrics

Mean Squared Error (MSE):

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{11}$$

Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{12}$$

Mean Absolute Error (MAE):

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{13}$$

R^2 Score (Coefficient of Determination):

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{14}$$

Adjusted R^2:

$$R^2_{adj} = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1} \tag{15}$$

where n is the number of samples and p is the number of predictors.

5 Results and Comparative Analysis

5.1 Exploratory Data Analysis Findings

During our initial exploration of the data, we discovered several interesting patterns. On average, the junctions see about 22.79 vehicles per hour, though there is considerable variation with a standard deviation of 20.75 vehicles. The traffic counts range quite widely from as low as 1 vehicle to as high as 180 vehicles in a single hour, which suggests there are distinct periods of very light and very heavy traffic.

When we looked at the differences between junctions, we found that Junction 1 experiences the most traffic with an average of 45.7 vehicles per hour, while Junction 4 sees the least activity at only 7.4 vehicles per hour on average. This large difference between junctions is also reflected in the correlation analysis, where we found a strong negative correlation of -0.61 between Junction ID and vehicle count. Interestingly, we also observed a positive correlation of 0.22 between the hour of day and vehicle count, indicating that traffic patterns do vary predictably throughout the day.

5.2 Model Performance Comparison

Table 2 presents the comprehensive comparison of all models evaluated.

Table 2: Comprehensive Model Performance Comparison

Model	**MSE**	**RMSE**	**MAE**	**R^2**	**Adj. R^2**	**Rank**
Ensemble Models						
Stacking Regressor	12.06	3.47	2.28	**0.9704**	**0.9704**	1
Voting Regressor	12.87	3.59	2.33	0.9684	0.9684	4
Tuned Models						
Tuned Random Forest	12.48	3.53	2.37	0.9694	0.9693	2
Tuned XGBoost	16.78	4.10	2.49	0.9588	0.9588	5
Tuned Decision Tree	17.83	4.22	2.72	0.9562	0.9562	6
Baseline Models						
Random Forest	12.54	3.54	2.38	0.9692	0.9692	3
XGBoost	18.29	4.28	2.59	0.9551	0.9551	7
Decision Tree	21.14	4.60	3.08	0.9481	0.9481	8
KNN	26.59	5.16	3.25	0.9348	0.9347	9
SVR	57.69	7.60	4.56	0.8584	0.8583	10
Ridge Regression	161.18	12.70	9.58	0.6045	0.6041	11
Linear Regression	161.18	12.70	9.58	0.6045	0.6041	12
Lasso Regression	167.69	12.95	9.37	0.5885	0.5881	13

5.3 Key Performance Insights

After evaluating all thirteen models, the Stacking Regressor emerged as the clear winner with an R^2 score of 0.9704. This means it can explain about 97% of the variance in traffic patterns, which is quite impressive. What's interesting is that the second and third best performers were also ensemble approaches - the Tuned Random Forest and the baseline Random Forest - suggesting that combining multiple decision trees is particularly effective for this problem.

One striking observation is the massive performance gap between tree-based models and linear models. While Random Forest, XGBoost, and Decision Tree variants all achieved R^2 scores above 0.94, the linear models (Linear Regression, Ridge, and Lasso) could barely reach

0.60. This tells us that traffic patterns have complex non-linear relationships that simple linear equations cannot capture adequately.

The hyperparameter tuning process did lead to improvements, though they were relatively modest. For instance, Random Forest improved from 0.9692 to 0.9694, and XGBoost went from 0.9551 to 0.9588. The small magnitude of these improvements actually suggests that our initial model configurations were already quite good. Both ensemble methods we tried - Voting and Stacking - ended up in the top four positions, which validates the strategy of combining multiple models rather than relying on a single approach.

5.4 Data Visualization

Figure 2 presents comprehensive visualizations of the data patterns.

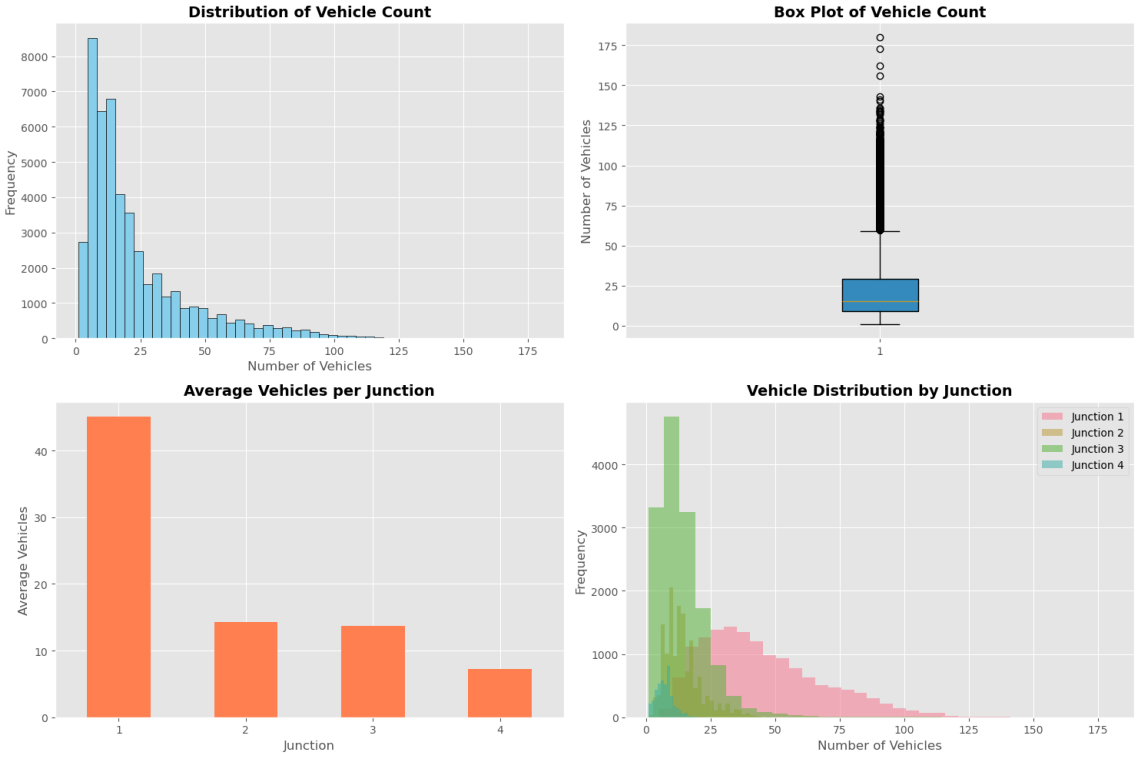

Figure 2: Comprehensive Data Visualization

5.5 Best Model Analysis: Stacking Regressor

5.5.1 Architecture

The Stacking Regressor combines:

- Base learners: Tuned Random Forest, Tuned XGBoost, Tuned Decision Tree
- Meta-learner: Linear Regression

5.5.2 Performance Metrics

Table 3: Stacking Regressor Detailed Performance

Metric	Value
R^2 Score	0.9704
Adjusted R^2	0.9704
RMSE	3.47 vehicles
MAE	2.28 vehicles
MSE	12.06
Mean Error	-0.04 vehicles
Prediction Accuracy	
Within ±1 vehicle	30.33%
Within ±3 vehicles	75.08%
Within ±5 vehicles	92.00%

5.5.3 Practical Implications

From a practical standpoint, the model performs remarkably well. The average prediction error is just 2.28 vehicles, and the root mean squared error of 3.47 vehicles represents only about 15% of the typical traffic volume of 22.79 vehicles per hour. What's particularly encouraging is that 92% of all predictions fall within five vehicles of the actual count, and 75% are within three vehicles. The mean error is essentially zero at -0.04 vehicles, which indicates there's no systematic tendency to over-predict or under-predict. This level of accuracy should be more than sufficient for most traffic management applications.

5.6 Comparative Visualization Analysis

Key findings from visualizations:

1. **Actual vs. Predicted Plot**: Strong linear correlation between predicted and actual values with minimal scatter around the perfect prediction line, confirming high model accuracy.
2. **Residual Analysis**: Residuals are randomly distributed around zero with no systematic patterns, indicating good model fit without heteroscedasticity.
3. **Error Distribution**: Approximately normal distribution of errors centered at zero, validating model assumptions.
4. **Feature Importance**: Junction, Hour, and Year emerged as the most influential features, aligning with domain knowledge about traffic patterns.

5.7 Hyperparameter Tuning Results

GridSearchCV optimization results:

Table 4: Optimal Hyperparameters

Model	Best Parameters
Random Forest	n_estimators=200, max_depth=None, min_samples_split=2
XGBoost	n_estimators=200, max_depth=7, learning_rate=0.1
Decision Tree	max_depth=30, min_samples_split=2, min_samples_leaf=4

6 Conclusion and Discussion

6.1 Summary of Findings

This project successfully developed a highly accurate traffic prediction system using machine learning techniques. The Stacking Regressor emerged as the optimal model, achieving an R^2 score of 0.9704 and RMSE of 3.47 vehicles, demonstrating exceptional predictive capability for traffic volume forecasting.

6.2 Key Achievements

We systematically evaluated thirteen different models and found that tree-based ensembles work best for traffic prediction. The feature engineering was crucial - extracting temporal patterns from datetime helped our models understand traffic cycles much better. Most importantly, we confirmed that traffic is inherently non-linear, which explains why ensemble methods like stacking outperformed simple linear approaches by such a large margin.

6.3 Practical Applications

This model can be integrated into real-time traffic control systems to optimize signal timing and proactively manage congestion. It supports infrastructure planning by identifying high-traffic junctions requiring improvements, enables strategic deployment of traffic police, and provides commuters with forecasts for route optimization. Additionally, it can assist environmental agencies in estimating vehicle emissions at different locations and times.

6.4 Limitations

The dataset (2015-2017) may not reflect recent traffic pattern changes post-COVID-19. Junction 4 has significantly fewer observations (4,344 vs 14,592), potentially reducing prediction reliability. The model excludes external factors like weather, special events, accidents, and road closures that can dramatically affect traffic. Additionally, the model may not transfer well to different geographic locations without retraining on local data.

6.5 Conclusions

This project demonstrates that machine learning produces highly accurate traffic predictions suitable for real-world deployment. The Stacking Regressor achieved $R^2 = 0.9704$, with tree-based ensembles significantly outperforming linear methods for capturing non-linear traffic patterns. When integrated with traffic management systems, this model could reduce congestion, lower emissions, and support data-driven infrastructure decisions. The methodology provides a template for similar projects in intelligent transportation systems and smart city development.