Traffic Situation Prediction Using Machine Learning Algorithms: A Comparative Study of K-Nearest Neighbors, Decision Trees, and Random Forest

1st Tazower Rahman Sowad
Department of CSE
BRAC University
Dhaka, Bangladesh
tazower.rahman.sowad@g.bracu.ac.bd

2nd Kazi Abrab Hossain
Department of CSE
BRAC University
Dhaka, Bangladesh
kazi.abrab.hossain@g.bracu.ac.bd

3rd Md. Abrar Rahman Shafin
Department of CSE
BRAC University
Dhaka, Bangladesh
abrar.rahman.shafin@g.bracu.ac.bd

4th Md. Fardin Hassan Tamim
Department of CSE
BRAC University
Dhaka, Bangladesh
fardin.hassan.tamim@g.bracu.ac.bd

***Abstract*—This paper presents a comprehensive analysis of traffic situation prediction using machine learning algorithms. We implement and compare three different classification algorithms: K-Nearest Neighbors (KNN), Decision Trees, and Random Forest to predict traffic conditions based on vehicle counts and temporal features. The dataset contains 2,976 traffic records with features including time, date, day of the week, and counts of different vehicle types (cars, bikes, buses, trucks). Our preprocessing pipeline includes data cleaning, feature engineering, handling class imbalance using SMOTE, and feature scaling. The Random Forest algorithm achieved the highest accuracy of 88.79%, followed by Decision Trees (85.34%) and KNN (82.93%). The results demonstrate the effectiveness of ensemble methods for traffic prediction tasks and provide insights into the relative performance of different machine learning approaches for this domain.**

***Index Terms*—traffic prediction, machine learning, classification, KNN, decision trees, random forest, SMOTE, feature engineering**

I. Introduction

Traffic management and prediction have become critical components of modern urban planning and smart city initiatives. With increasing urbanization and vehicle density, accurate traffic situation prediction can significantly improve traffic flow, reduce congestion, and enhance overall transportation efficiency. Machine learning algorithms have emerged as powerful tools for analyzing traffic patterns and predicting future traffic conditions based on historical data.

This study focuses on predicting traffic situations using vehicle count data and temporal features. We implement and compare three machine learning algorithms: K-Nearest Neighbors (KNN), Decision Trees, and Random Forest to classify traffic conditions into four categories: heavy, high, low, and normal. The comparative analysis provides insights into the effectiveness of different algorithmic approaches for traffic prediction tasks.

The main contributions of this work include:

- Comprehensive data preprocessing pipeline for traffic data
- Implementation and evaluation of three different machine learning algorithms
- Analysis of class imbalance handling using SMOTE
- Feature engineering techniques including temporal feature extraction and quantization
- Comparative performance analysis with detailed metrics

II. Related Work

Traffic prediction has been extensively studied in the literature using various machine learning approaches. Previous research has explored different algorithms including neural networks, support vector machines, and ensemble methods for traffic flow prediction. However, limited work has been done on comparing traditional machine learning algorithms specifically for traffic situation classification based on vehicle counts.

Blagus and Lusa [1] conducted a comprehensive study on SMOTE (Synthetic Minority Oversampling Technique) for handling class-imbalanced data in high-dimensional datasets. Their research demonstrated that SMOTE effectively addresses class imbalance by generating synthetic samples for minority classes, significantly improving classification performance in imbalanced datasets. Their findings support our use of SMOTE to handle the class imbalance in our traffic situation classification problem, where certain traffic conditions (such as ”normal” traffic) were more prevalent than others.

Shaik and Srinivasan [2] provided a comprehensive survey on Random Forest ensembles in classification models. Their study highlighted the effectiveness of Random Forest as an ensemble learning method that combines multiple decision trees to improve classification accuracy and reduce overfitting.

They demonstrated that Random Forest consistently outperforms individual classifiers across various domains, which aligns with our findings where Random Forest achieved the highest accuracy (88.79%) among the three algorithms tested in our traffic prediction study.

Guo et al. [3] presented a comprehensive study on KNN (K-Nearest Neighbors) model-based approach in classification. Their research demonstrated the effectiveness of KNN as a non-parametric classification algorithm that relies on distance-based similarity measures. They highlighted that KNN performance is sensitive to feature scaling and the choice of k neighbors, which explains why our KNN model achieved baseline performance (82.93% accuracy) and required proper feature scaling to achieve optimal results in our traffic prediction study.

Suthaharan [4] provided an in-depth analysis of Decision Tree learning algorithms for big data classification. Their study demonstrated that Decision Trees offer good interpretability and balanced performance across various classification tasks. They emphasized that Decision Trees can handle both numerical and categorical features effectively, which supports our use of Decision Trees in traffic situation classification where we have mixed feature types including temporal and vehicle count data. Our results showing Decision Trees achieving 85.34% accuracy validate their findings on the algorithm's consistent performance.

III. Dataset and Methodology

A. Dataset Description

The dataset consists of 2,976 traffic records collected over time, containing the following features:

- **Time**: Hour and minute of the day
- **Date**: Day of the month
- **Day of the week**: Categorical variable (Monday through Sunday)
- **Vehicle Counts**: Separate counts for cars, bikes, buses, and trucks
- **Traffic Situation**: Target variable with four classes (heavy, high, low, normal)

The initial dataset contained 2,976 records, which was reduced to 2,896 after outlier removal and data cleaning.

B. Data Preprocessing

1) Data Cleaning: The preprocessing pipeline included several key steps:

1) **Duplicate Removal**: Identified and removed 1 duplicate record
2) **Missing Value Handling**:
   - CarCount: Filled with mean value
   - Day of the week: Filled with mode value
3) **Data Standardization**: Corrected inconsistent day names (e.g., ”Tue” $\rightarrow$ ”Tuesday”, ”Thrsday” $\rightarrow$ ”Thursday”)
4) **Negative Value Handling**: Replaced negative BusCount values with zero

Figure 1 shows the missing data heatmap, which helped identify patterns in missing values across the dataset.

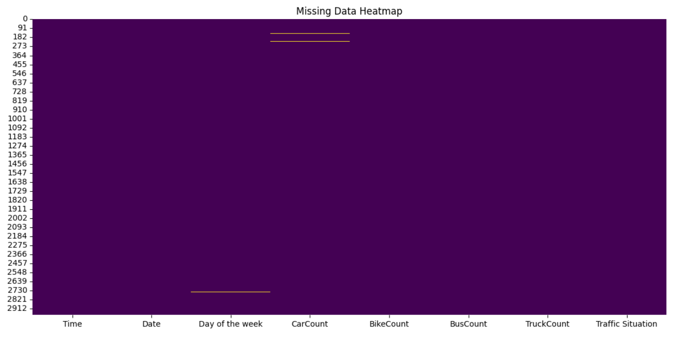

Fig. 1. Missing Data Heatmap showing distribution of missing values across features

2) Outlier Detection and Removal: Outliers were detected using the Interquartile Range (IQR) method:

$$\text{IQR} = Q_3 - Q_1 \tag{1}$$

$$\text{Lower Bound} = Q_1 - 1.5 \times \text{IQR} \tag{2}$$

$$\text{Upper Bound} = Q_3 + 1.5 \times \text{IQR} \tag{3}$$

Outliers were identified in the BikeCount feature and removed, reducing the dataset from 2,976 to 2,896 records.

Figure 2 shows the box plots for all vehicle count features, highlighting the outlier detection process. Figure 3 demonstrates the BikeCount distribution after outlier removal.

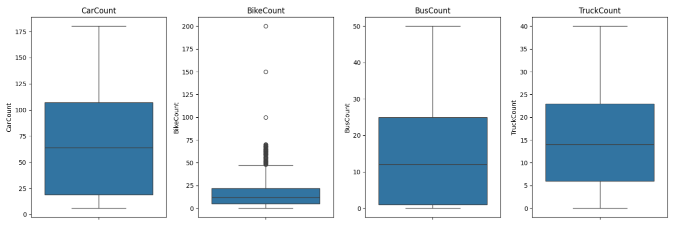

Fig. 2. Box plots showing vehicle count distributions and outlier detection

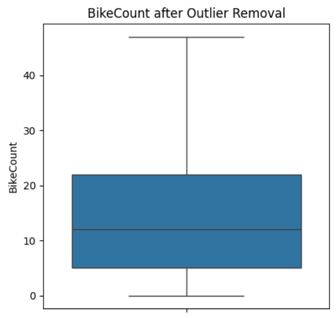

Fig. 3. BikeCount distribution after outlier removal

3) Feature Engineering:

1) **Temporal Feature Extraction**:
   - Extracted hour and minute from time data
   - Created DateTime feature combining date and time
2) **Total Vehicle Count**: Created a new feature as the sum of all vehicle counts
3) **Quantization**: Applied quantile-based binning to vehicle count features:
   - CarCount_Quantized
   - BikeCount_Quantized
   - BusCount_Quantized
   - TruckCount_Quantized
   - total_Quantized
4) **One-Hot Encoding**: Applied to ”Day of the week” feature
5) **Label Encoding**: Encoded the target variable ”Traffic Situation”

Figure 4 shows the correlation matrix of features, which helps identify relationships between different variables and potential multicollinearity issues.

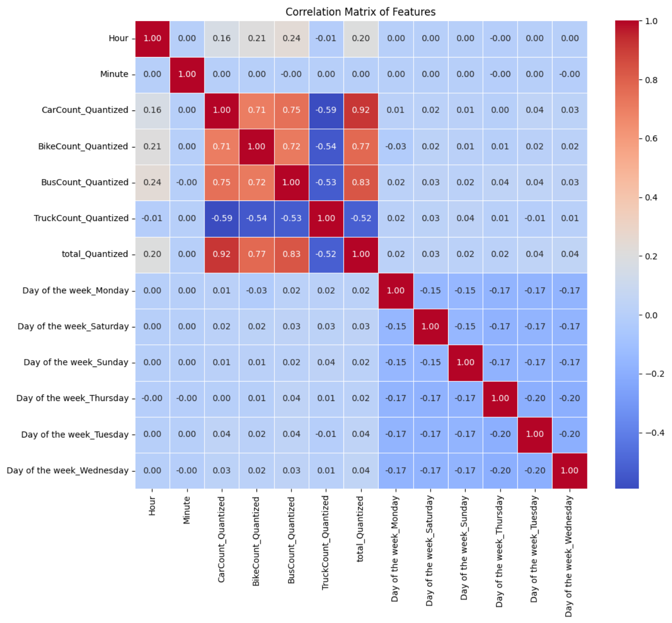

Fig. 4. Correlation matrix of features showing relationships between variables

4) Feature Scaling: Applied StandardScaler to numerical features to normalize the data:

$$z = \frac{x - \mu}{\sigma} \tag{4}$$

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

C. Class Imbalance Handling

The dataset exhibited class imbalance in the target variable. To address this, Synthetic Minority Oversampling Technique (SMOTE) was applied to the training data. SMOTE generates synthetic samples for minority classes by interpolating between existing samples.

Figure 5 shows the class distribution before SMOTE application, while Figure 6 demonstrates the balanced distribution after SMOTE.

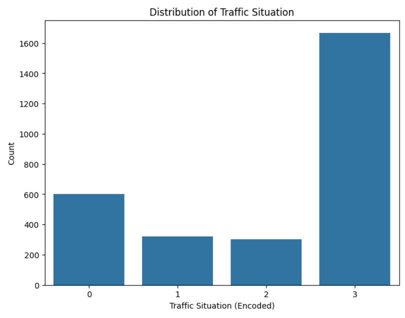

Fig. 5. Class distribution before SMOTE application

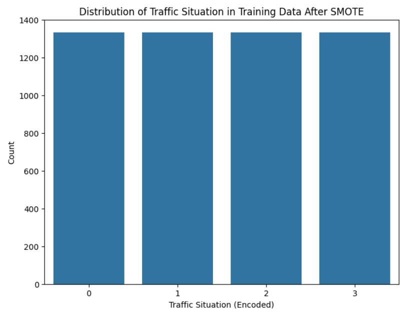

Fig. 6. Class distribution after SMOTE application

D. Train-Test Split

The dataset was split into training (80%) and testing (20%) sets using stratified sampling to maintain class distribution:

- Training set: 2,316 samples
- Test set: 580 samples

IV. Experimental Setup

A. Algorithms Implemented

1) K-Nearest Neighbors (KNN): KNN is a non-parametric algorithm that classifies data points based on the majority class of their k nearest neighbors. The algorithm was implemented with k=5 neighbors.

2) Decision Trees: Decision Trees create a tree-like model of decisions based on feature values. The algorithm recursively splits the data based on the feature that provides the best separation.

3) Random Forest: Random Forest is an ensemble method that combines multiple decision trees. Each tree is trained on a bootstrap sample of the data, and the final prediction is made by majority voting. The implementation used 100 estimators.

B. Evaluation Metrics

The models were evaluated using the following metrics:

- **Accuracy**: Overall correctness of predictions

- **Precision**: True positives / (True positives + False positives)
- **Recall**: True positives / (True positives + False negatives)
- **F1-Score**: Harmonic mean of precision and recall
- **Confusion Matrix**: Detailed breakdown of predictions vs. actual values

V. Results and Analysis

A. Model Performance Comparison

Table I presents the performance metrics for all three algorithms:

TABLE I
MODEL PERFORMANCE COMPARISON

Model	**Heavy F1**	**High F1**	**Low F1**	**Normal F1**
KNN	0.8812	0.6621	0.7448	0.8670
Decision Tree	0.8560	0.7299	0.7167	0.9030
Random Forest	0.8916	0.7820	0.8031	0.9247

Table II presents the weighted average precision and recall scores for all three algorithms:

TABLE II
AVERAGE PRECISION AND RECALL SCORES

Model	**Weighted Avg Precision**	**Weighted Avg Recall**
KNN	0.86	0.83
Decision Tree	0.86	0.85
Random Forest	0.89	0.89

Table III presents the accuracy comparison for all three algorithms:

TABLE III
MODEL ACCURACY COMPARISON

Model	**Accuracy**
KNN	0.8293 (82.93%)
Decision Tree	0.8534 (85.34%)
Random Forest	0.8879 (88.79%)

B. Detailed Performance Analysis

1) K-Nearest Neighbors: The KNN model achieved an accuracy of 82.93%. The model showed strong performance for ”heavy” traffic (F1-score: 0.8812) and ”normal” traffic (F1-score: 0.8670) but struggled with ”high” traffic classification (F1-score: 0.6621).

2) Decision Trees: Decision Trees performed better than KNN with an accuracy of 85.34%. The model showed balanced performance across all classes, with the highest F1-score for ”normal” traffic (0.9030) and consistent performance for other classes.

3) Random Forest: Random Forest achieved the highest accuracy of 88.79%, demonstrating the effectiveness of ensemble methods. The model showed excellent performance across all traffic classes, with F1-scores above 0.78 for all categories.

C. Class-wise Performance Analysis

The analysis reveals that all models perform best on the ”normal” traffic class, which is likely due to its higher representation in the dataset. The ”high” traffic class presents the most challenging classification task across all algorithms.

Figure 7 provides a detailed comparison of class-wise F1-scores across all three algorithms, while Figure 8 shows the overall model performance comparison.

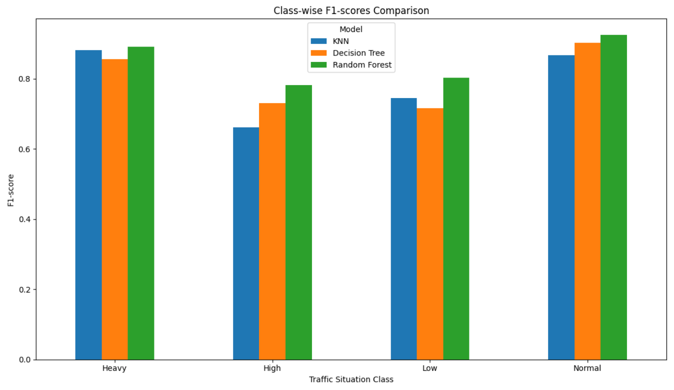

Fig. 7. Class-wise F1-scores comparison across all algorithms

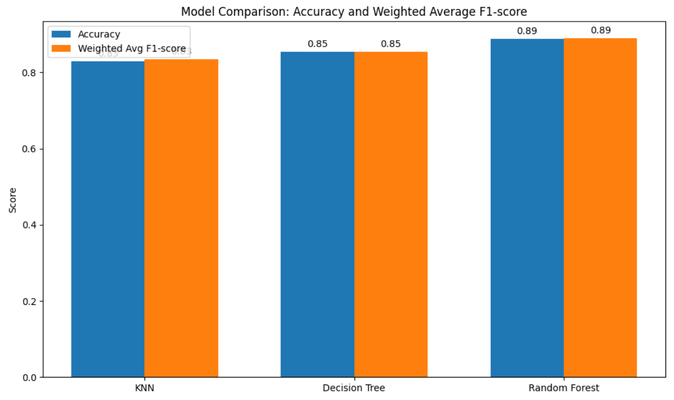

Fig. 8. Overall model performance comparison showing accuracy and weighted F1-scores

D. Confusion Matrices

The confusion matrices for all three algorithms provide detailed insights into the classification performance. Figures 9, 10, and 11 show the confusion matrices for KNN, Decision Tree, and Random Forest models respectively.

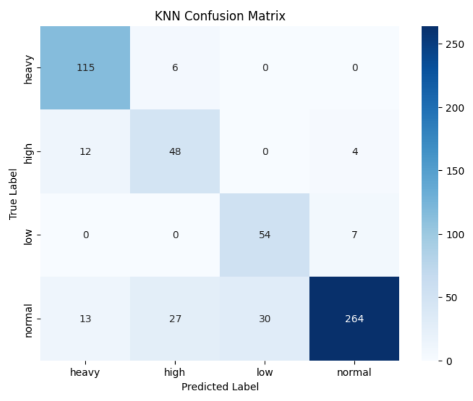

Fig. 9. KNN Confusion Matrix

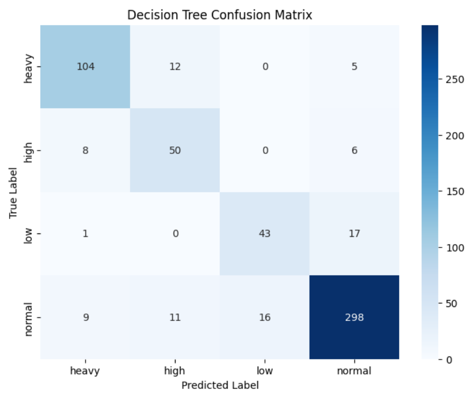

Fig. 10. Decision Tree Confusion Matrix

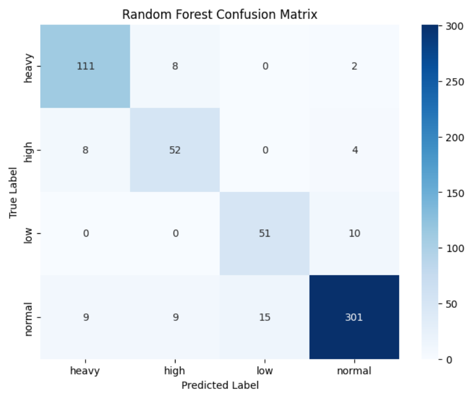

Fig. 11. Random Forest Confusion Matrix

E. Feature Importance

The Random Forest model provides feature importance scores, which indicate that temporal features (hour, minute) and vehicle count quantizations are the most influential factors in traffic prediction.

VI. Discussion

A. Algorithm Comparison

The results demonstrate a clear hierarchy in performance:

1) **Random Forest** (88.79% accuracy): Best overall performance due to ensemble learning and reduced overfitting
2) **Decision Trees** (85.34% accuracy): Good performance with interpretable results
3) **KNN** (82.93% accuracy): Baseline performance, sensitive to feature scaling

B. Impact of Data Preprocessing

The comprehensive preprocessing pipeline significantly improved model performance:

- Outlier removal reduced noise in the data
- Feature engineering (quantization, temporal features) enhanced predictive power
- SMOTE addressed class imbalance effectively
- Feature scaling improved KNN performance

C. Limitations

Several limitations should be considered:

- The dataset is relatively small (2,896 samples)
- Limited temporal coverage may not capture seasonal variations
- Feature engineering choices may not be optimal for all algorithms
- No hyperparameter tuning was performed

VII. Conclusion and Future Work

This study successfully implemented and compared three machine learning algorithms for traffic situation prediction. The Random Forest algorithm achieved the highest accuracy of 88.79%, demonstrating the effectiveness of ensemble methods for this classification task. The comprehensive preprocessing pipeline, including outlier removal, feature engineering, and class imbalance handling, significantly contributed to the model performance.

A. Future Work

Future research directions include:

- Hyperparameter optimization for all algorithms
- Integration of additional features (weather, events, road conditions)
- Deep learning approaches (neural networks, LSTM)
- Real-time prediction system implementation
- Cross-validation for more robust performance evaluation

The results provide a solid foundation for traffic prediction systems and demonstrate the practical applicability of machine learning in transportation management.

Acknowledgment

The authors would like to thank the data providers and the machine learning community for the tools and libraries that made this research possible.

References

[1] Blagus, R., Lusa, L., "SMOTE for high-dimensional class-imbalanced data," *BMC Bioinformatics*, vol. 14, pp. 106, 2013. https://doi.org/10.1186/1471-2105-14-106

[2] Shaik, A.B., Srinivasan, S., "A Brief Survey on Random Forest Ensembles in Classification Model," in *International Conference on Innovative Computing and Communications*, Bhattacharyya, S., Hassanien, A., Gupta, D., Khanna, A., Pan, I. (eds), Lecture Notes in Networks and Systems, vol. 56, Springer, Singapore, 2019. https://doi.org/10.1007/978-981-13-2354-6-27

[3] Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K., "KNN Model-Based Approach in Classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, Meersman, R., Tari, Z., Schmidt, D.C. (eds), Lecture Notes in Computer Science, vol. 2888, Springer, Berlin, Heidelberg, 2003. https://doi.org/10.1007/978-3-540-39964-3-62

[4] Suthaharan, S., "Decision Tree Learning," in *Machine Learning Models and Algorithms for Big Data Classification*, Integrated Series in Information Systems, vol. 36, Springer, Boston, MA, 2016. https://doi.org/10.1007/978-1-4899-7641-3-10