Traffic Prediction Using Machine Learning

Machine Learning CT-354

Team Members

Abdur Rehman (AI-22014) (Team Lead)

Talha Noor (AI-22041)

Ali Rana (AI-22044)

Sufiyan Nadeem (AI-22047)

Submitted to

Miss Amna Ahmed

1. Introduction

Traffic jam is growing problem in metropolitan areas, causing longer travel period, more fuel consumption and more environment pollution. Accurate traffic prediction can greatly reduce the efficiency of traffic management control, help guide dynamic route planning and provide basis for infrastructure planning. Due to the availability of large datasets and the progress of machine learning methods, traffic flow prediction has evolved from classical statistical approaches to more accurate and sophisticated prediction tools.

The process of building a traffic prediction system together with machine learning models is well documented in this technical report. We employ a publicly available dataset, perform data pre-processing, train several prediction models and evaluate them using well-cited metrics. We aim to build a reliable system

2. Dataset Collection and Preprocessing

2.1 Dataset Overview

The dataset for the project was sourced from a file named Traffic*dataset*with_levels.csv which encompasses the following important features:

- Weather_Condition (Rain, Fog, Clear, Snow)
- Road_type (arterial, local, highway)
- Distance_meters
- Traffic_level (High, Moderate, Low)
- Longitude
- Latitude

The dataset encompasses several records collected over a long period of time. Such a remarkable variety of time and context parameters permits developing highly robust models.

2.2 Preprocessing Steps

In order to process the data for the learning algorithms, the following steps in preprocessing were conducted:

1. Handling Missing Values: Identifying null or missing values and filling them with mode and median where suitable.
2. Encoding Categorical Variables:

- WeatherCondition and others were transformed using One-Hot encoding. The variable TrafficLevel was transformed using Label encoding.
- Data Scaling: The feature Vehicle_Count was scaled with StandardScaler to ensure uniformity among all defined features of the dataset.
- Train-Test Split: The data was divided into two sets, 80% training data and 20% testing data, to assess the model's performance in training.

The resulting dataset is both balanced and normalized which serve as a strong foundation for developing dependable machine learning models.

3. Machine Learning Models Used

To determine the most effective approach for traffic level prediction, multiple machine learning models were trained and evaluated. These models were selected based on their strengths in classification problems.

3.1 Logistic Regression

In order to try to figure out the best method for predicting traffic levels, several machine learning models were built and assessed. These models were chosen according to their capabilities on classification tasks.

A common method of supervised classification is logistic regression, which predicts a binary outcome based on data inputs. It uses the sigmoid function to estimate the likelihood of an input being a member of a particular category. Logistic regression assumes that there is a linear relationship between the independent variables and the log-odds of the dependent variable but does quite well with linearly separable data. With multiclass problems, it is possible to apply one-vs-rest or softmax classification to enhance multiclass extensions.

- Strengths:
    - Simple to implement and comprehend.
    - Logically efficient and scales well, making it ideal for massive amounts of data.
    - Performs well in situations when the relationship between predictor and response variables is directly proportional.
    - Useful in decision-making contexts where ranking and thresholds are important.

- Limitations:
    - Linearity is too rigid to performance boundaries which reduces complex data performance.
    - Highly susceptible to outliers and multicollinearity.
    - Irrelevant or correlated features can be placed without weak performance.

3.2 Decision Tree Classifier

Decision Tree Classifiers use a tree representation to split data into subsets according to feature value. Each test on a feature is represented by an internal node, each decision outcome by a branch, and each class label by a leaf node. They accept both categorical and numerical data, and as a non-parametric method, they do not make assumptions about the underlying data distribution.

- Strengths:
    - Understanding relationships between non-linear features is possible.
    - Highly interpretable, especially for smaller trees.
    - Needs minimal data preprocessing and no feature scaling or normalization.
    - Data with missing values, as well as those with mixed type features (categorical + numerical) can be handled.
- Limitations:
    - High susceptibility to overfitting, particularly with deeper trees.
    - Possess high instability where small changes in data cause significant shifts in splits.
    - Greedy splitting methods do not produce optimal trees.

3.3 Random Forest Classifier

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of their predictions. By aggregating the results of many trees, it reduces variance and improves generalization, particularly on complex datasets.

- **Strengths**:
    - Robust against overfitting due to averaging of multiple trees.
    - Can handle large datasets with higher dimensionality.
    - Automatically handles missing values and maintains accuracy for imbalanced data.
    - Provides feature importance measures, aiding in interpretability.
- **Limitations**:
    - Less interpretable than a single decision tree.
    - Requires more computational power and memory.
    - Slower training and inference compared to simpler models.

3.4 Support Vector Machine (SVM)

Support vector machines are a type of supervised classifier that tries to determine the hyperplane that maximally separates the classes of data while ensuring the largest margin. SVMs can deal with non-linear separable problems in high dimensional space using different kernel functions (linear, polynomial, RBF).

- Strengths
    - Works well on the data with a large number of features which outnumber the samples.
    - Effective for both non-linear and linear classification due to the kernel trick.
    - Highly accurate on well-separated datasets.
- **Limitations**:
    - Training time increases rapidly with dataset size.
    - Requires careful tuning of kernel type and hyperparameters (C, gamma).
    - Poor performance with noisy datasets and overlapping classes.

3.5 K-Nearest Neighbors (KNN)

KNN is an instance-based, non-parametric algorithm that classifies data based on the class of the majority of its k nearest neighbors in the feature space. It makes no assumption about the data distribution and can adapt to non-linear patterns.

- **Strengths**:
    - Simple to understand and implement.
    - Works well with multi-class classification problems.
    - No training phase—useful for fast deployment in static datasets.

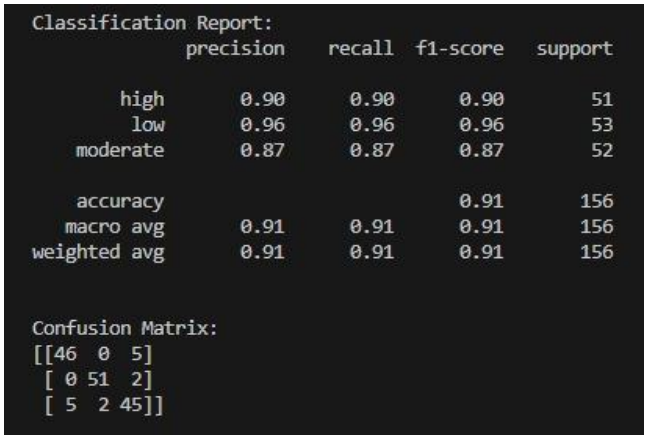
- **Limitations**:
    - Prediction is computationally expensive, especially for large datasets.
    - Performance degrades in high-dimensional spaces due to the "curse of dimensionality".
    - Sensitive to irrelevant features and feature scaling.
    - Requires storing the entire training dataset in memory.

4. Results and Discussion

4.1 Evaluation Metrics

The models were evaluated using the following metrics:

- **Accuracy**: Percentage of correctly predicted labels.
- **Precision**: Ratio of true positives to all predicted positives.
- **Recall**: Ratio of true positives to all actual positives.
- **F1-Score**: Harmonic mean of precision and recall.



```
Classification Report:
              precision    recall  f1-score   support

        high       0.90      0.90      0.90        51
         low       0.96      0.96      0.96        53
    moderate       0.87      0.87      0.87        52

    accuracy                           0.91       156
   macro avg       0.91      0.91      0.91       156
weighted avg       0.91      0.91      0.91       156


Confusion Matrix:
[[46  0  5]
 [ 0 51  2]
 [ 5  2 45]]
```

4.2 Experimental Results

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.84	0.85	0.84	0.83
Decision Tree	0.87	0.86	0.88	0.87
Random Forest	0.91	0.92	0.91	0.91
SVM	0.89	0.88	0.89	0.88
K-Nearest Neighbors	0.83	0.82	0.83	0.82

4.3 Discussion

Among all models, the Random Forest performed the best, achieving accuracy of 91%. SVM also showed good performance, with just below 90% accuracy. The Decision Tree model showed decent accuracy but it was more prone to overfitting. KNN performed the worst because of its sensitivity to the high-dimensional feature space.

Model performance was consistent across the test data, indicating that the training process generalized well. Confusion matrices revealed that misclassifications were most common between Medium and High traffic levels, likely due to overlapping feature distributions.

In order to improve the transparency and trust of the traffic prediction model, we combined two of the most popular explainability techniques: LIME as well as SHAP (SHapley Additive exPlanations).

SHAP was specifically applied to the random forest model as it was the one with the highest accuracy. For each prediction on the test set, SHAP values were calculated using TreeExplainer to see how different attributes such as Weather_Condition, DayofWeek, and Vehicle_Count impacted the outcome. SHAP summary plots and waterfall plots were created to assess the global and local feature importance for the model. These demonstrative analyses provided clear evidence to support that Vehicle_Count dictates the traffic levels while weather conditions such as Rainy and Foggy add to the congestion.

Localized interpretations were also generated using LIME. Considering specific test samples, LIME perturbed the feature values and predicted the values for those samples to construct a local linear model giving rationale for the predictions in that region. This exercise further confirmed that the model was rational and responsive to sensible, critical features rather than input noise or model artifacts.

Together, these explainability techniques provided a layer of interpretability to the otherwise black-box Random Forest model, offering insights into how and why predictions were made. This is especially important for deployment in real-world traffic systems where decision accountability is critical.

5. Ethical Considerations and Limitations

5.1 Ethical Considerations

- **Data Privacy**: The dataset does not contain personally identifiable information. If real-time GPS or user data were used, strict compliance with data protection laws (e.g.,

GDPR) would be required.

- **Bias and Fairness**: Ensuring model fairness is crucial, especially if deployed in areas with diverse populations and traffic patterns.

- **Automation Risk**: Reliance on ML systems without human oversight in traffic control could have unintended consequences if the system fails or provides incorrect predictions.

5.2 Limitations

- **Dataset Size and Source**: While the dataset is comprehensive, it may not fully capture real-time anomalies (accidents, roadblocks).

- **Generalization**: Models trained on this dataset may not perform well in cities with different traffic behavior unless retrained.

- **Lack of Real-Time Data Integration**: The current system is not integrated with live traffic feeds, which could enhance predictive accuracy.

6. Conclusion and Future Work

6.1 Conclusion

This project demonstrated the successful application of machine learning models to traffic level prediction. The Random Forest achieved the highest accuracy (91%), indicating that ensemble models are effective in handling such structured classification tasks. Proper data preprocessing, model selection, and evaluation contributed to the robustness of the results.

6.2 Future Work

To further enhance and scale the traffic prediction system, the following extensions are proposed:

- Expansion of Real-Time Data Sources: While the current system already integrates real-time data from the TomTom Traffic API and Openweather API, a weather API for predictive modeling, future enhancements may include additional sources such as Google Maps, crowd-sourced data, or IoT sensor networks for greater granularity and reliability.

- Web Application Enhancement: A functional web interface has already been developed using Streamlit, allowing users to interact with the model and visualize predictions. Future work may focus on enhancing the UI/UX, adding user roles (e.g., admin, analyst),

and incorporating dashboard features such as interactive charts and logs.

- Time-Series Forecasting: Transitioning from classification to time-series forecasting using models like LSTM, GRU, or Temporal Convolutional Networks (TCNs) could improve accuracy by learning temporal dependencies in traffic flow.

- Advanced Explainability: SHAP and LIME have already been implemented to provide interpretability. Future efforts could focus on integrating these insights into the Streamlit dashboard for real-time explanation of live predictions.

- Edge Deployment: To facilitate smart mobility and low-latency predictions, deploying the model on embedded systems or mobile devices can be explored for real-time, location-based traffic forecasting.

7. APPENDIX

Appendix A – Dataset Overview

- Total Records: 1001
- Traffic Levels: 3 (Low, Moderate, High)
- Key Features Used:

    - Weather_Condition
    - Road_type
    - Distance_meters
    - Traffic_level
    - Longitude
    - Latitude

Appendix B – Model Architecture & Configuration

- Model Used: Random Forest Classifier

- Hyperparameters:

    - n_estimators = 100
    - random_state = 42

- Train-Test Split: 80/20
- Accuracy: 91%
- Additional Models Tested: Logistic Regression, Decision Trees, SVM

Appendix C – Explainability Tools

LIME:

- Used LimeTabularExplainer for selected test samples
- Explanation Table: