<u>TrafficTelligence:Advanced Traffic Volume Estimation with Machine Learning</u>

By

(KRISHANMURTI KUMAR)

(RAJANI MUDIMADUGU)

(SHASHI KUMAR)

(VIKASH KUMAR UPADHAYA)

Guided by

Prof. Ms Raasha

A Dissertation Submitted to SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY, An Autonomous Institution affiliated to 'JNTU Ananthapur' in Partial Fulfilment of the Bachelor of Technology (***Computer Engineering***) with Specialization in ***Artificial Intelligence and Machine Learning*.**

May 2024

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY

R.V.S. Nagar Tirupathi Road, Andhra Pradesh– 517127

MODEL OPTIMIZATION AND TUNING REPORT:

Introduction:

In this report, we will outline the model optimization and tuning process for the project "Traffic Intelligence: Advanced Traffic Volume Estimation with Machine Learning." The objective of this project is to accurately estimate traffic volume using various machine learning techniques. We will cover the initial model selection, hyperparameter tuning, and performance evaluation.

1. Initial Model Selection

To begin with, we selected several machine learning models that are commonly used for regression tasks:

1. Linear Regression
2. Decision Tree Regressor
3. Random Forest Regressor
4. Gradient Boosting Regressor
5. Support Vector Regressor (SVR)
6. Neural Networks (MLPRegressor)

These models were chosen due to their varying capabilities and suitability for different data characteristics.

2. Data Preprocessing

Before model training, the data was preprocessed as follows:

Data Cleaning: Handling missing values and outliers.
Feature Engineering: Creating new features based on domain knowledge.
Normalization/Scaling: Normalizing features to ensure all have similar scales, which is crucial for models like SVR and Neural Networks.
Train-Test Split: Splitting the dataset into training (80%) and testing (20%) sets to evaluate the models' performance on unseen data.

3. Baseline Model Performance

We trained the initial models using default hyperparameters to establish baseline performance metrics. The evaluation metrics used were:

*Mean Absolute Error (MAE)
*Mean Squared Error (MSE)
*Root Mean Squared Error (RMSE)
*R-squared (R^2) Score

4. Hyperparameter Tuning

To optimize the models, we performed hyperparameter tuning using techniques like Grid Search and Random Search along with cross-validation.

4.1. Random Forest Regressor

Parameters tuned:
*n_estimators: Number of trees in the forest.
*max_depth: Maximum depth of the tree.
*min_samples_split: Minimum number of samples required to split an internal node.
*min_samples_leaf: Minimum number of samples required to be at a leaf node.

Best parameters found:
*n_estimators: 200
*max_depth: 20
*min_samples_split: 5
*min_samples_leaf: 2

4.2. Gradient Boosting Regressor

Parameters tuned:

n_estimators: Number of boosting stages.
learning_rate: Step size shrinkage.
max_depth: Maximum depth of the individual estimators.
subsample: Fraction of samples used for fitting the individual base learners.

Best parameters found:
n_estimators: 150
learning_rate: 0.05
max_depth: 8
subsample: 0.8

4.3. MLPRegressor

Parameters tuned:
hidden_layer_sizes: Number of neurons in the hidden layers.
activation: Activation function for the hidden layer.
solver: The solver for weight optimization.
alpha: L2 penalty (regularization term) parameter.

Best parameters found:
hidden_layer_sizes: (100, 50)
activation: 'relu'
solver: 'adam'
alpha: 0.0001

5. Final Model Performance

After tuning the hyperparameters, the models' performance improved as follows:

The Gradient Boosting Regressor achieved the best overall performance with the lowest MAE, MSE, and RMSE, and the highest R^2 score.

6. Conclusion

Through systematic model optimization and tuning, we improved the accuracy of traffic volume estimation significantly. The Gradient Boosting Regressor emerged as the best-performing model for this task, demonstrating the importance of hyperparameter tuning in enhancing model performance. Future work may involve exploring more advanced techniques like ensemble methods, stacking, or deep learning models, as well as incorporating real-time data for further improvements.