Interstate Traffic Volume Prediction - Random Forest Regressor

This project focuses on predicting **traffic volume** on interstate highways using historical traffic data. The model is built using a **Random Forest Regressor**, which is an ensemble learning method that combines multiple decision trees to improve the prediction accuracy and reduce overfitting.

Overview

The primary goal of this project is to predict the traffic volume on interstate highways. The dataset includes several features such as **date**, **time**, and other weather or traffic-related information that influence traffic patterns.

We use a **Random Forest Regressor** as the model to predict the traffic volume, which aggregates the predictions of multiple decision trees trained on different subsets of the data. This helps improve the robustness of the model compared to a single decision tree.

Model Structure

The model consists of two main parts:

1. **DecisionTreeRegressor**: This is the basic building block of the Random Forest. Each decision tree is trained by recursively splitting the data based on the feature that minimizes the Mean Squared Error (MSE). The tree continues splitting until it reaches the maximum depth or cannot split further.

2. **RandomForestRegressor**: This model uses bagging (Bootstrap Aggregating) to train multiple decision trees on different random subsets of the data. It then averages the predictions from all the trees to make the final prediction.

Key Features:

- **Ensemble learning**: By using multiple decision trees, the random forest improves prediction accuracy and generalizes better than a single decision tree.
- **Bootstrap sampling**: Each tree is trained on a random sample of the data, which helps reduce variance.
- **Mean Squared Error (MSE)**: This is the primary metric used to evaluate the model's performance by minimizing the difference between predicted and actual traffic volumes.

Functions Explanation

`DecisionTreeRegressor`

The **DecisionTreeRegressor** class is a basic decision tree for regression tasks. It is used to find the best feature and threshold for splitting the data at each node in the tree.

- **fit(X, y)**: This method trains the decision tree on the given features (`X`) and target (`y`). It recursively splits the data to minimize the MSE.
- **build_tree(X, y, depth=0)**: This is a recursive function that builds the tree by finding the best feature and threshold at each node.
- **_find_best_split(X, y)**: This method determines the best feature and threshold to split the data by calculating the MSE for every possible split.
- **predict(X)**: This method makes predictions based on the trained tree, returning the predicted values for the input data.

`RandomForestRegressor`

The **RandomForestRegressor** class implements the Random Forest model using multiple decision trees. It combines the predictions of several trees to make more robust predictions.

- **fit(X, y)**: This method trains the random forest by creating multiple decision trees using bootstrap sampling. Each tree is trained on a random subset of the data.
- **predict(X)**: This method predicts the traffic volume by averaging the predictions from all decision trees in the forest.
- **save_model(filename)**: This method saves the trained model to a file using `joblib` for later use.
- **load_model(filename)**: This method loads a previously saved model from a file.

Input/Output

Input

- **X (features)**: A dataset containing the features such as **date**, **time**, **weather conditions**, **traffic-related data**, etc.
- **y (target)**: A dataset containing the actual **traffic volume** values corresponding to the features in `X`.

Output

- The model outputs the predicted **traffic volume** for a given set of input features.
- Performance metrics like **Mean Squared Error (MSE)** and **R-squared (R^2)** are also evaluated to measure the model's prediction accuracy.

Conclusion

This **Random Forest Regressor** implementation provides a robust method for predicting traffic volume on interstate highways. By aggregating the predictions of multiple decision trees, the Random Forest model reduces overfitting and improves the accuracy of the traffic volume predictions.

Future Work

- **Hyperparameter Tuning**: The model can be further tuned by adjusting parameters such as `n_estimators`, `max_depth`, and `max_features`.
- **Model Comparison**: You can compare the performance of this custom Random Forest implementation with scikit-learn's built-in Random Forest Regressor.
- **Feature Engineering**: Additional features like holiday, weather, and road closures could improve model performance.