Traffic Analysis and Prediction Report

Name: Jiawen Miao

NetID: jm2354

Section: 07

The data set is from the Kaggle website:

https://www.kaggle.com/datasets/hasibullahaman/traffic-prediction-dataset

GitHub repo :

https://github.com/Tai5ui/Traffic-Analysis-and-Prediction

Traffic congestion and its associated challenges are significant concerns in urban areas worldwide, impacting daily commutes, economic productivity, and environmental quality. The complexity of traffic systems, influenced by factors such as population growth, urbanization, and inadequate infrastructure, necessitates a thorough understanding of traffic patterns to address these issues effectively. By analyzing traffic data, transportation planners can identify bottlenecks, predict peak congestion periods, and evaluate the impact of infrastructure projects.

Project Definition:

This project addresses the problem of understanding traffic density patterns to support urban planning and traffic management. By predicting traffic situations based on vehicle counts and times of the day, the project provides insights that can be used to improve public safety, reduce congestion, and optimize infrastructure.

The strategic aspects involve leveraging machine learning and data science techniques for predictive analysis. This includes feature engineering, model evaluation, and ensuring data integrity, which are directly related to the methodologies discussed in lectures.

The project is significant because effective traffic analysis can reduce economic costs associated with delays and accidents while enhancing public safety. It contributes to data-driven decision-making in urban management.

Code explanation:

[3]:
```python
import pandas as pd

# Load the dataset
data = pd.read_csv('Traffic.csv')

# Display the first few rows
print(data.head())

# Display the structure of the dataset
print(data.info())

# Check for missing values
print(data.isnull().sum())
```

This code represents the data loading process. The Traffic.csv dataset serves as a database, providing structured information for the project.

pd.read_csv() loads the data into a pandas DataFrame, analogous to querying a relational database.

data.head() and data.info() ensure the data's structure is correct and ready for further preprocessing.

```python
# Encode categorical variables
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
data['DayOfWeekEncoded'] = encoder.fit_transform(data['Day of the week'])
data['TrafficSituationEncoded'] = encoder.fit_transform(data['Traffic Situation'])

# Extract 'Hour' from 'Time'
data['Hour'] = pd.to_datetime(data['Time'], format='%I:%M:%S %p').dt.hour

# Drop unnecessary columns
data = data.drop(columns=['Time', 'Day of the week', 'Traffic Situation'])

print(data.head())
```

This code performs feature engineering, preparing the data for analysis and modeling.

Label Encoding converts categorical variables (Day of the week, Traffic

Situation) into numeric formats, enabling machine learning algorithms to process them.

The Hour feature is derived from the Time column, capturing time-based traffic patterns.

Dropping columns that are no longer needed simplifies the dataset for further analysis.

```
import matplotlib.pyplot as plt

# Group by hour and calculate average traffic
time_traffic = data.groupby('Hour')['Total'].mean()
time_traffic.plot(kind='line', figsize=(10, 6))
plt.title('Average Traffic by Hour')
plt.xlabel('Hour')
plt.ylabel('Average Traffic')
plt.grid(True)
plt.show()
```

This code visualizes traffic patterns by hour, an essential step in Exploratory Data Analysis (EDA).

Grouping data by Hour provides insights into how traffic density varies throughout the day.

Line plots highlight trends, showing peak traffic hours and helping stakeholders make data-driven decisions.

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Define features and target
X = data[['CarCount', 'BikeCount', 'BusCount', 'TruckCount', 'Hour', 'DayOfWeekEncoded', 'TrafficSituationEncoded']]
y = data['Total']

# Split the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Evaluate the model
y_pred_lr = lr_model.predict(X_test)
print(f"Linear Regression MAE: {mean_absolute_error(y_test, y_pred_lr)}")
print(f"Linear Regression R2: {r2_score(y_test, y_pred_lr)}")
```

Linear Regression provides a baseline for traffic prediction. It models the relationship between the features (e.g., vehicle counts, hour) and the target (Total traffic).

Model evaluation uses metrics like Mean Absolute Error (MAE) and R^2, assessing the accuracy of predictions.

```python
from sklearn.ensemble import RandomForestRegressor

# Train the Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Evaluate the model
y_pred_rf = rf_model.predict(X_test)
print(f"Random Forest MAE: {mean_absolute_error(y_test, y_pred_rf)}")
print(f"Random Forest R2: {r2_score(y_test, y_pred_rf)}")
```

Random Forest captures non-linear relationships in the data, improving prediction accuracy compared to Linear Regression.

The model’ s performance is evaluated similarly, and feature importance can be analyzed to identify the most influential predictors.

Existing Issues in Data Management:

Real-world datasets often contain missing values or poorly formatted data. Extracting meaningful features from time-based or categorical data is often complex. Large-scale traffic systems require models that generalize well.

Previous studies have focused on traffic prediction using techniques like deep learning or simulation models. However, this project emphasizes interpretable machine learning techniques (e.g., Random Forest), balancing predictive performance with understandability.

Models/Techniques:

Linear Regression: A baseline model to predict total traffic counts.

Random Forest Regressor: A more advanced model that captured non-linear relationships and delivered better results.

Experiments:

Models were evaluated on an 80-20 train-test split. Metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 were used for evaluation. Feature importance was analyzed to understand the contributions of individual predictors.

Key Findings:

Random Forest performed better than Linear Regression with an MAE of 2.66 and an R^2 of 1.0. Features like Hour and CarCount significantly impacted predictions, highlighting time-based traffic patterns.

Removing the Total column as a feature avoided data leakage and improved model robustness. The models were validated on a separate test set, and visualizations (e.g., scatter plots for actual vs. predicted values) were used to assess their performance.

Advantages:

Scalability: The approach can handle similar datasets in other regions or contexts.

Interpretability: Feature importance analysis provided actionable insights.

Robustness: Preprocessing steps ensured data integrity and avoided leakage.

Limitations:

Dependency on Data Quality: Poor-quality data could reduce model performance.

Overfitting Risk: Tree-based models like Random Forest may overfit small

datasets.

Changes After Proposal

Summary:

This project successfully demonstrated the use of machine learning models, particularly Random Forest, to predict traffic situations accurately. The insights gained from the feature importance analysis and model evaluation can aid traffic authorities in managing congestion and optimizing infrastructure. Future work could include more complex features and advanced techniques to further enhance the model's utility in real-world scenarios.