FINAL PROJECT REPORT

INTRODUCTION

Project Overview: TrafficTelligence is an advanced traffic management system designed to estimate and predict traffic volumes using cutting-edge machine learning algorithms. By analyzing a combination of historical traffic data, weather conditions, special events, and other pertinent factors, TrafficTelligence delivers precise traffic forecasts. These insights are aimed at optimizing traffic management, facilitating urban planning, and enhancing commuter experiences. **Key Features:**

1. **Historical Data Analysis:** Leverages past traffic data to identify patterns and trends.
2. **Weather Pattern Integration:** Incorporates weather forecasts to adjust traffic predictions.
3. **Event Impact Assessment:** Analyzes the impact of events (e.g., sports games, concerts) on traffic flow.
4. **Real-time Updates:** Provides current traffic volume estimates for immediate traffic management decisions.
5. **Machine Learning Algorithms:** Utilizes advanced algorithms for accurate and dynamic traffic predictions.

Scenarios:

1. **Dynamic Traffic Management:** Enables real-time traffic control adjustments to reduce congestion.
2. **Urban Development Planning:** Assists city planners in designing efficient infrastructure based on future traffic predictions.
3. **Commuter Guidance and Navigation:** Offers commuters optimal route recommendations and realtime traffic updates.

Deliverables

1. **Comprehensive Project Report:**

   - Detailed documentation of the project's objectives, methodology, and results.
   - Analysis of historical traffic data, weather patterns, and event impacts.
   - Insights into machine learning models used for traffic prediction.

2. **Traffic Prediction Models:**

   - Trained machine learning models capable of predicting traffic volumes.
   - Documentation of model selection, training process, and evaluation metrics.

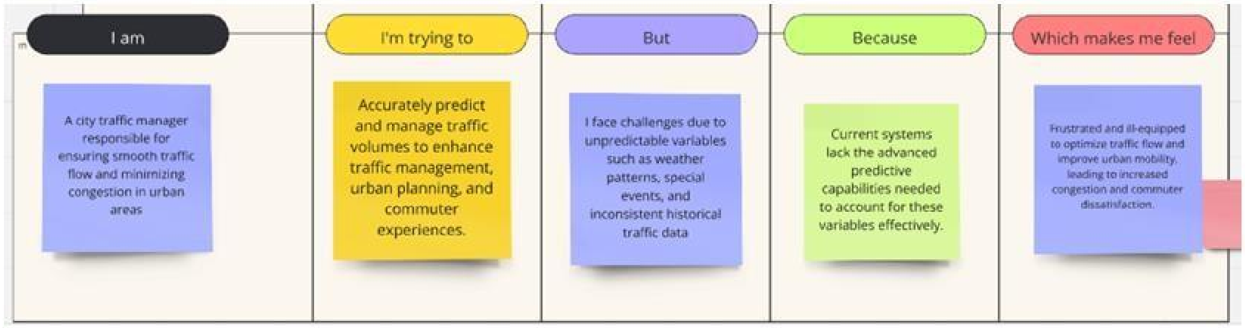
Project Initialization and Planning Phase

Define Problem Statements (Customer Problem Statement Template):

Urban areas are experiencing increasing challenges in managing traffic flow due to growing populations, frequent events, and variable weather conditions. Traditional traffic management systems struggle to provide accurate and timely predictions, leading to congestion, delays, and inefficiencies. These issues impact urban planning, commuter experiences, and overall quality of life.

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A city traffic manager responsible for ensuring smooth traffic flow and minimizing congestion in urban areas.	Accurately predict and manage traffic volumes to enhance traffic manageme nt, urban planning, and commuter experiences	I face challenges due to unpredicta ble variables such as weather patterns, special events, and inconsisten t historical traffic data.	Current systems lack the advanced predictive capabilitie s needed to account for these variables effectively.	Frustrated and ill equipped to optimize traffic flow and improve urban mobility, leading to increased congestion and commuter dissatisfaction.

Objective	To develop TrafficTelligence, an advanced system that uses machine learning algorithms to estimate and predict traffic volume with high precision, enhancing traffic management, urban planning, and commuter experiences.
Scope	The project will include: • Analyzing historical traffic data.



Project Proposal (Proposed Solution)

This project proposal outlines a solution to address a specific problem. With a clear objective, defined scope, and a concise problem statement, the proposed solution details the approach, key features, and resource requirements, including hardware, software, and personnel.

Project Overview

Problem Statement

	• Integrating weather patterns and event impacts. • Providing real-time traffic monitoring and predictive modeling.

Proposed Solution

Description	Urban areas face increasing traffic management challenges due to population growth, frequent events, and variable weather conditions. Traditional systems are inadequate for accurate, timely traffic predictions, leading to congestion, delays, and inefficiencies in urban planning and commuter experiences.
Impact	Solving this problem will lead to: • Improved traffic flow and reduced congestion. • Enhanced urban planning with accurate traffic forecasts. • Better commuter experiences with real-time traffic updates and predictive insights.
Approach	Use machine learning algorithms to analyze historical traffic data, weather patterns, and events. Develop predictive models to forecast traffic volumes. Create a real-time traffic monitoring system.
Key Features	High precision traffic volume predictions. Integration of weather and event data. Real-time traffic monitoring and updates. Predictive traffic modeling for future scenarios.

	Description	**Specification/Allocation**
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	e.g., scikit-learn, pandas, numpy
Development Environment	IDE, version control	e.g., Jupyter Notebook, Git

Software

Data

Given dataset from the portal.

Initial Project Planning Report

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Team Members	Sprint Start Date	Spr Dat (Pla
Sprint 1	Data Collection and Preprocessing	SL-1	Understanding & loading data	Low	Rishab	2024/06/20	202
Sprint 1	Data Collection and Preprocessing	SL-2	Data cleaning	High	Jerwin	2024/06/20	202
Sprint 1	Data Collection and Preprocessing	SL-3	EDA	Medium	Kaushik S	2024/06/21	202
Sprint 4	Project Report	SL-4	Report	Medium	Jeya Madhavan	2024/07/19	202
Sprint 2	Model Development	SL-5	Training the model	Medium	Jerwin	2024/07/02	202
Sprint 2	Model Development	SL-6	Evaluating the model	Medium	Jeya Madhavan	2024/07/04	202
Sprint 2	Model tuning and testing	SL-7	Model tuning	High	Rishab	2024/07/05	202
Sprint 2	Model tuning and	SL-8	Model testing	Medium	Kaushik S	2024/07/06	202

	testing						
Sprint 3	Web integration and Deployment	SL-9	Building HTML templates	Low	Jerwin	2024/07/12	202
Sprint 3	Web integration and Deployment	SL-10	Local deployment	Medium	Rishab	2024/07/16	202

Data Collection and Preprocessing Phase

Data Collection

Section	Description
Data Overview	**Dimension**: 48204 rows × 8 columns **Descriptive statistics:**

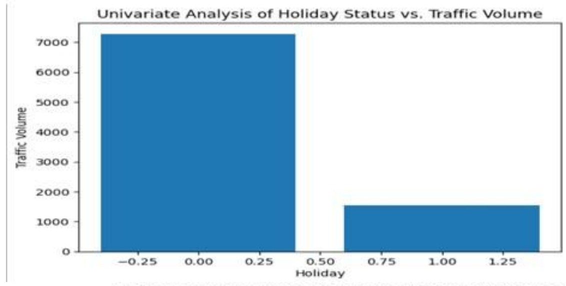
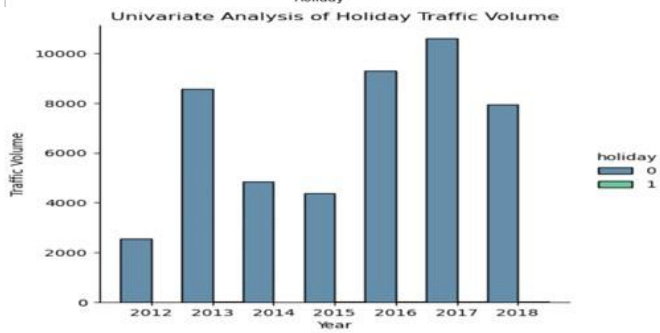
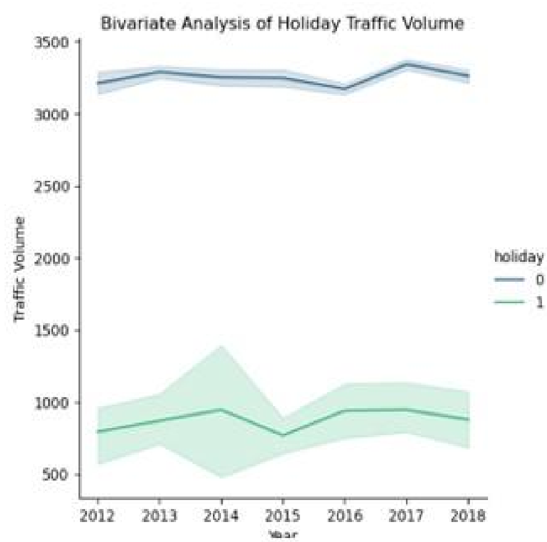
	temp	rain	snow	traffic_volume
count	48151.000000	48202.000000	48192.000000	48204.000000
mean	281.205351	0.334278	0.000222	3259.818355
std	13.343675	44.790062	0.008169	1986.860670
min	0.000000	0.000000	0.000000	0.000000
25%	272.160000	0.000000	0.000000	1193.000000
50%	282.460000	0.000000	0.000000	3380.000000
75%	291.810000	0.000000	0.000000	4933.000000
max	310.070000	9831.300000	0.510000	7280.000000

Data Quality

Project Overview	TrafficTelligence is an advanced system that uses machine learning algorithms to estimate and predict traffic volume with precision. By analyzing historical traffic data, weather patterns, events, and other relevant factors, TrafficTelligence provides accurate forecasts and insights to enhance traffic management, urban planning, and commuter experiences.
Data Collection Plan	Data was taken from the given pre default dataset, given in the guided workspace.
Raw Data Sources Identified	No raw dataset was chosen during this project.

Data Source	**Data Quality Issue**	**Severity**	**Resolution Plan**
Given default dataset	Missing values in 'temp', 'rain', 'snow', 'weather'	Moderate	Use mean/median imputation
Given default dataset	Categorical data in the dataset	Moderate	Encoding has to be done in the data

Data Exploration and Preprocessing

Univariate Analysis	 
Bivariate Analysis	

Data Preprocessing Code Screenshots

Loading Data

	holiday	temp	rain	snow	weather	date	Time	traffic_volume
0	NaN	288.28	0.0	0.0	Clouds	02-10-2012	09:00:00	5545
1	NaN	289.36	0.0	0.0	Clouds	02-10-2012	10:00:00	4516
2	NaN	289.58	0.0	0.0	Clouds	02-10-2012	11:00:00	4767
3	NaN	290.13	0.0	0.0	Clouds	02-10-2012	12:00:00	5026
4	NaN	291.14	0.0	0.0	Clouds	02-10-2012	13:00:00	4918

Handling Missing Data

```python
#fill the missing cells with the mean of the whole column
data['temp']=data['temp'].fillna(data['temp'].mean())
data['rain']=data['rain'].fillna(data['rain'].mean())
data['snow']=data['snow'].fillna(data['snow'].mean())
data['weather'].fillna('Clouds',inplace=True)
data['weather'].fillna('NaN',inplace=True)
```

Data Transformation

```python
holiday_list = ['Labor Day', 'Thanksgiving Day', 'Christmas Day', 'New Years Day',
                'Martin Luther King Jr Day', 'Columbus Day', 'Veterans Day',
               'Memorial Day', 'Independence Day', 'State Fair']

data['holiday'] = data['holiday'].apply(lambda x: '1' if x in holiday_list else '0')
```

```python
data[["day","month","year"]] = data["date"].str.split("-", expand=True)
data[["hours","minutes","seconds"]] = data["Time"].str.split(":", expand=True)
data.drop(columns=['date','Time'],axis=1,inplace=True)
```

Feature Engineering

```python
from sklearn.preprocessing import LabelEncoder

# Assuming your data is in a DataFrame called 'data'
# Assuming the weather column is named 'weather'

# Create a LabelEncoder object
le = LabelEncoder()

# Fit the LabelEncoder to the weather data (learn the categories)
le.fit(data['weather'])

# Transform the 'weather' column to numerical labels
data['weather'] = le.transform(data['weather'])
```

Save Processed Data

	holiday	temp	rain	snow	weather	traffic_volume	day	month	year	hours	minutes	seconds
39346	0	277.44	0.0	0.0	0	2859	29	11	2017	20	00	00
23628	0	296.46	0.0	0.0	1	4603	25	05	2016	18	00	00
6563	0	294.84	0.0	0.0	1	5635	31	05	2013	13	00	00
44041	0	279.69	0.0	0.0	0	622	13	05	2018	02	00	00
43918	0	290.46	0.0	0.0	6	3274	08	05	2018	19	00	00

Model Development Phase

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model	Description	Perform Metri F
Linear Regressor	A linear regressor is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data.	13%
Decision Tree Regressor	A Decision Tree Regressor is a machine learning model that predicts the value of a target variable by learning decision rules from features, recursively splitting the data into subsets based on feature values.	71%
Random Forest Regressor	A Random Forest Regressor is an ensemble learning method that uses multiple decision trees to improve the accuracy and robustness of predictions by averaging the outputs of individual trees.	84%
SVR	Support Vector Regression (SVR) is a machine learning model that uses the principles of support vector machines to predict continuous values by finding the best-fit hyperplane within a specified margin of tolerance.	0%

Feature	Description	Selected (Yes/No)	Reasoning
holiday	Tells whether a particular day is a holiday or not	No	Converted it to numerical data using lambda apply function in the form of 0’s and 1’s

MODEL SELECTION

temp	Describes about the temperature	No	Already in numerical data
rain	Whether it is raining or not	No	Already in numerical data
snow	Whether it is snowing or not	No	Already in numerical data
weather	Tells about the weather condition	Yes	To give different weather conditions a particular number using label encoding for easy processing
date	Particular date	No	Already in numeric data
time	Particular time	No	Already in numeric data
traffic volume	About traffic volume	No	Already in numeric data

FEATURE SELECTION

Model Optimization and Tuning Phase

Initial Model Training Code:

```
#splitting into independant and dependant variables
y=data['traffic_volume']
x=data.drop(columns=['traffic_volume'],axis=1)
print(x.head())

#splitting the data into train data and test data
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
lin_reg = linear_model. LinearRegression()
Dtree = tree. DecisionTreeRegressor()
Rand = ensemble. RandomForestRegressor()
svr = svm. SVR( )
#XGB = xgboost . XGBRegressor ()
```

```
from sklearn import linear_model
from sklearn import tree
from sklearn import ensemble
from sklearn import svm
```

```
lin_reg.fit(x_train,y_train)
Dtree.fit(x_train,y_train)
Rand.fit(x_train,y_train)
svr.fit(x_train,y_train)
#XGB.fit(x_train,y_train)
p1 = lin_reg.predict(x_train)
p2 = Dtree.predict(x_train)
p3 = Rand.predict(x_train)
p4 = svr.predict(x_train)
#p5 = XGB.predict(x_train)
```

```python
p1 = lin_reg.predict(x_test)
p2 = Dtree.predict(x_test)
p3 = Rand.predict(x_test)
p4 = svr.predict(x_test)
print(metrics. r2_score(p1,y_test))
print(metrics. r2_score(p2,y_test))
print(metrics. r2_score(p3,y_test))
print(metrics.r2_score(p4,y_test))
#print(metrics. r2_score(p5,y_test))
```

```
-5.491461561547912
0.7130190373733469
0.8117988884163669
-15966000.275938746
```

Model Validation and Evaluation Report:

Model	Regression Report	R2_score

Linear Regression	`p1 = lin_reg.predict(x_test)`
	`regression_report(y_test,p1)`
	
	`{'Mean Absolute Error (MAE)': 1637.9870039113694,`
	`'Mean Squared Error (MSE)': 3402975.5125765526,`
	`'Root Mean Squared Error (RMSE)': 1844.7155641389684,`
	`'R-squared (R²)': 0.1392528540190069,`
	`'Explained Variance Score': 0.13930894538755123}`
		13%

Decision Tree Regressor	`p2 = Dtree.predict(x_test)`
	`regression_report(y_test,p2)`
	
	`{'Mean Absolute Error (MAE)': 556.0734363655223,`
	`'Mean Squared Error (MSE)': 1118141.6407011722,`
	`'Root Mean Squared Error (RMSE)': 1057.4221676800482,`
	`'R-squared (R²)': 0.7171777397518407,`
	`'Explained Variance Score': 0.7173099360906563}`
		71%

Random Forest Regressor	`p3 = Rand.predict(x_test)`
	`regression_report(y_test,p3)`
	
	`{'Mean Absolute Error (MAE)': 494.5744746395602,`
	`'Mean Squared Error (MSE)': 612380.9824446529,`
	`'Root Mean Squared Error (RMSE)': 782.5477509038365,`
	`'R-squared (R²)': 0.8451046206638214,`
	`'Explained Variance Score': 0.8452203153920186}`
		84%

SVR	`p4 = svr.predict(x_test)`
	`regression_report(y_test,p4)`
	
	`{'Mean Absolute Error (MAE)': 1745.497301318169,`
	`'Mean Squared Error (MSE)': 3962326.1639990797,`
	`'Root Mean Squared Error (RMSE)': 1990.559259102597,`
	`'R-squared (R²)': -0.002229056454693401,`
	`'Explained Variance Score': 0.00012691427202182748}`
		0%

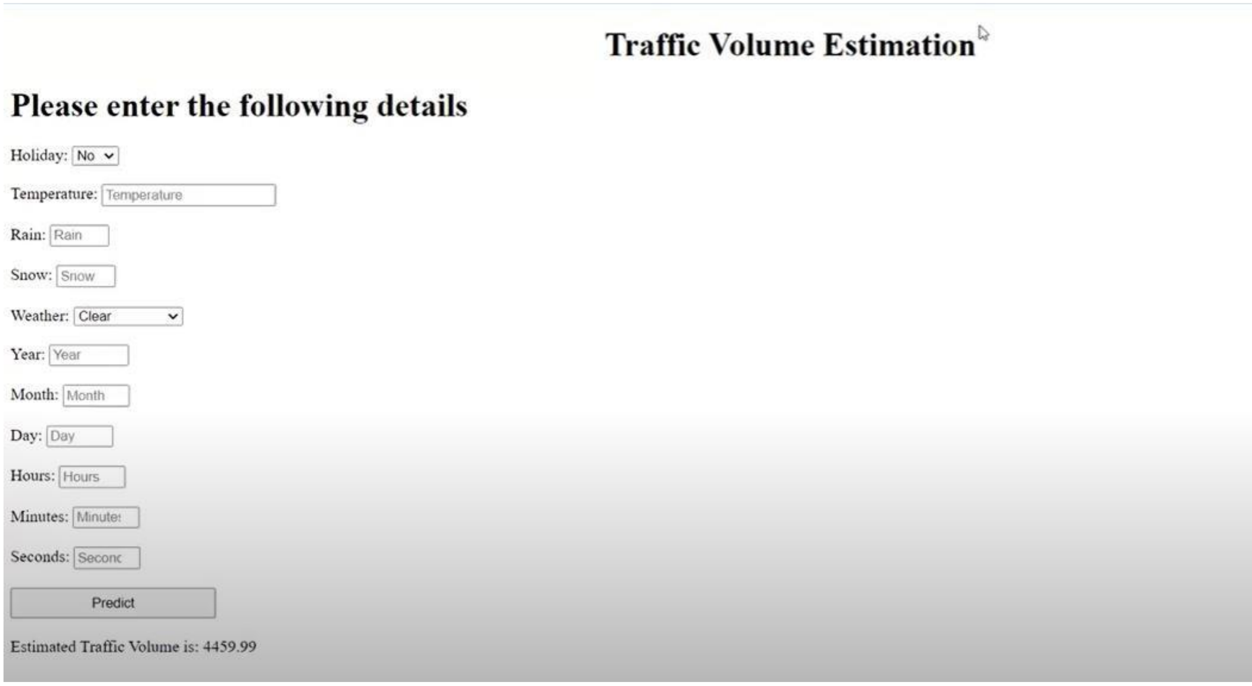
Final Model Selection Justifications

After extensive evaluation of various machine learning algorithms, we have selected the Random Forest Regressor as the final model for TrafficTelligence's traffic volume estimation system. This decision is based on several critical factors, primarily focusing on performance metrics, interpretability, and robustness. Below are the detailed justifications for our choice:

1. **High R^2 Score:** The Random Forest Regressor consistently demonstrated a high R^2 score across multiple validation datasets, indicating a strong ability to explain the variance in traffic volume data. This high R^2 score signifies that the model provides accurate predictions, which is crucial for real-time traffic management and planning applications.
2. **Performance Consistency:** During our evaluation, the Random

Forest Regressor outperformed other models such as Linear Regression, Decision Trees, and Support Vector Machines. It maintained high accuracy and low error rates across various scenarios, including different times of day, weather conditions, and special events. This consistency ensures reliable performance in diverse traffic situations.

3. **Scalability and Efficiency:** The Random Forest Regressor is highly scalable, capable of handling large datasets efficiently. Given the extensive historical traffic data and additional variables (such as weather patterns and event schedules), the model's ability to process large volumes of data swiftly is essential for real-time applications.



Results

FUTURE SCOPE

1. Integration of Real-time Data Sources: Expanding the system to integrate real-time data sources such as live traffic feeds, GPS data from connected vehicles, and IoT sensors installed at key traffic points can significantly enhance the accuracy and responsiveness of TrafficTelligence. This real-time data fusion will enable more dynamic and adaptive traffic management solutions.

2. Expansion to Multimodal Traffic Analysis: Future iterations of TrafficTelligence can include analysis and predictions for various modes of transportation, such as public transit, cycling, and pedestrian traffic. Incorporating multimodal traffic data will provide a holistic view of urban mobility, assisting in more comprehensive urban planning and improved multimodal transportation strategies.

3. Enhanced Predictive Analytics with Deep Learning: Exploring advanced deep learning techniques, such as recurrent neural networks (RNNs) or long short-term memory (LSTM) networks, could further improve the system's predictive capabilities. These models are particularly well-suited for time series data and can capture complex temporal dependencies in traffic patterns.

4. Integration with Smart City Infrastructure: TrafficTelligence can be integrated with broader smart city initiatives, collaborating with systems for energy management, environmental monitoring, and public safety. Such integration will contribute to creating more efficient, sustainable, and resilient urban environments.

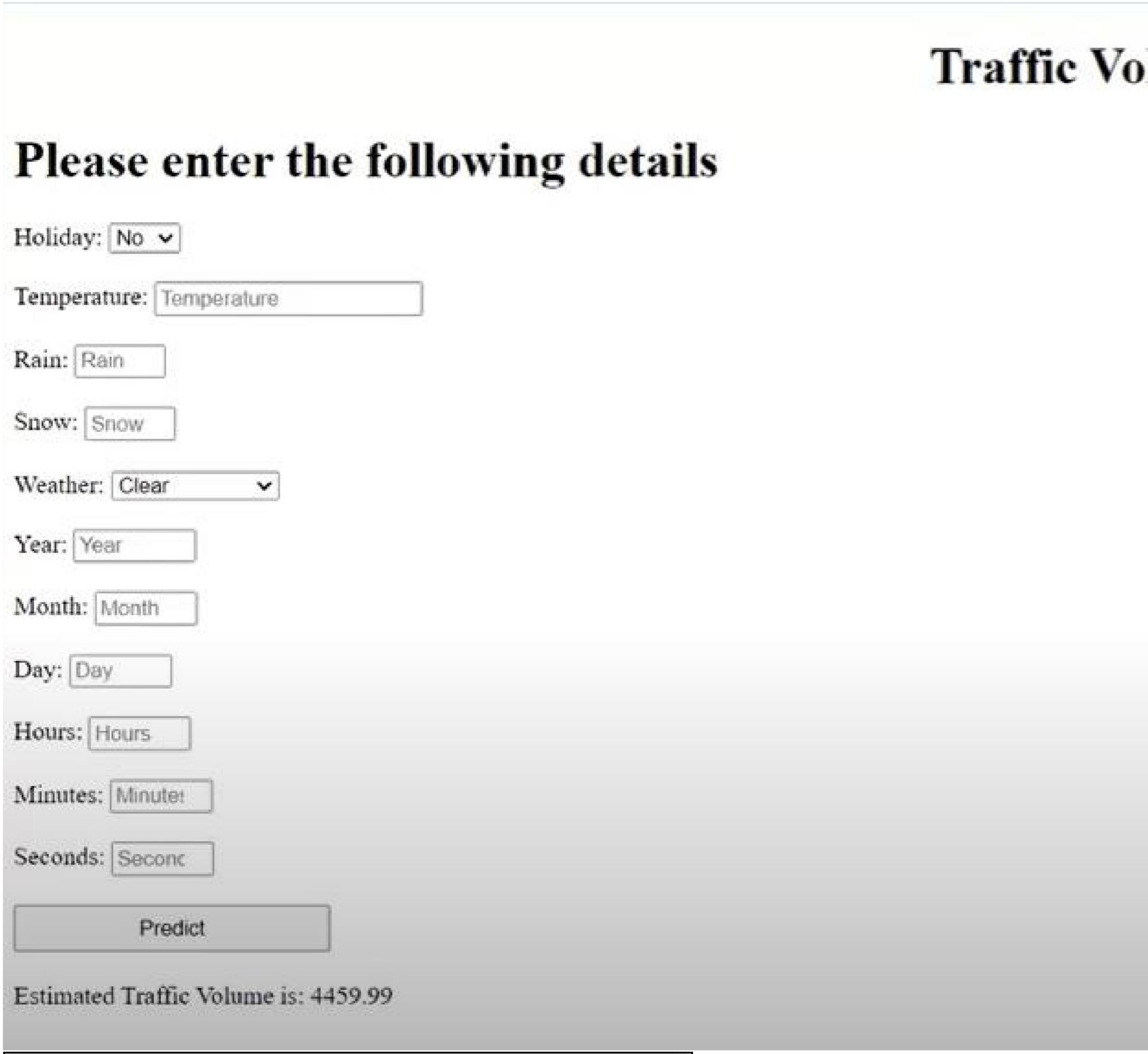
5. Predictive Maintenance for Traffic Infrastructure: Utilizing the traffic data to predict wear and tear on roads and traffic infrastructure can be a valuable addition. Predictive maintenance models can forecast when and where maintenance is needed, reducing disruptions and extending the lifespan of critical infrastructure.

6. User Personalization and Behavioral Insights: Developing personalized traffic predictions and recommendations based on individual commuter behavior and preferences can enhance user experience. Additionally, analyzing commuter behavior data can provide insights into travel habits and preferences, aiding in the design of more user-centric transportation solutions.

7. Scenario Planning and Simulation:
Incorporating simulation capabilities to model the impact of various scenarios, such as construction projects, policy changes, or emergency events, will help stakeholders make informed decisions. Scenario planning tools can provide valuable foresight and contingency strategies for effective traffic management.

APPENDIX

The project source code and various other requirements have been uploaded in our github. Demo link of our project is pasted at Project Delivarables.




```python
p4 = svr.predict(x_test)
regression_report(y_test,p4)
```

```
{'Mean Absolute Error (MAE)': 1745.497301318169,
 'Mean Squared Error (MSE)': 3962326.1639990797,
 'Root Mean Squared Error (RMSE)': 1990.559259102597,
 'R-squared (R²)': -0.002229056454693401,
 'Explained Variance Score': 0.00012691427202182748}
```

```python
p3 = Rand.predict(x_test)
regression_report(y_test,p3)
```

```
{'Mean Absolute Error (MAE)': 494.5744746395602,
 'Mean Squared Error (MSE)': 612380.9824446529,
 'Root Mean Squared Error (RMSE)': 782.5477509038365,
 'R-squared (R²)': 0.8451046206638214,
 'Explained Variance Score': 0.8452203153920186}
```

```python
p2 = Dtree.predict(x_test)
regression_report(y_test,p2)
```

```
{'Mean Absolute Error (MAE)': 556.0734363655223,
 'Mean Squared Error (MSE)': 1118141.6407011722,
 'Root Mean Squared Error (RMSE)': 1057.4221676800482,
 'R-squared (R²)': 0.7171777397518407,
 'Explained Variance Score': 0.7173099360906563}
```

```python
p1 = lin_reg.predict(x_test)
regression_report(y_test,p1)
```

```
{'Mean Absolute Error (MAE)': 1637.9870039113694,
 'Mean Squared Error (MSE)': 3402975.5125765526,
 'Root Mean Squared Error (RMSE)': 1844.7155641389684,
 'R-squared (R²)': 0.1392528540190069,
 'Explained Variance Score': 0.13930894538755123}
```

```
p1 = lin_reg.predict(x_test)
p2 = Dtree.predict(x_test)
p3 = Rand.predict(x_test)
p4 = svr.predict(x_test)
print(metrics. r2_score(p1,y_test))
print(metrics. r2_score(p2,y_test))
print(metrics. r2_score(p3,y_test))
print(metrics.r2_score(p4,y_test))
#print(metrics. r2_score(p5,y_test))
```

```
-5.491461561547912
0.713019037373346
0.811798884163669
-15966000.275938746
```

```
lin_reg.fit(x_train,y_train)
Dtree.fit(x_train,y_train)
Rand.fit(x_train,y_train)
svr.fit(x_train,y_train)
#XGB.fit(x_train,y_train)
p1 = lin_reg.predict(x_train)
p2 = Dtree.predict(x_train)
p3 = Rand.predict(x_train)
p4 = svr.predict(x_train)
#p5 = XGB.predict(x_train)
```

```python
from sklearn import linear_model
from sklearn import tree
from sklearn import ensemble
from sklearn import svm
```

```python
#splitting into independant and dependant variables
y=data['traffic_volume']
x=data.drop(columns=['traffic_volume'],axis=1)
print(x.head())

#splitting the data into train data and test data
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,rand
lin_reg = linear_model. LinearRegression()
Dtree = tree. DecisionTreeRegressor()
Rand = ensemble. RandomForestRegressor()
svr = svm. SVR( )
#XGB = xgboost . XGBRegressor ()
```

	holiday	temp	rain	snow	weather	traffic_volume	day	month	year	hours	minutes	seconds
39346	0	277.44	0.0	0.0	0	2859	29	11	2017	20	00	00
23628	0	296.46	0.0	0.0	1	4603	25	05	2016	18	00	00
6563	0	294.84	0.0	0.0	1	5635	31	05	2013	13	00	00
44041	0	279.69	0.0	0.0	0	622	13	05	2018	02	00	00
43918	0	290.46	0.0	0.0	6	3274	08	05	2018	19	00	00

```python
from sklearn.preprocessing import LabelEncoder

# Assuming your data is in a DataFrame called 'data'
# Assuming the weather column is named 'weather'

# Create a LabelEncoder object
le = LabelEncoder()

# Fit the LabelEncoder to the weather data (learn the categories)
le.fit(data['weather'])

# Transform the 'weather' column to numerical labels
data['weather'] = le.transform(data['weather'])
```

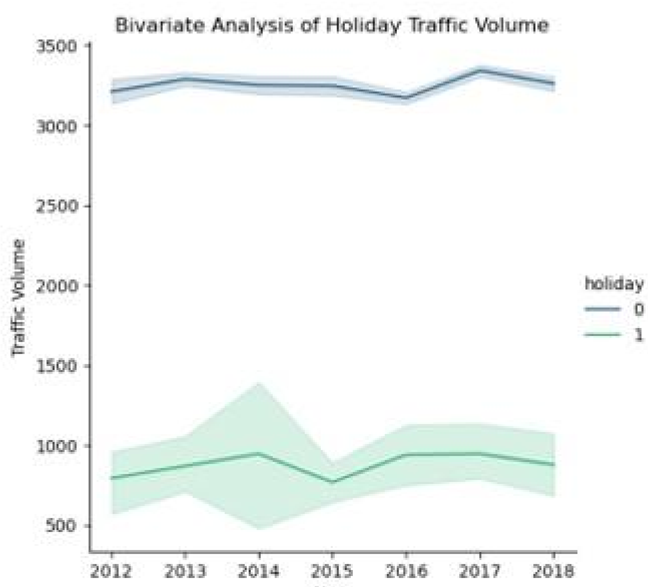
```python
holiday_list = ['Labor Day', 'Thanksgiving Day', 'Christmas Day', 'New Years Day',
                'Martin Luther King Jr Day', 'Columbus Day', 'Veterans Day',
              'Memorial Day', 'Independence Day', 'State Fair']
```

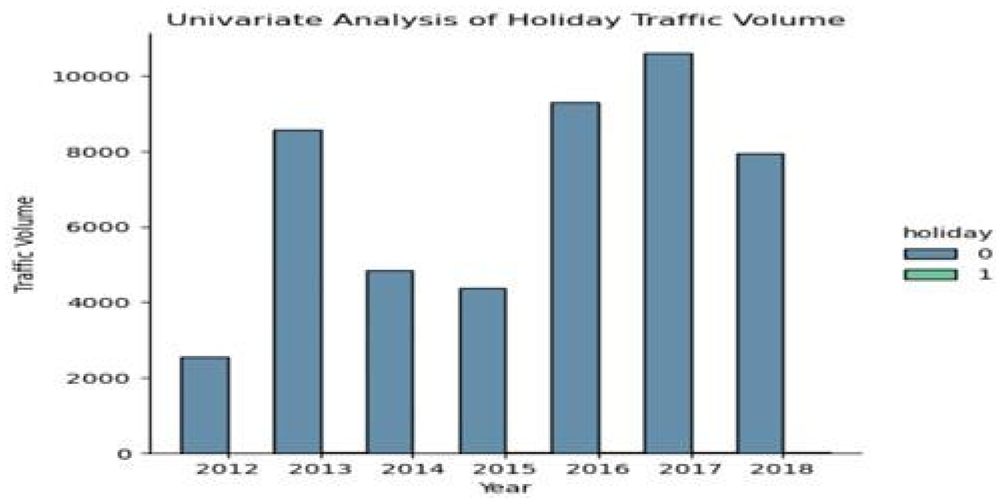
```python
data['holiday'] = data['holiday'].apply(lambda x: '1' if x in holiday_list else '0')
```

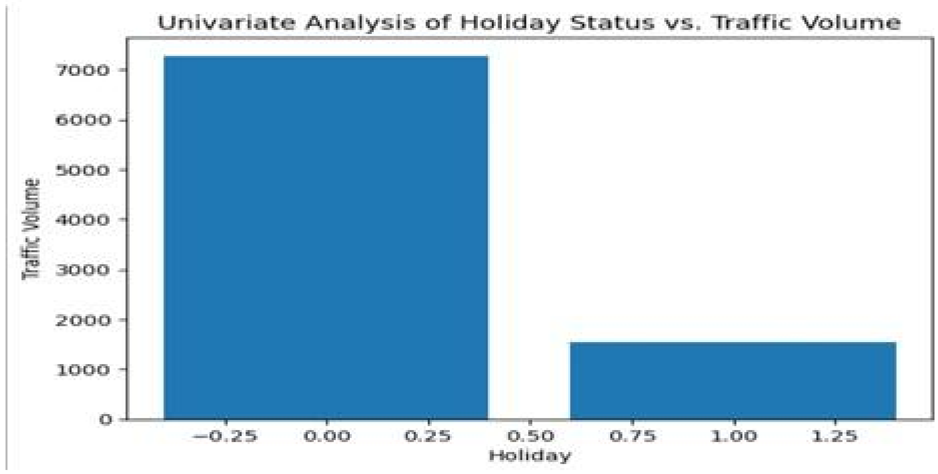
```python
data[["day","month","year"]] = data["date"].str.split("-", expand=True)
data[["hours","minutes","seconds"]] = data["Time"].str.split(":", expand=True)
data.drop(columns=['date','Time'],axis=1,inplace=True)
```

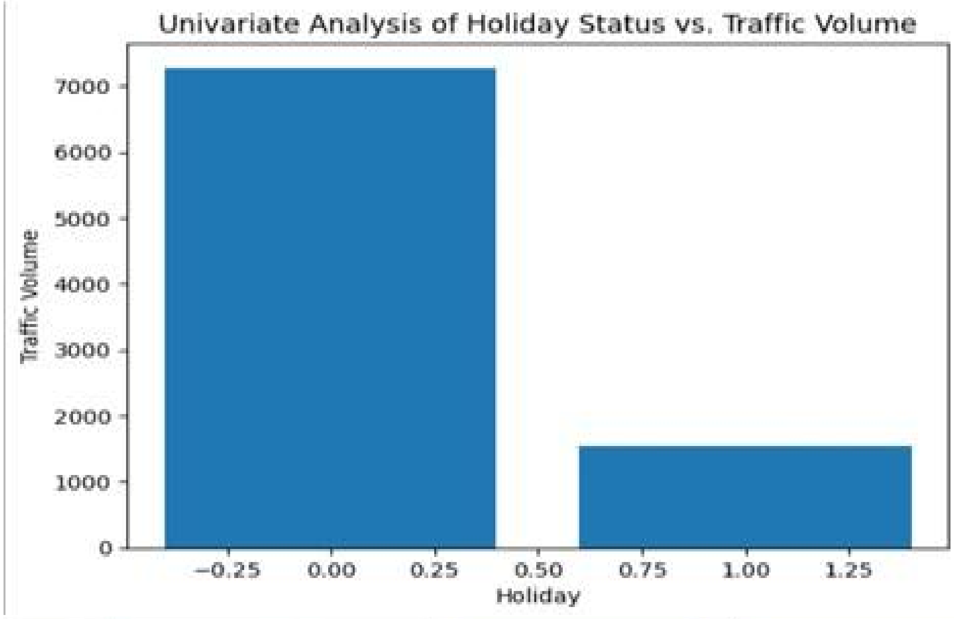
	holiday	temp	rain	snow	weather	traffic_volume	day	month	year	hours	minutes	seconds
39346	0	277.44	0.0	0.0	0	2859	29	11	2017	20	00	00
23628	0	296.46	0.0	0.0	1	4603	25	05	2016	18	00	00
6563	0	294.84	0.0	0.0	1	5635	31	05	2013	13	00	00
44041	0	279.69	0.0	0.0	0	622	13	05	2018	02	00	00
43918	0	290.46	0.0	0.0	6	3274	08	05	2018	19	00	00

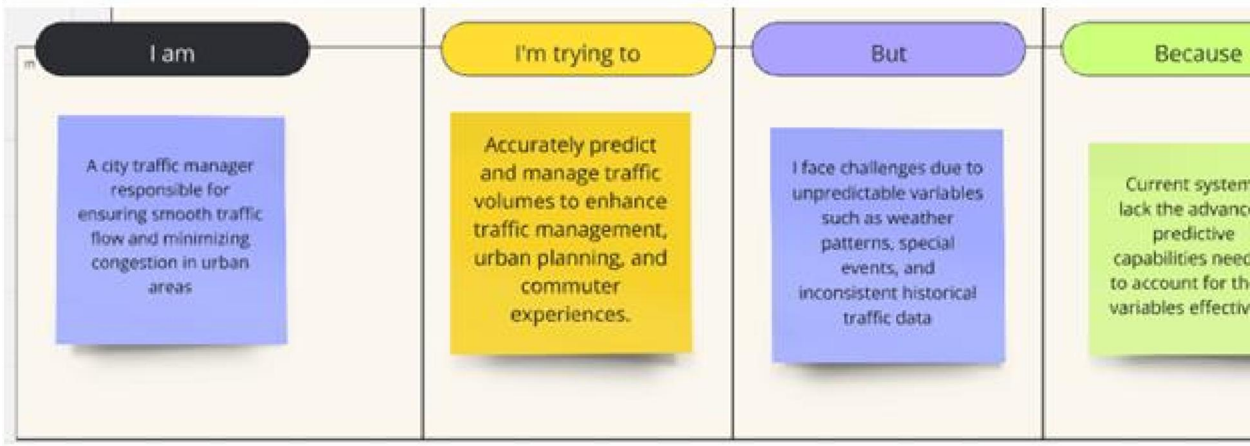
	holiday	temp	rain	snow	weather	date	Time	traffic_volume
0	NaN	288.28	0.0	0.0	Clouds	02-10-2012	09:00:00	5545
1	NaN	289.36	0.0	0.0	Clouds	02-10-2012	10:00:00	4516
2	NaN	289.58	0.0	0.0	Clouds	02-10-2012	11:00:00	4767
3	NaN	290.13	0.0	0.0	Clouds	02-10-2012	12:00:00	5026
4	NaN	291.14	0.0	0.0	Clouds	02-10-2012	13:00:00	4918




Univariate Analysis of Holiday Traffic Volume
10000
8000
6000
4000
2000
0
Traffic Volume
2012
2013
2014
2015
2016
2017
2018
Year
holiday
0
1


Univariate Analysis of Holiday Status vs. Traffic Volume
7000
6000
5000
4000
3000
2000
1000
0
Traffic Volume
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
Holiday


Univariate Analysis of Holiday Status vs. Traffic Volume
Traffic Volume
7000
6000
5000
4000
3000
2000
1000
0
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
Holiday


I am
I'm trying to
But
Because
A city traffic manager responsible for ensuring smooth traffic flow and minimizing congestion in urban areas
Accurately predict and manage traffic volumes to enhance traffic management, urban planning, and commuter experiences.
I face challenges due to unpredictable variables such as weather patterns, special events, and inconsistent historical traffic data