Final Project Report Template

1. Introduction

1.1. Project overviews

This project aims to assist fleet managers in optimizing fuel consumption and preventing fraud through a predictive machine learning model. By accurately estimating fuel usage for each vehicle, the solution addresses the inefficiencies and high costs associated with current reactive and inaccurate methods. The model leverages data to provide actionable insights, enabling proactive management of fleet operations. Key benefits include reduced operational costs, enhanced efficiency, and minimized fraudulent activities. Ultimately, this project provides a robust tool for fleet managers to optimize their fleets, ensuring cost-effectiveness and operational integrity.

1.2. Objectives

Accurate Fuel Consumption Prediction:

Develop a machine learning model to accurately predict fuel consumption for each vehicle in a fleet based on historical data and relevant features.

Fraud Detection and Prevention:

Implement mechanisms to identify discrepancies between predicted and actual fuel usage, detecting potential fraudulent activities.

Operational Optimization:

Enable fleet managers to optimize routes, driving behaviors, and maintenance schedules based on fuel consumption predictions, leading to reduced operational costs.

Data-Driven Decision Making:

Provide actionable insights derived from data to empower fleet managers in making informed decisions regarding fleet management.

Cost Reduction:

Achieve significant reductions in fuel costs by improving fuel efficiency and preventing fraud, ultimately enhancing the overall cost-effectiveness of fleet operations.

Environmental Impact:

Contribute to sustainability efforts by optimizing fuel usage and reducing carbon emissions from the fleet.

Scalability and Flexibility:

Design the solution to be scalable and adaptable to different fleet sizes and types, ensuring broad applicability and effectiveness.

User-Friendly Interface:

Develop an intuitive interface for fleet managers and drivers to easily access and interpret fuel consumption data and insights.

2. Project Initialization and Planning Phase

2.1. Define Problem Statement

Fleet management faces significant challenges in effectively monitoring and optimizing fuel consumption, which is a major operational cost. Additionally, fraudulent activities related to fuel usage can result in substantial financial losses and inefficiencies. Current methods of tracking fuel consumption are often inaccurate and reactive, lacking the predictive capabilities necessary for proactive management and fraud prevention. As a result, there is a pressing need for a robust and predictive solution to accurately estimate fuel consumption for each vehicle in a fleet. Such a solution will enable fleet managers to optimize operations, reduce costs, and prevent fraudulent activities, ensuring efficient and cost-effective fleet management.

2.2. Project Proposal (Proposed Solution)

The project involves a wide use of preprocessing and data cleaning techniques so as to deal with null values and outliers. It also uses tools to visualize the data using libraries such as numpy, seaborn and scikit-learn. It also uses aRandom Forest for prediction of results, leading to a good deal of accuracy. The unique aspects of the solution are that the interface is very simple to easy and straightforward to use, and that fleet managers who are not tech savvy, can use it without any issues. It also provides good accuracy on predictions.

2.3. Initial Project Planning

Sprint	Functional Requirement	User Story	User Story / Task	Story Points	Priority	Team Members	Sprint Start	Sprint End Date
Sprint-1	Data collection and	USN-1	Downloading the data	2	Medium	Y.Lakshmi Nikhitha	04-07-2024	05-07-2024
Sprint-1	Data collection and	USN-2	Data cleaning and EDA	2	Medium	Sai Divya	05-07-2024	06-07-2024
Sprint-	Model	USN-3	Training the	2	High	Sayuj Sunil	06-07-	07-07-
Sprint-	Model	USN-4	Evaluating the	2	Medium	Sai Divya	07-07-	07-07-
Sprint-	Model Tuning	USN-5	Model tuning	2	Medium	Y.Lakshmi	08-07-	08-07-
Sprint-	Model Tuning	USN-6	Model tuning	2	Medium	Vaibhav D	08-07-	11-07-
Sprint-4	Web integration and	USN-7	Creating HTML pages	2	High	Vaibhav D	08-07-2024	08-07-2024

Sprint-4	Web integration and	USN-8	Deployment	2	High	Sayuj Sunil	11-07-2024	11-07-202411-

3. Data Collection and Preprocessing Phase

3.1. Data Collection Plan and Raw Data Sources Identified

The two datasets used for analysis were picked up from kaggle, and the data sources that were identified were:

Source Name	**Description**	**Location/URL**	**Format**	**Size**	**Access Permissions**
measurements.csv	Contains the information of a fleet, i.e. various features of the individual vehicle.	Link of Dataset 1	CSV	14.2 kB	Public
Measurement2.xlsx	Similar to the previous	Link of Dataset 2	Excel	26.8 kB	Public

3.2. Data Quality Report

Data Source	**Data Quality Issue**	**Severity**	**Resolution Plan**
Dataset 1	The float values in the dataset were represented with commas, instead of decimal points.	Moderate	Used a for loop to traverse through the columns and check that If the type is ‘object’ instead of ‘float’, replace the comma (if any) with a full stop.
Dataset 2	Poor performance on second dataset as compared to the first.	High	Using only the first dataset for training and testing the model.

3.3. Data Exploration and Preprocessing

Data overview:

The dataset contains 488 records with 12 column names (12 prospective features before cleaning). The columns are distance, consume, speed, temp_inside, temp_outside, specials, gas_type, AC, rain, sun, refill liters (contains majority null values), refill gas (contains majority null values)

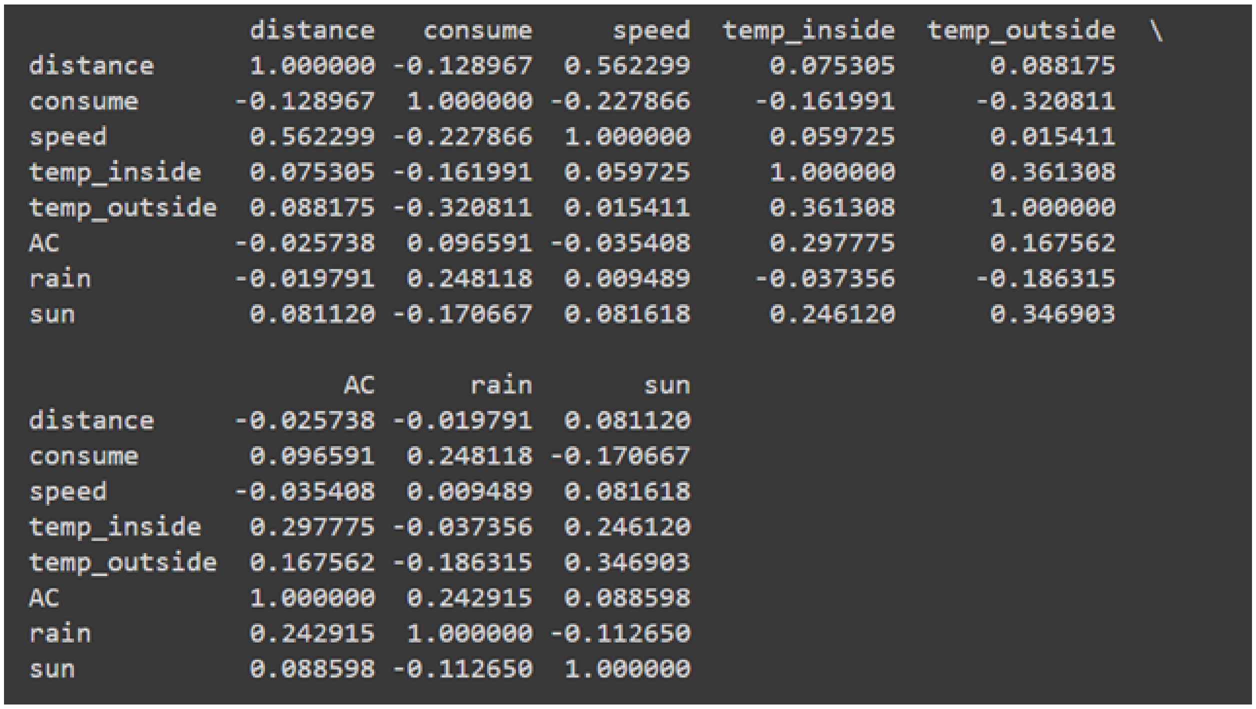
Univariate Analysis:

The mean, median and mode of the corresponding columns are as follows:

```
                  Mean       median   mode
distance      19.652835    14.6  11.8
consume        4.912371     4.7   4.5
speed         41.927835    40.5  42.0
temp_inside   21.929521    22.0  21.5
temp_outside  11.358247    10.0   8.0
AC             0.077320     0.0   0.0
```

rain	0.123711	0.0	0.0
sun	0.082474	0.0	0.0

Bivariate Analysis:



```
             distance   consume     speed  temp_inside  temp_outside  \
distance     1.000000 -0.128967  0.562299     0.075305      0.088175
consume     -0.128967  1.000000 -0.227866    -0.161991     -0.320811
speed        0.562299 -0.227866  1.000000     0.059725      0.015411
temp_inside  0.075305 -0.161991  0.059725     1.000000      0.361308
temp_outside 0.088175 -0.320811  0.015411     0.361308      1.000000
AC          -0.025738  0.096591 -0.035408     0.297775      0.167562
rain        -0.019791  0.248118  0.009489    -0.037356     -0.186315
sun          0.081120 -0.170667  0.081618     0.246120      0.346903

                   AC      rain       sun
distance    -0.025738 -0.019791  0.081120
consume      0.096591  0.248118 -0.170667
speed       -0.035408  0.009489  0.081618
temp_inside  0.297775 -0.037356  0.246120
temp_outside 0.167562 -0.186315  0.346903
AC           1.000000  0.242915  0.088598
rain         0.242915  1.000000 -0.112650
sun          0.088598 -0.112650  1.000000
```

Multivariate Analysis:

On performing feature selection with SelectKBest and f_regression, we get the following result:

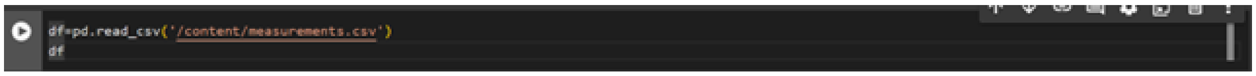
```
Selected features: Index(['speed', 'temp_inside', 'temp_outside', 'rain',
'sun'], dtype='object')
```

Outliers and Anomalies:

Identification is done by using the IQR method. We define the first and third quartile’s boundaries and check for the values that lie outside and inside the boundaries.

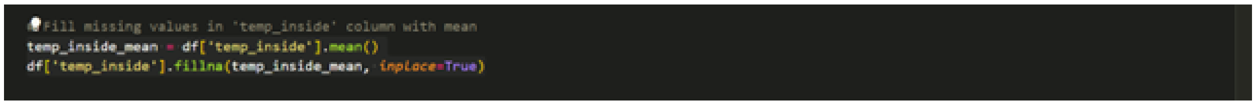
Data preprocessing code screenshots:

Loading data:



```
df=pd.read_csv('/content/measurements.csv')
df
```

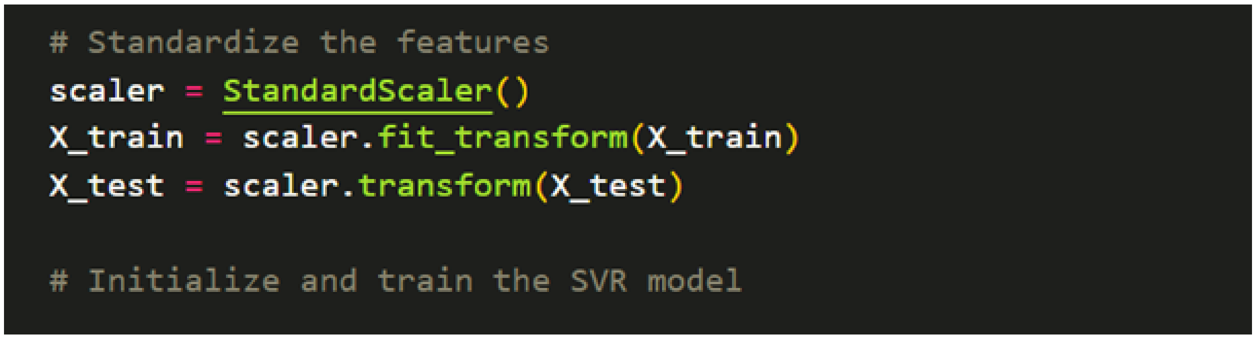
Handling missing data:



```
# Fill missing values in 'temp_inside' column with mean
temp_inside_mean = df['temp_inside'].mean()
df['temp_inside'].fillna(temp_inside_mean, inplace=True)
```

Data Transformation:

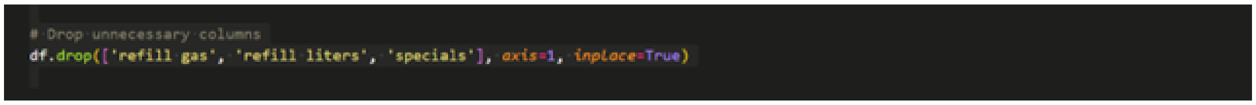
Code for transforming variables (scaling, normalization).



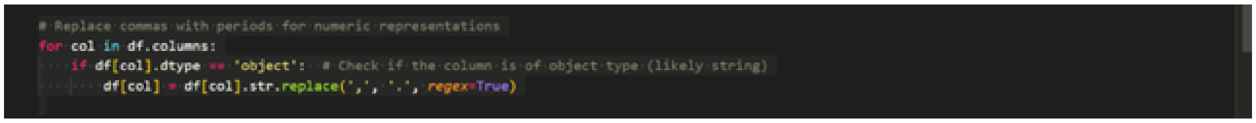
```
# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize and train the SVR model
```

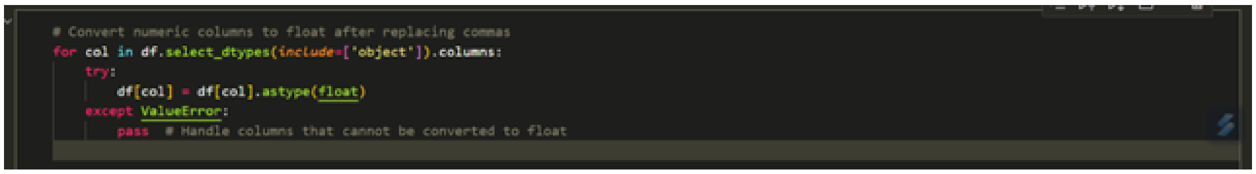
Feature Engineering:



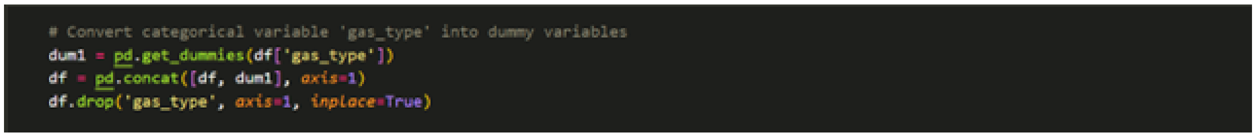
```
# Drop unnecessary columns
df.drop(['refill gas', 'refill liters', 'specials'], axis=1, inplace=True)
```



```
# Replace commas with periods for numeric representations
for col in df.columns:
    if df[col].dtype == 'object':  # Check if the column is of object type (likely string)
        df[col] = df[col].str.replace(',', '.', regex=True)
```

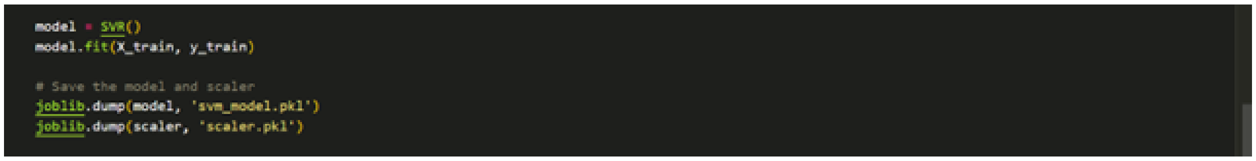


```
# Convert numeric columns to float after replacing commas
for col in df.select_dtypes(include=['object']).columns:
    try:
        df[col] = df[col].astype(float)
    except ValueError:
        pass  # Handle columns that cannot be converted to float
```



```
# Convert categorical variable 'gas_type' into dummy variables
dum1 = pd.get_dummies(df['gas_type'])
df = pd.concat([df, dum1], axis=1)
df.drop('gas_type', axis=1, inplace=True)
```

Save processed data:



```
model = SVR()
model.fit(X_train, y_train)

# Save the model and scaler
joblib.dump(model, 'svm_model.pkl')
joblib.dump(scaler, 'scaler.pkl')
```

4. Model Development Phase

4.1. Feature Selection Report

Feature	**Description**	**Selected (Yes/No)**	**Reasoning**
Distance	This is the distance that the fleet has travelled.	Yes	On performing multivariate analysis, it was found that this feature had the most influence on the target variable.
consume	Consumption of each vehicle in the fleet.	Yes (target variable)	The target variable that is to be predicted.
speed	**Speed of the vehicle in the fleet**	**Yes**	**Multivariate analysis results**
Temp_inside	**Temperature inside the vehicle**	**Yes**	**Multivariate analyisis result**

Temp_outside	**Atmospheric temperature**	**Yes**	**Multivariate analyisis result**
specials	**The weather on the day the vehicle was travelling.**	**No**	**Too many values, it was an unnecessary column.**
Gas_type	**Gas type either E10 or SP98**	**Yes**	**Amount of fuel depends on what gas type**
Rain	**Whether it was raining (denoted by 0 or 1)**	**Yes**	**Multivariate analyisis result**
Sun	**Whether it was sunny (denoted by 0 or 1)**	**Yes**	**Multivariate analyisis result**
Refill liters	**Liters of fuel refilled**	**No**	**Too many null values**
Refill gas	**Type of gas used to refill**	**No**	**Too many null values**
AC	**Whether the AC was on in the car (denoted by 0 or 1)**	**Yes**	**Less important feature but still important In bivariate analysis**

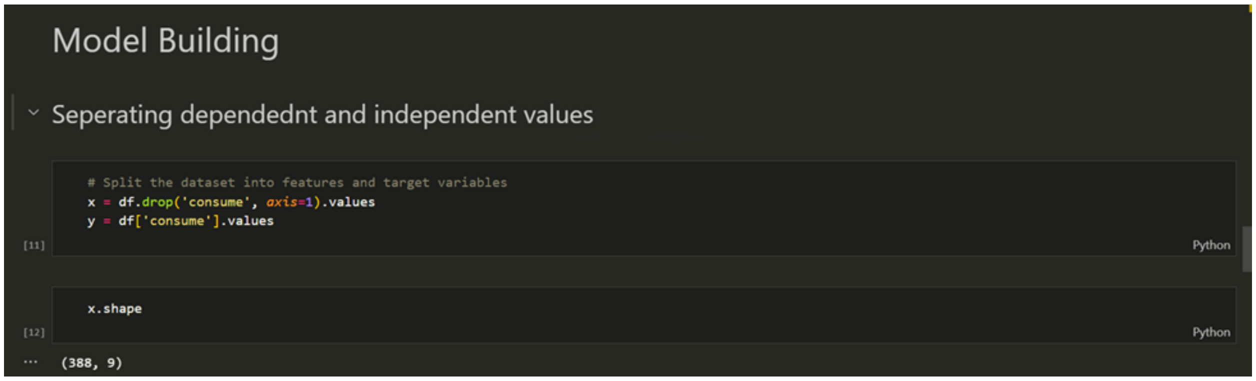
4.2. Model Selection Report

Model	Description	Hyperparameters	Performance Metric (e.g., Accuracy, F1 Score)
Random Forest	Random Forest is an ensemble learning method used for both regression and classification tasks. It operates by constructing a multitude of decision trees during training and outputting the average prediction (regression) or the mode of the classes (classification) of the individual trees.	N/A	Average RMSE: 0.6887571449768667 Standard Deviation of RMSE: 0.0919128258979233 Mean Absolute Error: 0.4446656898656903 Mean Squared Error: 0.3821795521791004 R^2 Score: 0.5436583449392025

Support Vector Regression	Support Vector Regression (SVR) is a type of Support Vector Machine (SVM) that is used for regression tasks. It works by finding the hyperplane that best fits the data points within a certain margin of tolerance. SVR aims to minimize the error while maintaining the complexity of the model within a certain threshold.	N/A	Performance metric value
…	**…**	**…**	**…**

4.3. Initial Model Training Code, Model Validation and Evaluation Report

Screenshots of the code:



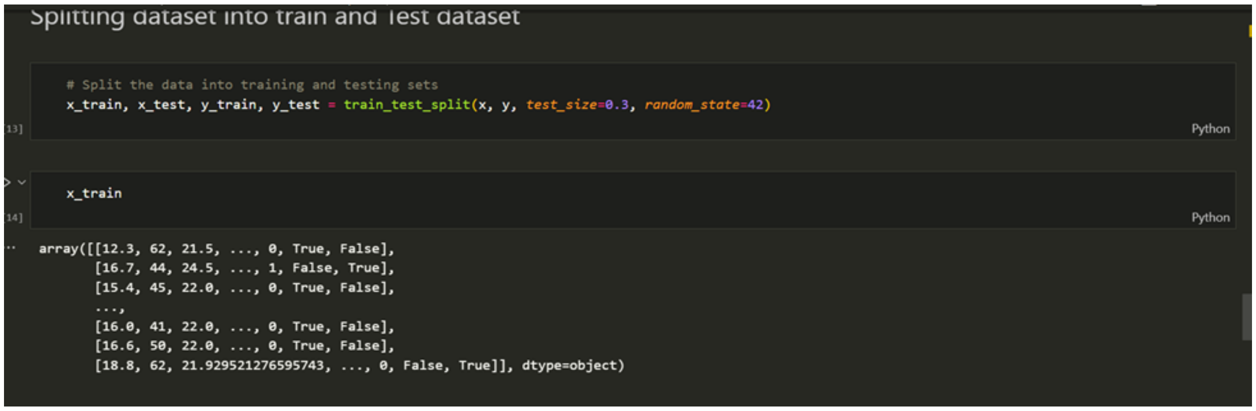
Model Building

Seperating dependednt and independent values

```
[11]
# Split the dataset into features and target variables
x = df.drop('consume', axis=1).values
y = df['consume'].values
```

```
[12]
x.shape
```

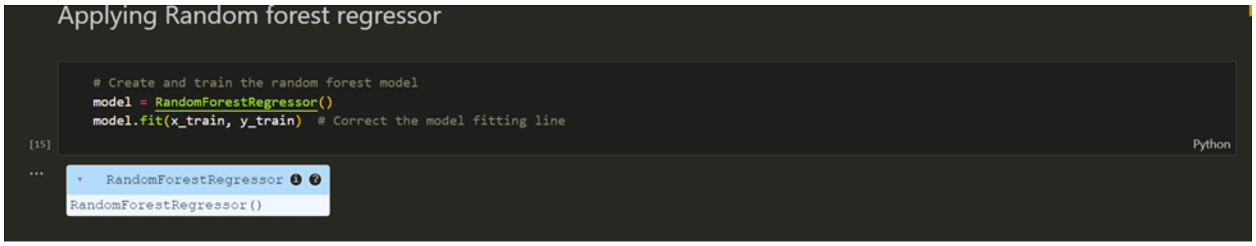
```
(388, 9)
```


Splitting dataset into train and Test dataset

```python
# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

```python
x_train
```

```
array([[12.3, 62, 21.5, ..., 0, True, False],
       [16.7, 44, 24.5, ..., 1, False, True],
       [15.4, 45, 22.0, ..., 0, True, False],
       ...,
       [16.0, 41, 22.0, ..., 0, True, False],
       [16.6, 50, 22.0, ..., 0, True, False],
       [18.8, 62, 21.929521276595743, ..., 0, False, True]], dtype=object)
```


Applying Random forest regressor

```python
# Create and train the random forest model
model = RandomForestRegressor()
model.fit(x_train, y_train)  # Correct the model fitting line
```

```
RandomForestRegressor
RandomForestRegressor()
```

Model validation and evaluation report:

Model	Classification Report	Accuracy	Confusion Matrix
Random Forest	Not a classification problem so invalid	Error metrics: Mean Absolute Error: 0.4446656898656903 Mean Squared Error: 0.3821795521791004 R^2 Score: 0.5436583449392025	Invalid
Support Vector	Not a classification	-	Invalid

Regression	problem so invalid		

5. Model Optimization and Tuning Phase

5.1. Hyperparameter Tuning Documentation

We did using SVM,Linear Regression and Random Forest.

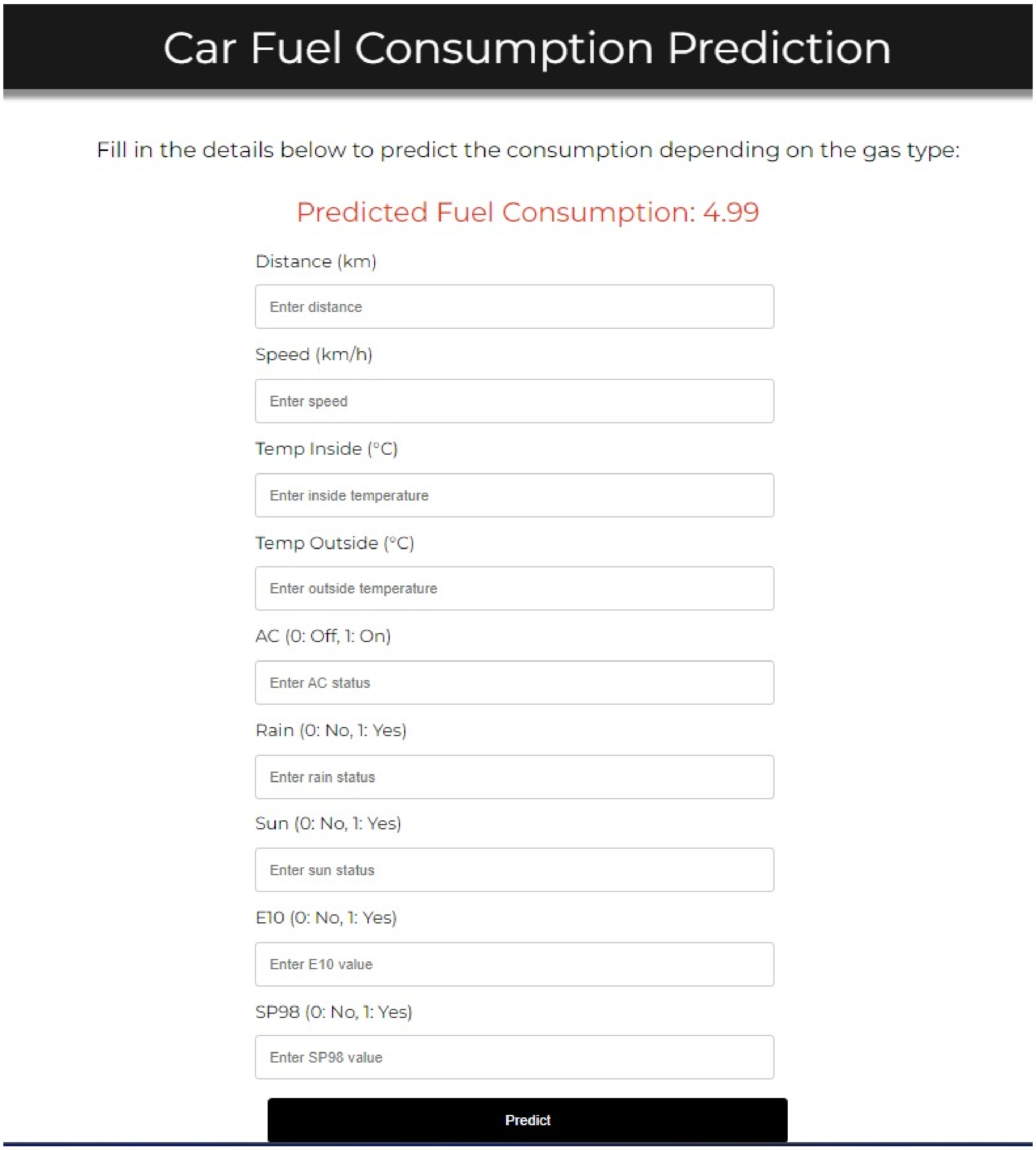
5.2. Performance Metrics Comparison Report

5.3. Final Model Selection Justification

The final model was chosen because it presented better accuracy as compared to the other options, namely linear regression and support vector regression.

6. Results

6.1. Output Screenshots


Car Fuel Consumption Prediction

Fill in the details below to predict the consumption depending on the gas type:

Predicted Fuel Consumption: 4.99

Distance (km)	Enter distance
Speed (km/h)	Enter speed
Temp Inside (°C)	Enter inside temperature
Temp Outside (°C)	Enter outside temperature
AC (0: Off, 1: On)	Enter AC status
Rain (0: No, 1: Yes)	Enter rain status
Sun (0: No, 1: Yes)	Enter sun status
E10 (0: No, 1: Yes)	Enter E10 value
SP98 (0: No, 1: Yes)	Enter SP98 value

7. Advantages & Disadvantages

Advantages:

1. Increased Accuracy:

   - Machine learning models can analyze complex patterns and relationships in data, leading to more accurate fuel consumption predictions than traditional methods.

2. Proactive Management:

   - Predictive capabilities enable fleet managers to anticipate fuel needs and optimize routes and schedules in advance, reducing wastage and improving efficiency.

3. Fraud Detection:

   - The model can identify anomalies in fuel consumption, helping to detect and prevent fraudulent activities.

4. Cost Reduction:

   - By optimizing fuel usage, the model helps in significantly reducing operational costs associated with fuel consumption.

5. Data-Driven Decisions:

   - Provides fleet managers with actionable insights derived from data, enabling informed decision-making and strategic planning.

6. Scalability:

   - The solution can be scaled to accommodate fleets of different sizes and types, making it adaptable to various operational contexts.

7. Environmental Impact:

   - Optimized fuel consumption leads to lower carbon emissions, contributing to environmental sustainability efforts.

8. Real-Time Monitoring:

   - Continuous data analysis allows for real-time monitoring and adjustments, ensuring ongoing optimization of fuel usage.

Disadvantages:

1. Initial Setup Cost:

   - Implementing a machine learning model involves initial costs for data collection, infrastructure, and model development.

2. Data Quality and Availability:

   - The accuracy of the model depends on the quality and quantity of historical data available. Incomplete or inaccurate data can affect predictions.

3. Complexity:

   - Developing and maintaining a machine learning model requires specialized knowledge and skills, which may necessitate hiring experts or training existing staff.

4. Integration Challenges:

   - Integrating the model with existing fleet management systems and processes can be complex and time-consuming.

5. Maintenance and Updates:

   - The model will require regular updates and maintenance to ensure its accuracy and relevance as operational conditions and data patterns change.

6. Dependence on Technology:

   - Reliance on a machine learning model means that technical issues or system failures could disrupt fuel consumption monitoring and management.

7. Resistance to Change:

   - Fleet managers and drivers may resist adopting new technology and changing established practices, requiring change management efforts.

8. Privacy and Security Concerns:

- Handling large amounts of data, including potentially sensitive information, raises concerns about data privacy and security that must be managed effectively.

8. Conclusion:

Implementing a machine learning model to predict fleet fuel consumption offers significant benefits, including increased accuracy, cost reduction, fraud detection, and enhanced operational efficiency. The model's predictive capabilities empower fleet managers with data-driven insights, enabling proactive management and strategic decision-making. Additionally, optimized fuel usage contributes to environmental sustainability by reducing carbon emissions.

However, the project also presents challenges, such as initial setup costs, data quality requirements, and integration complexities. Ongoing maintenance and updates are necessary to ensure the model remains effective and relevant. Despite these challenges, the advantages of adopting a predictive fuel consumption model outweigh the disadvantages, making it a valuable investment for fleet management.

Overall, the project promises to transform fleet operations by providing a robust tool for optimizing fuel consumption, reducing costs, and preventing fraud, ultimately leading to more efficient and cost-effective fleet management.

9. Future Scope

1. IoT Integration:Incorporate real-time data from IoT devices to enhance prediction accuracy.
2. Advanced Analytics: Utilize deep learning for improved predictive capabilities.
3. Expanded Metrics: Extend predictions to other KPIs like maintenance and tire wear.
4. Personalized Insights: Provide tailored feedback to drivers for fuel-efficient driving.
5. Seamless Integration: Integrate with existing fleet management systems.
6. Geographic Adaptation: Adapt the model for various regions and driving conditions.
7. Predictive Maintenance: Forecast and schedule vehicle maintenance.
8. Regulatory Compliance: Assist in meeting fuel and emissions regulations.

9. Mobile Access: Develop a user-friendly mobile app for real-time insights.

10. Appendix

10.1. Source Code

```
 import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
import joblib

df = pd.read_csv("measurements.csv")
print(df.head())
# Iterate through all columns and replace commas with periods for numeric
representations
for col in df.columns:
    if df[col].dtype == 'object':  # Check if the column is of object type (likely string)
        df[col] = df[col].str.replace(',', '.', regex=True)
## finding count of missing values
df.isna().sum()

# Drop unnecessary columns
df.drop(['refill gas', 'refill liters', 'specials'], axis=1, inplace=True)

# Convert numeric columns to float after replacing commas
for col in df.select_dtypes(include=['object']).columns:
    try:
        df[col] = df[col].astype(float)
    except ValueError:
        pass  # Handle columns that cannot be converted to float
# Convert numeric columns to float after replacing commas
for col in df.select_dtypes(include=['object']).columns:
    try:
        df[col] = df[col].astype(float)
    except ValueError:
        pass  # Handle columns that cannot be converted to float
# Convert categorical variable 'gas_type' into dummy variables
dum1 = pd.get_dummies(df['gas_type'])
df = pd.concat([df, dum1], axis=1)
df.drop('gas_type', axis=1, inplace=True)
# Split the dataset into features and target variables
x = df.drop('consume', axis=1).values
y = df['consume'].values
x.shape
# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
# Create and train the random forest model
model = RandomForestRegressor()
model.fit(x_train, y_train)  # Correct the model fitting line
# Make predictions on the test set
y_pred = model.predict(x_test)
print(y_pred)
# Evaluate the model
mse = metrics.mean_squared_error(y_test, y_pred)
mae = metrics.mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)
```

```
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("Root Mean Squared Error:", rmse)
# Save the model
joblib.dump(model, 'fleet_fuel.pkl')
from sklearn.model_selection import KFold, cross_val_score

# Choose a model (replace with your desired algorithm)
model = RandomForestRegressor()

# Set up k-fold cross-validation (e.g., 5 folds)
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Calculate cross-validated RMSE (you can also use MAE or other metrics)
rmse_scores = cross_val_score(model, x, y, cv=kf,
scoring='neg_root_mean_squared_error')

# Convert negative RMSE scores to positive
rmse_scores = -rmse_scores

# Print average RMSE and standard deviation
print("Average RMSE:", rmse_scores.mean())
print("Standard Deviation of RMSE:", rmse_scores.std())

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Mean Absolute Error
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error: {mae}")
```

```
# Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")

# R² Score
r2 = r2_score(y_test, y_pred)
print(f"R² Score: {r2}")
```

10.2. GitHub & Project Demo Link

Github link:

https://github.com/Saidivyakarnati11/Predicting-Modelling-for-Feet-Fuel-Management-using-machine-learning-techniques

Project demo link:

https://www.youtube.com/watch?v=mgYD-LQWc8c&t=11s