Assessment Report

on

"Vehicle Emission Classification Report"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Name : Aditi Verma

Roll Number : 202401100400014

Section: A

Under the supervision of

"BIKKI KUMAR"

KIET Group of Institutions, Ghaziabad

APRIL, 2025

---

1. Introduction

This report focuses on predicting the emission category of vehicles based on engine and fuel features. The dataset includes factors such as engine size, fuel type, and CO_2 emissions, which are used to classify vehicles into different emission categories. The goal of this analysis is to predict the emission category of vehicles and to explore which factors most influence this classification through exploratory data analysis (EDA).

---

2. Problem Statement

Predict emission category of a vehicle based on engine and fuel features

---

3. Objectives

- Preprocess the dataset for training a machine learning model.
- Train a Logistic Regression model to classify loan defaults.
- Evaluate model performance using standard classification metrics.
- Visualize the confusion matrix using a heatmap for interpretability.

---

4. Methodology

Data Collection: The user uploads a CSV file containing the vehicle emissions dataset.

Data Preprocessing:

- **Handling missing values**: Missing numerical values are filled using the mean of respective columns.

- **Label encoding**: Categorical variables like 'fuel type' and 'emission category' are encoded using LabelEncoder.
- **Feature scaling**: The data is scaled using StandardScaler to normalize the feature values.

Model Building:

- **Splitting the dataset** into training (80%) and testing (20%) sets using train_test_split.
- **Training a Random Forest Classifier** to classify vehicles into emission categories based on the preprocessed data.

Model Evaluation:

- **Evaluating performance** using accuracy, precision, recall, and F1-score.
- **Confusion Matrix**: A confusion matrix is generated and visualized with a heatmap to understand prediction errors.

---

5. Data Preprocessing

The dataset is cleaned and prepared as follows:

- **Handling Missing Values**: Missing numerical values are filled using the mean of their respective columns.
- **Categorical Encoding**: The categorical variable 'fuel type' is encoded using LabelEncoder for compatibility with machine learning models.
- **Feature Scaling**: The features are normalized using `StandardScaler` to scale the values into a similar range.
- **Splitting the Data**: The dataset is split into 80% training and 20% testing sets for model evaluation.

---

6. Model Implementation

A **Random Forest Classifier** was chosen due to its effectiveness in handling complex datasets and its ability to provide robust predictions. The classifier is trained using the processed dataset to predict the emission category of vehicles. After training, the model is evaluated on the test set to assess its performance.

---

7. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy**: Measures the overall correctness of the model by calculating the percentage of correctly predicted classifications.
- **Precision**: Indicates the proportion of predicted emission categories that are correctly classified.
- **Recall**: Shows the proportion of actual emission categories that were correctly identified.
- **F1-Score**: The harmonic mean of precision and recall, providing a balance between the two.
- **Confusion Matrix**: The confusion matrix is visualized using Seaborn heatmap to understand prediction errors.

---

8. Results and Analysis

- The **Random Forest Classifier** provided reasonable performance on the test set, with strong metrics for both precision and recall.
- The **Confusion Matrix heatmap** helped identify the balance between true positives (correct predictions) and false negatives (incorrect predictions).
- **Precision and recall** indicated that the model performed well at detecting vehicles within the correct emission categories, with some minor misclassifications.

---

9. Conclusion

The **Random Forest Classifier** successfully predicted the emission category of vehicles with satisfactory performance metrics. The project demonstrates the potential of machine learning in classifying vehicles based on engine and fuel characteristics. However, improvements can be made by exploring more advanced models and handling imbalanced data to further improve the model's performance.

---

10. References

Scikit-learn documentation: https://scikit-learn.org

- **Pandas documentation**: https://pandas.pydata.org
- **Seaborn visualization library**: https://seaborn.pydata.org
- **Research articles on vehicle emission prediction**: Used for referencing relevant studies and methodologies.

```
[1] import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
```

```
[2] df=pd.read_csv('/content/vehicle_emissions.csv')
```

```
df.sample(5)
```

	engine_size	fuel_type	co2_emissions	emission_category
13	4.320879	electric	52.371358	A
4	1.416434	diesel	269.166344	A
83	1.981396	electric	282.897372	C
49	3.533913	electric	259.456434	C
39	4.054423	electric	114.745663	A

```
[4] print(df['fuel_type'].unique())
```

```
['petrol' 'electric' 'diesel']
```

```
[5] from sklearn.preprocessing import LabelEncoder, StandardScaler
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import classification_report, confusion_matrix
```

```
[6] le_fuel = LabelEncoder()
    df['fuel_type'] = le_fuel.fit_transform(df['fuel_type'])
```

```
[7] df.sample(5)
```

	engine_size	fuel_type	co2_emissions	emission_category
81	1.331195	1	58.855592	A
60	3.306066	0	104.081006	C
78	1.187864	1	169.169207	B
23	1.725742	0	180.629798	A
43	4.578209	2	200.199612	A

```
# Print the mapping of original labels to numbers
for i, class_label in enumerate(le_fuel.classes_):
    print(f"{class_label} → {i}")
```

```
diesel → 0
electric → 1
petrol → 2
```

```
[9] le_emission = LabelEncoder()
    df['emission_category'] = le_emission.fit_transform(df['emission_category'])
```

```
[10] X = df[['engine_size', 'fuel_type', 'co2_emissions']]
```

```
[10] X = df[['engine_size', 'fuel_type', 'co2_emissions']]
     y = df['emission_category']
```

```
[11] # Scale features
     scaler = StandardScaler()
     X_scaled = scaler.fit_transform(X)
```

```
[12] # Step 3: Train-Test Split
     X_train, X_test, y_train, y_test = train_test_split(
         X_scaled, y, test_size=0.2, random_state=42
     )
```

```
# Step 4: Train Classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
[14]
     # Step 5: Predictions
     y_pred = clf.predict(X_test)
```

```
[15] # Step 6: Evaluation
     # Classification Report
     report = classification_report(
```


```
[14] # Step 5: Predictions
     y_pred = clf.predict(X_test)
```


```
[15] # Step 6: Evaluation
     # Classification Report
     report = classification_report(
         y_test, y_pred, target_names=le_emission.classes_, output_dict=True
     )
     report_df = pd.DataFrame(report).transpose()
     print("Classification Report:\n", report_df[['precision', 'recall', 'f1-score', 'support']])
```

```
Classification Report:
               precision    recall  f1-score  support
A              0.400000  0.400000  0.400000      5.0
B              0.250000  0.333333  0.285714      6.0
C              0.571429  0.444444  0.500000      9.0
accuracy       0.400000  0.400000  0.400000      0.4
macro avg      0.407143  0.392593  0.395238     20.0
weighted avg   0.432143  0.400000  0.410714     20.0
```

```
[16] # Confusion Matrix
     cm = confusion_matrix(y_test, y_pred)
```

```
cm
```

```
array([[2, 2, 1],
       [2, 2, 2],
       [1, 4, 4]])
```

```python
# Heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(
    cm, annot=True, fmt='d', cmap='Blues',
    xticklabels=le_emission.classes_,
    yticklabels=le_emission.classes_
)
plt.title("Confusion Matrix Heatmap")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```