Name: Anish Kulkarni Roll No.: 29 Class: D15C AY: 2024-25

Experiment 6

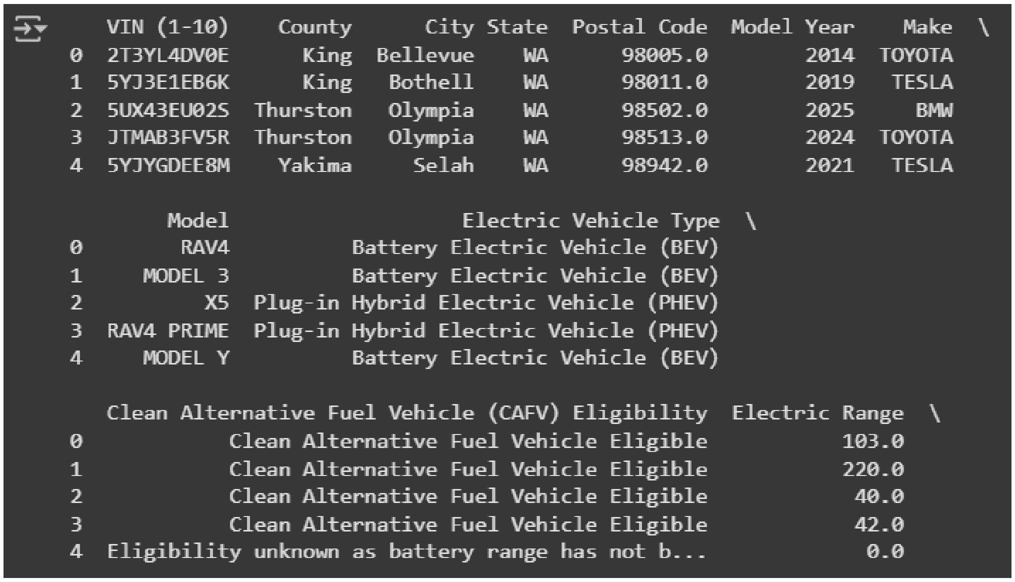
Aim: Perform Classification Modeling:
a) Choose a classifier for a classification problem.
b) Evaluate the performance of the classifier.
Perform Classification using the following 3 classifiers:

1. K-Nearest Neighbors (KNN)
2. Naive Bayes
3. Decision Tree

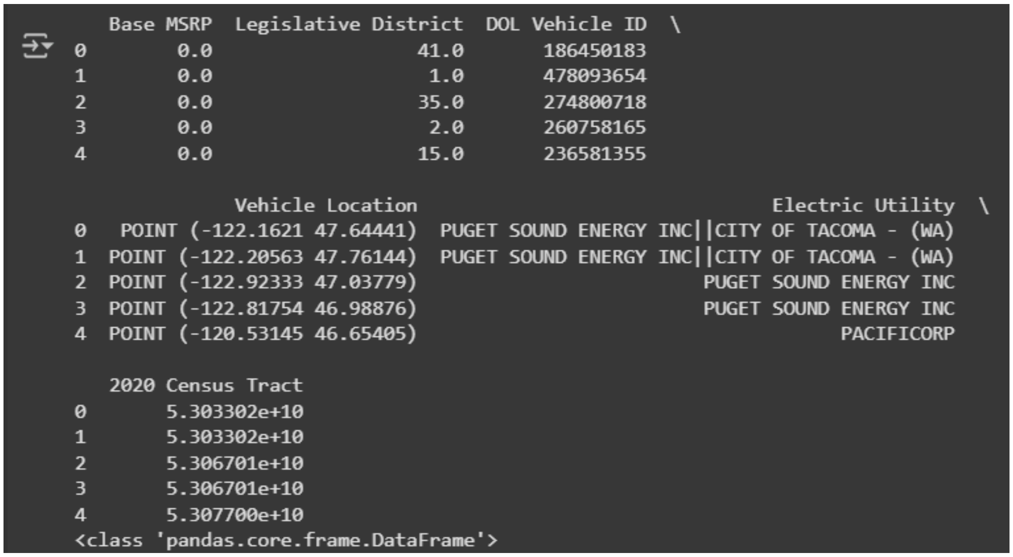
Performance:

- Prerequisite: Import essential libraries: pandas for data manipulation, numpy for numerical computations, seaborn and matplotlib.pyplot for data visualization, and sklearn modules for dataset splitting, preprocessing, and classification using K-Nearest Neighbors, Naïve Bayes, SVM, and Decision Tree models. Load the Electric Vehicle Population Dataset into a Pandas DataFrame using pd.read_csv(). Finally, explore the dataset by displaying the first few rows with df.head() and checking column names, data types, and missing values using df.info():

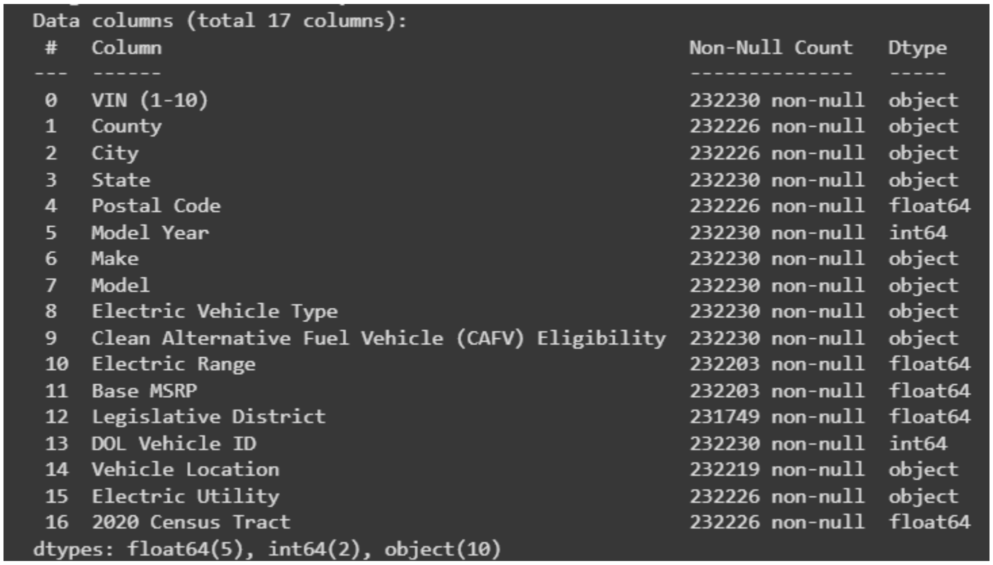
Command:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
df = pd.read_csv('Electric_Vehicle_Population_Data.csv')
print(df.head())
print(df.info())
```



```
   VIN (1-10)    County      City State  Postal Code  Model Year    Make  \
0  2T3YL4DV0E      King  Bellevue    WA      98005.0        2014  TOYOTA
1  5YJ3E1EB6K      King   Bothell    WA      98011.0        2019   TESLA
2  5UX43EU02S  Thurston   Olympia    WA      98502.0        2025     BMW
3  JTMAB3FV5R  Thurston   Olympia    WA      98513.0        2024  TOYOTA
4  5YJYGDEE8M    Yakima     Selah    WA      98942.0        2021   TESLA

        Model                   Electric Vehicle Type  \
0        RAV4          Battery Electric Vehicle (BEV)
1     MODEL 3          Battery Electric Vehicle (BEV)
2          X5  Plug-in Hybrid Electric Vehicle (PHEV)
3  RAV4 PRIME  Plug-in Hybrid Electric Vehicle (PHEV)
4     MODEL Y          Battery Electric Vehicle (BEV)

   Clean Alternative Fuel Vehicle (CAFV) Eligibility  Electric Range  \
0            Clean Alternative Fuel Vehicle Eligible           103.0
1            Clean Alternative Fuel Vehicle Eligible           220.0
2            Clean Alternative Fuel Vehicle Eligible            40.0
3            Clean Alternative Fuel Vehicle Eligible            42.0
4  Eligibility unknown as battery range has not b...             0.0
```



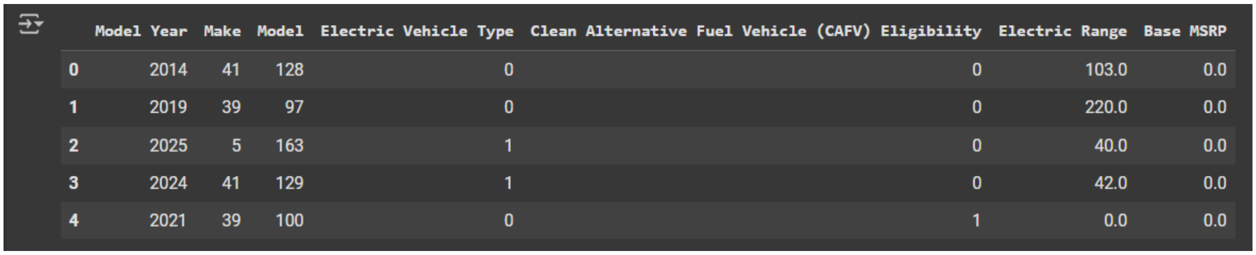
```
   Base MSRP  Legislative District  DOL Vehicle ID  \
0        0.0                  41.0       186450183
1        0.0                   1.0       478093654
2        0.0                  35.0       274800718
3        0.0                   2.0       260758165
4        0.0                  15.0       236581355

           Vehicle Location                             Electric Utility  \
0   POINT (-122.1621 47.64441)  PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA)
1  POINT (-122.20563 47.76144)  PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA)
2  POINT (-122.92333 47.03779)                         PUGET SOUND ENERGY INC
3  POINT (-122.81754 46.98876)                         PUGET SOUND ENERGY INC
4  POINT (-120.53145 46.65405)                                     PACIFICORP

   2020 Census Tract
0       5.303302e+10
1       5.303302e+10
2       5.306701e+10
3       5.306701e+10
4       5.307700e+10
<class 'pandas.core.frame.DataFrame'>
```



```
Data columns (total 17 columns):
 #   Column                                             Non-Null Count   Dtype
---  ------                                             --------------   -----
 0   VIN (1-10)                                         232230 non-null  object
 1   County                                             232226 non-null  object
 2   City                                               232226 non-null  object
 3   State                                              232230 non-null  object
 4   Postal Code                                        232226 non-null  float64
 5   Model Year                                         232230 non-null  int64
 6   Make                                               232230 non-null  object
 7   Model                                              232230 non-null  object
 8   Electric Vehicle Type                              232230 non-null  object
 9   Clean Alternative Fuel Vehicle (CAFV) Eligibility  232230 non-null  object
 10  Electric Range                                     232203 non-null  float64
 11  Base MSRP                                          232203 non-null  float64
 12  Legislative District                               231749 non-null  float64
 13  DOL Vehicle ID                                     232230 non-null  int64
 14  Vehicle Location                                   232219 non-null  object
 15  Electric Utility                                   232226 non-null  object
 16  2020 Census Tract                                  232226 non-null  float64
dtypes: float64(5), int64(2), object(10)
```

Step 1: Data Preprocessing and Encoding:-

```
Command: df_filtered = df[["Model Year", "Make", "Model", "Electric Vehicle Type",
                "Clean Alternative Fuel Vehicle (CAFV) Eligibility",
                "Electric Range", "Base MSRP"]].dropna()
label_encoders = {}
for col in ["Make", "Model", "Clean Alternative Fuel Vehicle (CAFV) Eligibility"]:
   le = LabelEncoder()
   df_filtered[col] = le.fit_transform(df_filtered[col])
   label_encoders[col] = le
target_encoder = LabelEncoder()
df_filtered["Electric Vehicle Type"] = target_encoder.fit_transform(df_filtered["Electric Vehicle
Type"])
df_filtered.head()
```



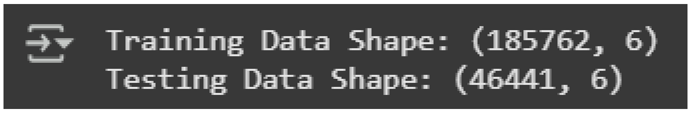
	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP
0	2014	41	128	0	0	103.0	0.0
1	2019	39	97	0	0	220.0	0.0
2	2025	5	163	1	0	40.0	0.0
3	2024	41	129	1	0	42.0	0.0
4	2021	39	100	0	1	0.0	0.0

The code selects key columns for classification, including Model Year, Make, Model, Electric Vehicle Type, CAFV Eligibility, Electric Range, and Base MSRP, while removing rows with missing values using dropna(). Categorical variables (Make, Model, CAFV Eligibility) are encoded into numerical values with LabelEncoder, making them suitable for machine learning. The target variable (Electric Vehicle Type) is also encoded for classification. Finally, the first five rows of the preprocessed dataset are displayed using df_filtered.head().

Step 2: Splitting Data into Training and Testing Sets:-

Command:

```
X = df_filtered.drop(columns=["Electric Vehicle Type"])
y = df_filtered["Electric Vehicle Type"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(f"Training Data Shape: {X_train.shape}")
print(f"Testing Data Shape: {X_test.shape}")
```

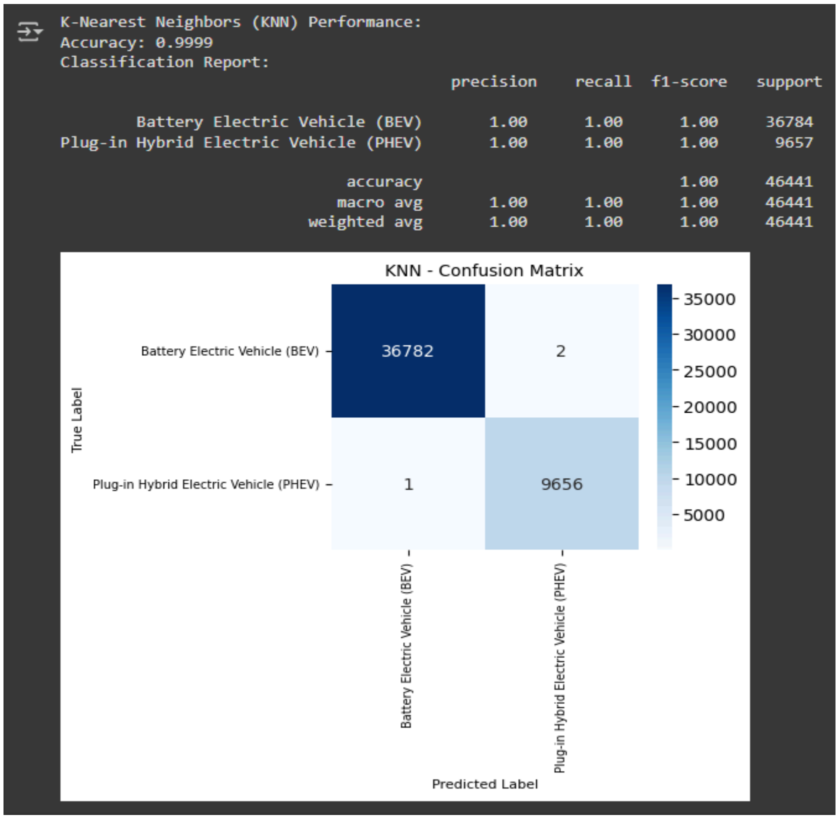


```
Training Data Shape: (185762, 6)
Testing Data Shape: (46441, 6)
```

The code defines features (X) by dropping "Electric Vehicle Type" and sets y as the target variable. It then splits the dataset into 80% training and 20% testing sets using train_test_split(), ensuring reproducibility with random_state=42. Finally, it prints the shapes of the training and testing sets.

Step 3: Training and Evaluating K-Nearest Neighbours (KNN):-

```
Command: knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
print("\nK-Nearest Neighbors (KNN) Performance:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_knn):.4f}")
print("Classification Report:\n", classification_report(y_test, y_pred_knn,
target_names=target_encoder.classes_))
plt.figure(figsize=(4, 3))  # Compact size
cm = confusion_matrix(y_test, y_pred_knn)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=target_encoder.classes_,
yticklabels=target_encoder.classes_)
plt.xlabel("Predicted Label", fontsize=8)
plt.ylabel("True Label", fontsize=8)
plt.title("KNN - Confusion Matrix", fontsize=10)
plt.xticks(fontsize=7)
plt.yticks(fontsize=7)
plt.show()
```



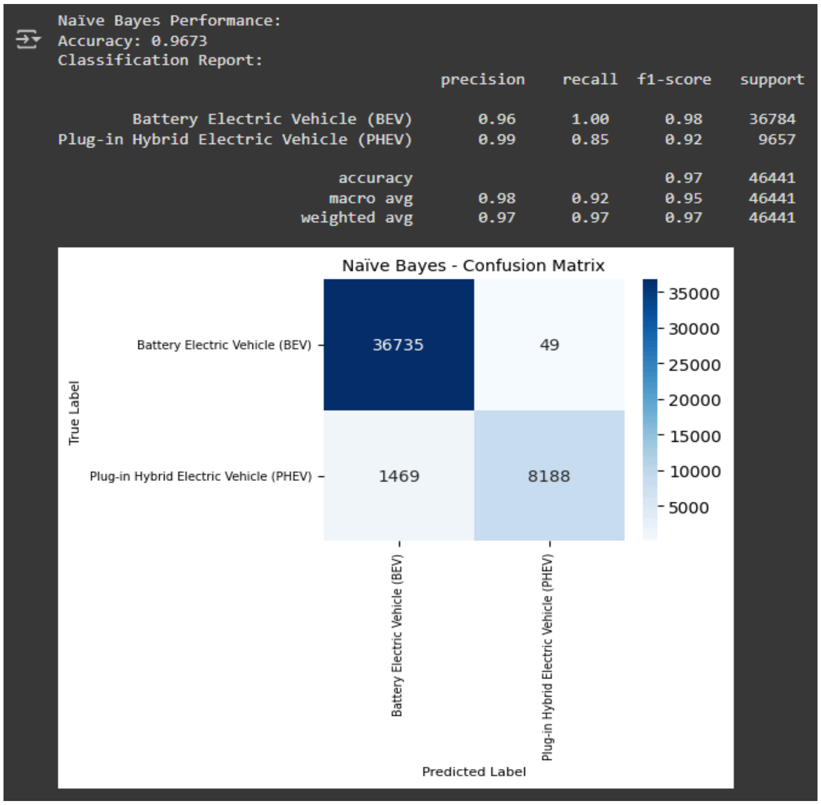
```
K-Nearest Neighbors (KNN) Performance:
Accuracy: 0.9999
Classification Report:
                                         precision    recall  f1-score   support

        Battery Electric Vehicle (BEV)       1.00      1.00      1.00     36784
Plug-in Hybrid Electric Vehicle (PHEV)       1.00      1.00      1.00      9657

                              accuracy                           1.00     46441
                             macro avg       1.00      1.00      1.00     46441
                          weighted avg       1.00      1.00      1.00     46441
```

KNN - Confusion Matrix
Battery Electric Vehicle (BEV)
Plug-in Hybrid Electric Vehicle (PHEV)
True Label
36782
2
1
9656
Battery Electric Vehicle (BEV)
Plug-in Hybrid Electric Vehicle (PHEV)
Predicted Label
35000
30000
25000
20000
15000
10000
5000

The code trains a K-Nearest Neighbors (KNN) model with n_neighbors=5 using the training data and makes predictions on the test set. It evaluates performance by computing accuracy and displaying a classification report. Additionally, it visualizes the confusion matrix using a heatmap to show the distribution of correct and incorrect predictions, making model performance easier to interpret.

Step 4: Training and Evaluating Naive Bayes:

```
Command: nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)
print("\nNaïve Bayes Performance:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_nb):.4f}")
print("Classification Report:\n", classification_report(y_test, y_pred_nb,
target_names=target_encoder.classes_))
plt.figure(figsize=(4, 3))  # Smaller size
cm = confusion_matrix(y_test, y_pred_nb)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=target_encoder.classes_,
yticklabels=target_encoder.classes_)
plt.xlabel("Predicted Label", fontsize=8)
plt.ylabel("True Label", fontsize=8)
plt.title("Naïve Bayes - Confusion Matrix", fontsize=10)
plt.xticks(fontsize=7)
plt.yticks(fontsize=7)
plt.show()
```



```
Naïve Bayes Performance:
Accuracy: 0.9673
Classification Report:
                                        precision    recall  f1-score   support

          Battery Electric Vehicle (BEV)       0.96      1.00      0.98     36784
Plug-in Hybrid Electric Vehicle (PHEV)       0.99      0.85      0.92      9657

                              accuracy                           0.97     46441
                             macro avg       0.98      0.92      0.95     46441
                          weighted avg       0.97      0.97      0.97     46441
```

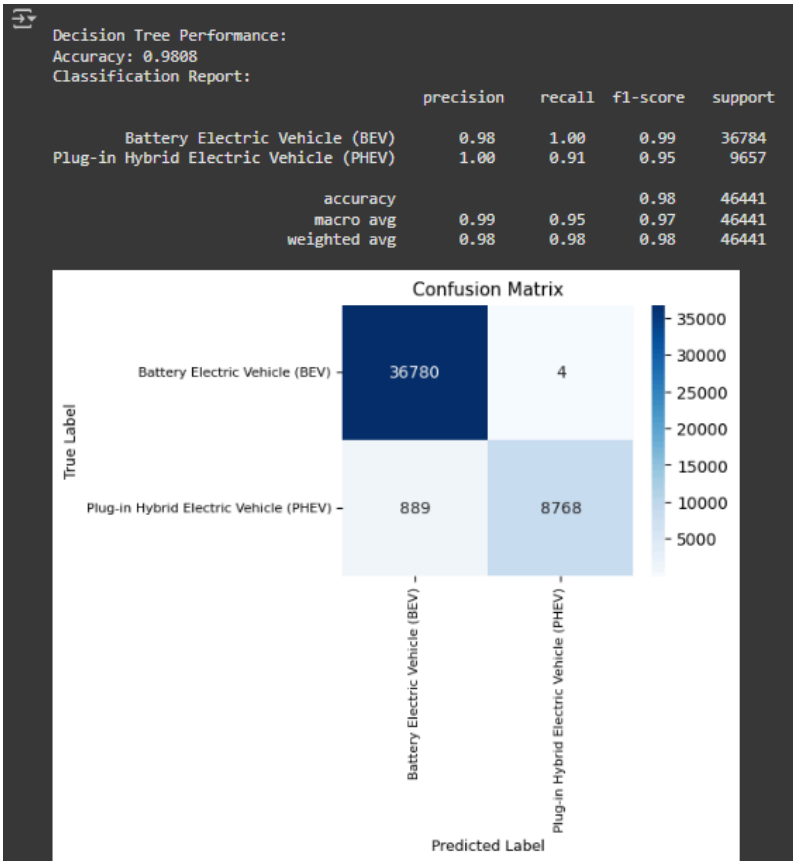
Naïve Bayes - Confusion Matrix
True Label
Battery Electric Vehicle (BEV)
Plug-in Hybrid Electric Vehicle (PHEV)
36735
49
1469
8188
35000
30000
25000
20000
15000
10000
5000
Battery Electric Vehicle (BEV)
Plug-in Hybrid Electric Vehicle (PHEV)
Predicted Label

The code trains a Naïve Bayes classifier using the training dataset and makes predictions on the test set. Model performance is evaluated using accuracy and a classification report, which provide insights into precision, recall, and F1-score. Additionally, a confusion matrix is visualized with a heatmap to highlight correct and incorrect predictions, helping to assess the classifier's effectiveness and potential areas for improvement.

Step 5: Decision Tree Model Training, Evaluation and Visualization:-

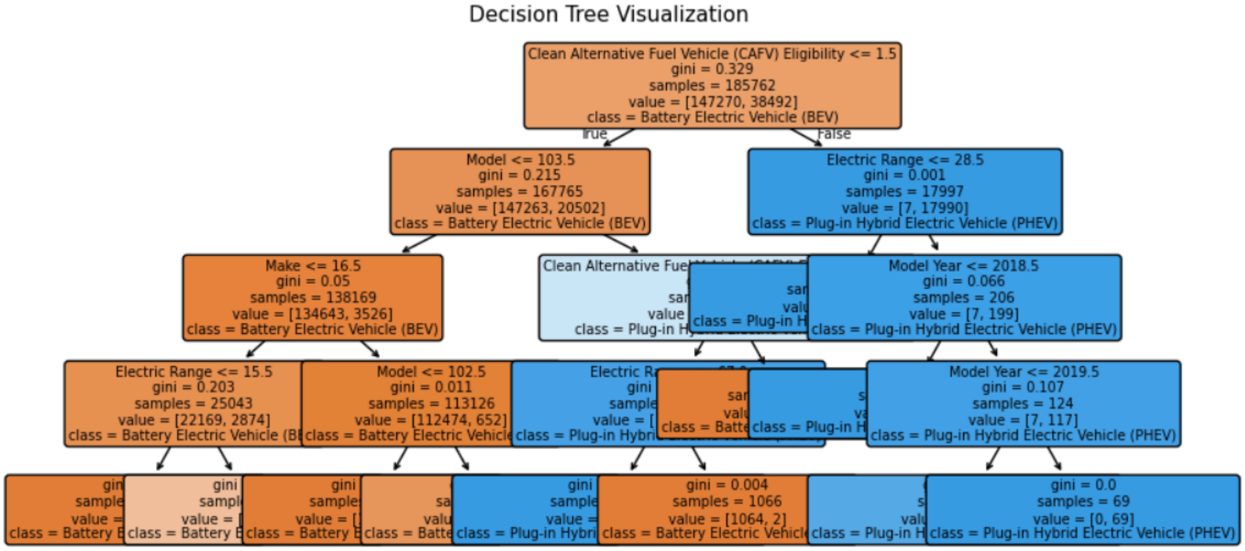
Command: from sklearn.tree import plot_tree

```
dt = DecisionTreeClassifier(random_state=42, max_depth=3, min_samples_split=10,
min_samples_leaf=5)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
print("\nDecision Tree Performance:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_dt):.4f}")
print("Classification Report:\n", classification_report(y_test, y_pred_dt,
target_names=target_encoder.classes_))
plt.figure(figsize=(4, 3))
cm = confusion_matrix(y_test, y_pred_dt)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=target_encoder.classes_,
yticklabels=target_encoder.classes_)
plt.xlabel("Predicted Label", fontsize=9)
plt.ylabel("True Label", fontsize=9)
plt.title("Confusion Matrix", fontsize=11)
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.show()
plt.figure(figsize=(10, 5))
plot_tree(dt, filled=True, feature_names=X.columns, class_names=target_encoder.classes_,
fontsize=7, rounded=True)
plt.title("Decision Tree Visualization", fontsize=11)
plt.show()
```



```
Decision Tree Performance:
Accuracy: 0.9808
Classification Report:
                                        precision    recall  f1-score   support

          Battery Electric Vehicle (BEV)       0.98      1.00      0.99     36784
Plug-in Hybrid Electric Vehicle (PHEV)       1.00      0.91      0.95      9657

                              accuracy                           0.98     46441
                             macro avg       0.99      0.95      0.97     46441
                          weighted avg       0.98      0.98      0.98     46441
```

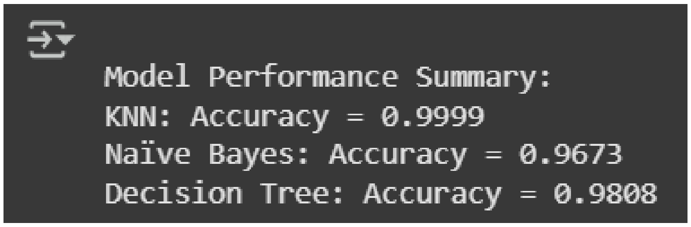
Confusion Matrix
Battery Electric Vehicle (BEV)
Plug-in Hybrid Electric Vehicle (PHEV)
True Label
36780
4
889
8768
35000
30000
25000
20000
15000
10000
5000
Battery Electric Vehicle (BEV)
Plug-in Hybrid Electric Vehicle (PHEV)
Predicted Label



The code trains a Decision Tree classifier on the training dataset and makes predictions on the test set. Model performance is assessed using accuracy and a classification report to evaluate precision, recall, and F1-score. A confusion matrix is visualized using a heatmap to analyze correct and incorrect predictions. Additionally, a decision tree diagram is plotted, offering a detailed view of the model's decision-making process by showing feature splits and class assignments, aiding in model interpretability.

Step 6: Model Performance Comparison:-

```
Command: model_performances = {
   "KNN": accuracy_score(y_test, y_pred_knn),
   "Naïve Bayes": accuracy_score(y_test, y_pred_nb),
   "Decision Tree": accuracy_score(y_test, y_pred_dt)
}
print("\nModel Performance Summary:")
for model, acc in model_performances.items():
   print(f"{model}: Accuracy = {acc:.4f}")
```



```
Model Performance Summary:
KNN: Accuracy = 0.9999
Naïve Bayes: Accuracy = 0.9673
Decision Tree: Accuracy = 0.9808
```

The code stores and compares the accuracy scores of different machine learning models, including K-Nearest Neighbors (KNN), Naïve Bayes and Decision Tree. These accuracy values are saved in a dictionary, model_performances, and then printed in a structured format to provide a quick overview of how well each model performed on the test dataset. This step helps in identifying the most effective model for classification.

Conclusion:

1. In this experiment, we learned how to perform classification modeling using different classifiers.
2. Decision Tree and KNN performed exceptionally well, achieving near perfect accuracy (0.9808 and 0.9999) respectively.
3. Naive Bayes had the lowest accuracy (0.9673), struggling with Plug-in Hybrid Electric Vehicles (PHEVs), misclassifying 1469 instances.
4. Decision Tree showed perfect classification, with only 4 misclassified BEVs, indicating clear decision boundaries.
5. KNN also performed nearly perfectly, misclassifying just 3 instances, proving its effectiveness for our dataset.
6. The Decision Tree visualization highlights key decision factors like ‘Electric Range’ and ‘CAFV Eligibility’ for classification.
7. Overall, Decision Tree and KNN are the best models while Naive Bayes is less suitable due to lower recall for PHEVs.