Q1. Implement binary image classification using the CIFAR-10 dataset.

a. Import the CIFAR-10 dataset using the following command:

from keras.datasets import cifar10

The CIFAR-10 dataset comprises 60,000 color images (32x32 pixels) across 10 different classes, with each class containing 6,000 images. The dataset is divided into 50,000 training images and 10,000 test images. Focus only on the training images. Label all "Automobile" class images as 1 and select an equal number of images from another class, labeling them as 0 (Non-Automobile).

In [12]:
```
from keras.datasets import cifar10
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [13]:
```
# Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

In [14]:
```
# Print the shape of the datasets
print("Shape of x_train:", x_train.shape)  # (50000, 32, 32, 3)
print("Shape of y_train:", y_train.shape)  # (50000, 1)
print("Shape of x_test:", x_test.shape)    # (10000, 32, 32, 3)
print("Shape of y_test:", y_test.shape)    # (10000, 1)

# Print information on the number of samples per class in the training data
classes, counts = np.unique(y_train, return_counts=True)
print("Number of samples per class in the training set:")
for cls, count in zip(classes, counts):
    print(f"Class {cls}: {count} samples")

# Print the first image shape and label
print("Shape of the first image in x_train:", x_train[0].shape)
print("Label of the first image in y_train:", y_train[0][0])
```

```
Shape of x_train: (50000, 32, 32, 3)
Shape of y_train: (50000, 1)
Shape of x_test: (10000, 32, 32, 3)
Shape of y_test: (10000, 1)
Number of samples per class in the training set:
Class 0: 5000 samples
Class 1: 5000 samples
Class 2: 5000 samples
Class 3: 5000 samples
Class 4: 5000 samples
Class 5: 5000 samples
Class 6: 5000 samples
Class 7: 5000 samples
Class 8: 5000 samples
Class 9: 5000 samples
Shape of the first image in x_train: (32, 32, 3)
Label of the first image in y_train: 6
```

In [18]:
```
# Define the class labels
automobile_label = 1
non_automobile_class = 0  # You can choose any other class (e.g., class 0 for 'airplane'

# Extract the automobile class (label = 1)
automobile_images = x_train[y_train.flatten() == automobile_label]
```

```python
automobile_labels = y_train[y_train.flatten() == automobile_label]

# Extract an equal number of images from the non-automobile class
non_automobile_images = x_train[y_train.flatten() == non_automobile_class][:len(automobi
non_automobile_labels = y_train[y_train.flatten() == non_automobile_class][:len(automobi

# Combine the images and labels for binary classification
x_binary = np.concatenate((automobile_images, non_automobile_images), axis=0)
y_binary = np.concatenate((np.ones(len(automobile_images)), np.zeros(len(non_automobile_

# Shuffle the dataset
indices = np.arange(x_binary.shape[0])
np.random.shuffle(indices)
x_binary = x_binary[indices]
y_binary = y_binary[indices]
```

In [ ]:

```python
# Print the shape and some information about the dataset after preprocessing
print("Shape of x_binary:", x_binary.shape)
print("Shape of y_binary:", y_binary.shape)
print("Number of samples (automobiles):", np.sum(y_binary == 1))
print("Number of samples (non-automobiles):", np.sum(y_binary == 0))

# Example of printing the shape of the first image and some sample data
print("Shape of the first image:", x_binary[0].shape)
```

In [20]:

```python
# Normalize the images
x_binary = x_binary.astype('float32') / 255.0

# Flatten the images
x_binary = x_binary.reshape(x_binary.shape[0], -1)
```

In [21]:

```python
# Print the shape and some information about the dataset after preprocessing
print("Shape of x_binary:", x_binary.shape)
print("Shape of y_binary:", y_binary.shape)
print("Number of samples (automobiles):", np.sum(y_binary == 1))
print("Number of samples (non-automobiles):", np.sum(y_binary == 0))

# Example of printing the shape of the first image and some sample data
print("Shape of the first image:", x_binary[0].shape)
print("First image (flattened):", x_binary[0])
```

```
Shape of x_binary: (10000, 3072)
Shape of y_binary: (10000,)
Number of samples (automobiles): 5000
Number of samples (non-automobiles): 5000
Shape of the first image: (3072,)
First image (flattened): [0.00239908 0.00173779 0.00141484 ... 0.00339869 0.00290657 0.0
0218378]
```

In [22]:

```python
# Split the data into training and validation sets
x_train_binary, x_val_binary, y_train_binary, y_val_binary = train_test_split(
    x_binary, y_binary, test_size=0.2, random_state=42
)
```

In [3]:

```python
# Initialize the classifiers
svc = SVC()
knn = KNeighborsClassifier()
decision_tree = DecisionTreeClassifier()
logistic_regression = LogisticRegression()

# Train and evaluate SVC
svc.fit(x_train_binary, y_train_binary)
svc_predictions = svc.predict(x_val_binary)
svc_accuracy = accuracy_score(y_val_binary, svc_predictions)
```

```python
print(f"SVC Accuracy: {svc_accuracy * 100:.2f}%")

# Train and evaluate kNN
knn.fit(x_train_binary, y_train_binary)
knn_predictions = knn.predict(x_val_binary)
knn_accuracy = accuracy_score(y_val_binary, knn_predictions)
print(f"kNN Accuracy: {knn_accuracy * 100:.2f}%")

# Train and evaluate Decision Tree
decision_tree.fit(x_train_binary, y_train_binary)
decision_tree_predictions = decision_tree.predict(x_val_binary)
decision_tree_accuracy = accuracy_score(y_val_binary, decision_tree_predictions)
print(f"Decision Tree Accuracy: {decision_tree_accuracy * 100:.2f}%")

# Train and evaluate Logistic Regression
logistic_regression.fit(x_train_binary, y_train_binary)
logistic_regression_predictions = logistic_regression.predict(x_val_binary)
logistic_regression_accuracy = accuracy_score(y_val_binary, logistic_regression_predicti
print(f"Logistic Regression Accuracy: {logistic_regression_accuracy * 100:.2f}%")
```

```
SVC Accuracy: 89.05%
kNN Accuracy: 67.45%
Decision Tree Accuracy: 76.35%
Logistic Regression Accuracy: 79.90%
```

```
c:\Users\satch\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\linear_
model\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Q1 b. Build a binary classifier that predicts whether a given TEN images are "Automobile"(1) or "Non-Automobile" (0).

Implement the following classification algorithms:

- Support Vector Classifier (SVC)
- k-Nearest Neighbors (kNN)
- Decision Tree
- Logistic Regression

In [24]:

```python
from PIL import Image
import numpy as np
import os

# Directory containing your 10 images
image_directory = "D:/MCA/4th Trm/ADA/Practical/Lab 7/testimages"  # Update this with th

# List of image filenames
image_filenames = ['image1.jpeg', 'image2.jpeg', 'image3.jpeg', 'image4.jpeg', 'image5.j
                   'image6.jpeg', 'image7.jpeg', 'image8.jpeg', 'image9.jpeg', 'image10.

# Initialize list to hold image data
x_custom_test = []

# Load and preprocess each image
for filename in image_filenames:
    # Load the image
    image_path = os.path.join(image_directory, filename)
    img = Image.open(image_path)

    # Resize the image to 32x32 pixels as required by the model
```

```python
    img = img.resize((32, 32))

    # Convert the image to a numpy array and normalize it
    img_array = np.array(img).astype('float32') / 255.0

    # Add the image array to the list
    x_custom_test.append(img_array)

# Convert the list to a numpy array and adjust dimensions
x_custom_test = np.array(x_custom_test)

# Flatten the images for classification
x_custom_test_flattened = x_custom_test.reshape(len(x_custom_test), -1)

# Print the shapes to confirm
print("Shape of x_custom_test:", x_custom_test.shape)
print("Shape of flattened x_custom_test:", x_custom_test_flattened.shape)
```

```
Shape of x_custom_test: (10, 32, 32, 3)
Shape of flattened x_custom_test: (10, 3072)
```

In [25]:

```python
# Predict using SVC
svc_predictions = svc.predict(x_custom_test_flattened)

# Predict using kNN
knn_predictions = knn.predict(x_custom_test_flattened)

# Predict using Decision Tree
decision_tree_predictions = decision_tree.predict(x_custom_test_flattened)

# Predict using Logistic Regression
logistic_regression_predictions = logistic_regression.predict(x_custom_test_flattened)

# Print the results for each custom image
for i, filename in enumerate(image_filenames):
    print(f"Image {filename}:")
    print(f"  SVC Prediction: {'Automobile' if svc_predictions[i] == 1 else 'Non-Automob
    print(f"  kNN Prediction: {'Automobile' if knn_predictions[i] == 1 else 'Non-Automob
    print(f"  Decision Tree Prediction: {'Automobile' if decision_tree_predictions[i] ==
    print(f"  Logistic Regression Prediction: {'Automobile' if logistic_regression_predi
    print()
```

```
Image image1.jpeg:
  SVC Prediction: Automobile
  kNN Prediction: Automobile
  Decision Tree Prediction: Automobile
  Logistic Regression Prediction: Automobile

Image image2.jpeg:
  SVC Prediction: Automobile
  kNN Prediction: Automobile
  Decision Tree Prediction: Non-Automobile
  Logistic Regression Prediction: Non-Automobile

Image image3.jpeg:
  SVC Prediction: Automobile
  kNN Prediction: Automobile
  Decision Tree Prediction: Automobile
  Logistic Regression Prediction: Automobile

Image image4.jpeg:
  SVC Prediction: Automobile
  kNN Prediction: Automobile
  Decision Tree Prediction: Automobile
  Logistic Regression Prediction: Automobile
```

```
Image image5.jpeg:
  SVC Prediction: Automobile
  kNN Prediction: Non-Automobile
  Decision Tree Prediction: Automobile
  Logistic Regression Prediction: Automobile

Image image6.jpeg:
  SVC Prediction: Automobile
  kNN Prediction: Automobile
  Decision Tree Prediction: Automobile
  Logistic Regression Prediction: Automobile

Image image7.jpeg:
  SVC Prediction: Automobile
  kNN Prediction: Automobile
  Decision Tree Prediction: Automobile
  Logistic Regression Prediction: Automobile

Image image8.jpeg:
  SVC Prediction: Automobile
  kNN Prediction: Non-Automobile
  Decision Tree Prediction: Automobile
  Logistic Regression Prediction: Automobile

Image image9.jpeg:
  SVC Prediction: Non-Automobile
  kNN Prediction: Non-Automobile
  Decision Tree Prediction: Non-Automobile
  Logistic Regression Prediction: Non-Automobile

Image image10.jpeg:
  SVC Prediction: Automobile
  kNN Prediction: Automobile
  Decision Tree Prediction: Automobile
  Logistic Regression Prediction: Non-Automobile
```

Image Filename	Original Label	SVC Prediction	kNN Prediction	Decision Tree Prediction	Logistic Regression Prediction
image1.jpg	Automobile	Automobile	Automobile	Automobile	Automobile
image2.jpg	Automobile	Automobile	Automobile	Non-Automobile	Non-Automobile
image3.jpg	Automobile	Automobile	Automobile	Automobile	Automobile
image4.jpg	Automobile	Automobile	Automobile	Automobile	Automobile
image5.jpg	Automobile	Automobile	Non-Automobile	Automobile	Automobile
image6.jpg	Automobile	Automobile	Automobile	Automobile	Automobile
image7.jpg	Automobile	Automobile	Automobile	Automobile	Automobile
image8.jpg	Automobile	Automobile	Non-Automobile	Automobile	Automobile
image9.jpg	Non-Automobile	Non-Automobile	Non-Automobile	Non-Automobile	Non-Automobile
image10.jpg	Automobile	Automobile	Automobile	Automobile	Non-Automobile

Models	SVC Accuracy	kNN Accuracy	Decision Tree Accuracy	Logistic Regression Accuracy
Correct	9	9	8	9
Incorrect	1	1	2	1
Accuracy	89.05%	67.45%	76.35%	79.90%