Numerical Dataset : **Car Price Prediction**

The goal of this project is creation a model by linear regression method and KNN model that predict the selling price of a car

About Dataset

This dataset contains information about used cars. This data can be used for a lot of purposes such as price prediction to exemplify the use of linear regression in Machine Learning. The columns in the given dataset are as follows:

1. name: Name of the cars.
2. year: Year of the car when it was bought.
3. selling_price: selling price of the car.
4. km_driven: number of kilometers the car is driven.
5. fuel: fuel type of the car.
6. seller_type: tells if car is sold by individual or dealer.
7. transmission: Gear transmission of the car (Automatic/Manual).
8. Owner: number of previous owners.
9. mileage: mileage of the car.
10. engine: engine capacity of the car.
11. max_power: max power of engine.
12. torque: torque of the car.
13. seats: number of seats in the car.

Total number of samples in dataset : 8128
Number of samples used in training : 5503
Number of samples used in testing : 1376

Algorithms used:

1 - Linear Regression

Linear Regression is a simple and widely used algorithm for predicting continuous values. It works by finding the best-fitting straight line that shows the relationship between input features and the target output. The goal is to minimize the difference between the predicted values and the actual values. Linear Regression is efficient, interpretable, and often used as a baseline model for regression tasks.

2 - K-Nearest Neighbors (KNN)

K-Nearest Neighbors is a versatile algorithm used for both classification and regression. It makes predictions by comparing a new data point to existing data points and selecting the closest ones (neighbors). For classification, it identifies the most common class among the neighbors. For regression, it averages the values of

the neighbors. KNN is simple to understand but can be slower for large datasets since it requires comparing the query to all data points.

```
    # Evaluate the Linear Regression model
    mse_lr = mean_squared_error(y_test, y_pred_lr)  # Mean Squared Error
    rmse_lr = mse_lr ** 0.5  # Root Mean Squared Error
    r2_lr = r2_score(y_test, y_pred_lr)  # R2 Score

    print("Linear Regression Results:")
    print(f"Mean Squared Error (MSE): {mse_lr:f}")
    print(f"Root Mean Squared Error (RMSE): {rmse_lr:f}")
    print(f"R² Score: {r2_lr:f}")
```

```
Linear Regression Results:
Mean Squared Error (MSE): 0.003065
Root Mean Squared Error (RMSE): 0.055360
R² Score: 0.823362
```

```
    # Evaluate the KNN Regression model
    mse_knn = mean_squared_error(y_test, y_pred_knn)  # Mean Squared Error
    rmse_knn = mse_knn ** 0.5  # Root Mean Squared Error
    r2_knn = r2_score(y_test, y_pred_knn)  # R² Score

    print("\nKNN Regression Results:")
    print(f"Mean Squared Error (MSE): {mse_knn:f}")
    print(f"Root Mean Squared Error (RMSE): {rmse_knn:f}")
    print(f"R² Score: {r2_knn:f}")
```

```
KNN Regression Results:
Mean Squared Error (MSE): 0.002070
Root Mean Squared Error (RMSE): 0.045496
R² Score: 0.880700
```

Image Dataset : Fruits 360

Fruits-360: A dataset of images containing fruits and vegetables

A high-quality, dataset of images containing fruits and vegetables. The following fruits and vegetables are included:

Apples (different varieties: Crimson Snow, Golden, Golden-Red, Granny Smith, Pink Lady, Red, Red Delicious), Apricot, Avocado, Avocado ripe, Banana (Yellow, Red, Lady Finger), Beetroot Red, Blueberry, Cactus fruit, Cantaloupe (2 varieties), Carambula, Carrot, Cauliflower, Cherry (different varieties, Rainier), Cherry Wax (Yellow, Red, Black), Chestnut, Clementine, Cocos, Corn (with husk), Cucumber (ripened, various), Dates, Eggplant (normal and long), Fig, Ginger Root, Granadilla, Grape (Blue, Pink, White (different varieties)), Grapefruit (Pink, White), Guava, Hazelnut, Huckleberry, Kiwi, Kaki, Kohlrabi, Kumsquats, Lemon (normal, Meyer), Lime, Lychee, Mandarine, Mango (Green, Red), Mangostan, Maracuja, Melon Piel de Sapo, Mulberry, Nectarine (Regular, Flat), Nut (Forest, Pecan), Onion (Red, White), Orange, Papaya, Passion fruit, Peach (different varieties), Pepino, Pear (different varieties, Abate, Forelle, Kaiser, Monster, Red, Stone, Williams), Pepper (Red, Green, Orange, Yellow), Physalis (normal, with Husk), Pineapple (normal, Mini), Pitahaya Red, Plum (different varieties), Pomegranate, Pomelo Sweetie, Potato (Red, Sweet, White), Quince, Rambutan, Raspberry, Redcurrant, Salak, Strawberry (normal, Wedge), Tamarillo, Tangelo, Tomato (different varieties, Maroon, Cherry Red, Yellow, not ripened, Heart), Walnut, Watermelon, Zucchini (white and dark).

Total number of images: 94110.

Training set size: 70491 images (one fruit or vegetable per image).

Test set size: 23619 images (one fruit or vegetable per image).

Number of classes: 141 (fruits, vegetables and nuts) : Used(5).

Image size: 100x100 pixels.

Algorithms used:

1 - Logistic Regression:

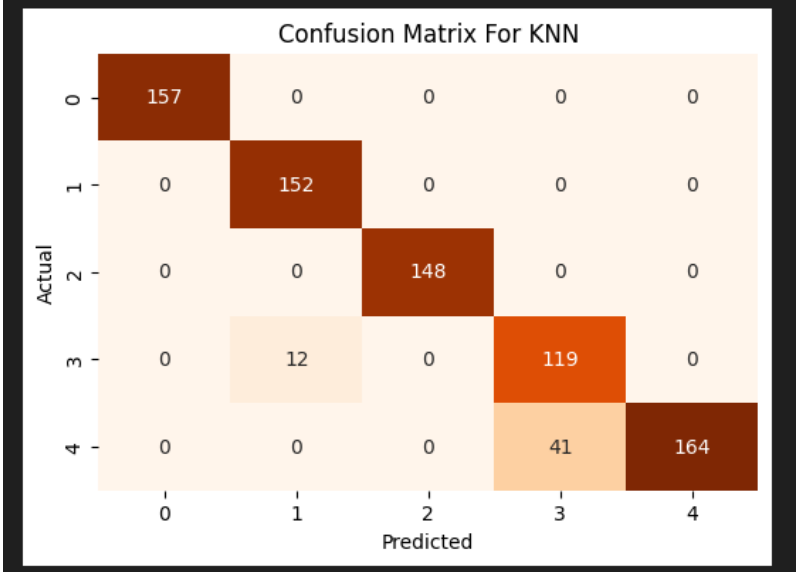
Logistic Regression is a supervised learning algorithm used for classification tasks. Unlike linear regression, it predicts the probability that a given input belongs to a particular class. The output is a value between 0 and 1, which is then used to assign the data to a specific class (e.g., 0 or 1). Logistic Regression is widely used for binary classification problems, such as determining whether an email is spam or not. It is simple, efficient, and interpretable, making it a popular choice for many classification tasks.

2 - K-Nearest Neighbors (KNN) :

K-Nearest Neighbors is a versatile algorithm used for both classification and regression. It makes predictions by comparing a new data point to existing data points and selecting the closest ones (neighbors). For classification, it identifies the most common class among the neighbors. For regression, it averages the values of

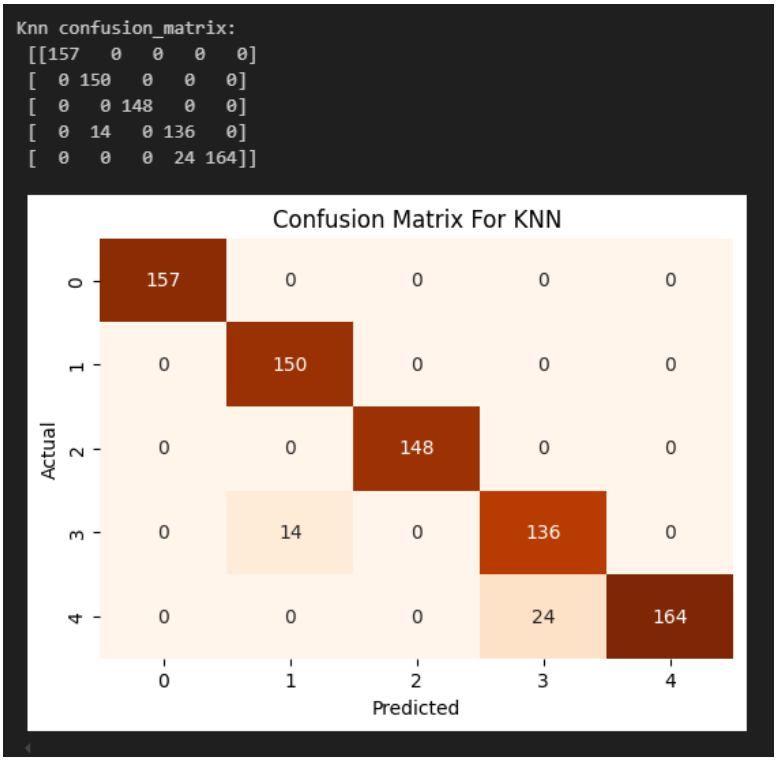
Logistic Regression:

```
Logistic Regression confusion_matrix:
 [[157   0   0   0   0]
 [  0 152   0   0   0]
 [  0   0 148   0   0]
 [  0  12   0 119   0]
 [  0   0   0  41 164]]
```

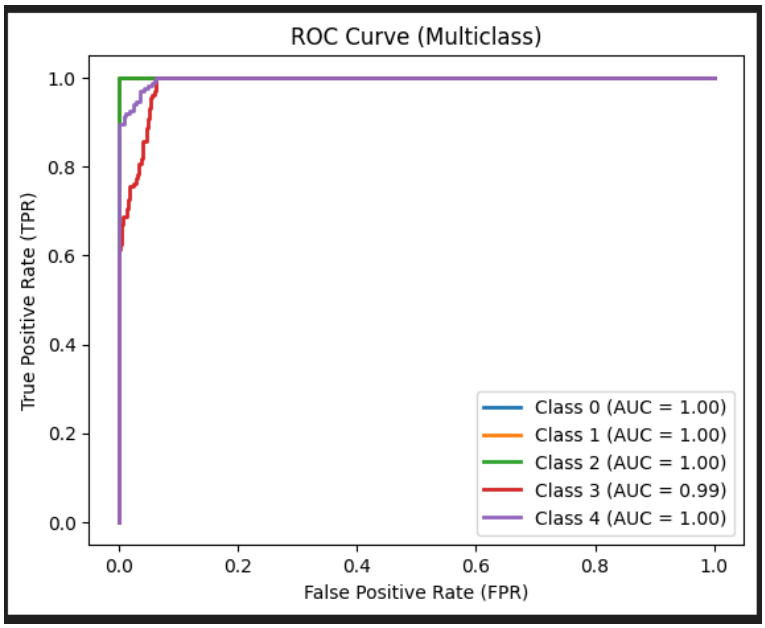


KNN:

```
Knn confusion_matrix:
 [[157   0   0   0   0]
 [  0 150   0   0   0]
 [  0   0 148   0   0]
 [  0  14   0 136   0]
 [  0   0   0  24 164]]
```



Logistic Regression:



KNN :