Load the dataset

In [114…
```
import pandas as pd

df = pd.read_csv("car_details.csv")
df.head()
```

Out[114…

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

Shape of the dataset

In [115…
```
df.shape
```

Out[115… (4340, 8)

Number of duplicates

In [116…
```
print(df.duplicated().sum())
```

763

Drop the duplicates

In [117…
```
df = df.drop_duplicates()
df.shape
```

Out[117… (3577, 8)

Number of null values

In [118…
```
df.isnull().sum()
```

Out[118…
```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
dtype: int64
```

Data types

In [119…
```
df.dtypes
```

Out[119…
```
name              str
year            int64
selling_price   int64
km_driven       int64
fuel              str
seller_type       str
transmission      str
owner             str
dtype: object
```

Handling "name" column

In [120…
```
df["name"].value_counts()
```

Out[120…
```
name
Maruti Swift Dzire VDI                          54
Maruti Alto 800 LXI                             48
Maruti Alto LXi                                 42
Maruti Alto LX                                  30
Hyundai EON Era Plus                            28
                                                ..
Maruti Swift LDI                                 1
Tata Nano XM                                     1
Mahindra Verito 1.5 D6 BSIII                     1
Toyota Innova 2.5 VX (Diesel) 8 Seater BS IV     1
Hyundai i20 Magna 1.4 CRDi                       1
Name: count, Length: 1491, dtype: int64
```

In [121…
```
df["name"].nunique()
```

Out[121…
```
1491
```

Handling "fuel", "seller_type", "transmission", "brand" columns

In [122…
```
df["brand"] = df["name"].str.split(" ").str[0]

df.drop("name", axis=1, inplace=True)

df["brand"].value_counts()
```

Out[122…
```
brand
Maruti           1072
Hyundai           637
Mahindra          328
Tata              308
Ford              220
Honda             216
Toyota            170
Chevrolet         151
Renault           110
Volkswagen         93
Nissan             52
Skoda              49
Fiat               32
Audi               31
Datsun             29
BMW                25
Mercedes-Benz      21
Jaguar              5
Mitsubishi          5
Land                5
Volvo               4
Jeep                3
Ambassador          3
MG                  2
OpelCorsa           2
Daewoo              1
Force               1
Isuzu               1
Kia                 1
Name: count, dtype: int64
```

In [123…
```
df["brand"].nunique()
```

Out[123…
```
29
```

In [124…
```
df["fuel"].value_counts()
```

Out[124…
```
fuel
Diesel      1800
Petrol      1717
CNG           37
LPG           22
Electric       1
Name: count, dtype: int64
```

In [125…
```
df["seller_type"].value_counts()
```

Out[125…
```
seller_type
Individual          2832
Dealer               712
Trustmark Dealer      33
Name: count, dtype: int64
```

In [126…
```
df["transmission"].value_counts()
```

Out[126…
```
transmission
Manual       3265
Automatic     312
Name: count, dtype: int64
```

In [127…
```
df = pd.get_dummies(df, columns=["fuel", "seller_type", "transmission", "
```

Handling "owner" column

In [128…
```
df["owner"].value_counts()
```

Out[128…
```
owner
First Owner             2218
Second Owner             978
Third Owner              289
Fourth & Above Owner      75
Test Drive Car            17
Name: count, dtype: int64
```

In [129…
```
owner_mapping = {
    "Test Drive Car": 0,
    "First Owner": 1,
    "Second Owner": 2,
    "Third Owner": 3,
    "Fourth & Above Owner": 4
}

df["owner"] = df["owner"].map(owner_mapping)
```

In [130…
```
df.head()
```

Out[130…

	year	selling_price	km_driven	owner	fuel_Diesel	fuel_Electric	fuel_LPG	fuel_Pe
0	2007	60000	70000	1	0	0	0	
1	2007	135000	50000	1	0	0	0	
2	2012	600000	100000	1	1	0	0	
3	2017	250000	46000	1	0	0	0	
4	2014	450000	141000	2	1	0	0	

5 rows × 39 columns

Split the dataset

In [131…
```
from sklearn.model_selection import train_test_split

X = df.drop("selling_price", axis=1)
y = df["selling_price"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

print("Training data shape:", X_train.shape)
print("Testing data shape:", X_test.shape)
```

```
Training data shape: (2861, 38)
Testing data shape: (716, 38)
```

Linear Regression Model

In [132…

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, r2_score

linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

linear_y_pred = linear_model.predict(X_test)

linear_mae = mean_absolute_error(y_test, linear_y_pred)
linear_r2 = r2_score(y_test, linear_y_pred)

print("Linear Regression\n")
print(f"MAE: {linear_mae:,.4f}\n")
print(f"R2 Score: {linear_r2:.4f}")
```

```
Linear Regression

MAE: 180,314.2719

R2 Score: 0.5373
```

Decision Tree Regressor Model

In [133…

```
from sklearn.tree import DecisionTreeRegressor

tree_model = DecisionTreeRegressor(random_state=42)
tree_model.fit(X_train, y_train)

tree_y_pred = tree_model.predict(X_test)

tree_mae = mean_absolute_error(y_test, tree_y_pred)
tree_r2 = r2_score(y_test, tree_y_pred)

print("DecisionTreeRegressor\n")
print(f"MAE: {tree_mae:,.4f}\n")
print(f"R2 Score: {tree_r2:.4f}")
```

```
DecisionTreeRegressor

MAE: 203,422.6013

R2 Score: 0.2580
```

Random Forest Regressor Model

In [134…
```
from sklearn.ensemble import RandomForestRegressor

forest_model = RandomForestRegressor(random_state=42)
forest_model.fit(X_train, y_train)

forest_y_pred = forest_model.predict(X_test)

forest_mae = mean_absolute_error(y_test, forest_y_pred)
forest_r2 = r2_score(y_test, forest_y_pred)

print("DecisionTreeRegressor\n")
print(f"MAE: {forest_mae:,.4f}\n")
print(f"R2 Score: {forest_r2:.4f}")
```

```
DecisionTreeRegressor

MAE: 159,718.9318

R2 Score: 0.5842
```

Gradient Boosting Regressor Model

In [135…
```
from sklearn.ensemble import GradientBoostingRegressor

gradient_model = GradientBoostingRegressor(random_state=42)
gradient_model.fit(X_train, y_train)

gradient_y_pred = gradient_model.predict(X_test)

gradient_mae = mean_absolute_error(y_test, gradient_y_pred)
gradient_r2 = r2_score(y_test, gradient_y_pred)

print("DecisionTreeRegressor\n")
print(f"MAE: {gradient_mae:,.4f}\n")
print(f"R2 Score: {gradient_r2:.4f}")
```

```
DecisionTreeRegressor

MAE: 157,631.8637

R2 Score: 0.5608
```

Hyper Parameter Tuning the Random Forest Regressor

In [138…
```
from sklearn.model_selection import RandomizedSearchCV
import numpy as np

random_grid = {
    'n_estimators': [100, 200, 300, 500],
    'max_features': ['sqrt', 'log2', None],
    'max_depth': [10, 20, 30, 40, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
grid_search = RandomizedSearchCV(estimator=forest_model, param_distributi
grid_search.fit(X_train, y_train)

best_forest_model = grid_search.best_estimator_
best_y_pred = best_forest_model.predict(X_test)

print("Hyper Tuned Random Forest Regressor Model\n")
print(f"Best Parameters: {grid_search.best_params_}\n")
print(f"Best MAE: {mean_absolute_error(y_test, best_y_pred):,.4f}\n")
print(f"Best R2 Score: {r2_score(y_test, best_y_pred):.4f}")
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
Hyper Tuned Random Forest Regressor Model

Best Parameters: {'n_estimators': 200, 'min_samples_split': 2, 'min_sample
s_leaf': 1, 'max_features': 'log2', 'max_depth': None}

Best MAE: 154,199.1606

Best R2 Score: 0.5519
```

Save the final Random Forest Classifier Model

In [139…

```
import joblib

joblib.dump(forest_model, "car_model.pkl")

model_columns = list(X.columns)
joblib.dump(model_columns, "model_columns.pkl")

print("Model and columns successfully saved!")
```

```
Model and columns successfully saved!
```