```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
df = pd.read_csv('/content/vehicle price prediction data set.csv')
df.head()
```

	name	description	make	model	year	price	engine	cylinders	fuel	mileage	transmission	trim	body	c
0	2024 Jeep Wagoneer Series II	\n \n Heated Leather Seats, Nav Sy...	Jeep	Wagoneer	2024	74600.0	24V GDI DOHC Twin Turbo	6.0	Gasoline	10.0	8-Speed Automatic	Series II	SUV	
1	2024 Jeep Grand Cherokee Laredo	Al West is committed to offering every custome...	Jeep	Grand Cherokee	2024	50170.0	OHV	6.0	Gasoline	1.0	8-Speed Automatic	Laredo	SUV	
2	2024 GMC Yukon XL Denali	NaN	GMC	Yukon XL	2024	96410.0	6.2L V-8 gasoline direct injection, variable v...	8.0	Gasoline	0.0	Automatic	Denali	SUV	
3	2023 Dodge Durango Pursuit	White Knuckle Clearcoat 2023 Dodge Durango Pur...	Dodge	Durango	2023	46835.0	16V MPFI OHV	8.0	Gasoline	32.0	8-Speed Automatic	Pursuit	SUV	
4	2024 RAM 3500 Laramie	\n \n 2024 Ram 3500 Laramie Billet...	RAM	3500	2024	81663.0	24V DDI OHV Turbo Diesel	6.0	Diesel	10.0	6-Speed Automatic	Laramie	Pickup Truck	

Next steps: Generate code with df New interactive sheet

```
df.shape
```

```
(1002, 17)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1002 entries, 0 to 1001
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   name            1002 non-null   object
 1   description     946 non-null    object
 2   make            1002 non-null   object
 3   model           1002 non-null   object
 4   year            1002 non-null   int64
 5   price           979 non-null    float64
 6   engine          1000 non-null   object
 7   cylinders       897 non-null    float64
 8   fuel            995 non-null    object
 9   mileage         968 non-null    float64
 10  transmission    1000 non-null   object
 11  trim            1001 non-null   object
 12  body            999 non-null    object
 13  doors           995 non-null    float64
 14  exterior_color  997 non-null    object
 15  interior_color  964 non-null    object
 16  drivetrain      1002 non-null   object
dtypes: float64(4), int64(1), object(12)
memory usage: 133.2+ KB
```

```
df.describe()
```

	year	price	cylinders	mileage	doors
count	1002.000000	979.000000	897.000000	968.000000	995.000000
mean	2023.916168	50202.985700	4.975474	69.033058	3.943719
std	0.298109	18700.392062	1.392526	507.435745	0.274409
min	2023.000000	0.000000	0.000000	0.000000	2.000000
25%	2024.000000	36600.000000	4.000000	4.000000	4.000000
50%	2024.000000	47165.000000	4.000000	8.000000	4.000000
75%	2024.000000	58919.500000	6.000000	13.000000	4.000000
max	2025.000000	195895.000000	8.000000	9711.000000	5.000000

```
df.isnull().sum()
```

	0
name	0
description	56
make	0
model	0
year	0
price	23
engine	2
cylinders	105
fuel	7
mileage	34
transmission	2
trim	1
body	3
doors	7
exterior_color	5
interior_color	38
drivetrain	0

dtype: int64

```
df = df.dropna()
```

```
df = df.drop(['name', 'description'], axis=1)
```

```
le = LabelEncoder()

categorical_columns = [
    'make', 'model', 'engine', 'fuel', 'transmission',
    'trim', 'body', 'exterior_color', 'interior_color', 'drivetrain'
]
```

```
]

for col in categorical_columns:
    df[col] = le.fit_transform(df[col])
```

```
X = df.drop('price', axis=1)
y = df['price']
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
model = RandomForestRegressor(
    n_estimators=200,
    random_state=42
)

model.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor(n_estimators=200, random_state=42)
```

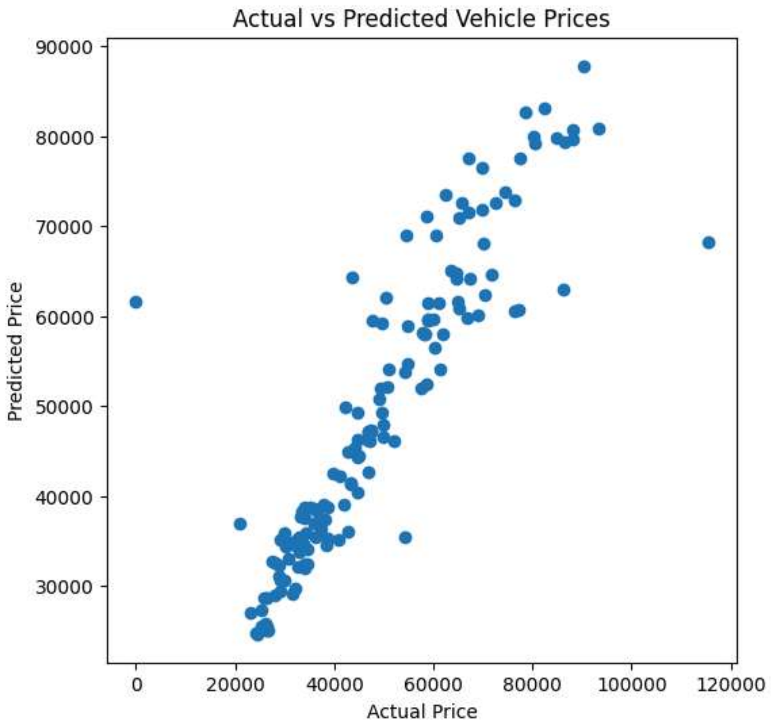
```
y_pred = model.predict(X_test)
```

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("R2 Score:", r2)
```

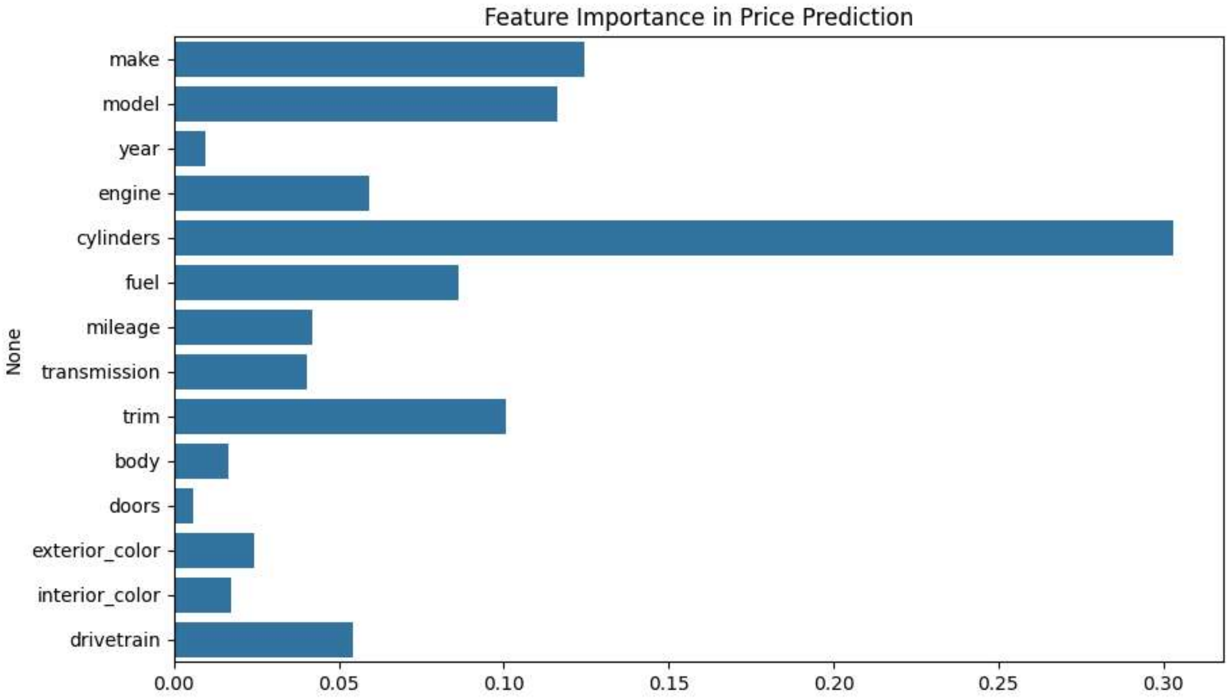
```
Mean Absolute Error (MAE): 4453.061633472823
Mean Squared Error (MSE): 72541272.79104285
Root Mean Squared Error (RMSE): 8517.116459873192
R2 Score: 0.7938057528150575
```

```
plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted Vehicle Prices")
plt.show()
```



```
importance = model.feature_importances_
features = X.columns

plt.figure(figsize=(10,6))
sns.barplot(x=importance, y=features)
plt.title("Feature Importance in Price Prediction")
plt.show()
```



```
sample_vehicle = X.iloc[0:1]
predicted_price = model.predict(sample_vehicle)

print("Predicted Vehicle Price:", predicted_price)
```

```
Predicted Vehicle Price: [77222.155]
```