```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler  #for feature scaling
from sklearn.model_selection import train_test_split  #train-test
split
#for simple linear regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv('/content/dataset.zip')
print("Shape of the dataset:", df.shape)
df.head()
```

```
Shape of the dataset: (15915, 23)
```

{"type":"dataframe","variable_name":"df"}

```python
df.info()
df.describe()

# Check for missing values
missing_values = df.isnull().sum()
missing_values[missing_values > 0]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15915 entries, 0 to 15914
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   make_model            15915 non-null  object 
 1   body_type             15915 non-null  object 
 2   price                 15915 non-null  int64  
 3   vat                   15915 non-null  object 
 4   km                    15915 non-null  float64
 5   Type                  15915 non-null  object 
 6   Fuel                  15915 non-null  object 
 7   Gears                 15915 non-null  float64
 8   Comfort_Convenience   15915 non-null  object 
 9   Entertainment_Media   15915 non-null  object 
 10  Extras                15915 non-null  object 
 11  Safety_Security       15915 non-null  object 
 12  age                   15915 non-null  float64
 13  Previous_Owners       15915 non-null  float64
 14  hp_kW                 15915 non-null  float64
 15  Inspection_new        15915 non-null  int64  
 16  Paint_Type            15915 non-null  object 
 17  Upholstery_type       15915 non-null  object 
 18  Gearing_Type          15915 non-null  object 
```

```
 19  Displacement_cc      15915 non-null  float64
 20  Weight_kg            15915 non-null  float64
 21  Drive_chain          15915 non-null  object
 22  cons_comb            15915 non-null  float64
dtypes: float64(8), int64(2), object(13)
memory usage: 2.8+ MB
```

```
Series([], dtype: int64)
```

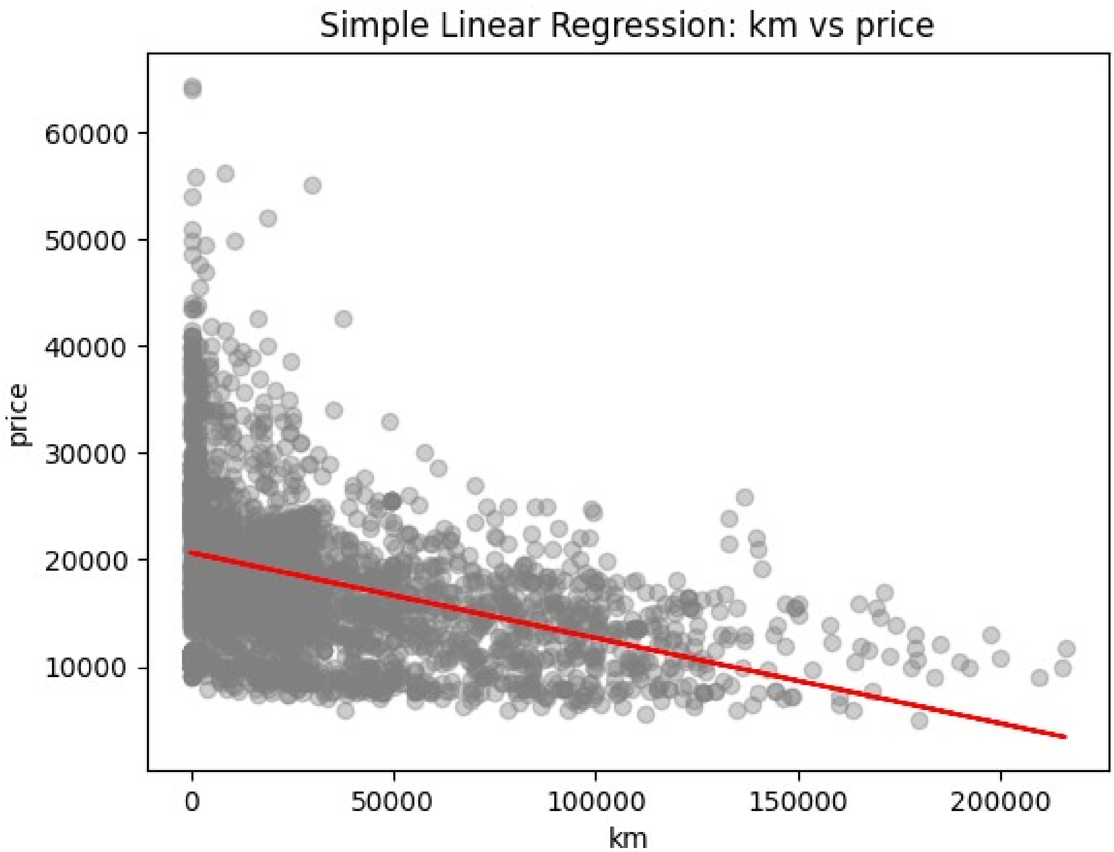
```
# Drop rows with missing values
df_cleaned = df.dropna()
# Select only numeric columns
numeric_df = df_cleaned.select_dtypes(include=[np.number])
# Check again
numeric_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15915 entries, 0 to 15914
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   price            15915 non-null  int64
 1   km               15915 non-null  float64
 2   Gears            15915 non-null  float64
 3   age              15915 non-null  float64
 4   Previous_Owners  15915 non-null  float64
 5   hp_kW            15915 non-null  float64
 6   Inspection_new   15915 non-null  int64
 7   Displacement_cc  15915 non-null  float64
 8   Weight_kg        15915 non-null  float64
 9   cons_comb        15915 non-null  float64
dtypes: float64(8), int64(2)
memory usage: 1.2 MB
```

```
features = ['km', 'hp_kW', 'Displacement_cc', 'Weight_kg',
'Previous_Owners', 'cons_comb', 'Gears', 'age']
target = 'price'

X = df[features]
y = df[target]

# Splitting dataset
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
#Simple Linear Regression:km vs price
X_km = df[['km']]  #one feature that most correlates with the target variable
y_price = df['price']

X_train_km, X_test_km, y_train_km, y_test_km = train_test_split(X_km, y_price, test_size=0.2, random_state=42)

# Model
lr_simple = LinearRegression()
lr_simple.fit(X_train_km, y_train_km)

# Predict and evaluate
y_pred_km = lr_simple.predict(X_test_km)

print("Simple Linear Regression:")
print("R^2 Score:", r2_score(y_test_km, y_pred_km))
print("MSE:", mean_squared_error(y_test_km, y_pred_km))

# Visualization
plt.scatter(X_test_km, y_test_km, color='gray', alpha=0.4)
plt.plot(X_test_km, y_pred_km, color='red')
plt.xlabel('km')
plt.ylabel('price')
plt.title('Simple Linear Regression: km vs price')
plt.show()
```

```
Simple Linear Regression:
R^2 Score: 0.15181949886405333
MSE: 45774249.476677515
```




```
# Use already scaled and split data from earlier
# X_scaled, y, X_train, X_test, y_train, y_test

lr_multi = LinearRegression()
lr_multi.fit(X_train_scaled, y_train)

# Predict and evaluate
y_pred_multi = lr_multi.predict(X_test_scaled)

print("Multiple Linear Regression:")
print("R^2 Score:", r2_score(y_test, y_pred_multi))
print("MSE:", mean_squared_error(y_test, y_pred_multi))
```

```
Multiple Linear Regression:
R^2 Score: 0.757866741684256
MSE: 13067346.110765168
```

```
comparison_df = pd.DataFrame({
    "Model": ["Simple Linear Regression", "Multiple Linear
Regression"],
    "R² Score": [r2, r2_m],
    "MSE": [mse, mse_m]
})
```

```
comparison_df
```

{"summary":"{\n \"name\": \"comparison_df\",\n \"rows\": 2,\n \"fields\": [\n   {\n   \"column\": \"Model\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"Multiple Linear Regression\",\n   \"Simple Linear Regression\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"R\\u00b2 Score\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.1738736253066655,\n   \"min\": 0.5122324660161586,\n   \"max\": 0.7581269050638227,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   0.7581269050638227,\n   0.5122324660161586\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"MSE\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 9383538.871198356,\n   \"min\": 13053305.7226343,\n   \"max\": 26323633.65733814,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   13053305.7226343,\n   26323633.65733814\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"comparison_df"}