```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn import metrics
```

```
df = pd.read_csv('car data.csv.csv')
df
```

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

Next steps: Generate code with df | View recommended plots | New interactive shee

```
from google.colab import drive
drive.mount('/content/drive')
```

```
df.shape
```

```
(4340, 8)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           4340 non-null   object
 1   year           4340 non-null   int64
 2   selling_price  4340 non-null   int64
 3   km_driven      4340 non-null   int64
 4   fuel           4340 non-null   object
 5   seller_type    4340 non-null   object
 6   transmission   4340 non-null   object
 7   owner          4340 non-null   object
dtypes: int64(3), object(5)
memory usage: 271.4+ KB
```

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

	0
name	0
year	0
selling_price	0
km_driven	0
fuel	0
seller_type	0
transmission	0
owner	0

dtype: int64

Table of contents

+ Section

```
# Check the distribution of categorical columns
print(df[['fuel', 'seller_type', 'transmission', 'owner']].apply(pd.Series
```

```
                        fuel  seller_type  transmission   owner
Automatic                NaN          NaN         448.0     NaN
CNG                     40.0          NaN           NaN     NaN
Dealer                   NaN        994.0           NaN     NaN
Diesel                2153.0          NaN           NaN     NaN
Electric                 1.0          NaN           NaN     NaN
First Owner              NaN          NaN           NaN  2832.0
Fourth & Above Owner     NaN          NaN           NaN    81.0
Individual               NaN       3244.0           NaN     NaN
LPG                     23.0          NaN           NaN     NaN
Manual                   NaN          NaN        3892.0     NaN
Petrol                2123.0          NaN           NaN     NaN
Second Owner             NaN          NaN           NaN  1106.0
Test Drive Car           NaN          NaN           NaN    17.0
Third Owner              NaN          NaN           NaN   304.0
Trustmark Dealer         NaN        102.0           NaN     NaN
```

```
# Encoding categorical columns using map() function
df['transmission'] = df['transmission'].map({'Manual': 0, 'Automatic': 1})
df['seller_type'] = df['seller_type'].map({'Individual': 0, 'Dealer': 1})
df['fuel'] = df['fuel'].map({'Petrol': 0, 'Diesel': 1})

# Check the result
print(df[['transmission', 'seller_type', 'fuel']].head())
```

```
   transmission  seller_type  fuel
0             0          0.0   0.0
1             0          0.0   0.0
2             0          0.0   1.0
3             0          0.0   0.0
4             0          0.0   1.0
```

```
from sklearn.model_selection import train_test_split

# Define the features (X) and the target variable (y)
X = df[['year', 'km_driven', 'transmission', 'seller_type', 'fuel']]  # Fe
y = df['selling_price']  # Target variable

# Split the data into training and testing sets (90% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, r

# Check the shape of the splits
print(f"Training data shape: X_train={X_train.shape}, y_train={y_train.sha
print(f"Testing data shape: X_test={X_test.shape}, y_test={y_test.shape}")
```

```
Training data shape: X_train=(3906, 5), y_train=(3906,)
Testing data shape: X_test=(434, 5), y_test=(434,)
```

```
# Remove rows with missing values
df = df.dropna()

# Redefine X and y after removing NaN rows
X = df[['year', 'km_driven', 'transmission', 'seller_type', 'fuel']]
y = df['selling_price']

# Split the data into training and testing sets again
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r

# Train the model
model.fit(X_train, y_train)

# Make predictions and evaluate the model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Output the results
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2): {r2}")
```

```
Mean Squared Error (MSE): 159636917548.56778
R-squared (R2): 0.4752797615156025
```

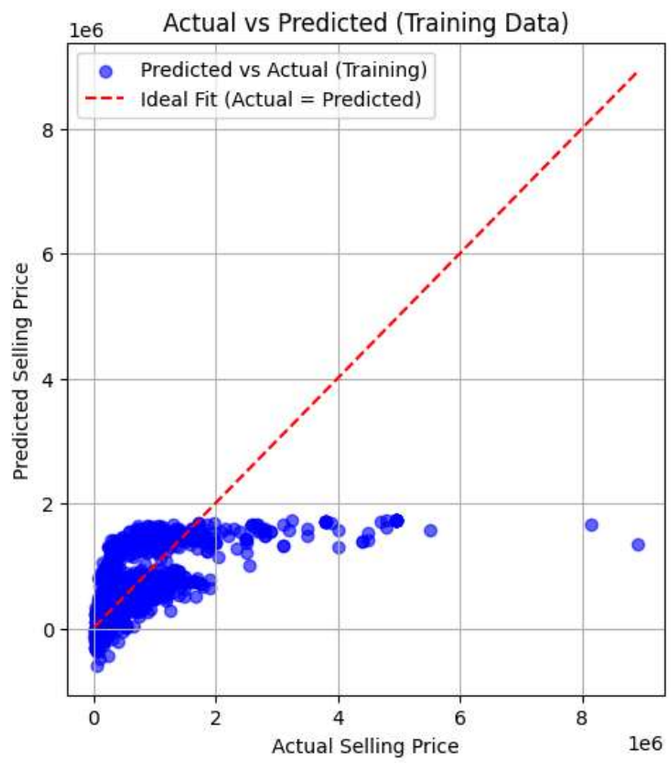
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Initialize the Linear Regression model
model = LinearRegression()
```

```
# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Output the results
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2): {r2}")
```

```
Mean Squared Error (MSE): 159636917548.56778
R-squared (R2): 0.4752797615156025
```

```
import matplotlib.pyplot as plt

# Plot the actual vs predicted selling prices
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.6, label='Predicted vs A

# Plot a line where actual = predicted (diagonal line)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='re

# Add labels and title
plt.title('Actual vs Predicted Selling Prices')
plt.xlabel('Actual Selling Price')
plt.ylabel('Predicted Selling Price')
plt.legend()
plt.grid(True)
plt.show()
```


```
y_train_pred = model.predict(X_train)
plt.figure(figsize=(12, 6))

# Plot for training data
plt.subplot(1, 2, 1)
plt.scatter(y_train, y_train_pred, color='blue', alpha=0.6, label='Predicted
plt.plot([min(y_train), max(y_train)], [min(y_train), max(y_train)], color='
plt.title('Actual vs Predicted (Training Data)')
plt.xlabel('Actual Selling Price')
plt.ylabel('Predicted Selling Price')
plt.legend()
plt.grid(True)
```




```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error, r2_score

# Initialize the Lasso Regression model
lasso_model = Lasso(alpha=0.1)  # You can adjust alpha for regularization st

# Train the Lasso Regression model
lasso_model.fit(X_train, y_train)

# Make predictions on both the training and test data
y_train_lasso_pred = lasso_model.predict(X_train)  # Predictions for trainir
y_test_lasso_pred = lasso_model.predict(X_test)    # Predictions for test da

# Evaluate the model's performance
mse_train_lasso = mean_squared_error(y_train, y_train_lasso_pred)
mse_test_lasso = mean_squared_error(y_test, y_test_lasso_pred)
r2_train_lasso = r2_score(y_train, y_train_lasso_pred)
r2_test_lasso = r2_score(y_test, y_test_lasso_pred)

# Output the results
print(f"Lasso Regression - Training MSE: {mse_train_lasso}")
print(f"Lasso Regression - Test MSE: {mse_test_lasso}")
print(f"Lasso Regression - Training R2: {r2_train_lasso}")
print(f"Lasso Regression - Test R2: {r2_test_lasso}")

# Plot the actual vs predicted prices for both training and test data (Lassc
plt.figure(figsize=(12, 6))

# Plot for training data
plt.subplot(1, 2, 1)
plt.scatter(y_train, y_train_lasso_pred, color='blue', alpha=0.6, label='Pre
plt.plot([min(y_train), max(y_train)], [min(y_train), max(y_train)], color='
plt.title('Actual vs Predicted (Training Data - Lasso Regression)')
plt.xlabel('Actual Selling Price')
plt.ylabel('Predicted Selling Price')
plt.legend()
plt.grid(True)

# Plot for test data
plt.subplot(1, 2, 2)
plt.scatter(y_test, y_test_lasso_pred, color='green', alpha=0.6, label='Prec
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red'
plt.title('Actual vs Predicted (Test Data - Lasso Regression)')
plt.xlabel('Actual Selling Price')
plt.ylabel('Predicted Selling Price')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

```
Lasso Regression - Training MSE: 191225935587.077
Lasso Regression - Test MSE: 159636923601.57385
Lasso Regression - Training R2: 0.4370259521172015
Lasso Regression - Test R2: 0.47527974161961073
```