Car Price Prediction

> About This Project

Here's a summary of the code and the conclusions you can find it:

1. **Importing Dependencies:** You've imported the required libraries, including pandas, matplotlib, seaborn, scikit-learn's LinearRegression, Lasso, and metrics modules.
2. **Data Collection and Processing:** You've loaded a car dataset from a CSV file, explored its shape, and checked for missing values. You've also displayed information about the dataset, including data types of columns and non-null counts.
3. **Encoding the Data:** You've encoded categorical columns like "Fuel_Type," "Seller_Type," and "Transmission" into numerical values for the machine learning models to work with.
4. **Data Splitting:** You've split the data into features (X) and target (Y), and then further split it into training and testing sets using the `train_test_split` function.
5. **Linear Regression Model:** You've created a Linear Regression model, fitted it to the training data, and evaluated its performance using the R-squared score. Additionally, you've plotted scatter plots of actual vs. predicted prices for both training and testing data.
6. **Lasso Regression Model:** You've created a Lasso Regression model, fitted it to the training data, evaluated its performance using the R-squared score, and plotted scatter plots for both training and testing data.

Conclusions:

- **Model Performance:** Both the Linear Regression and Lasso Regression models have been trained and evaluated. The R-squared scores on both the training and testing data are measures of how well the models fit the data. R-squared values close to 1 indicate a good fit, while values closer to 0 indicate less accurate predictions.
- **Scatter Plots:** The scatter plots of actual vs. predicted prices for both training and testing data show how well the models' predictions align with the actual prices. Points close to the diagonal line indicate accurate predictions.
- **Comparison:** By comparing the R-squared scores and scatter plots of both models, you can assess which model performs better on this particular dataset. Higher R-squared scores and closer alignment of points to the diagonal line in scatter plots indicate better model performance.
- **Limitations:** It's important to note that these models might not capture all factors influencing car prices, and their performance could vary with different datasets. Feature engineering, model tuning, and considering more advanced algorithms might further improve predictions.

Remember to include additional details about the dataset, the significance of the chosen features, and the implications of the results when writing the conclusions for a comprehensive analysis.

Importing the Dependencies

```
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
```

Data Collection and Processing

```
# data Load
car_dataset = pd.read_csv('/content/car data.csv')
```

```
car_dataset.head()
```

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer

```
car_dataset.shape
```

```
(301, 9)
```

3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer

```
car_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Seller_Type    301 non-null    object 
 7   Transmission   301 non-null    object 
 8   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

```
car_dataset.isnull().sum()
```

```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

```
# distribution of Categorical data
print(car_dataset.Fuel_Type.value_counts())
print(car_dataset.Seller_Type.value_counts())
print(car_dataset.Transmission.value_counts())
```

```
Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64
Dealer        195
Individual    106
Name: Seller_Type, dtype: int64
Manual       261
Automatic     40
Name: Transmission, dtype: int64
```

Encoding The Data

```
# encoding "Fuel_Type" Column
car_dataset.replace({'Fuel_Type':{'Petrol':0,'Diesel':1,'CNG':2}},inplace=True)

# encoding "Seller_Type" Column
car_dataset.replace({'Seller_Type':{'Dealer':0,'Individual':1}},inplace=True)

# encoding "Transmission" Column
car_dataset.replace({'Transmission':{'Manual':0,'Automatic':1}},inplace=True)
```

```
car_dataset.head()
```

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type
0	ritz	2014	3.35	5.59	27000	0	0
1	sx4	2013	4.75	9.54	43000	1	0
2	ciaz	2017	7.25	9.85	6900	0	0
3	wagon r	2011	2.85	4.15	5200	0	0
4	swift	2014	4.60	6.87	42450	1	0

Test & Training Data

```
X = car_dataset.drop(['Car_Name','Selling_Price'],axis=1)
Y = car_dataset['Selling_Price']
```

```
print(X)
```

```
     Year  Present_Price  Kms_Driven  Fuel_Type  Seller_Type  Transmission  \
0    2014           5.59       27000          0            0             0
1    2013           9.54       43000          1            0             0
2    2017           9.85        6900          0            0             0
3    2011           4.15        5200          0            0             0
4    2014           6.87       42450          1            0             0
..    ...            ...         ...        ...          ...           ...
296  2016          11.60       33988          1            0             0
297  2015           5.90       60000          0            0             0
298  2009          11.00       87934          0            0             0
299  2017          12.50        9000          1            0             0
300  2016           5.90        5464          0            0             0

     Owner
0        0
1        0
2        0
3        0
4        0
..     ...
296      0
297      0
298      0
299      0
300      0

[301 rows x 7 columns]
```

```
print(Y)
```

```
0       3.35
1       4.75
2       7.25
3       2.85
4       4.60
       ...
296     9.50
297     4.00
298     3.35
299    11.50
300     5.30
Name: Selling_Price, Length: 301, dtype: float64
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.1, random_state=2)
```

```
# loading the linear regression model
lin_reg_model = LinearRegression()
```

```
lin_reg_model.fit(X_train,Y_train)
```

```
▾ LinearRegression
LinearRegression()
```

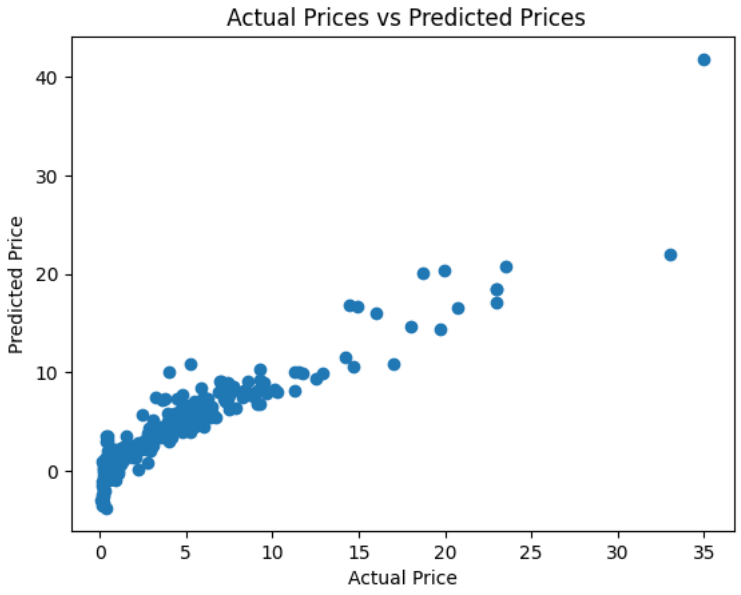
Model Evaluation

```
training_data_prediction = lin_reg_model.predict(X_train)
```

```
error_score = metrics.r2_score(Y_train, training_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.8799451660493711
```

```
plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")
plt.show()
```




```
# prediction on Training data
test_data_prediction = lin_reg_model.predict(X_test)
```

```
# R squared Error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.8365766715027051
```

```
plt.scatter(Y_test, test_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")
plt.show()
```


```
lass_reg_model = Lasso()
```

```
lass_reg_model.fit(X_train,Y_train)
```

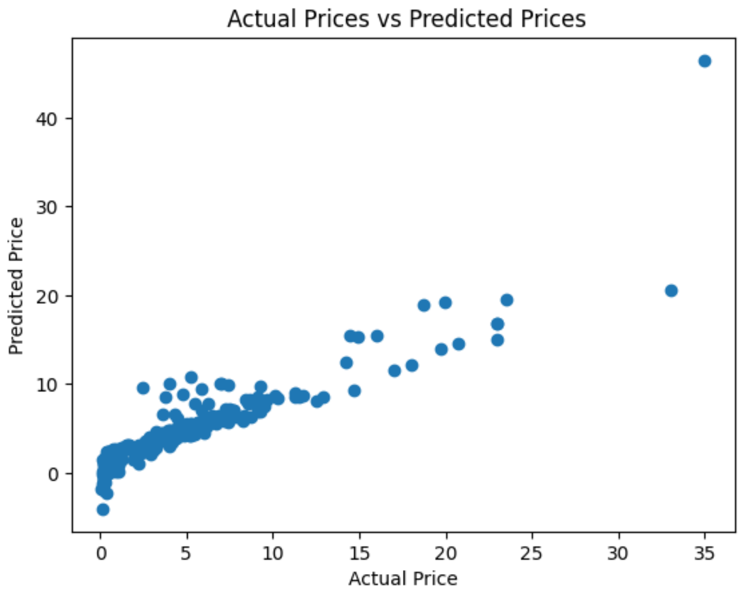
▾ Lasso
Lasso()

```
# prediction on Training data
training_data_prediction = lass_reg_model.predict(X_train)
```

```
# R squared Error
error_score = metrics.r2_score(Y_train, training_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.8427856123435794
```

```
plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")
plt.show()
```

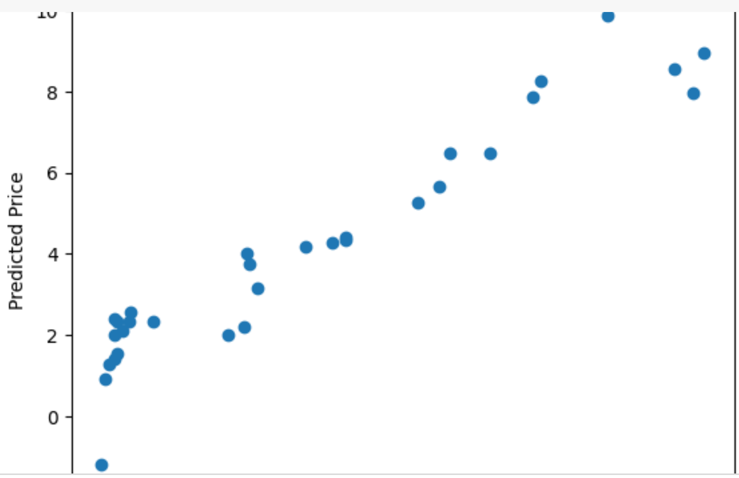



```
# prediction on Training data
test_data_prediction = lass_reg_model.predict(X_test)
```

```
# R squared Error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.8709167941173195
```

```
plt.scatter(Y_test, test_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title(" Actual Prices vs Predicted Prices")
plt.show()
```



✓ 0s completed at 9:49 PM