car-prediction

November 5, 2023

Import Required Libraries

[50]:
```
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

[51]:
```
from google.colab import files

uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving car.csv to car (1).csv
```

[52]:
```
car_dataset = pd.read_csv('car.csv')
```

[53]:
```
car_dataset.head()
```

[53]:

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

	Selling_type	Transmission	Owner
0	Dealer	Manual	0
1	Dealer	Manual	0
2	Dealer	Manual	0
3	Dealer	Manual	0
4	Dealer	Manual	0

[54]:
```
car_dataset.shape
```

[54]: (301, 9)

[55]:
```
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   Driven_kms     301 non-null    int64
 5   Fuel_Type      301 non-null    object
 6   Selling_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
```

[56]:
```
car_dataset.isnull().sum()
```

[56]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Driven_kms       0
Fuel_Type        0
Selling_type     0
Transmission     0
Owner            0
dtype: int64
```

[57]:
```
categorical_columns = ['Car_Name', 'Fuel_Type', 'Selling_type', 'Transmission',␣
 ↪'Owner']

for col in categorical_columns:
    print(car_dataset[col].value_counts())
```

```
city                        26
corolla altis               16
verna                       14
fortuner                    11
brio                        10
                            ..
Honda CB Trigger             1
Yamaha FZ S                  1
```

```
Bajaj Pulsar 135 LS          1
Activa 4g                    1
Bajaj Avenger Street 220     1
Name: Car_Name, Length: 98, dtype: int64
Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64
Dealer        195
Individual    106
Name: Selling_type, dtype: int64
Manual       261
Automatic     40
Name: Transmission, dtype: int64
0    290
1     10
3      1
Name: Owner, dtype: int64
```

One-Hot Encoding

[58]:
```
categorical_columns = ['Fuel_Type', 'Selling_type', 'Transmission']

car_dataset_encoded = pd.get_dummies(car_dataset, columns=categorical_columns,␣
 ↪drop_first=True)
```

[59]:
```
car_dataset_encoded
```

[59]:
```
     Car_Name  Year  Selling_Price  Present_Price  Driven_kms  Owner  \
0        ritz  2014           3.35           5.59       27000      0
1         sx4  2013           4.75           9.54       43000      0
2        ciaz  2017           7.25           9.85        6900      0
3    wagon r   2011           2.85           4.15        5200      0
4       swift  2014           4.60           6.87       42450      0
..        ...   ...            ...            ...         ...    ...
296      city  2016           9.50          11.60       33988      0
297     brio   2015           4.00           5.90       60000      0
298      city  2009           3.35          11.00       87934      0
299      city  2017          11.50          12.50        9000      0
300     brio   2016           5.30           5.90        5464      0

     Fuel_Type_Diesel  Fuel_Type_Petrol  Selling_type_Individual  \
0                   0                 1                        0
1                   1                 0                        0
2                   0                 1                        0
3                   0                 1                        0
4                   1                 0                        0
..                ...               ...                      ...
```

```
296                  1                  0                        0
297                  0                  1                        0
298                  0                  1                        0
299                  1                  0                        0
300                  0                  1                        0

     Transmission_Manual
0                      1
1                      1
2                      1
3                      1
4                      1
..                   …
296                    1
297                    1
298                    1
299                    1
300                    1

[301 rows x 10 columns]
```

Label Encoding:

[24]:
```
from sklearn.preprocessing import LabelEncoder

categorical_columns = ['Car_Name']

label_encoder = LabelEncoder()

car_dataset['Car_Name'] = label_encoder.fit_transform(car_dataset['Car_Name'])
```

[25]:
```
car_dataset.head()
```

[25]:
```
   Car_Name  Year  Selling_Price  Present_Price  Driven_kms Fuel_Type  \
0        90  2014           3.35           5.59       27000    Petrol
1        93  2013           4.75           9.54       43000    Diesel
2        68  2017           7.25           9.85        6900    Petrol
3        96  2011           2.85           4.15        5200    Petrol
4        92  2014           4.60           6.87       42450    Diesel

  Selling_type Transmission  Owner
0       Dealer       Manual      0
1       Dealer       Manual      0
2       Dealer       Manual      0
3       Dealer       Manual      0
4       Dealer       Manual      0
```

Splitting the data and Target

[60]:

[61]:
```
print(X)
```

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

     Fuel_Type_Petrol  Selling_type_Individual  Transmission_Manual
0                   1                        0                    1
1                   0                        0                    1
2                   1                        0                    1
3                   1                        0                    1
4                   0                        0                    1
..                …                        …                    …
296                 0                        0                    1
297                 1                        0                    1
298                 1                        0                    1
299                 0                        0                    1
300                 1                        0                    1

[301 rows x 8 columns]
```

[62]:
```
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
```

Splitting Training and Test Data

Model Training

1. Linear Regression

[67]:

Model Evaluation

[77]:
```
from sklearn.model_selection import train_test_split

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,␣
 ↪random_state=2)
```

[78]:
```
X_train
```

[78]:
```
     Year  Present_Price  Driven_kms  Owner  Fuel_Type_Diesel  \
204  2015          4.430       28282      0                 0
249  2016          7.600       17000      0                 0
277  2015         13.600       21780      0                 0
194  2008          0.787       50000      0                 0
244  2013          9.400       49000      0                 1
..    …              …          …     …                 …
75   2015          6.800       36000      0                 0
22   2011          8.010       50000      0                 0
72   2013         18.610       56001      0                 0
15   2016         10.790       43000      0                 1
168  2013          0.730       12000      0                 0

     Fuel_Type_Petrol  Selling_type_Individual  Transmission_Manual
204                 1                        0                    1
249                 1                        0                    1
277                 1                        0                    1
194                 1                        1                    1
244                 0                        0                    1
..                …                        …                    …
75                  1                        0                    1
22                  1                        0                    0
72                  1                        0                    1
15                  0                        0                    1
168                 1                        1                    1

[270 rows x 8 columns]
```

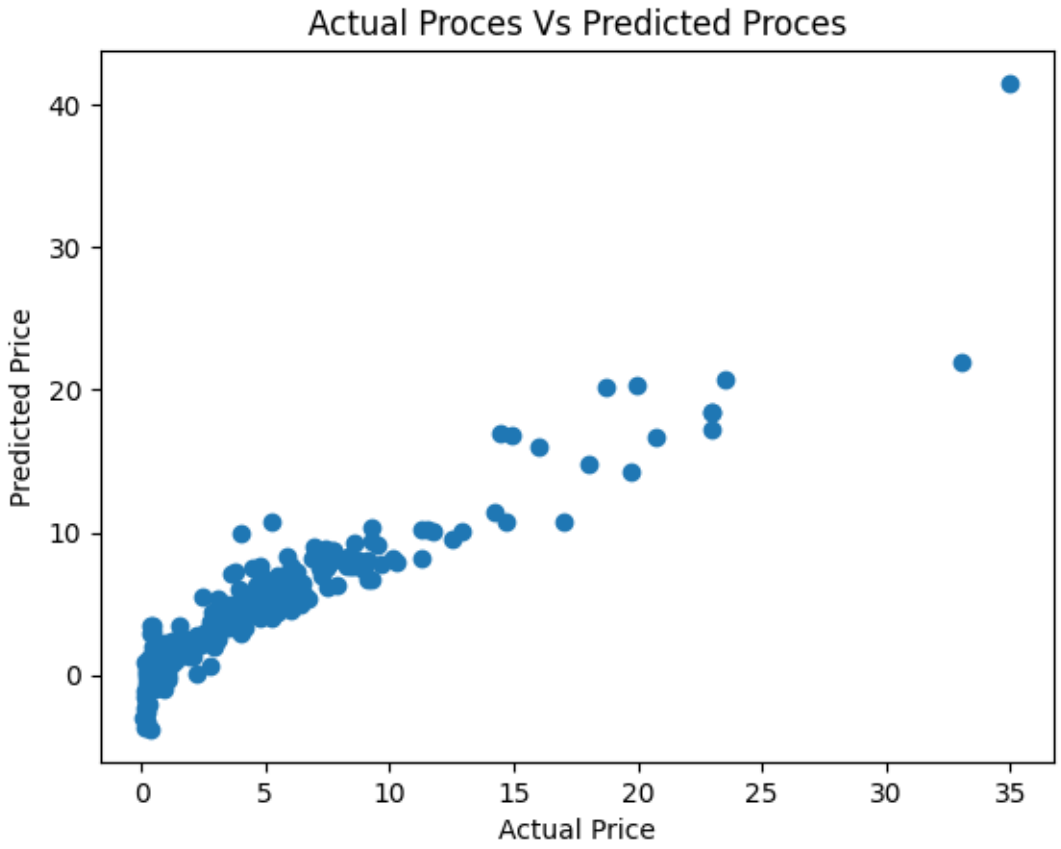
[80]:
```
lin_reg_model = LinearRegression()
lin_reg_model.fit(X_train, y_train)
```

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

[84]:
```
error_score = metrics.r2_score(y_train, training_data_prediction)
print("R squared Error :", error_score)
```

```
R squared Error : 0.8823856405331196
```

[85]:
```
plt.scatter(y_train, training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Proces Vs Predicted Proces")
plt.show()
```



[86]:
```
test_data_prediction = lin_reg_model.predict(X_test)
```

[87]:
```
error_score = metrics.r2_score(y_test, test_data_prediction)
print("R squared Error : ", error_score)
```

```
R squared Error :  0.8694567179819735
```

[88]:
```
plt.scatter(y_test, test_data_prediction)
plt.xlabel('Actual Price')
plt.ylabel("Predicted Price")
plt.title("Actual Proces vs Predicted Prices")
plt.show()
```

[89]:
```
lass_reg_model=Lasso()
```

[90]:
```
lass_reg_model.fit(X_train, y_train)
```

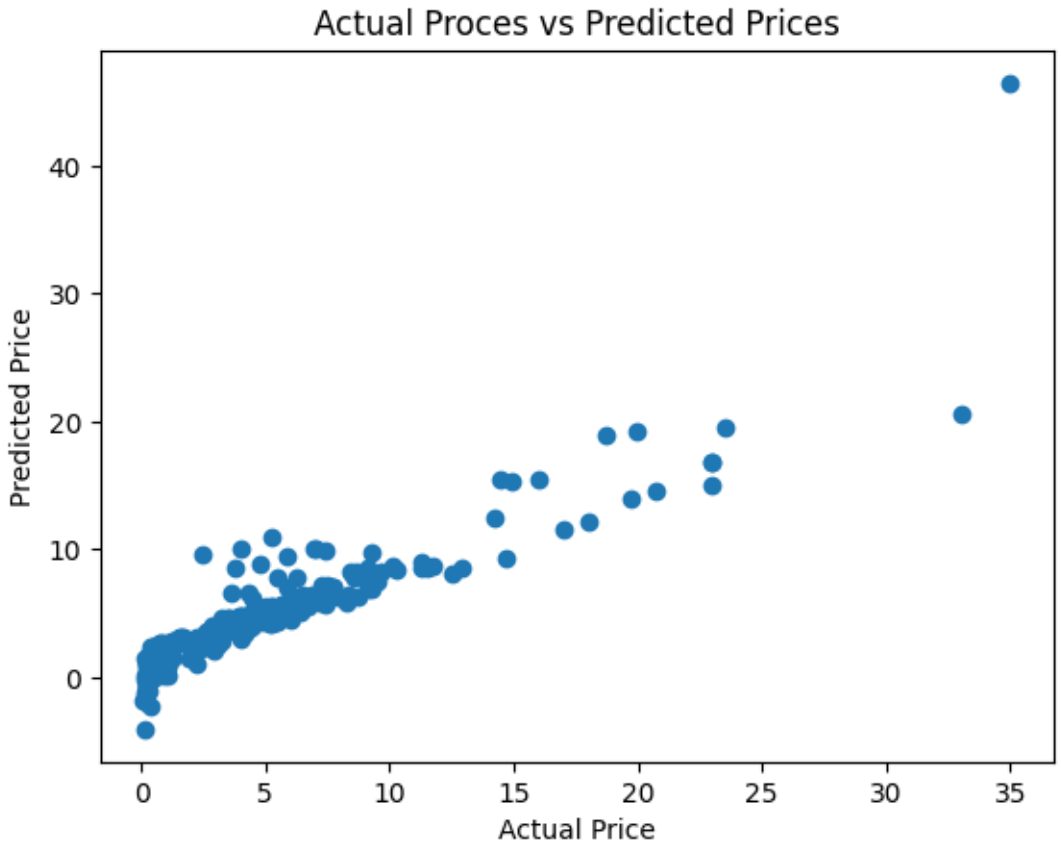
[90]:
```
Lasso()
```

[97]:
```
training_data_prediction = lass_reg_model.predict(X_train)
```

[98]:
```
error_score = metrics.r2_score(y_train, training_data_prediction)
print("R squared Error :", error_score)
```

```
R squared Error : 0.8424480718240743
```

[99]:
```
plt.scatter(y_train, training_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Prices vs Predicted Prices")
plt.show()
```



[100]:
```
#prediction on training data
test_data_prediction = lass_reg_model.predict(X_test)
```

[102]:
```
# R sqaured Error
error_score = metrics.r2_score(y_test, test_data_prediction)
print("R squared error : ", error_score)
```

```
R squared error :  0.8709763132343395
```

[104]:
```
plt.scatter(y_test, test_data_prediction)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Prices vs Predicted Prices")
plt.show()
```


Actual Prices vs Predicted Prices
Predicted Price
Actual Price
0
2
4
6
8
10
12

[ ]: