**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**

```
# loading the data from csv file to pandas dataframe
car_dataset = pd.read_csv('/content/car data.csv')
```

```
# inspecting the first 5 rows of the dataframe
car_dataset.head()
```

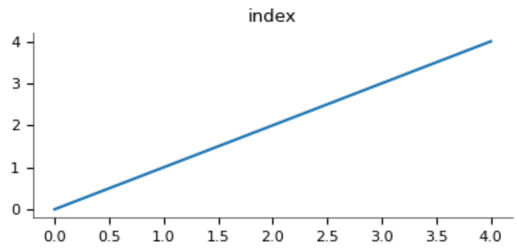
1 to 5 of 5 entries Filter

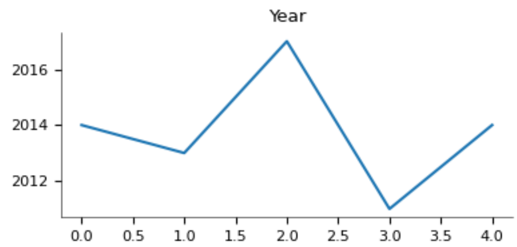
| index | Car_Name | Year | Selling_Price | Present_Price | Kms_Driven | Fuel_Type | Seller_Type | Transmission | Owner |
|---|---|---|---|---|---|---|---|---|---|
| 0 | ritz | 2014 | 3.35 | 5.59 | 27000 | Petrol | Dealer | Manual | 0 |
| 1 | sx4 | 2013 | 4.75 | 9.54 | 43000 | Diesel | Dealer | Manual | 0 |
| 2 | ciaz | 2017 | 7.25 | 9.85 | 6900 | Petrol | Dealer | Manual | 0 |
| 3 | wagon r | 2011 | 2.85 | 4.15 | 5200 | Petrol | Dealer | Manual | 0 |
| 4 | swift | 2014 | 4.6 | 6.87 | 42450 | Diesel | Dealer | Manual | 0 |

Show 25 per page

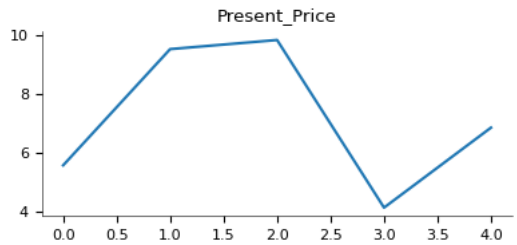
Like what you see? Visit the [data table notebook]() to learn more about interactive tables.

**Values**

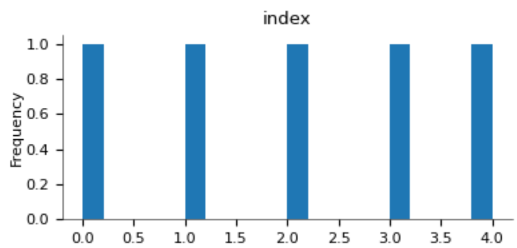


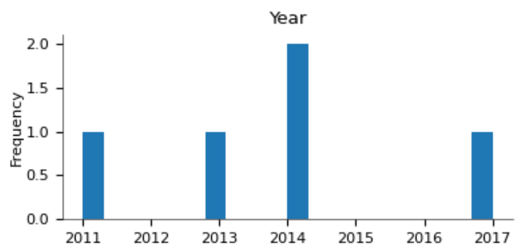



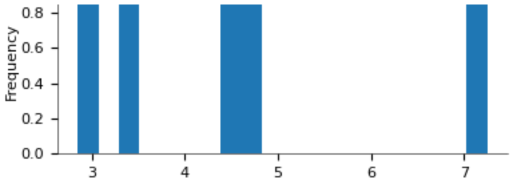


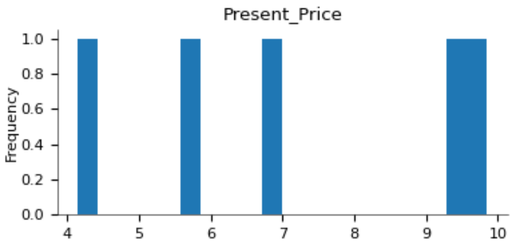



**Distributions**




**Categorical distributions**



```
# checking the number of rows and columns
car_dataset.shape
```

```
(301, 9)
```

```
# getting some information about the dataset
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
```

```
# checking the number of missing values
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
```

```
# checking the 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 Categorical 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()
```

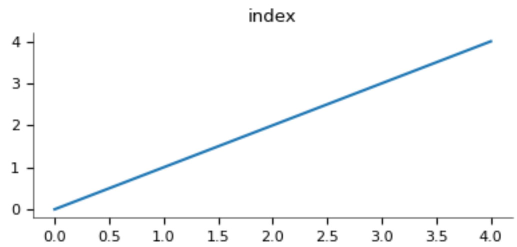
1 to 5 of 5 entries Filter

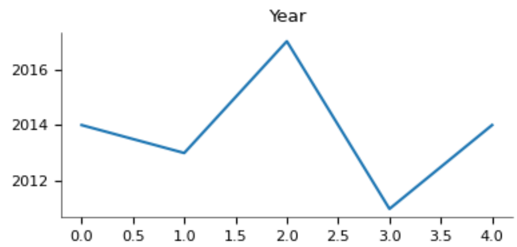
| index | Car_Name | Year | Selling_Price | Present_Price | Kms_Driven | Fuel_Type | Seller_Type | Transmission | Owner |
|---|---|---|---|---|---|---|---|---|---|
| 0 | ritz | 2014 | 3.35 | 5.59 | 27000 | 0 | 0 | 0 | 0 |
| 1 | sx4 | 2013 | 4.75 | 9.54 | 43000 | 1 | 0 | 0 | 0 |
| 2 | ciaz | 2017 | 7.25 | 9.85 | 6900 | 0 | 0 | 0 | 0 |
| 3 | wagon r | 2011 | 2.85 | 4.15 | 5200 | 0 | 0 | 0 | 0 |
| 4 | swift | 2014 | 4.6 | 6.87 | 42450 | 1 | 0 | 0 | 0 |

Show 25 per page

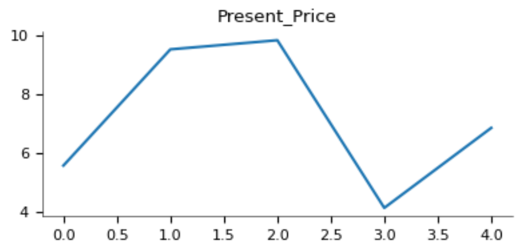
Like what you see? Visit the [data table notebook](data table notebook) to learn more about interactive tables.

**Values**

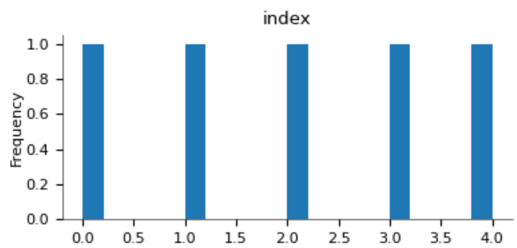


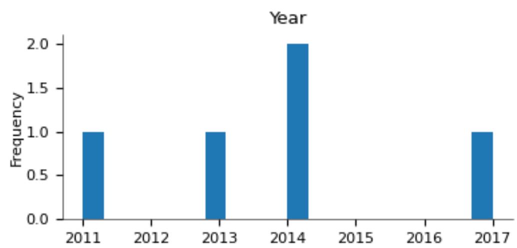








**Distributions**





### Splitting the data and Target

```
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
```

### Splitting Training and Test data

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

### Model Training

1. Linear Regression

```
# loading the linear regression model
lin_reg_model = LinearRegression()
lin_reg_model.fit(X_train,Y_train)
```

```
▾ LinearRegression
LinearRegression()
```

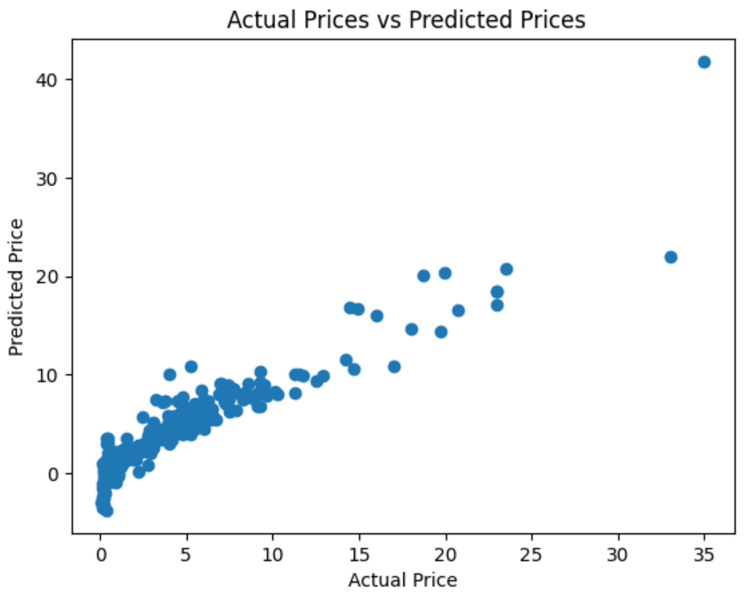
### Model Evaluation

```
# prediction on Training data
training_data_prediction = lin_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 :  0.8799451660493711
```

**Visualize the actual prices and Predicted prices**

```
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()
```


2. Lasso Regression

```
# loading the linear regression model
lass_reg_model = Lasso()
lass_reg_model.fit(X_train,Y_train)
```

```
▾ Lasso
Lasso()
```

**Model Evaluation**

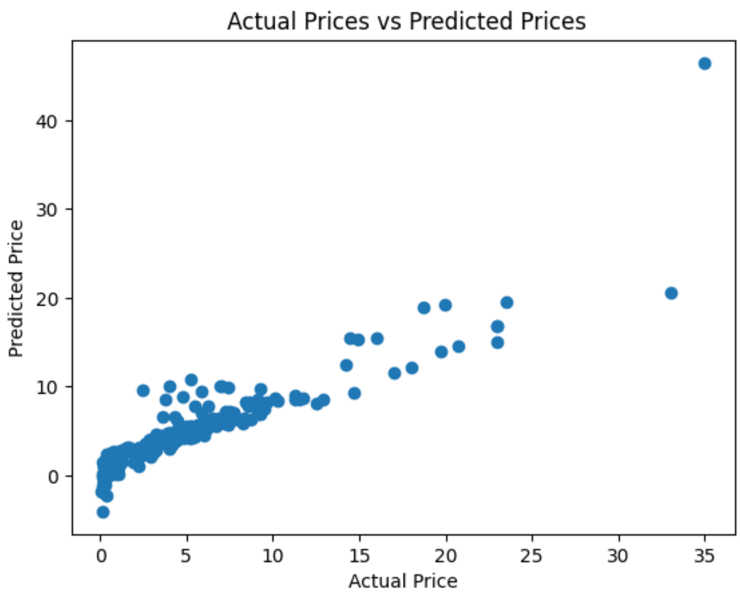
```
# prediction on Training data
training_data_prediction = lass_reg_model.predict(X_train)
```

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

```
R squared Error :  0.8427856123435794
```

**Visualize the actual prices and Predicted prices**

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

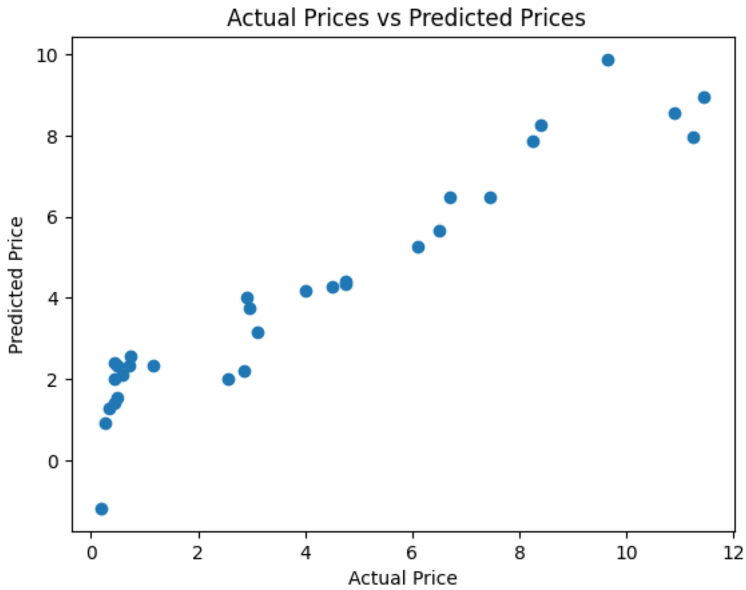



```
# prediction on Training data
test_data_prediction = lass_reg_model.predict(X_test)
```

```
# 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 3:35 PM