EXP2:PERFORM EDA ON THE DATASET AND PREPARE THE DATASET TO TRAIN AND TEST ML MODEL

For a given set of training data examples stored in a .CSV file, demonstrate Data Preprocessing in Machine learning with the following steps a) Getting the dataset. b) Importing libraries. c) Importing datasets. d) Finding Missing Data. e) Encoding Categorical Data.

```
#GETTING DATA SET
import pandas as pd
from google.colab import files
uploaded = files.upload()
```

Choose Files Toyota.csv

- **Toyota.csv**(text/csv) - 65631 bytes, last modified: 6/23/2022 - 100% done

```
Saving Toyota.csv to Toyota.csv
```

```
#IMPORTING DATASET
import pandas as pd
import io
toyotadf=pd.read_csv(io.BytesIO(uploaded['Toyota.csv']),index_col=0)
toyotadf
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23.0	46986	Diesel	90	1.0	0	2000	three	1165
1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...
1431	7500	NaN	20544	Petrol	86	1.0	0	1300	3	1025
1432	10845	72.0	??	Petrol	86	0.0	0	1300	3	1015
1433	8500	NaN	17016	Petrol	86	0.0	0	1300	3	1015
1434	7250	70.0	??	NaN	86	1.0	0	1300	3	1015
1435	6950	76.0	1	Petrol	110	0.0	0	1600	5	1114

1436 rows × 10 columns

```
#IMPORTING LIBRARIES
```

```
import numpy as np
import matplotlib.pyplot as plt

#FINDING DATA TYPES OF EACH COLUMN TO FIND MISSING DATA OR NULL VALUES
toyotadf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436 entries, 0 to 1435
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Price       1436 non-null   int64  
 1   Age         1336 non-null   float64
 2   KM          1436 non-null   object 
 3   FuelType    1336 non-null   object 
 4   HP          1436 non-null   object 
 5   MetColor    1286 non-null   float64
 6   Automatic   1436 non-null   int64  
 7   CC          1436 non-null   int64  
 8   Doors       1436 non-null   object 
 9   Weight      1436 non-null   int64  
dtypes: float64(2), int64(4), object(4)
memory usage: 123.4+ KB
```

```
#TO PRINT FIRST FIVE ROWS OF THE DATASET
toyotadf.head()
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23.0	46986	Diesel	90	1.0	0	2000	three	1165
1	13750	23.0	72937	Diesel	90	1.0	0	2000	3	1165
2	13950	24.0	41711	Diesel	90	NaN	0	2000	3	1165
3	14950	26.0	48000	Diesel	90	0.0	0	2000	3	1165
4	13750	30.0	38500	Diesel	90	0.0	0	2000	3	1170

```
#TO PRINT LAST FIVE ROWS OF THE DATASET
toyotadf.tail()
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
1431	7500	NaN	20544	Petrol	86	1.0	0	1300	3	1025
1432	10845	72.0	??	Petrol	86	0.0	0	1300	3	1015
1433	8500	NaN	17016	Petrol	86	0.0	0	1300	3	1015
1434	7250	70.0	??	NaN	86	1.0	0	1300	3	1015
1435	6950	76.0	1	Petrol	110	0.0	0	1600	5	1114

```
toyotadf["FuelType"].value_counts()
```

```
Petrol    1277
Diesel     144
CNG         15
Name: FuelType, dtype: int64
```

```
toyotadf.shape
```

```
(1436, 10)
```

```
#GIVES NUMBER OF NULL VALUES IN EACH COLUMN
toyotadf.isnull().sum()
```

```
Price          0
Age          100
KM             0
FuelType     100
HP             0
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

```
import numpy as np
np.unique(toyotadf['HP'])
```

```
array(['107', '110', '116', '192', '69', '71', '72', '73', '86', '90',
       '97', '98', '????'], dtype=object)
```

```
np.unique(toyotadf['KM'])
```

```
array(['1', '10000', '100123', ..., '99865', '99971', '??'], dtype=object)
```

```
np.unique(toyotadf['Doors'])
```

```
array(['2', '3', '4', '5', 'five', 'four', 'three'], dtype=object)
```

```
toyotadf['Automatic'].unique()
```

```
array([0, 1])
```

```
toyotadf['MetColor'].unique()
```

```
array([ 1., nan,  0.])
```

```
import pandas as pd
import io
toyotadf=pd.read_csv(io.BytesIO(uploaded['Toyota.csv']),index_col=0,na_values=["??","????"])
toyotadf
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23.0	46986.0	Diesel	90.0	1.0	0	2000	three	1165
1	13750	23.0	72937.0	Diesel	90.0	1.0	0	2000	3	1165
2	13950	24.0	41711.0	Diesel	90.0	NaN	0	2000	3	1165
3	14950	26.0	48000.0	Diesel	90.0	0.0	0	2000	3	1165
4	13750	30.0	38500.0	Diesel	90.0	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...
1431	7500	NaN	20544.0	Petrol	86.0	1.0	0	1300	3	1025
1432	10845	72.0	NaN	Petrol	86.0	0.0	0	1300	3	1015
1433	8500	NaN	17016.0	Petrol	86.0	0.0	0	1300	3	1015
1434	7250	70.0	NaN	NaN	86.0	1.0	0	1300	3	1015
1435	6950	76.0	1.0	Petrol	110.0	0.0	0	1600	5	1114

1436 rows × 10 columns

```
toyotadf['Doors'].unique()
```

```
array(['three', '3', '5', '4', 'four', 'five', '2'], dtype=object)
```

```
toyotadf['Doors'].replace('three',3,inplace=True)
toyotadf['Doors'].replace('four',4,inplace=True)
toyotadf['Doors'].replace('five',5,inplace=True)
toyotadf
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23.0	46986.0	Diesel	90.0	1.0	0	2000	3	1165
1	13750	23.0	72937.0	Diesel	90.0	1.0	0	2000	3	1165
2	13950	24.0	41711.0	Diesel	90.0	NaN	0	2000	3	1165
3	14950	26.0	48000.0	Diesel	90.0	0.0	0	2000	3	1165
4	13750	30.0	38500.0	Diesel	90.0	0.0	0	2000	3	1170

```
toyotadf['Doors'].unique()
```

```
array([3, '3', '5', '4', 4, 5, '2'], dtype=object)
```

1432	10845	72.0	NaN	Petrol	86.0	0.0	0	1300	3	1015

```
toyotadf['Doors'] = toyotadf['Doors'].astype('int')
```

```
toyotadf['Doors'].unique()
```

```
array([3, 5, 4, 2])
```

```
toyotadf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436 entries, 0 to 1435
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Price       1436 non-null   int64  
 1   Age         1336 non-null   float64
 2   KM          1421 non-null   float64
 3   FuelType    1336 non-null   object 
 4   HP          1430 non-null   float64
 5   MetColor    1286 non-null   float64
 6   Automatic   1436 non-null   int64  
 7   CC          1436 non-null   int64  
 8   Doors       1436 non-null   int64  
 9   Weight      1436 non-null   int64  
dtypes: float64(4), int64(5), object(1)
memory usage: 123.4+ KB
```

Let's focus on NULL values....

```
toyotadf.isnull().sum()
```

```
Price          0
Age          100
KM            15
FuelType     100
HP             6
```

```
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

```
toyotadf.isnull().sum().sum()
```

```
371
```

```
toyotadf.describe()
```

	Price	Age	KM	HP	MetColor	Automatic	
ount	1436.000000	1336.000000	1421.000000	1430.000000	1286.000000	1436.000000	1436.(
ean	10730.824513	55.672156	68647.239972	101.478322	0.674961	0.055710	1566.8
std	3626.964585	18.589804	37333.023589	14.768255	0.468572	0.229441	187.1
nin	4350.000000	1.000000	1.000000	69.000000	0.000000	0.000000	1300.(
5%	8450.000000	43.000000	43210.000000	90.000000	0.000000	0.000000	1400.(
0%	9900.000000	60.000000	63634.000000	110.000000	1.000000	0.000000	1600.(
5%	11950.000000	70.000000	87000.000000	110.000000	1.000000	0.000000	1600.(
nax	32500.000000	80.000000	243000.000000	192.000000	1.000000	1.000000	2000.(

```
toyotadf['Age'].tail(10)
```

```
1426    78.0
1427     NaN
1428    72.0
1429    78.0
1430    80.0
1431     NaN
1432    72.0
1433     NaN
1434    70.0
1435    76.0
Name: Age, dtype: float64
```

```
toyotadf['Age'].mean()
```

```
55.67215568862275
```

```
toyotadf['Age'].fillna(toyotadf['Age'].mean(), inplace=True)
```

```
toyotadf['Age'].tail(10)
```

```
1426    78.000000
1427    55.672156
1428    72.000000
1429    78.000000
1430    80.000000
1431    55.672156
1432    72.000000
1433    55.672156
1434    70.000000
1435    76.000000
Name: Age, dtype: float64
```

```
toyotadf['KM'].head(10)
```

```
0    46986.0
1    72937.0
2    41711.0
3    48000.0
4    38500.0
5    61000.0
6        NaN
7    75889.0
8    19700.0
9    71138.0
Name: KM, dtype: float64
```

```
toyotadf['KM'].fillna(toyotadf['KM'].median(), inplace=True)
toyotadf['KM'].median()
```

```
63634.0
```

```
toyotadf['KM'].head(10)
```

```
0    46986.0
1    72937.0
2    41711.0
3    48000.0
4    38500.0
5    61000.0
6    63634.0
7    75889.0
8    19700.0
9    71138.0
Name: KM, dtype: float64
```

```
toyotadf['FuelType'].value_counts()
```

```
Petrol    1177
Diesel     144
CNG         15
Name: FuelType, dtype: int64
```

```
toyotadf['FuelType'].fillna(toyotadf['FuelType'].value_counts().index[0], inplace=True)
```

```
toyotadf['MetColor'].value_counts()
```

```
1.0    868
0.0    418
Name: MetColor, dtype: int64
```

```
toyotadf['MetColor'].mode()
```

```
0    1.0
dtype: float64
```

```
toyotadf['MetColor'].fillna(toyotadf['MetColor'].mode().index[0], inplace=True)
```

```
toyotadf.isnull().sum()
```

```
Price        0
Age          0
KM           0
FuelType     0
HP           6
MetColor     0
Automatic    0
CC           0
Doors        0
Weight       0
dtype: int64
```

```
toyotadf['HP'].fillna(toyotadf['HP'].mean(), inplace=True)
```

```
toyotadf.isnull().sum()
```

```
Price        0
Age          0
KM           0
FuelType     0
HP           0
MetColor     0
Automatic    0
CC           0
Doors        0
Weight       0
dtype: int64
```

```
#ENCODING CATEGORICAL DATA
one_hot_encoded_data = pd.get_dummies(toyotadf, columns = ['FuelType'])
print(one_hot_encoded_data)
```

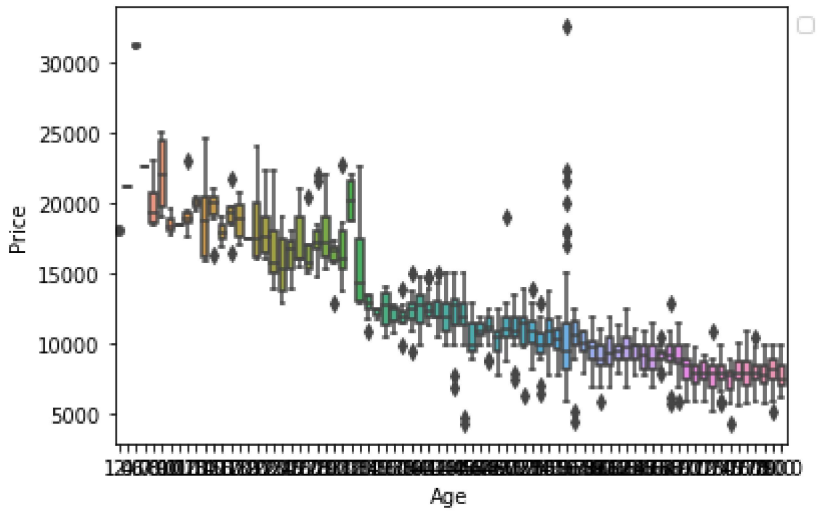
```
      Price        Age       KM     HP  MetColor  Automatic    CC  Doors  \
0     13500  23.000000  46986.0   90.0       1.0          0  2000      3
1     13750  23.000000  72937.0   90.0       1.0          0  2000      3
2     13950  24.000000  41711.0   90.0       0.0          0  2000      3
3     14950  26.000000  48000.0   90.0       0.0          0  2000      3
4     13750  30.000000  38500.0   90.0       0.0          0  2000      3
...     ...        ...      ...    ...       ...        ...   ...    ...
1431   7500  55.672156  20544.0   86.0       1.0          0  1300      3
1432  10845  72.000000  63634.0   86.0       0.0          0  1300      3
1433   8500  55.672156  17016.0   86.0       0.0          0  1300      3
1434   7250  70.000000  63634.0   86.0       1.0          0  1300      3
1435   6950  76.000000      1.0  110.0       0.0          0  1600      5

      Weight  FuelType_CNG  FuelType_Diesel  FuelType_Petrol
0       1165             0                1                0
1       1165             0                1                0
2       1165             0                1                0
3       1165             0                1                0
4       1170             0                1                0
...      ...           ...              ...              ...
1431    1025             0                0                1
1432    1015             0                0                1
1433    1015             0                0                1
1434    1015             0                0                1
1435    1114             0                0                1

[1436 rows x 12 columns]
```

```
a = one_hot_encoded_data['Price']
b = one_hot_encoded_data['Age']

plt.scatter(a,b, color = 'm')

plt.xlabel('Price')
plt.ylabel('Age')
```

```
Text(0, 0.5, 'Age')
```


```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.boxplot( x="Age", y='Price', data=one_hot_encoded_data )
plt.show()
```

```
WARNING:matplotlib.legend:No handles with labels found to put in legend.
```




```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.histplot(x='Age', data=one_hot_encoded_data )
plt.show()
```

Seaborn is a Python data visualization library based on matplotlib.

It provides a high-level interface for drawing attractive and informative statistical graphics.

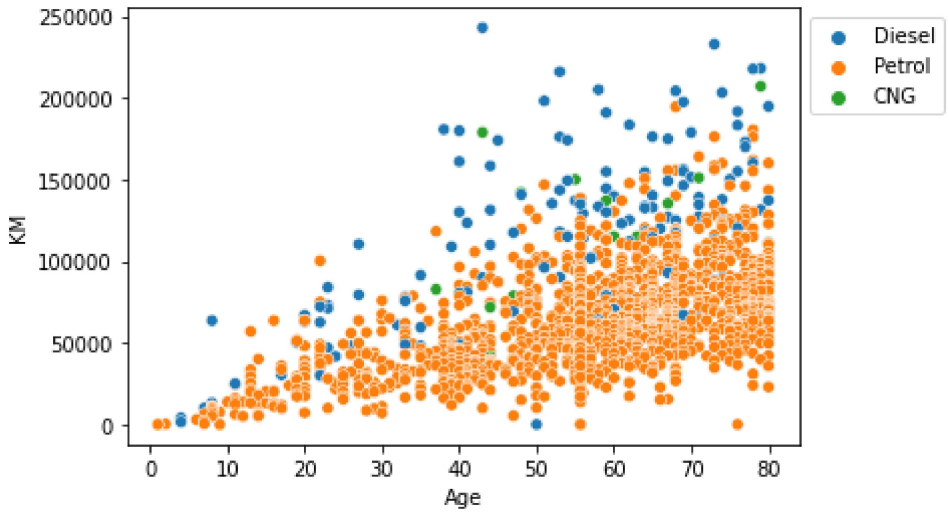
seaborn.pairplot() :

To plot multiple pairwise bivariate distributions in a dataset, you can use the .pairplot() function.

The diagonal plots are the univariate plots, and this displays the relationship for the (n, 2) combination of variables in a DataFrame as a matrix of plots.

**hue* *Variable in “data“ to map plot aspects to different colors.

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt


sns.scatterplot( x="Age", y='KM', data=toyotadf,
                hue='FuelType')

# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()
```




```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
```

```
sns.pairplot(toyotadf, hue='FuelType', height=2)
```

```
<seaborn.axisgrid.PairGrid at 0x7eff497e3b50>
```

Colab paid products - Cancel contracts here

✓ 0s completed at 10:52 AM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.