EDA

- Exploratory Data Analysis
- Knowledge Extraction
- Modeling

Descriptive Analysis

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# from google.colab import files
# uploaded = files.upload()
# D3data1.csv

import os
os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
os.getcwd()
```

```
'C:\\Users\\surya\\Downloads\\PG-DBDA-Mar23\\Datasets'
```

```
data = pd.read_csv('D3data1.csv')
data
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	TM195	18	Male	14	Single	3	4	29562	112
1	TM195	19	Male	15	Single	2	3	31836	75
2	TM195	19	Female	14	Partnered	4	3	30699	66
3	TM195	19	Male	12	Single	3	3	32973	85
175	TM798	40	Male	21	Single	6	5	83416	200

df.shape

```
1 data.shape
```

```
(180, 9)
```

df.describe()

180 rows × 9 columns

```
1 data.describe()
```

	Age	Education	Usage	Fitness	Income	Miles
count	180.000000	180.000000	180.000000	180.000000	180.000000	180.000000
mean	28.788889	15.572222	3.455556	3.311111	53719.577778	103.194444
std	6.943498	1.617055	1.084797	0.958869	16506.684226	51.863605
min	18.000000	12.000000	2.000000	1.000000	29562.000000	21.000000
25%	24.000000	14.000000	3.000000	3.000000	44058.750000	66.000000
50%	26.000000	16.000000	3.000000	3.000000	50596.500000	94.000000
75%	33.000000	16.000000	4.000000	4.000000	58668.000000	114.750000
max	50.000000	21.000000	7.000000	5.000000	104581.000000	360.000000

df.head()

```
1 data.head()
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	TM195	18	Male	14	Single	3	4	29562	112
1	TM195	19	Male	15	Single	2	3	31836	75
2	TM195	19	Female	14	Partnered	4	3	30699	66
3	TM195	19	Male	12	Single	3	3	32973	85

df.info()

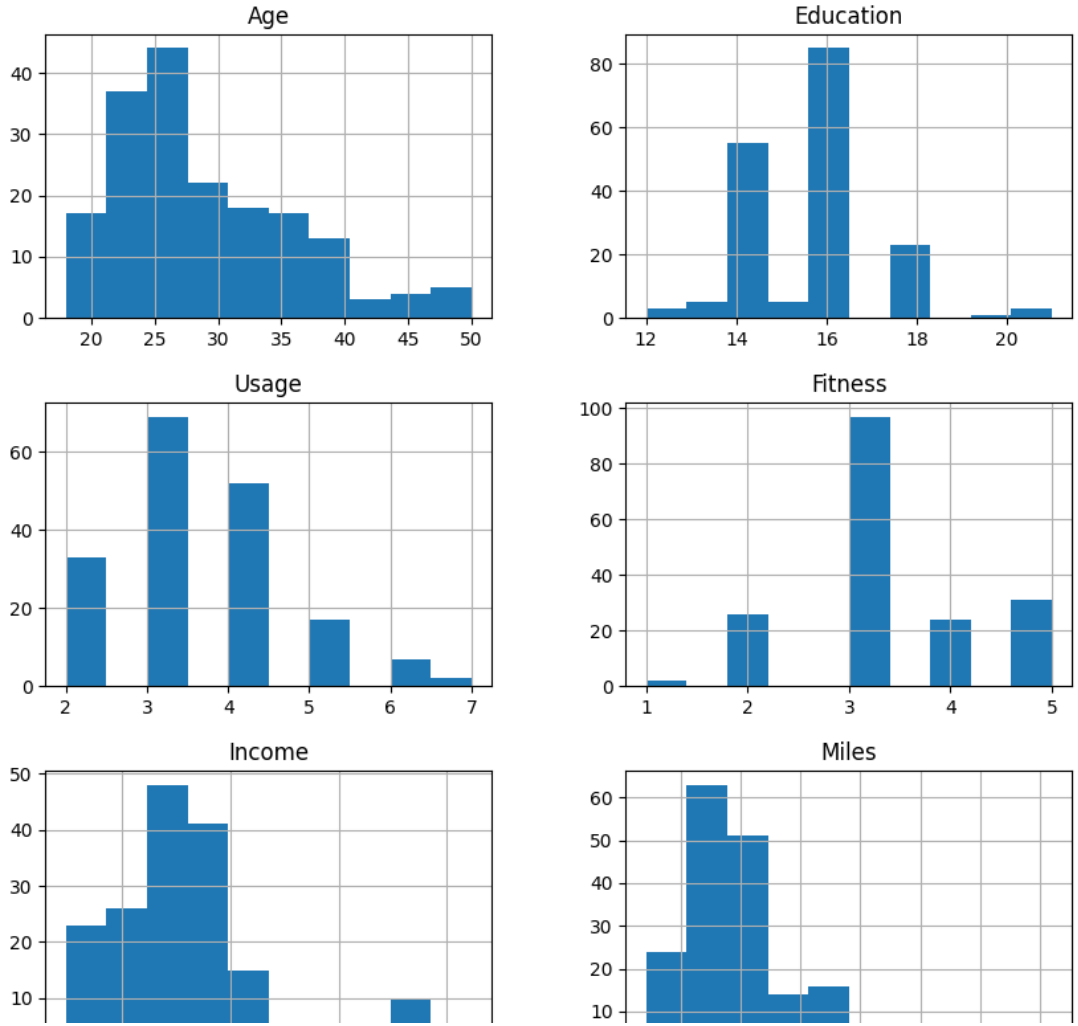
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Product        180 non-null    object
 1   Age            180 non-null    int64 
 2   Gender         180 non-null    object
 3   Education      180 non-null    int64 
 4   MaritalStatus  180 non-null    object
 5   Usage          180 non-null    int64 
 6   Fitness        180 non-null    int64 
 7   Income         180 non-null    int64 
 8   Miles          180 non-null    int64 
dtypes: int64(6), object(3)
memory usage: 12.8+ KB
```

df.hist(figsize(10, 10))

```
1 data.hist(figsize=(10, 10))
2 # pandas histogram function
```

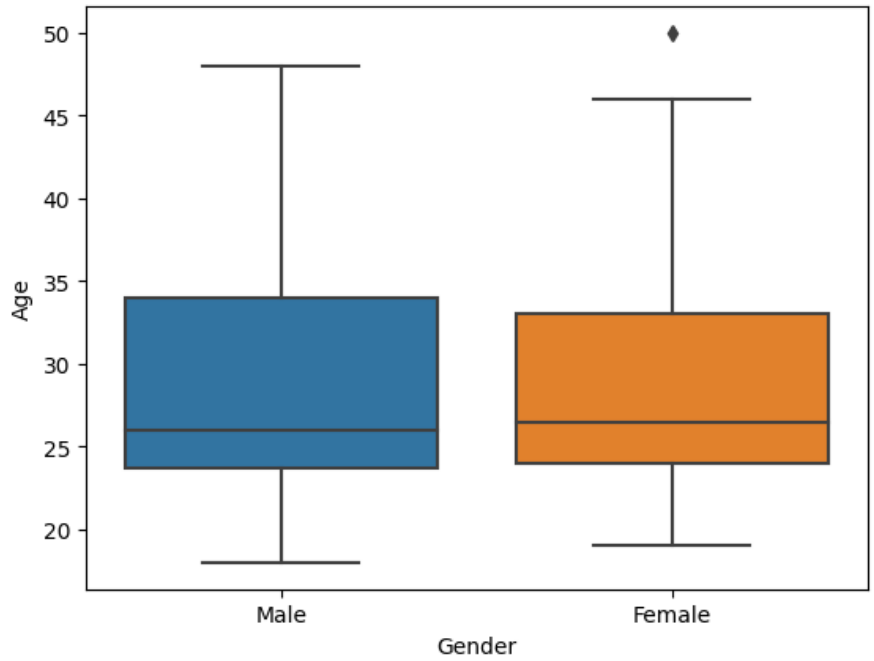
```
array([[<Axes: title={'center': 'Age'}>,
        <Axes: title={'center': 'Education'}>],
       [<Axes: title={'center': 'Usage'}>,
        <Axes: title={'center': 'Fitness'}>],
       [<Axes: title={'center': 'Income'}>,
        <Axes: title={'center': 'Miles'}>]], dtype=object)
```



seaborn.boxplot(x='a', y='b', data=df)

```
sns.boxplot(x='Gender', y='Age', data=data)
# seaborn boxplot
```

```
<Axes: xlabel='Gender', ylabel='Age'>
```

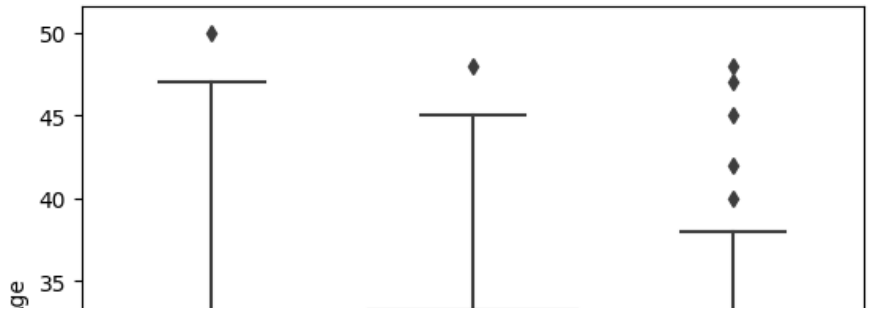


- observation
  - range, median, outliers, min, max

seaborn.boxplot(x='a', y='b', data=df)

```
1 sns.boxplot(x='Product', y='Age', data=data)
```

```
<Axes: xlabel='Product', ylabel='Age'>
```



- conclusion
  - three boxplots for three different categories of Product

pd.crosstab(df['a'], df['b'])

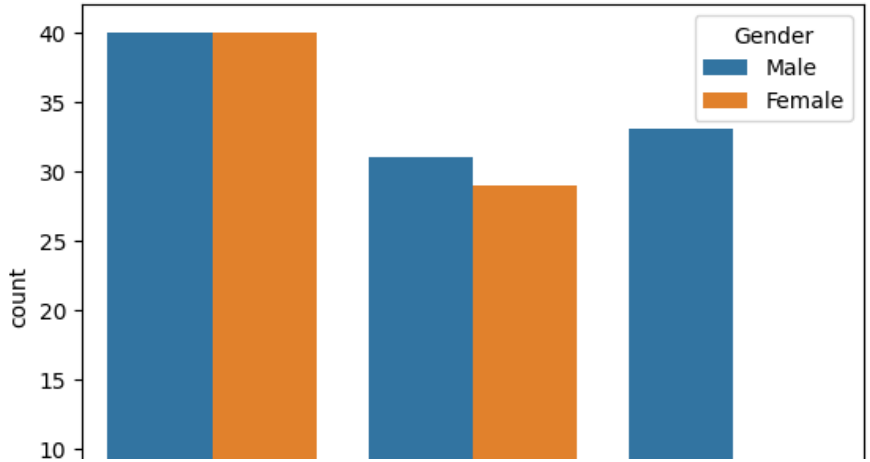
```
1 pd.crosstab(data['Product'], data['Gender'])
2 # Compute a simple cross tabulation of two (or more) factors
```

Gender	Female	Male
Product		
TM195	40	40
TM498	29	31
TM798	7	33

seaborn.countplot(x='a', y='b', data=df)

```
1 sns.countplot(x='Product', hue='Gender', data=data)
2 # countplot
```

```
<Axes: xlabel='Product', ylabel='count'>
```



pd.pivot_table(df, index=[ ], columns=[ ], aggfunc='len')

```
pd.pivot_table(data, index=['Product', 'Gender'], columns=['MaritalStatus'], aggfunc=len)
# length function
```

		Age		Education		Fitness		Income
	MaritalStatus	Partnered	Single	Partnered	Single	Partnered	Single	Partnered
Product	Gender							
TM195	Female	27	13	27	13	27	13	27
	Male	21	19	21	19	21	19	21
TM498	Female	15	14	15	14	15	14	15
	Male	21	10	21	10	21	10	21
TM798	Female	4	3	4	3	4	3	4
	Male	19	14	19	14	19	14	19

```
pd.pivot_table(data, index=['Product', 'Gender'], columns=['MaritalStatus'], aggfunc='mean')
# Create a spreadsheet-style pivot table as a DataFrame
# mean function
```

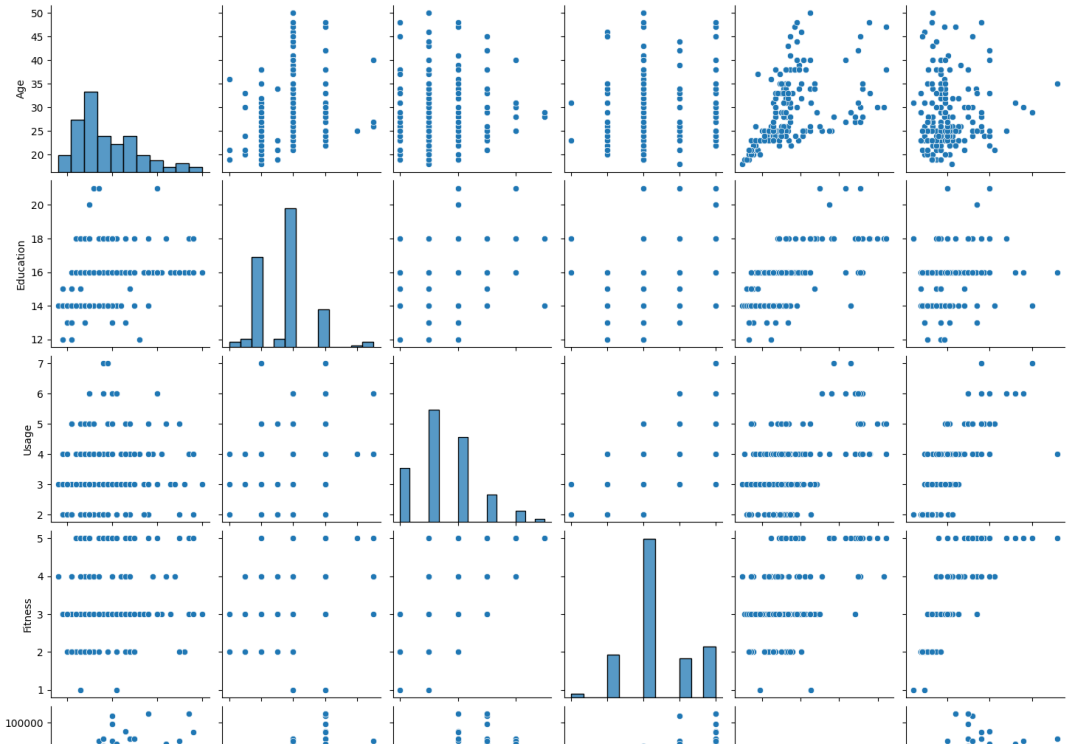
		Age		Education		Fitness		Inc
	MaritalStatus	Partnered	Single	Partnered	Single	Partnered	Single	Par
Product	Gender							
TM195	Female	28.333333	28.692308	14.888889	15.538462	2.851852	2.923077	461
	Male	31.380952	25.631579	15.428571	14.473684	2.857143	3.263158	500
TM498	Female	30.000000	28.142857	15.200000	15.214286	2.933333	2.785714	497
	Male	30.380952	25.200000	15.285714	14.500000	2.904762	3.000000	493

```
pd.pivot_table(data, 'Miles', columns=['MaritalStatus'], aggfunc='mean')
# mean of miles grouped by MaritalStatus
```

MaritalStatus	Partnered	Single
Miles	104.28972	101.589041

```
sns.pairplot(data)
```

```
c:\users\surya\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
  self._figure.tight_layout(*args, **kwargs)
<seaborn.axisgrid.PairGrid at 0x1e63599a700>
```



df.corr()

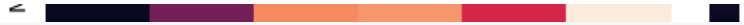
```
1 corr = data.iloc[ : , [1, 3, 5, 6, 7, 8]].corr()
```

```
1 sns.heatmap(corr, annot=True)
```

```
<Axes: >
```

Exploratory Data Analysis

EDA1 : Games



```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 %matplotlib inline
```

```
1 # from google.colab import files
2 # uploaded = files.upload()
3 # D3data2.csv
4
5 import os
6 os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
7 os.getcwd()
```

```
'C:\\Users\\surya\\Downloads\\PG-DBDA-Mar23\\Datasets'
```

```
1 data = pd.read_csv('D3data2.csv')
```

```
1 data.shape
```

```
(81312, 20)
```

```
1 data.describe()
```

	id	yearpublished	minplayers	maxplayers	playingtime	minplaytime
count	81312.000000	81309.000000	81309.000000	81309.000000	81309.000000	81309.000000
mean	72278.150138	1806.630668	1.992018	5.637703	51.634788	49.276833
std	58818.237742	588.517834	0.931034	56.076890	345.699969	334.483934
min	1.000000	-3500.000000	0.000000	0.000000	0.000000	0.000000
25%	21339.750000	1984.000000	2.000000	2.000000	8.000000	10.000000
50%	43258.000000	2003.000000	2.000000	4.000000	30.000000	30.000000
75%	128836.500000	2010.000000	2.000000	6.000000	60.000000	60.000000
max	184451.000000	2018.000000	99.000000	11299.000000	60120.000000	60120.000000

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81312 entries, 0 to 81311
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    81312 non-null  int64  
 1   type                  81312 non-null  object 
 2   name                  81271 non-null  object 
 3   yearpublished         81309 non-null  float64
 4   minplayers            81309 non-null  float64
 5   maxplayers            81309 non-null  float64
 6   playingtime           81309 non-null  float64
 7   minplaytime           81309 non-null  float64
 8   maxplaytime           81309 non-null  float64
 9   minage                81309 non-null  float64
 10  users_rated           81312 non-null  int64  
 11  average_rating        81312 non-null  float64
 12  bayes_average_rating  81312 non-null  float64
 13  total_owners          81312 non-null  int64  
 14  total_traders         81312 non-null  int64  
 15  total_wanters         81312 non-null  int64  
 16  total_wishers         81312 non-null  int64  
 17  total_comments        81312 non-null  int64  
 18  total_weights         81312 non-null  int64  
 19  average_weight        81312 non-null  float64
dtypes: float64(10), int64(8), object(2)
memory usage: 12.4+ MB
```

```
1 data.head()
```

	id	type	name	yearpublished	minplayers	maxplayers	playingtime	minpla
0	12333	boardgame	Twilight Struggle	2005.0	2.0	2.0	180.0	
1	120677	boardgame	Terra Mystica	2012.0	2.0	5.0	150.0	
2	102794	boardgame	Caverna: The Cave Farmers	2013.0	1.0	7.0	210.0	
3	25613	boardgame	Through the Ages: A Story of Civilization	2006.0	2.0	4.0	240.0	
4	3076	boardgame	Puerto Rico	2002.0	2.0	5.0	150.0	

null check

```
1 data.isnull().sum()
```

```
id                     0
type                   0
name                  41
yearpublished          3
minplayers             3
maxplayers             3
playingtime            3
minplaytime            3
maxplaytime            3
minage                 3
users_rated            0
average_rating         0
bayes_average_rating   0
total_owners           0
total_traders          0
total_wanters          0
total_wishers          0
total_comments         0
total_weights          0
average_weight         0
dtype: int64
```

df.dropna()

```
data.dropna(inplace=True)
# dropping records with null values
```

```
data.shape
```

```
(81268, 20)
```

```
data.head()
```

	id	type	name	yearpublished	minplayers	maxplayers	playingtime	minpl
0	12333	boardgame	Twilight Struggle	2005.0	2.0	2.0	180.0	
1	120677	boardgame	Terra Mystica	2012.0	2.0	5.0	150.0	
2	102794	boardgame	Caverna: The Cave Farmers	2013.0	1.0	7.0	210.0	
3	25613	boardgame	Through the Ages: A Story of Civilization	2006.0	2.0	4.0	240.0	
4	3076	boardgame	Puerto Rico	2002.0	2.0	5.0	150.0	

pair plot

- sns.pairplot(df)

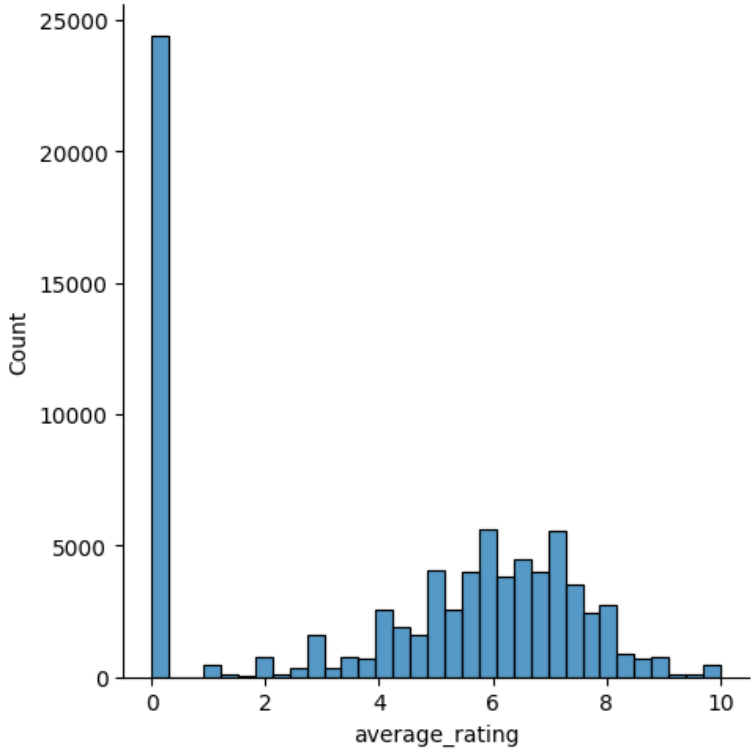
```
# sns.pairplot(data)
```

distribution plot

- sns.displot(df['col'])

```
sns.displot(data['average_rating'])
# displot - distribution plot
```

```
c:\users\surya\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
  self._figure.tight_layout(*args, **kwargs)
<seaborn.axisgrid.FacetGrid at 0x1f8c31a9eb0>
```

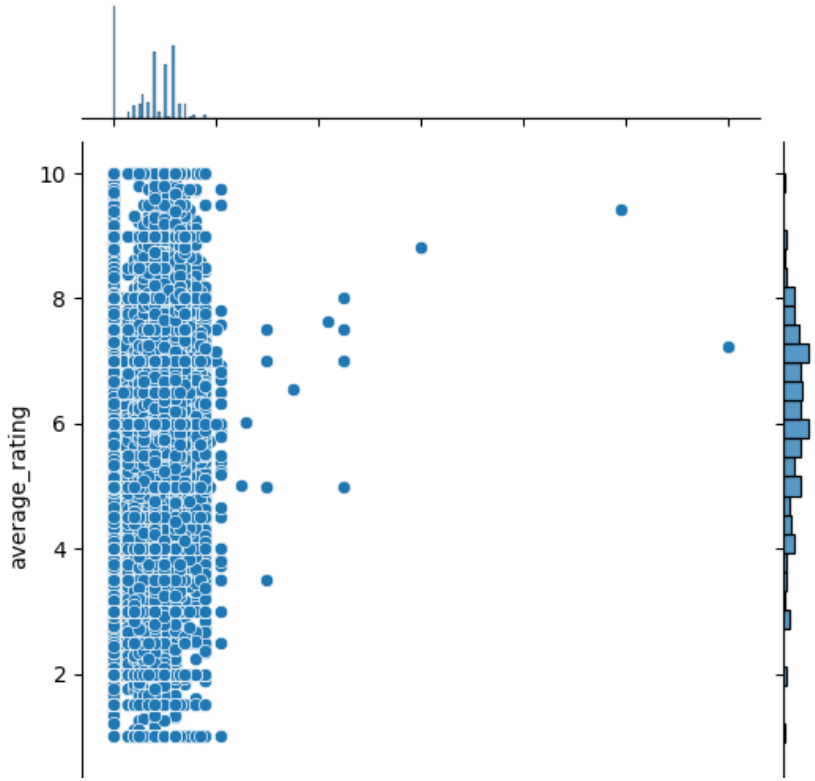


jointplot

- sns.jointplot(x=df['col1'], y = df['col2'])
- Draw a plot of two variables with bivariate and univariate graphs.

```
sns.jointplot(x = data['minage'], y = data['average_rating'])
# shows zero age with zero rating which is wrong logic
```

```
<seaborn.axisgrid.JointGrid at 0x1f8c6f15f10>
```

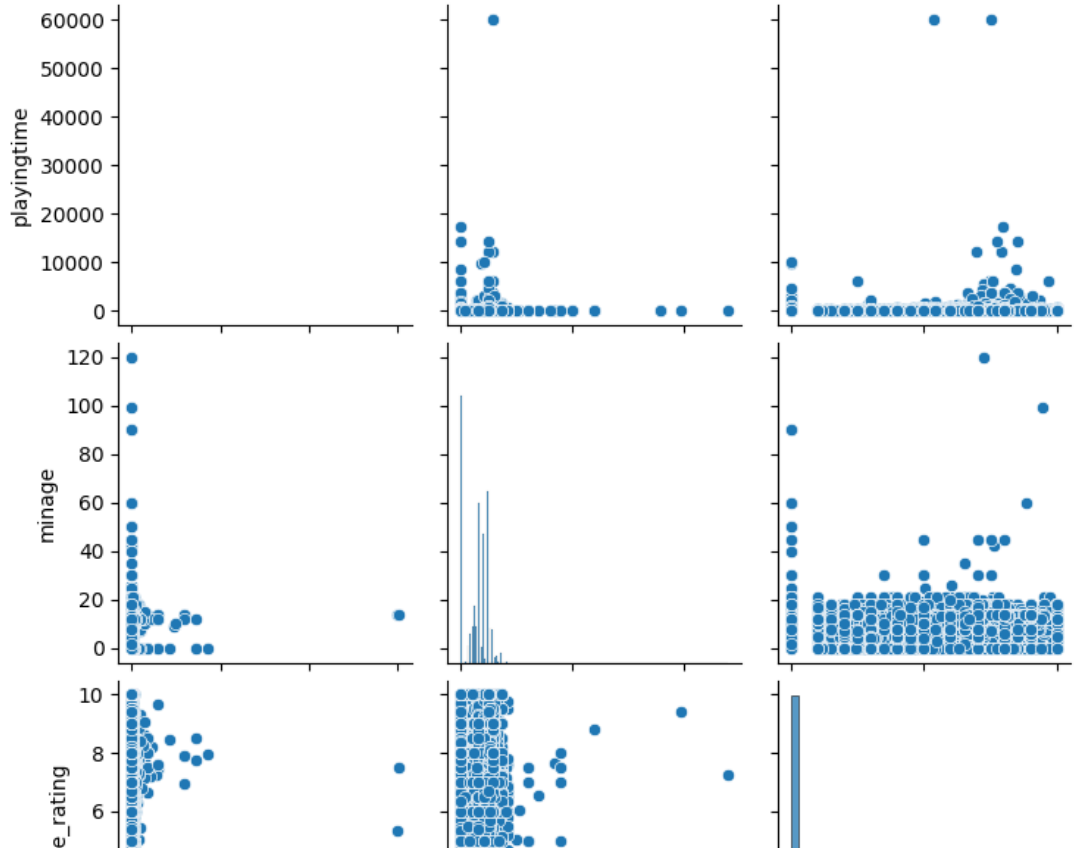


pair plot

- sns.pairplot(df)
- sns.pairplot(df[['col1', 'col2', 'col3']])

```
1 sns.pairplot(data[['playingtime', 'minage', 'average_rating']])
```

```
c:\users\surya\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
  self._figure.tight_layout(*args, **kwargs)
<seaborn.axisgrid.PairGrid at 0x1f8a4cce190>
```

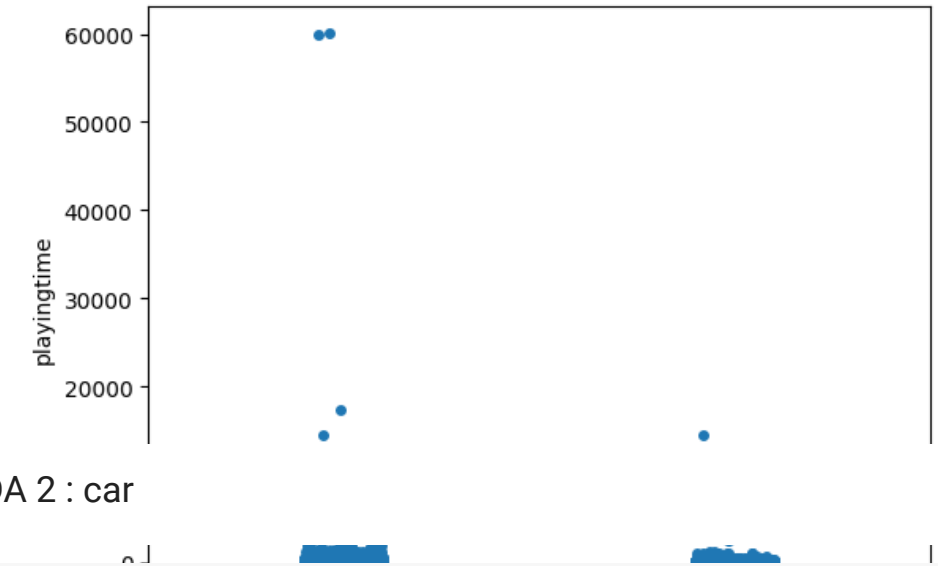


strip plot

- sns.stripplot(x = df['col1'], y = df['col2'], jitter=True)

```
1 sns.stripplot(x = data['type'], y = data['playingtime'], jitter=True)
```

```
<Axes: xlabel='type', ylabel='playingtime'>
```



EDA 2 : car

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# from google.colab import files
# uploaded = files.upload()
# D3data3.csv

import os
os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
os.getcwd()
```

```
'C:\\Users\\surya\\Downloads\\PG-DBDA-Mar23\\Datasets'
```

```
data = pd.read_csv('D3data3.csv')
```

```
data.shape
```

```
(201, 26)
```

```
data.describe()
```

	symboling	normalized_losses	wheel_base	length	width	height	curb_w
count	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.0
mean	0.840796	125.189055	98.797015	174.200995	65.889055	53.766667	2555.6
std	1.254802	33.572966	6.066366	12.322175	2.101471	2.447822	517.2
min	-2.000000	65.000000	86.600000	141.100000	60.300000	47.800000	1488.0
25%	0.000000	101.000000	94.500000	166.800000	64.100000	52.000000	2169.0
50%	1.000000	122.000000	97.000000	173.200000	65.500000	54.100000	2414.0
75%	2.000000	150.000000	102.400000	183.500000	66.600000	55.500000	2926.0
max	3.000000	256.000000	120.900000	208.100000	72.000000	59.800000	4066.0

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   symboling            201 non-null    int64  
 1   normalized_losses    201 non-null    int64  
 2   make                 201 non-null    object 
 3   fuel_type            201 non-null    object 
 4   aspiration           201 non-null    object 
 5   number_of_doors      201 non-null    object 
 6   body_style           201 non-null    object 
 7   drive_wheels         201 non-null    object 
 8   engine_location      201 non-null    object 
 9   wheel_base           201 non-null    float64
 10  length               201 non-null    float64
 11  width                201 non-null    float64
 12  height               201 non-null    float64
 13  curb_weight          201 non-null    int64  
 14  engine_type          201 non-null    object 
 15  number_of_cylinders  201 non-null    object 
 16  engine_size          201 non-null    int64  
 17  fuel_system          201 non-null    object 
 18  bore                 201 non-null    float64
 19  stroke               201 non-null    float64
 20  compression_ratio    201 non-null    float64
 21  horsepower           201 non-null    int64  
 22  peak_rpm             201 non-null    int64  
 23  city_mpg             201 non-null    int64  
 24  highway_mpg          201 non-null    int64  
 25  price                201 non-null    int64  
```

```
dtypes: float64(7), int64(9), object(10)
memory usage: 41.0+ KB
```

```
1 data.head()
```

	symboling	normalized_losses	make	fuel_type	aspiration	number_of_doors	body_style
0	3	168	alfa-romero	gas	std	two	convertible
1	3	168	alfa-romero	gas	std	two	convertible
2	1	168	alfa-romero	gas	std	two	hatchback
3	2	164	audi	gas	std	four	sedan
4	2	164	audi	gas	std	four	sedan

5 rows × 26 columns

null check

```
1 data.isnull().sum()
```

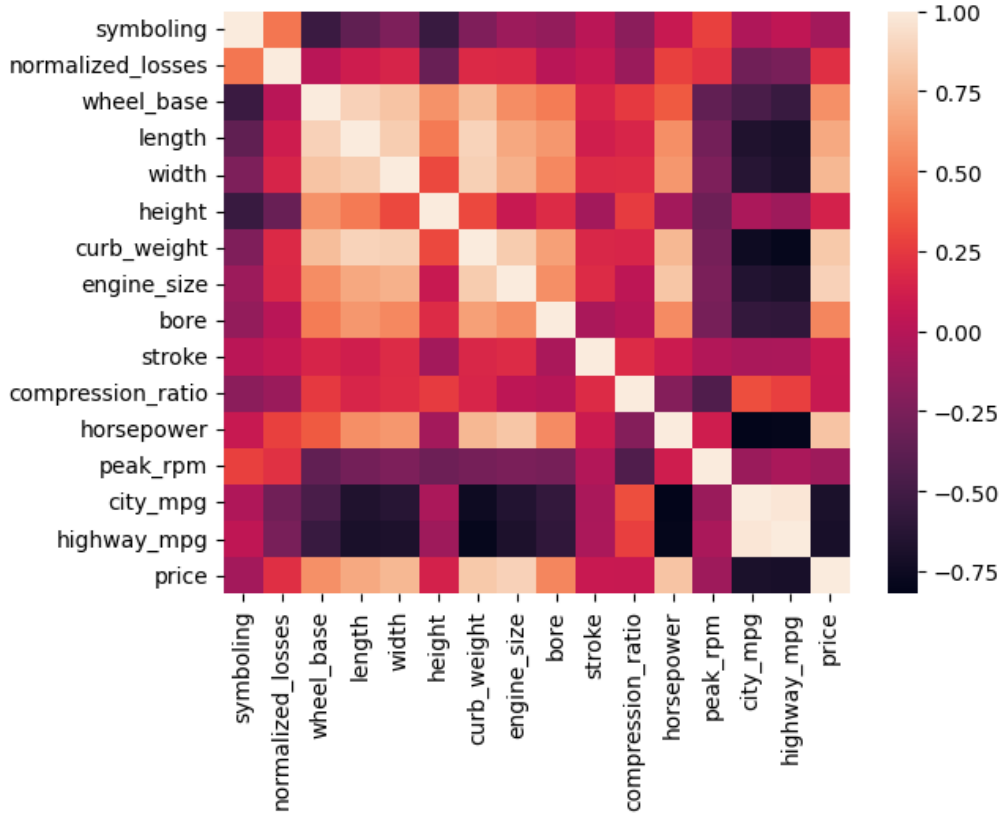
```
symboling             0
normalized_losses     0
make                  0
fuel_type             0
aspiration            0
number_of_doors       0
body_style            0
drive_wheels          0
engine_location       0
wheel_base            0
length                0
width                 0
height                0
curb_weight           0
engine_type           0
number_of_cylinders   0
engine_size           0
fuel_system           0
bore                  0
stroke                0
compression_ratio     0
```

```
horsepower          0
peak_rpm            0
city_mpg            0
highway_mpg         0
price               0
dtype: int64
```

df.corr()

```
1 sns.heatmap(data.iloc[ : , [0, 1, 9, 10, 11, 12, 13, 16, 18, 19, 20, 21, 22, 23, 24, 25]].corr())
```

```
<Axes: >
```

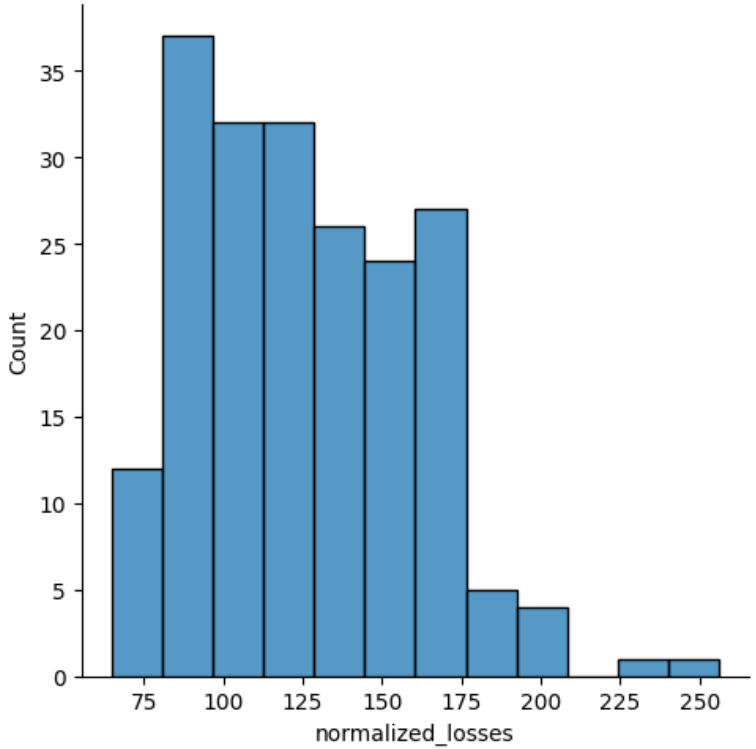


displot

- sns.displot(df['col1'])

```
plt.figure(figsize=(18, 7))
sns.displot(data['normalized_losses'])
plt.show()
```

```
c:\users\surya\appdata\local\programs\python\python39\lib\site-packages\seaborn\axisgrid.p
  self._figure.tight_layout(*args, **kwargs)
<Figure size 1800x700 with 0 Axes>
```

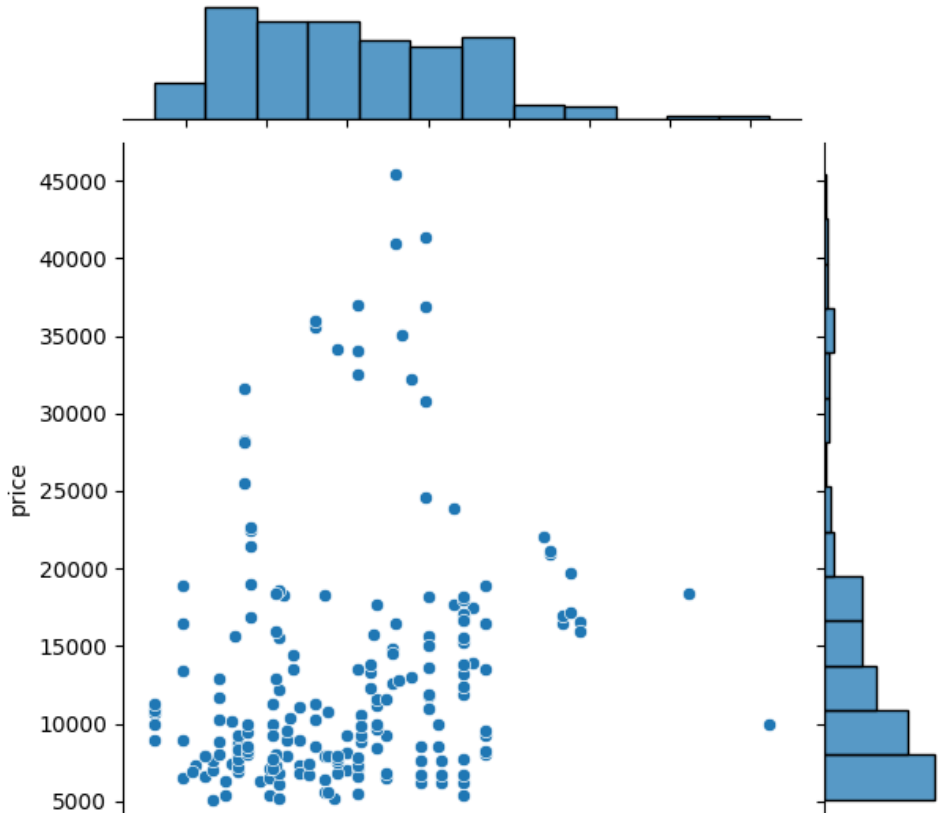


jointplot

- sns.jointplot(x=df['col1'], y=df['col2'])

```
plt.figure(figsize=(18, 7))
sns.jointplot(x=data['normalized_losses'], y=data['price'])
plt.show()
```

```
<Figure size 1800x700 with 0 Axes>
```

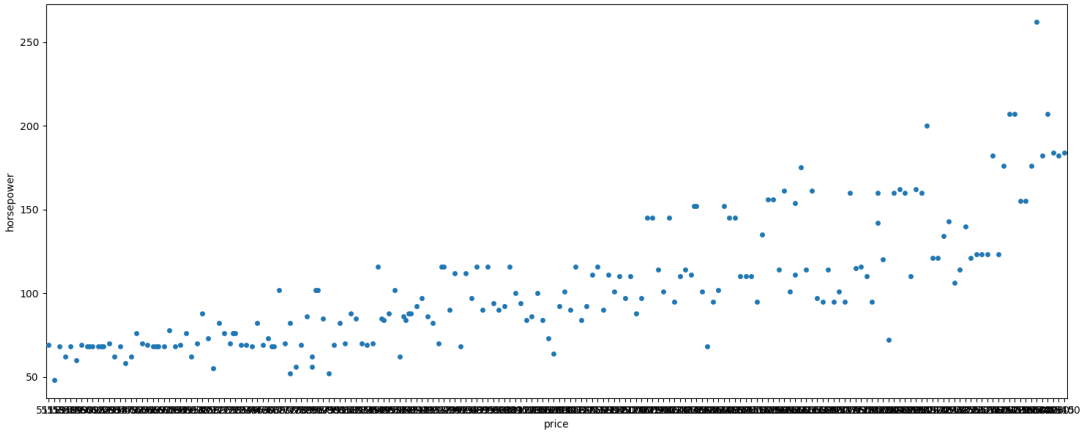


swarmplot

- sns.swarmplot(x=df['col1'], y=df['col2'])
- Draw a categorical scatterplot with points adjusted to be non-overlapping.

```
plt.figure(figsize=(18, 7))
sns.swarmplot(x=data['price'], y=data['horsepower'])
plt.show()
```

```
c:\users\surya\appdata\local\programs\python\python39\lib\site-packages\seaborn\categorica
  warnings.warn(msg, UserWarning)
```




```
1 sns.boxplot(data[['wheel_base', 'engine_size']])
```

```
<Axes: >
```

HW

1. Perform Data Preprocessing and EDA on D3data4.csv and D3data5.csv datasets.

```
1
```