Kmit

SONET

Data Science

(Level-I)

Ajeet K. Jain, M. Narsimlu

(ML TEAM)- SONET, KMIT, Hyderabad

Session - 26

This session deals with

Seaborn Library

Creating various graphs using seaborn

Exercises on Seaborn

Data Preprocessing

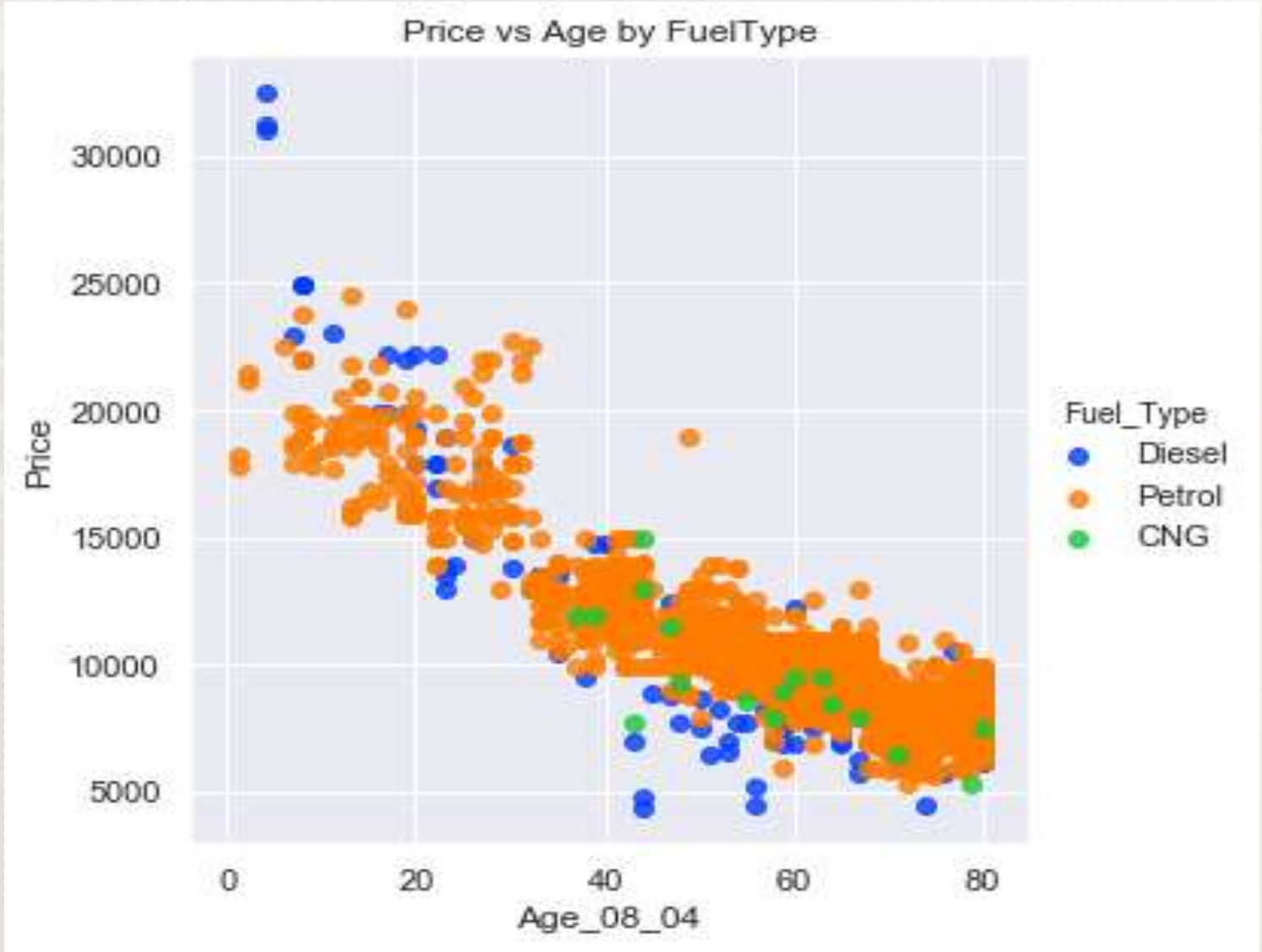
Exercise-4

- Read a Toyota Cars dataset and perform following tasks:
- 1.Set grid background color as darkgrid
- 2.Create a Scatter plot of *Price vs Age by FuelType*
- use Implot() function to create scatter plot
- Implot() methods will take feature attributes and dataset
- 3.Add hue parameter which includes another variable categories(“Fuel_Type” variable) with different colors

Solution

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data_cars=pd.read_csv("ToyotaCorolla.csv")
sns.set(style="darkgrid")
g=sns.lmplot(x="Age_08_04",y="Price",data=data_cars,fit_reg=False,
             hue="Fuel_Type",legend=True,palette="bright")
#bright- deep,muted,pastel,bright,dark,Set1 and colorblind
ax=plt.gca()
ax.set_title("Price vs Age by FuelType")
```

Output

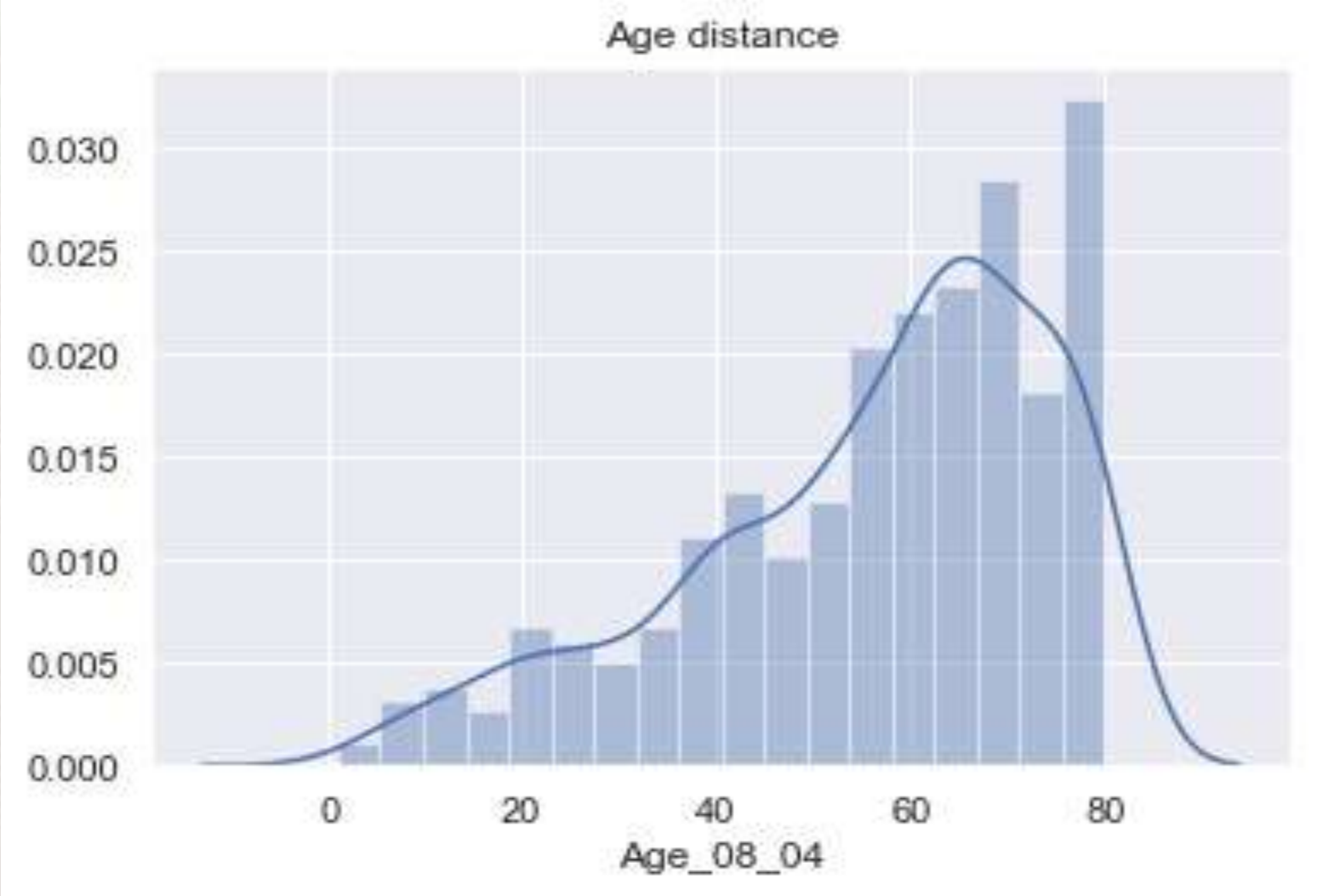

Price vs Age by FuelType
Price
30000
25000
20000
15000
10000
5000
0
20
40
60
80
Age_08_04
Fuel_Type
Diesel
Petrol
CNG

Histogram Exercise-5

- Create a Histogram with default kernel density estimate of Age feature in Toyota cars dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_cars=pd.read_csv("ToyotaCorolla.csv")
sns.distplot(data_cars["Age_08_04"])
g=plt.gca()
g.set_title("Age distance")
```

Output

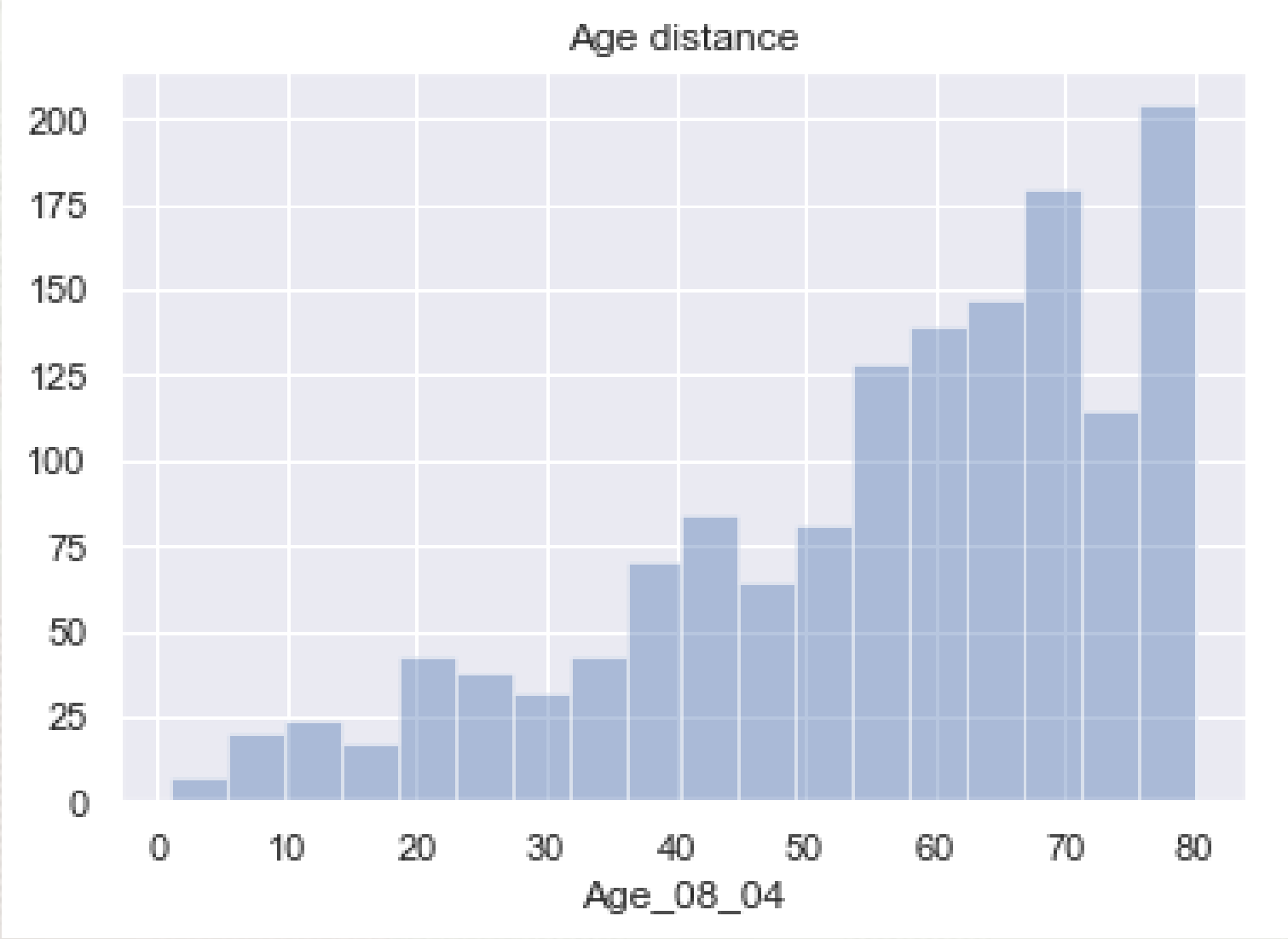

Age distance
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
Age_08_04

Exercise-6

- Create a Histogram without kernel density estimate of Age feature in Toyota cars dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_cars=pd.read_csv("ToyotaCorolla.csv")
sns.distplot(data_cars["Age_08_04"],kde=False)
g=plt.gca()
g.set_title("Age distance")
```

Output

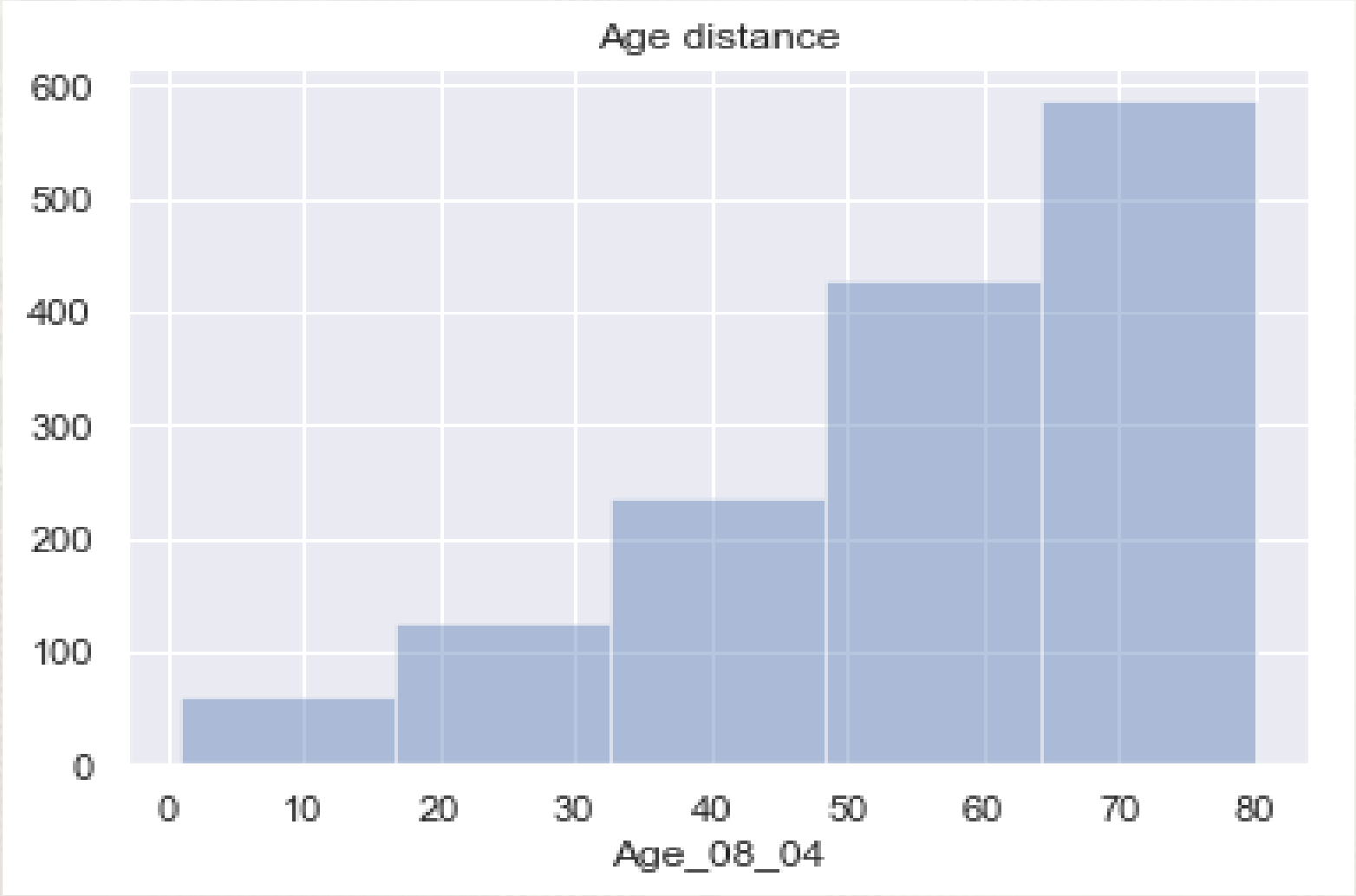

Age distance
200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
Age_08_04

Exercise-7

- Create a Histogram with fixed no. of bins without kernel density estimate of Age feature in Toyota cars dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_cars=pd.read_csv("ToyotaCorolla.csv")
sns.distplot(data_cars["Age_08_04"],kde=False,bins=5)
g=plt.gca()
g.set_title("Age distance")
```

Output

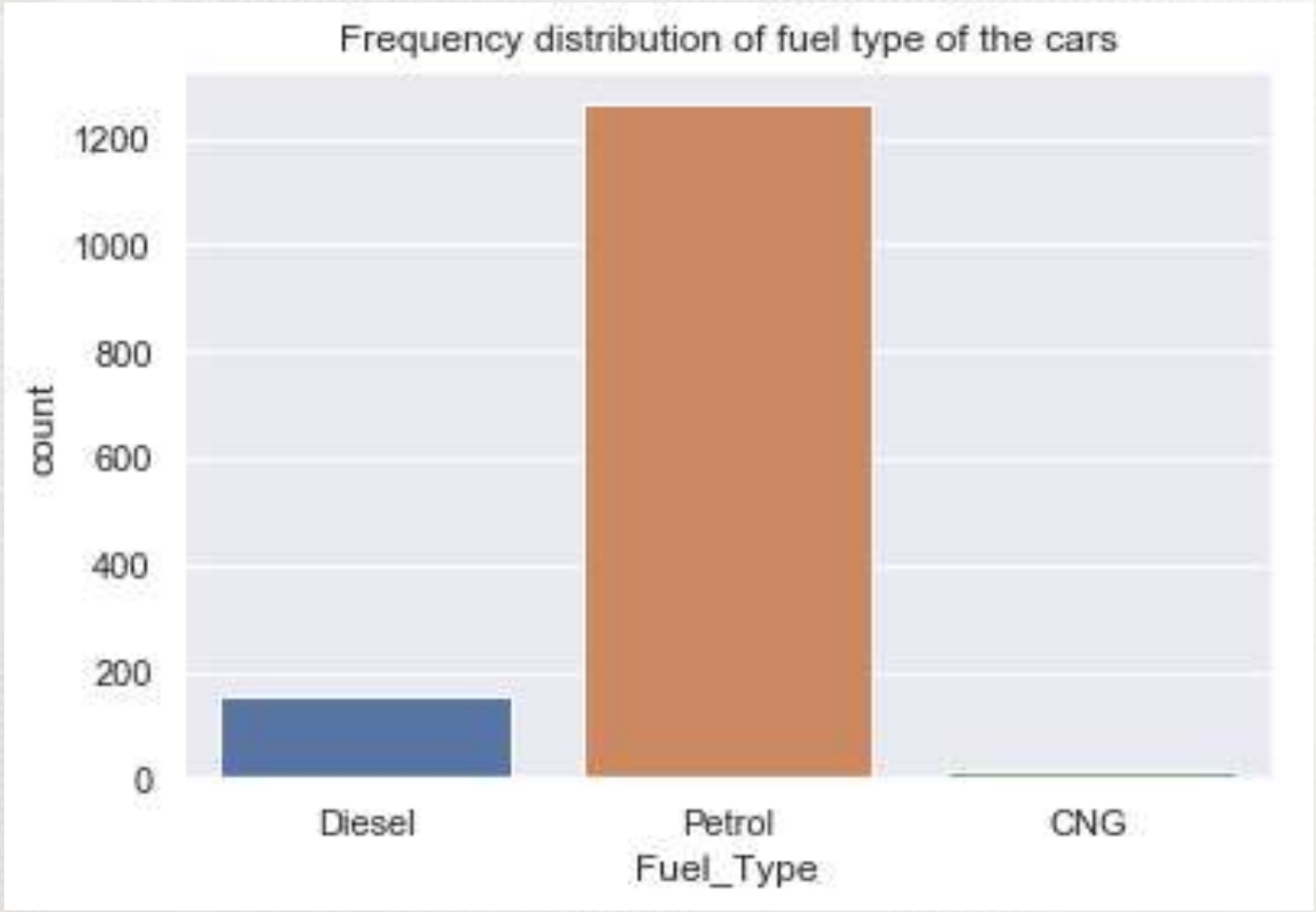

Age distance
600
500
400
300
200
100
0
0
10
20
30
40
50
60
70
80
Age_08_04

Bar plot Ex-8

- Read the Toyota cars dataset and perform the following tasks:
- Create count plot on "Fuel _Type" which says Frequency distribution of fuel type of the cars

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_cars=pd.read_csv("ToyotaCorolla.csv")
sns.countplot(x="Fuel_Type",data=data_cars)
g=plt.gca()
g.set_title("Frequency distribution of fuel type of the cars")
```

Output


Frequency distribution of fuel type of the cars
1200
1000
800
600
400
200
0
ount
Diesel
Petrol
CNG
Fuel_Type

Grouped Bar plot

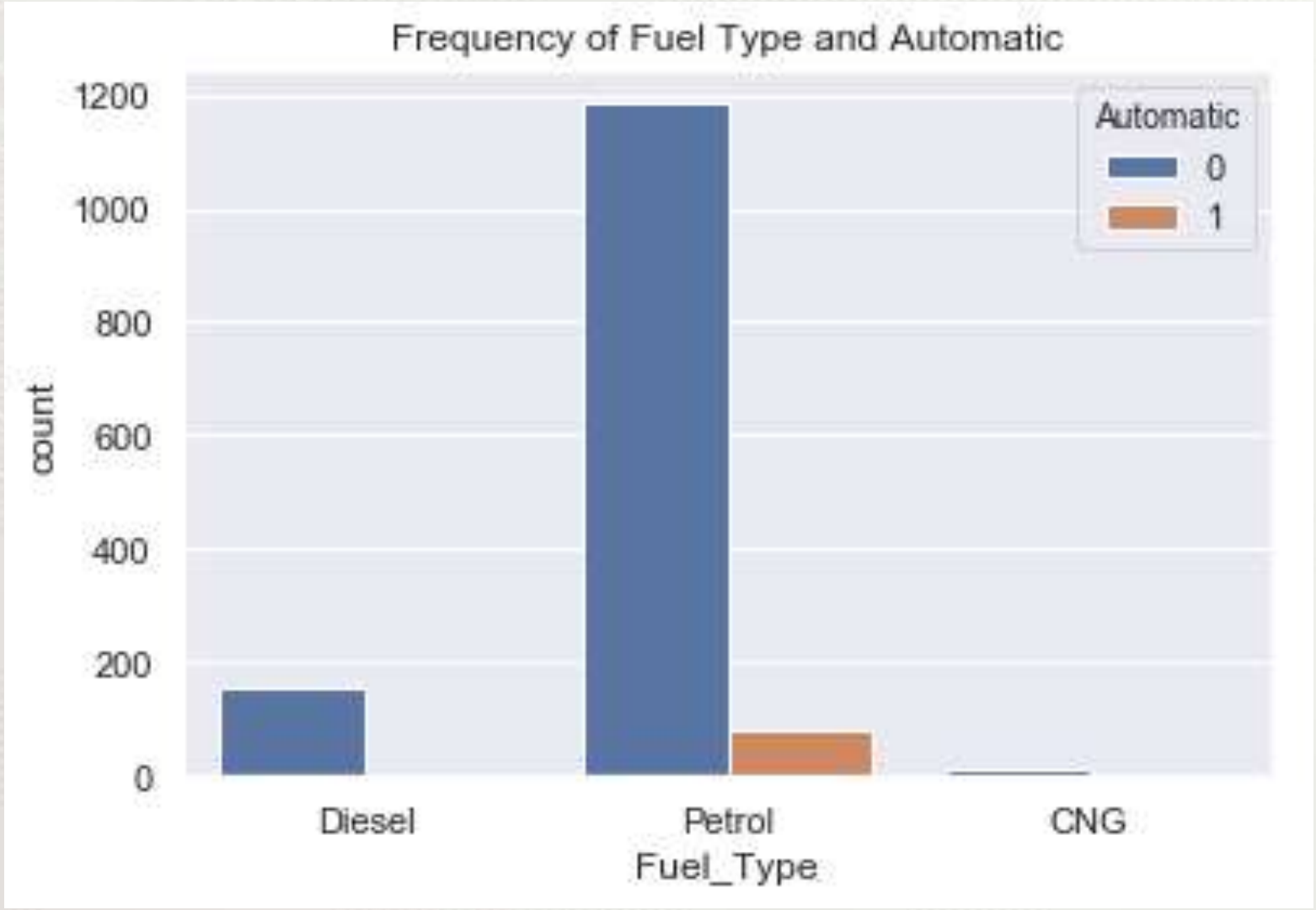
Read a Toyota cars dataset and perform the following tasks

1.Create a Two way table on "Fuel_Type“ and Automatic feature
2.Create a count plot by grouping Automatic by Fuel Type

Solution

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_cars=pd.read_csv("ToyotaCorolla.csv")
auto_tab=pd.crosstab(index=data_cars["Automatic"],
                     columns=data_cars["Fuel_Type"],dropna=True)
sns.countplot(x="Fuel_Type",data=data_cars,hue="Automatic")
g=plt.gca()
g.set_title("Frequency of Fuel Type and Automatic")
```

Output


Frequency of Fuel Type and Automatic
1200
1000
800
600
400
200
0
count
Automatic
0
1
Diesel
Petrol
CNG
Fuel_Type

Box and Whiskers plot – numerical variable

- Box and whiskers plot of *Price* to visually interpret the five-number summary

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_cars=pd.read_csv("ToyotaCorolla.csv")
sns.boxplot(y=data_cars["Price"])
g=plt.gca()
g.set_title("Outliers in Price Feature")
```

Output

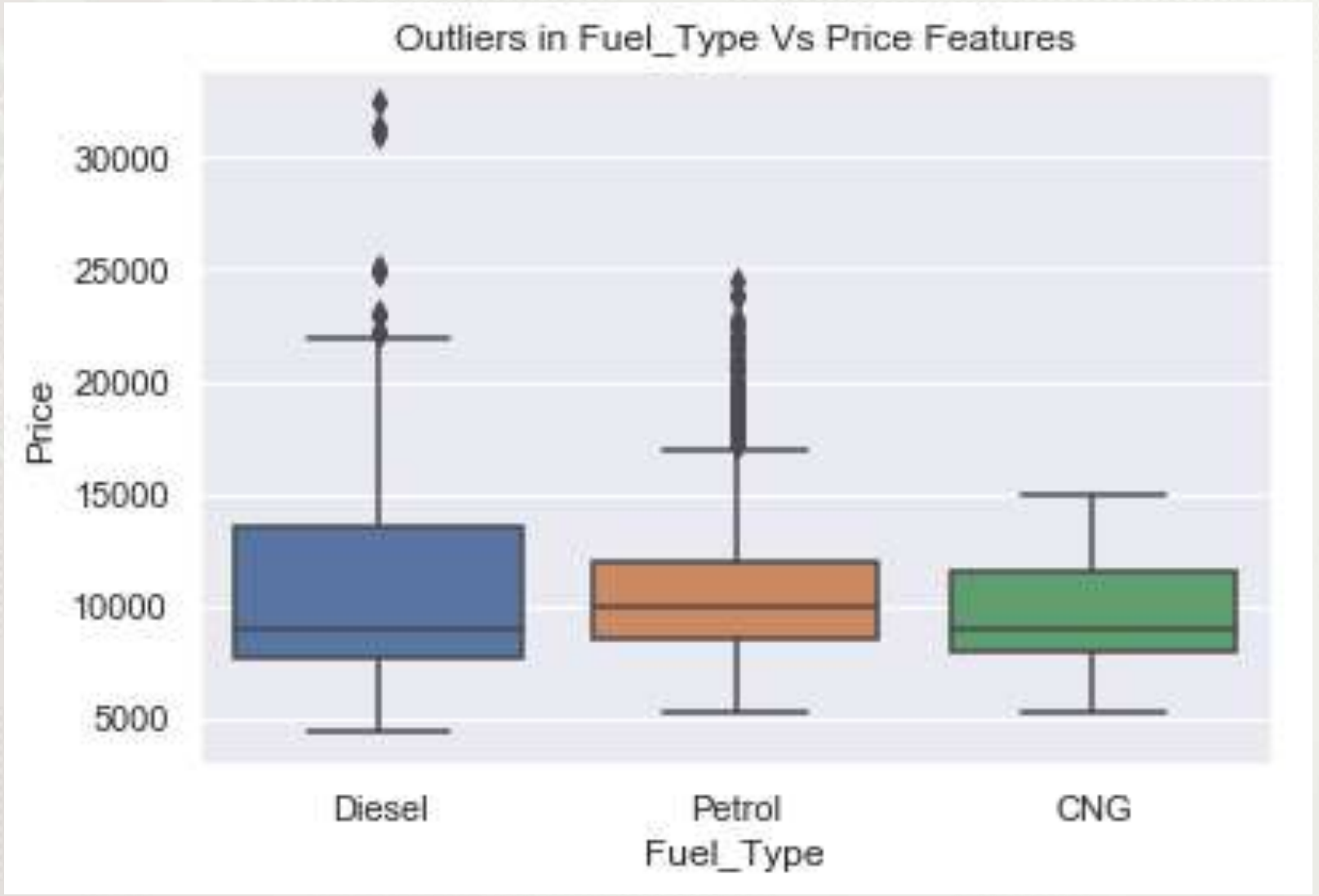

Outliers in Price Feature
Price
30000
25000
20000
15000
10000
5000

Box and Whiskers plot

- Box and whiskers plot for numerical vs categorical variable
- Price of the cars for various fuel types

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_cars=pd.read_csv("ToyotaCorolla.csv")
sns.boxplot(x=data_cars["Fuel_Type"],y=data_cars["Price"])
g=plt.gca()
g.set_title("Outliers in Fuel_Type Vs Price Features")
```

Output

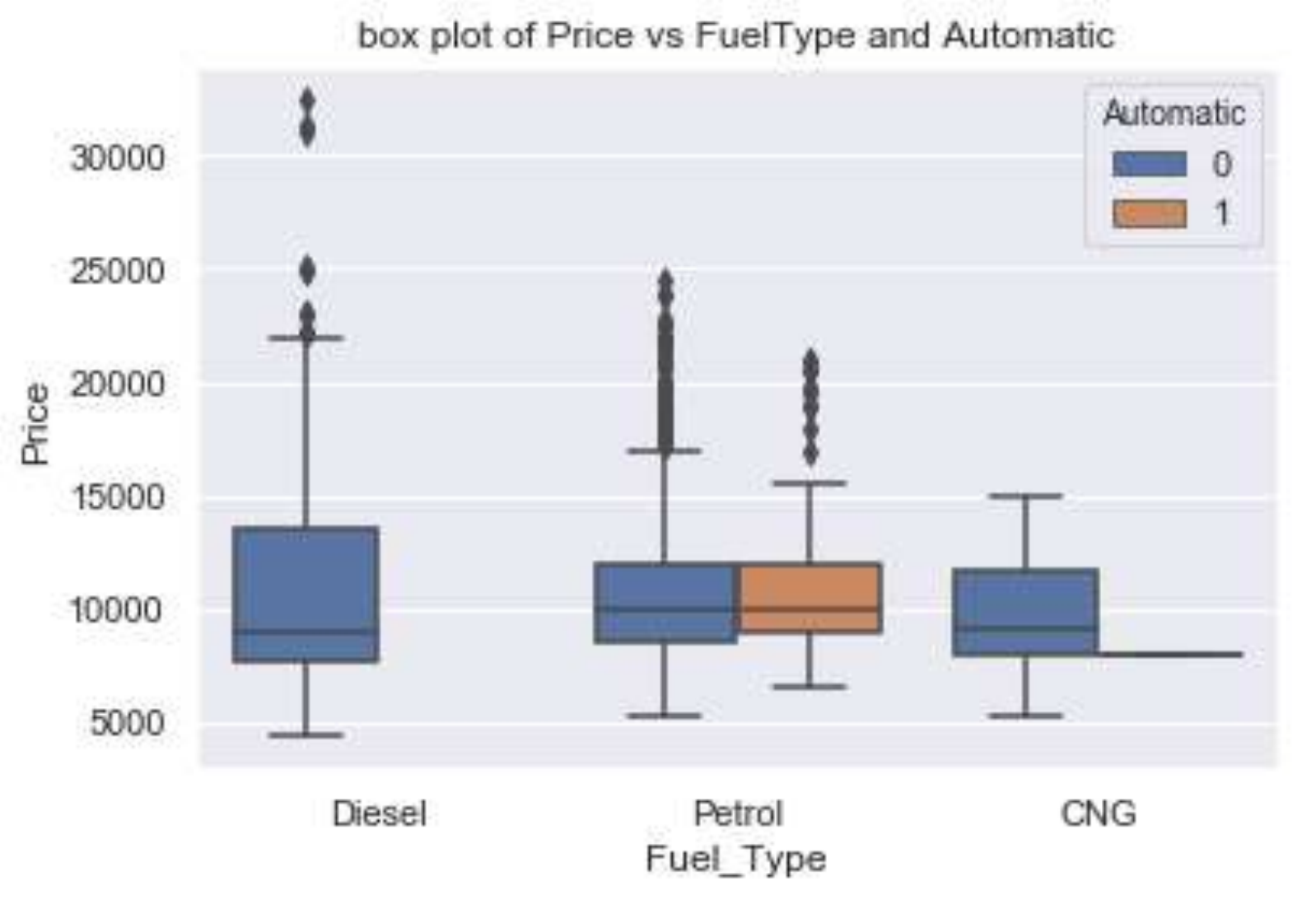

Outliers in Fuel_Type Vs Price Features
Price
30000
25000
20000
15000
10000
5000
Diesel
Petrol
CNG
Fuel_Type

Grouped box and whiskers plot

- Grouped box and whiskers plot of *Price* vs *FuelType* and *Automatic*

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_cars=pd.read_csv("ToyotaCorolla.csv")
sns.boxplot(x=data_cars["Fuel_Type"],y=data_cars["Price"],
            data=data_cars,hue="Automatic")
g=plt.gca()
g.set_title("box plot of Price vs FuelType and Automatic")
```

Output

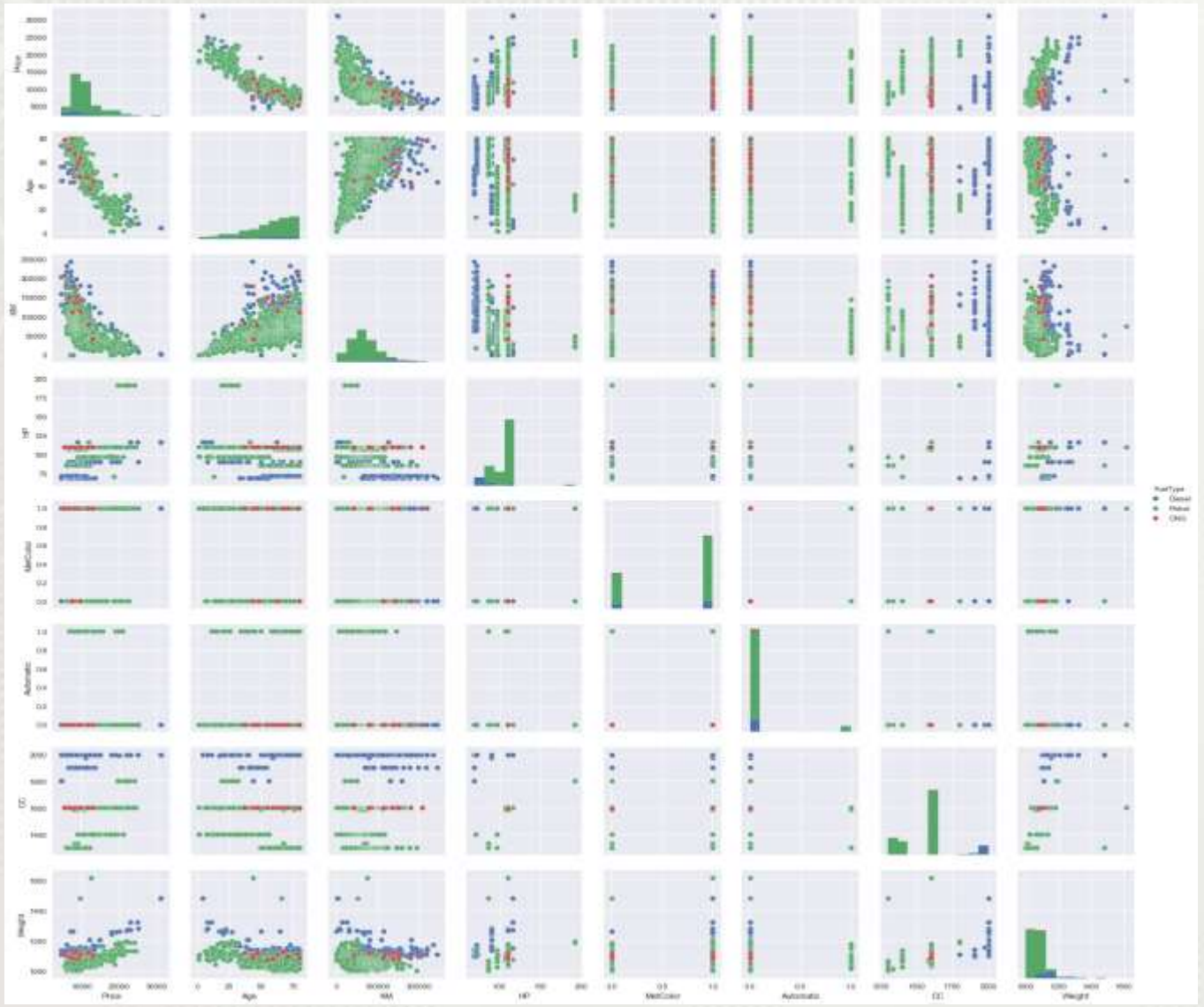

box plot of Price vs FuelType and Automatic
Automatic
0
1
30000
25000
20000
15000
10000
5000
Price
Diesel
Petrol
CNG
Fuel_Type

Pairwise plots

- It is used to plot pairwise relationships in a dataset
- Creates scatterplots for joint relationships and histograms for univariate distributions

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data_cars=pd.read_csv("ToyotaCorolla.csv")
sns.pairplot(data_cars, kind="scatter", hue="Fuel_Type")
g=plt.gca()
g.set_title("Pairwise plot")
```

Pairwise plots



Output:

Introduction

Why preprocessing ?

Real world data are generally

Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data

Noisy: containing errors or outliers

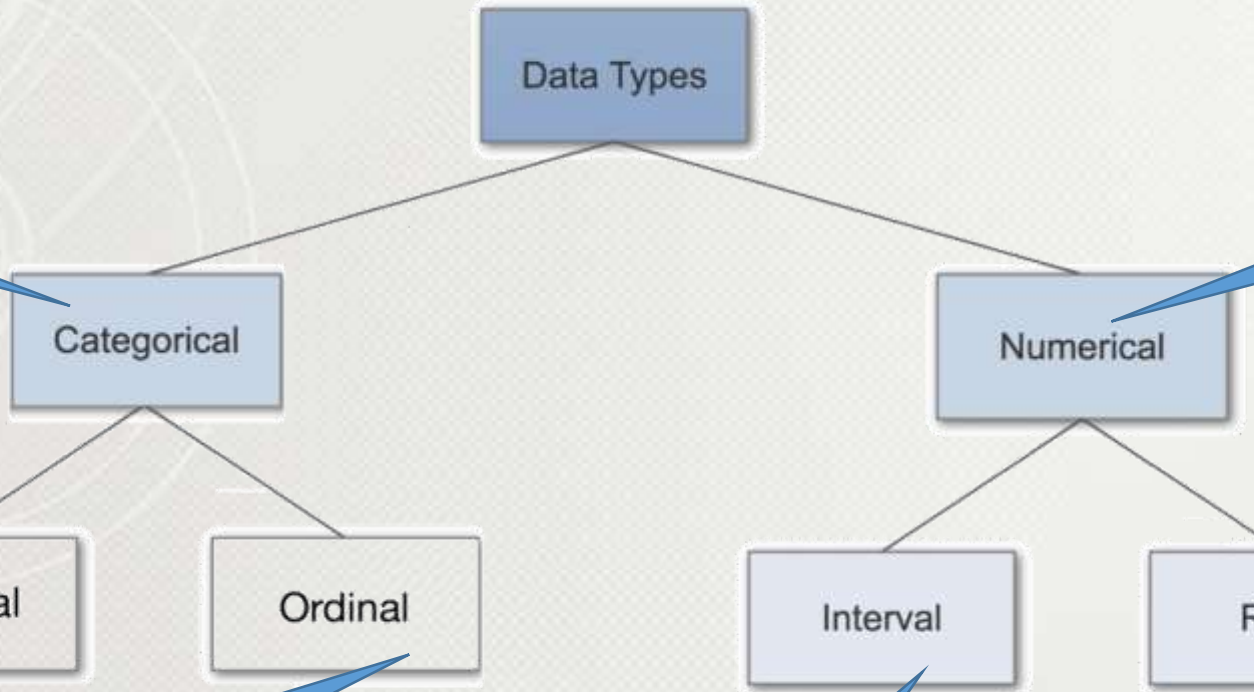
Inconsistent: containing discrepancies in codes or names Tasks in data preprocessing

Introduction

- Data pre-processing is an important step of solving every machine learning problem.
- Most of the datasets used with Machine Learning problems need to be processed / cleaned / transformed so that a Machine Learning algorithm can be trained on it.
- Most commonly used pre-processing techniques are very few like - missing value imputation, encoding categorical variables, scaling, etc.

Types of Data

It represents characteristics
Ex: a person's gender, language… etc.



It represents numerical values
Ex: a person's height, weight.. etc.

Nominal scales are like "names" or labels
Ex: Gender: Male or Female

It measures of non-numeric concepts like satisfaction, happiness, discomfort, etc.

It represent ordered units that have the same difference
Ex: Temperature

These are the same as interval values, with the difference that they do have an absolute zero
Ex:height,weight

Introduction

- It involves in every machine learning problem.
- Most of the datasets need to be processed / cleaned / transformed
- Pre-processing techniques - missing value imputation, encoding categorical variables, scaling, etc.

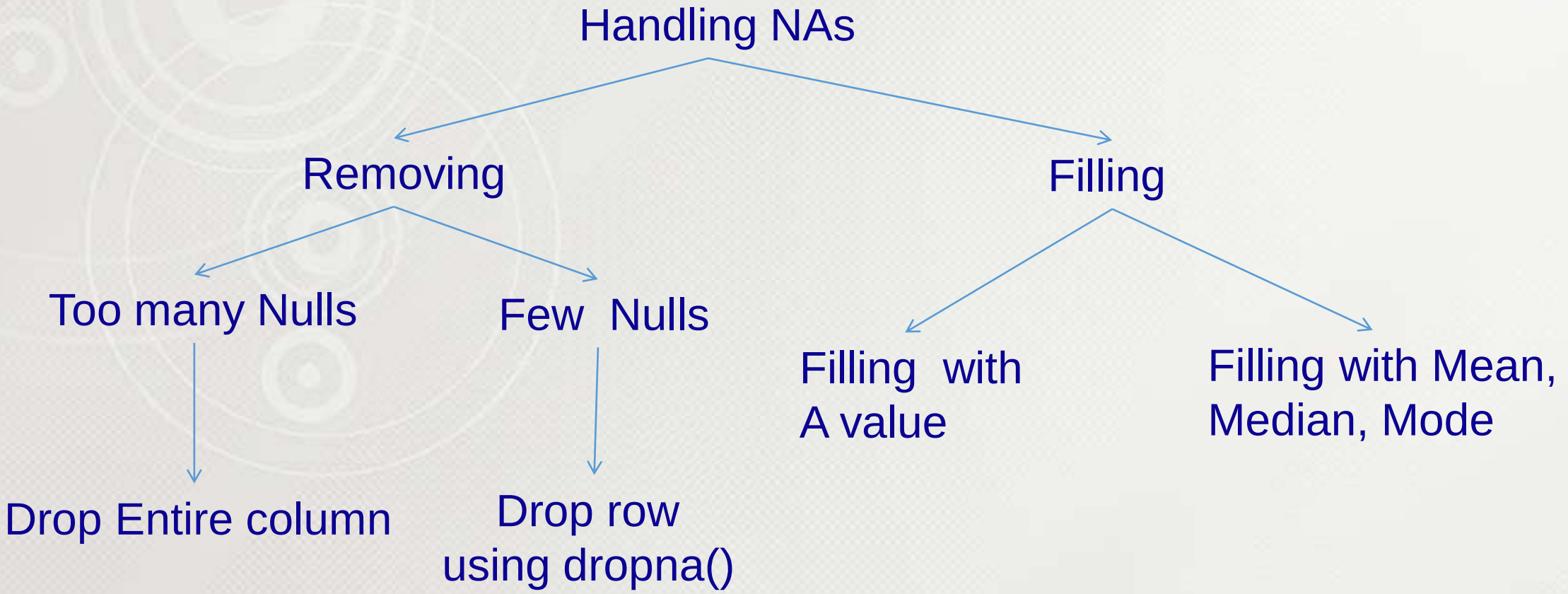
Handling Numeric data

Handling NAs

Outliers

Scaling / Normalization

Handling Numeric data



Handling Numerical data

1. Load Data Set and find the missing values

```
import pandas as pd
import matplotlib.pyplot as plt
data_loan=pd.read_csv("loan.csv")
print(data_loan.isnull().sum())
```

Output


```
Loan ID                                0
Customer ID                            0
Loan Status                            0
Current Loan Amount                    0
Term                                   0
Credit Score                       19154
Annual Income                      19154
Years in current job                4222
Home Ownership                         0
Purpose                                0
Monthly Debt                           0
Years of Credit History                0
Months since last delinquent       53141
Number of Open Accounts                0
Number of Credit Problems              0
Current Credit Balance                 0
Maximum Open Credit                    2
Bankruptcies                         204
Tax Liens                             10
dtype: int64
```

Handling Numerical data

Dropping Entire column

```
import pandas as pd
import matplotlib.pyplot as plt
data_loan=pd.read_csv("loan.csv")
df_col=data_loan.drop(["Months since last delinquent"],axis=1)
print(df_col.head())
```

Handling Numerical data

Dropping all null values

```
import pandas as pd
import matplotlib.pyplot as plt
data_loan=pd.read_csv("loan.csv")
data_loan.dropna(inplace=True)
print(data_loan.isnull().sum())
```

Handling Numerical data

2. Check is there any null values

3. fillna() method –filling null values with '1'

```
import pandas as pd
import matplotlib.pyplot as plt
data_loan=pd.read_csv("loan.csv")
print(data_loan.isnull().sum())
data_loan.fillna(1,inplace=True)
print(data_loan.isnull().sum())
```

Handling Numerical data

```
In [32]: df=data.fillna(data.mean())
         print(df.head())
```

	site	totabund	density	meandepth	year	sweptarea
0	1.000000	76	0.002070	804	1978	64992.287498
1	2.000000	161	0.003520	808	2001	45741.253910
2	3.000000	39	0.000981	809	2001	39775.000000
3	76.029197	410	0.008039	848	1979	51000.000000
4	5.000000	177	0.005933	853	2002	29831.251950

Filling missing data with mean

```
In [33]: df=data.fillna(data.mode())
         print(df.head())
```

	site	totabund	density	meandepth	year	sweptarea
0	1.0	76	0.002070	804	1978	59662.50391
1	2.0	161	0.003520	808	2001	45741.25391
2	3.0	39	0.000981	809	2001	39775.00000
3	5.0	410	0.008039	848	1979	51000.00000
4	5.0	177	0.005933	853	2002	29831.25195

Filling missing data with mode

Handling Numerical data

Handling Duplicated Values:

```
df1=df2.duplicated(['density'])
print(df1.sum())
```

```
12
```

Dropping Duplicated Values:

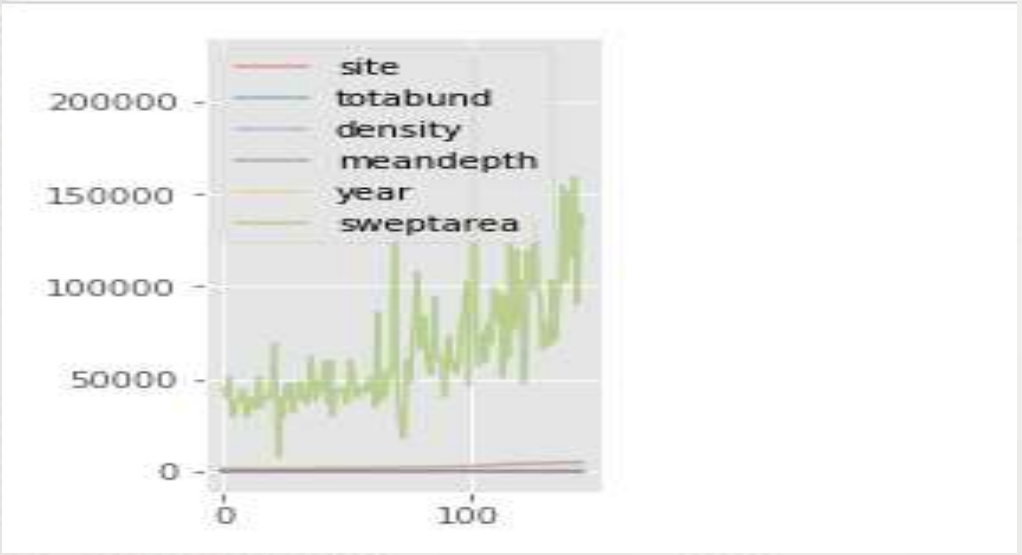
```
df2=df.drop_duplicates(['density'],keep="first")
df1=df2.duplicated(['density'])
print(df1.sum())
```

```
0
```

Handling Numerical data

Skewness : is a measure of the symmetry in a distribution. A symmetrical dataset will have a Skewness equal to 0. Skewness essentially measures the relative size of the two tails.

```
import matplotlib
import matplotlib.pyplot as plt
matplotlib.style.use('ggplot')


gf=df.skew()
df.plot(alpha=0.5, figsize=(2, 4))

plt.show()
```



Outliers

An *outlier* is an observation that lies an abnormal distance from other values in a random sample from a population.

To remove outliers we will use

- np.log() Natural logarithm, element-wise.
- np.sqrt() square root
- np.cbrt() cube root

Handling Outliers

Check the difference between minimum value and maximum value

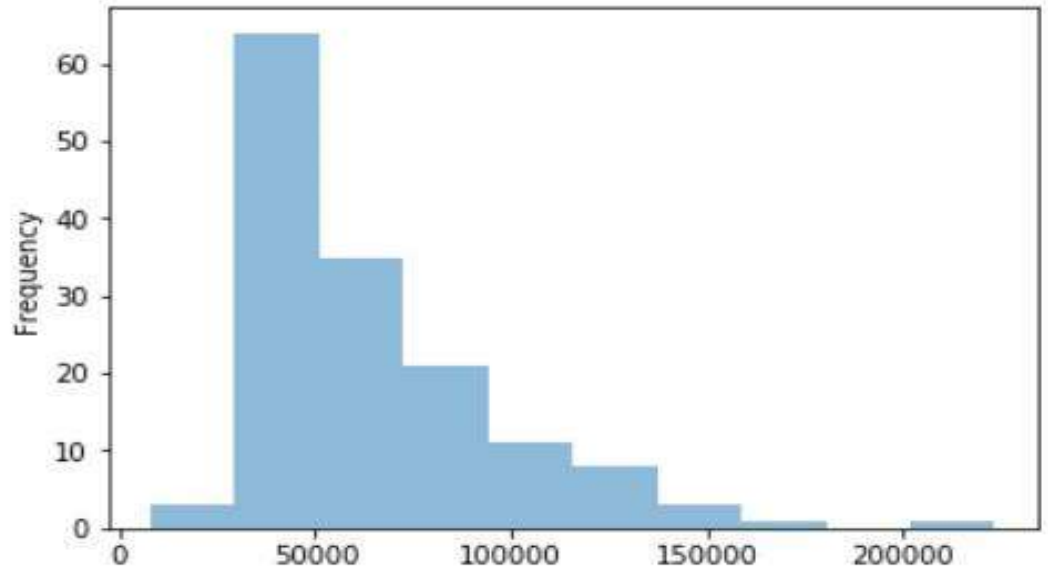
```
In [17]: df['sweptarea'].max()          # max = 223440.0
         df['sweptarea'].min()          # min = 7970.0
         df['sweptarea'].mean()         # mean = 64956.030466666685

Out[17]: 223440.0
```

Handling Outliers

Histogram for original ‘sweptarea’ data

```
In [20]: import matplotlib.pyplot as plt
         import pandas as pd
         data=pd.read_csv("C:\\Users\\Venkatesh\\Desktop\\Machine Learning\\datasets\\fishing.csv")
         dd=df['sweptarea']
         plt.figure()
         dd.plot.hist(alpha=0.5)
```

```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x1897017acc0>
```



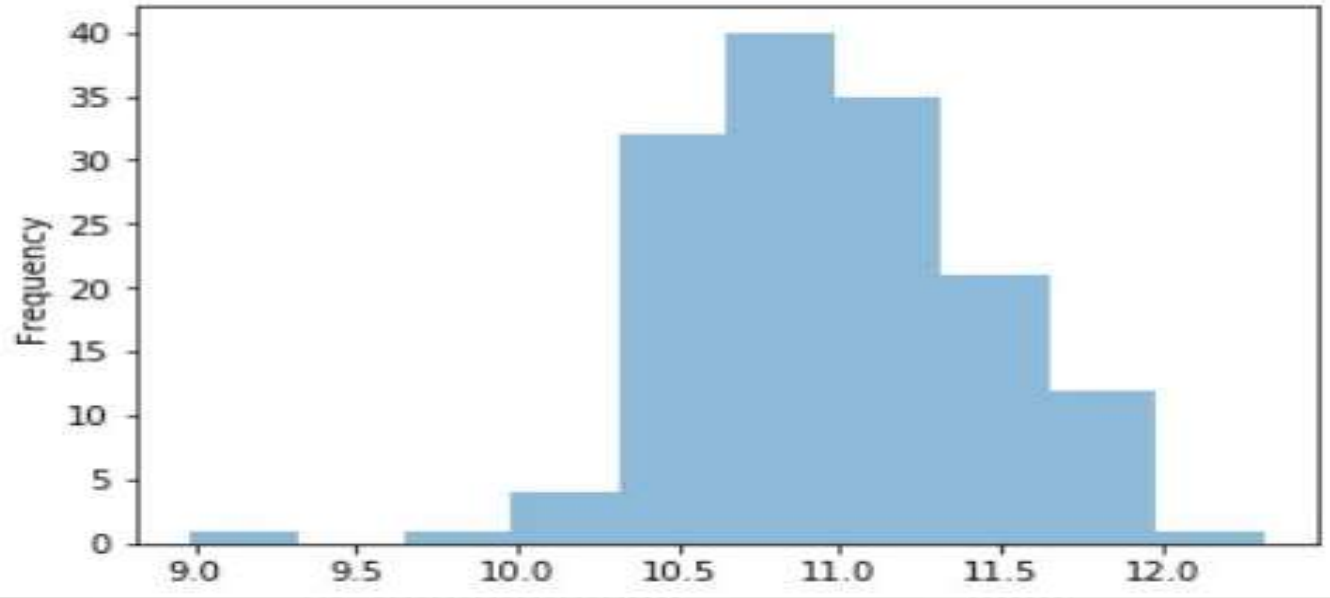
Handling Outliers

Handling outliers with numpy.log():

This mathematical function helps user to calculate **Natural logarithm of x** where x belongs to all the input elements.

```
import numpy as np
log=df['sweptarea_log']=np.log(df['sweptarea'])
plt.figure()
log.plot.hist(alpha=0.5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x189703705c0>
```

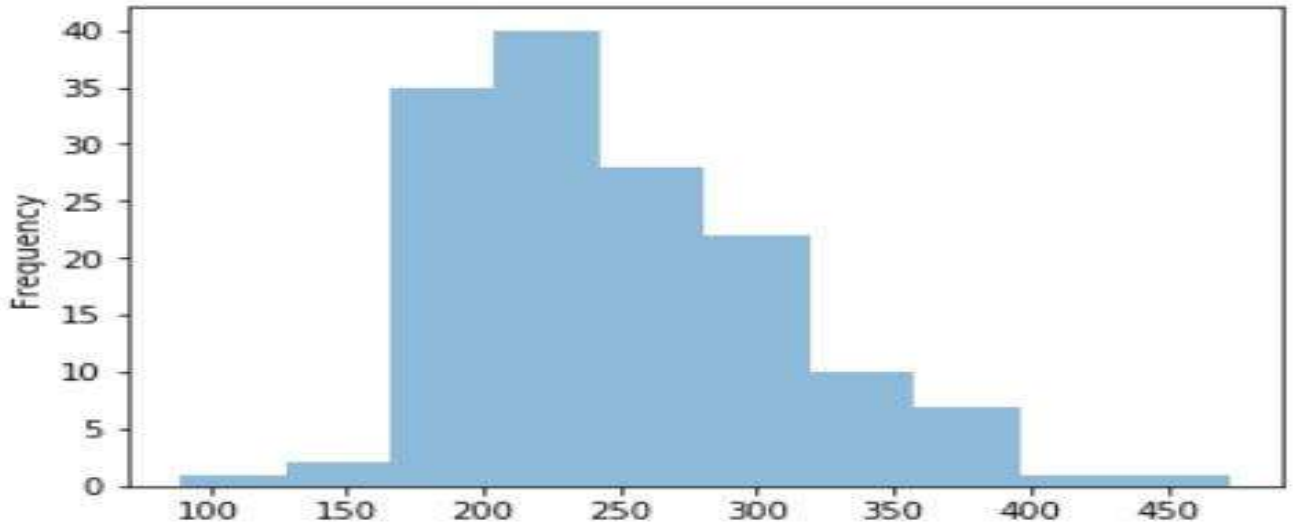


Handling Outliers

numpy.sqrt():
Return the positive square-root of an array(data), element-wise.

```
In [36]: sqrt=df['sweptarea_sqrt']=np.sqrt(df['sweptarea'])
         plt.figure()
         sqrt.plot.hist(alpha=0.5)
```

```
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x189702e9470>
```

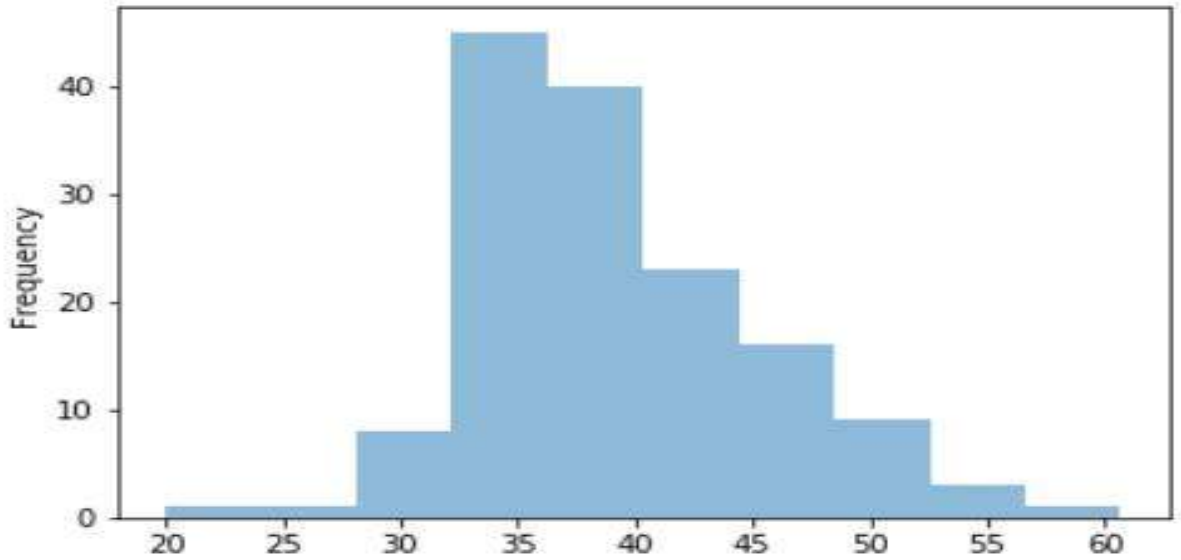


Handling Outliers

numpy.cbrt():
This mathematical function helps user to calculate cube root of x for all x being the array elements.

```
In [32]: cbrt=df['sweptarea_cbrt']=np.cbrt(df['sweptarea'])
         plt.figure()
         cbrt.plot.hist(alpha=0.5)
```

```
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x18971520dd8>
```

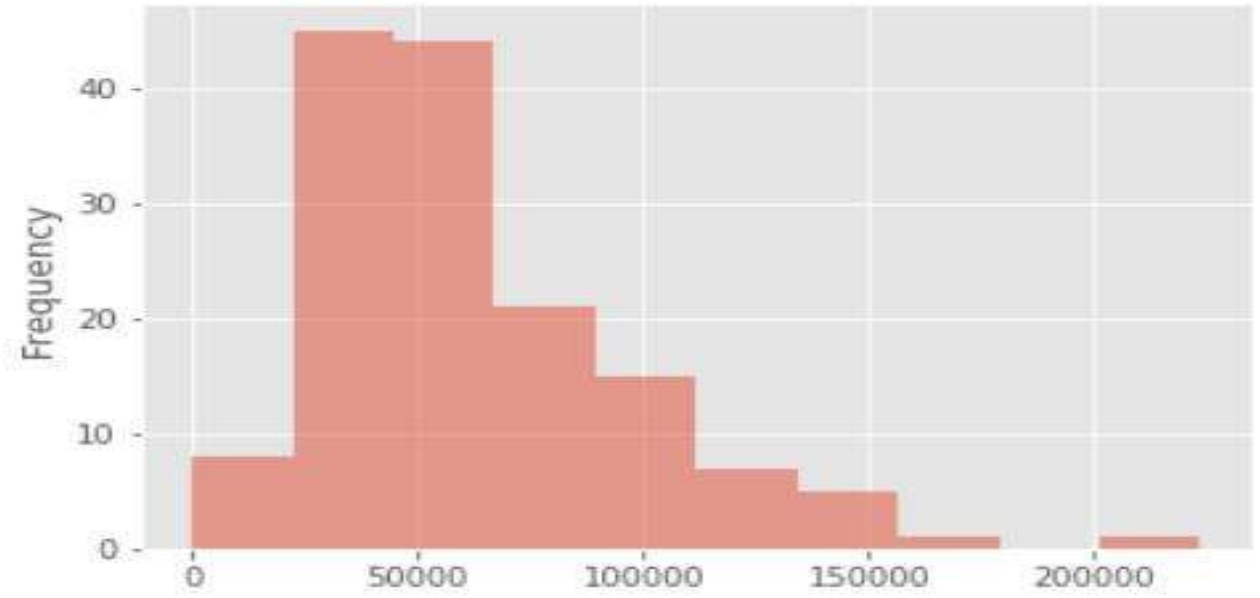


Handling Outliers

DATA SCIENCE

Using Percentile:

```
In [43]: import matplotlib.pyplot as plt
         import numpy as np
         per=df['sweptarea']
         per1=np.percentile(per,0.25)
         plt.style.use('ggplot')
         per.plot.hist(alpha=0.5)
         plt.plot(per1)
         plt.show()
```



Normalization

It is the process of reorganizing data in a dataset so that it meets two basic requirements:
(1)There is no redundancy of data (all data is stored in only one place), and
(2)data dependencies are logical (all related data items are stored together)

Normalizing in scikit-learn refers to rescaling each observation (row) to have a length of 1 (called a unit norm in linear algebra).
This preprocessing can be useful for sparse datasets (lots of zeros) with attributes of varying scales when using algorithms that weight input values such as neural networks and algorithms that use distance measures such as K-Nearest Neighbors.

Normalization

Normalizing in scikit-learn refers to rescaling each observation (row) to have a length of 0 to1

In [60]:
```
import pandas as pd
sweptarea=df['sweptarea']
def normalize_list(sweptarea):
    max_value = max(sweptarea)
    min_value = min(sweptarea)
    for i in range(0, len(sweptarea)):
        sweptarea[i] = (sweptarea[i] - min_value) / (max_value - min_value)
for i in range(0, len(sweptarea)):
    print(sweptarea[i])
```

```
0.17786849655999748
0.20457497464201144
0.0
0.2281144668108021
0.13335773765560585
0.0
0.1712660194000922
0.17108696917202698
0.19946643032036562
0.17786849655999748
0.0013115429205777952
0.17842802852270134
```

Categorical data

Categorical Attributes –

- When the number unique values in a categorical column are too high, check the value counts of each of those values. Replace rarely occurring values together into a single value like ‘Other’ before encoding.
- When number of unique values is huge and even the values are equally distributed, try to find some related values and see if the multiple categorical values can be clubbed into single (grouping), thereby reducing the count of categorical values.

Related Attributes –

- If there multiple attributes with same information with different granularity, like city and state, it’s better to keep columns like state and delete city column. Additionally, keeping both columns and assessing feature importance might help in eliminating one column.

Handling Categorical data

1.Label encoding

2.Range encoding

3.one-hot encoding

Handling Categorical data

1.Label encoding

Sex	New_sex
Male	1
Female	0
Female	0
Male	1
Male	1

2.Range encoding

Height	Avg_Height	Low_Height	High_Height
100-110	105	100	100
110-120	115	110	110
120-130	125	120	120
130-140	135	130	130
140-150	145	140	150

Handling Categorical data

2.one-hot encoding

Hyderabad
Guntur
Hyderabad
Vizag
Vizag

city	New_city_hyd	New_city_gnt	New_city_viz
Hyderabad	1	0	0
Guntur	0	1	0
Hyderabad	1	0	0
Vizag	0	0	1
Vizag	0	0	1

Categorical data

Label Encoder	One Hot Encoder
Numeric representation, ordinals	Binary representation
Loses uniqueness of values, single dimension in vector space	Individual values expressed as a different dimension in orthogonal vector space
Suitable with categorical values that are ordinal in nature, like – fog_level (low, medium, high)	Suitable with non-ordinal types of categorical attributes, like – car_type (hatchback, sedan, SUV, etc.)
Label encoded categorical attributes don't pose any further challenges	One hot encoded categorical attributes might dramatically increase the feature space (curse of dimensionality). When One hot encoding is used, it's often followed by PCA to tackle high-dimensionality

Conclusion

You are aware of

Data Visualization

Data Interpretation

We will proceed with

Data Preprocessing

Kmit

SONET

DATA SCIENCE

THANK YOU