Experiment 4

Name: Danyl Fernandes (72)
Class: TE COMPS
XIE ID: 2020012004
Date: 14-08-2021

Aim: Perform Data pre-processing on given data set

Theory:

Data Preprocessing:
Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.

Why Data Preprocessing?

- Data in the real world is dirty
  1. incomplete: missing attribute values, lack of certain attributes of interest, or containing only aggregate data
     e.g., occupation=""
  2. noisy: containing errors or outliers
     e.g., Salary="-10"
  3. inconsistent: containing discrepancies in codes or names
     e.g., Age="42" Birthday="03/07/1997"
     e.g., Was rating "1,2,3", now rating "A, B, C"
     e.g., discrepancy between duplicate records

Why is Data Preprocessing important?

- No quality data, no quality mining results!
  1. Quality decisions must be based on quality data
     e.g., duplicate or missing data may cause incorrect or even misleading statistics.
  2. Data preparation, cleaning, and transformation comprises the majority of the work in a data mining application (around 90%).

Steps Involved in Data Preprocessing:

1. Data Cleaning:
The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

(a). Missing Data:
This situation arises when some data is missing in the data. It can be handled in various ways. Some of them are:
Ignore the tuples:
This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

Fill the Missing values:
There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

Why Missing Values Exist?
- Faulty equipment, incorrect measurements, missing cells in manual data entry, censored/anonymous data
- Review scores for movies, books, etc.
- Very frequent in questionnaires for medical scenarios
- Censored/anonymous data
- Interview data

How to handle missing data?
- Ignore the tuple
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
- a global constant:mean,median,mode

Measuring the Central Tendency
Mean (algebraic measure) (sample vs. population):
Note: n is sample size and N is population size.
Weighted arithmetic mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \mu = \frac{\sum x}{N}$$

Trimmed mean: chopping extreme values
Empirical formula:

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

Median:
Middle value if odd number of values, or average of the middle two values otherwise.

Mode
Value that occurs most frequently in the data : Unimodal, bimodal, trimodal

Measuring Dispersion of Data:

- Quartiles, outliers and boxplots
- Quartiles: Q1 (25th percentile), Q3 (75th percentile)
- Interquartile range: IQR = Q3 – Q1
- Five number summary: min, Q1, median, Q3, max
- Boxplot: ends of the box are the quartiles; median is marked; add whiskers, and plot outliers individually
- Outlier: usually, a value higher/lower than 1.5 x IQR

Variance: Variance is a simple measure of dispersion. Variance measures how far each number in the dataset from the mean.

$$\sigma^2 = \frac{\sum(x-\mu)^2}{n}$$

Standard deviation: σ is the square root of variance σ2 .Low standard deviation indicates data points close to mean.

$$\sigma = \sqrt{\frac{\sum(X-\mu)^2}{n}}$$

where,

σ = population standard deviation
$\sum$ = sum of...
μ = population mean
n = number of scores in sample.

(b). Noisy Data:
Noisy data is meaningless data that can't be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :

Noise: random error or variance in a measured variable incorrect attribute values may be due to faulty data collection instruments, data entry problems, data transmission problems, technology limitation, inconsistency in naming convention. Other data problems which require data cleaning: duplicate records, incomplete data, inconsistent data.

Binning Method:
This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segment is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.

Regression:
Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

Clustering:
This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

2. Data Transformation and Integration:

- Data integration: Combines data from multiple sources

Schema integration: e.g., A.cust-id B.cust-#

- Integrate metadata from different sources

Entity identification problem:

- Identify real world entities from multiple data sources, e.g., Bombay = Mumbai

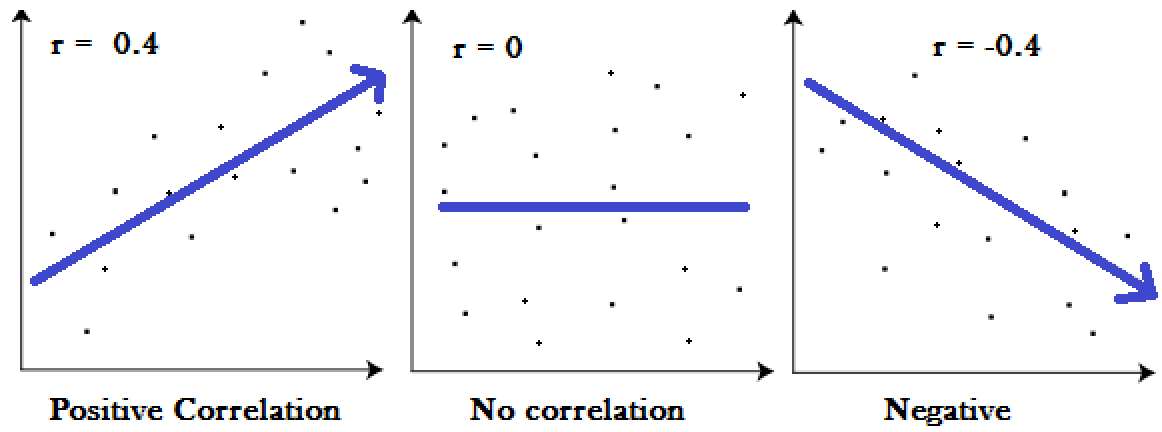
Detecting and resolving data value conflicts

- For the same real world entity, attribute values from different sources are different
- Possible reasons: different representations, different scales

Handling Redundancy in Data Integration

- Redundant data occur often when integration of multiple databases
- Redundant attributes may be able to be detected by correlation analysis and covariance analysis
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

Correlation Coefficient
Pearson's Correlation Coefficient is a linear correlation coefficient that returns a value of between -1 and +1. A -1 means there is a strong negative correlation and +1 means that there is a strong positive correlation. A 0 means that there is no correlation (this is also called zero correlation).



Data Transformation: mapping the entire old values of an attribute to a new values w.r.t. each old value

- Smoothing: Remove noise from data

Attribute/feature construction: New attributes constructed from the given ones
Aggregation: Summarization, data cube construction
Normalization: Scaled to fall within a smaller, specified range

- min-max normalization
- z-score normalization
- normalization by decimal scaling

Generalization: Concept hierarchy climbing
The attribute with the most distinct values is placed at the lowest level of the hierarchy
Exceptions, e.g., weekday, month, quarter, year

Normalization: Min-max normalization: to [new_minA, new_maxA]

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

Z-score normalization : (μ: mean, σ: standard deviation)

$$v' = \frac{v - \mu_A}{\sigma_A}$$

Normalization by decimal scaling

$$v' = \frac{v}{10^j}$$

Where j is the smallest integer such that Max(|v'|) < 1

3. Data Reduction:
Since data mining is a technique that is used to handle huge amounts of data. While working with a huge volume of data, analysis became harder in such cases. In order to get rid of this, we use data reduction techniques. It aims to increase the storage efficiency and reduce data storage and analysis costs.

The various steps to data reduction are:

Data Cube Aggregation:
Aggregation operation is applied to data for the construction of the data cube.

Attribute Subset Selection:
The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use the level of significance and p- value of the attribute.The attribute having p-value greater than significance level can be discarded.

Numerosity Reduction:
This enables us to store the model of data instead of whole data, for example: Regression Models.

Dimensionality Reduction:
This reduces the size of data by encoding mechanisms.It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction is called lossless reduction, else it is called lossy reduction. The two effective methods of dimensionality reduction are:Wavelet transforms and PCA (Principal Component Analysis).

Conclusion: Study of preprocessing of dataset, and application of the same on the suggested dataset was successfully completed**.**

```
1 import pandas as pd
2 import numpy as np
```

```
1 from google.colab import drive
2 drive.mount('/content/gdrive')
```

```
--NORMAL--
Mounted at /content/gdrive
```

```
1 df = pd.read_csv('/content/gdrive/MyDrive/SEM5/DWM/Datasets/car_datas
```

```
1 df.head(10)
```

	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
5	12950	32.0	61000.0	Diesel	90.0	0.0	0	2000	3	1170
6	16900	27.0	NaN	Diesel	NaN	NaN	0	2000	3	1245
7	18600	30.0	75889.0	NaN	90.0	1.0	0	2000	3	1245
8	21500	27.0	19700.0	Petrol	192.0	0.0	0	1800	3	1185
9	12950	23.0	71138.0	Diesel	NaN	NaN	0	1900	3	1105

```
1 df.describe()
```

	Price	Age	KM	HP	MetColor	Automat
count	1436.000000	1336.000000	1421.000000	1430.000000	1286.000000	1436.0000
mean	10730.824513	55.672156	68647.239972	101.478322	0.674961	0.0557
std	3626.964585	18.589804	37333.023589	14.768255	0.468572	0.2294
min	4350.000000	1.000000	1.000000	69.000000	0.000000	0.0000
25%	8450.000000	43.000000	43210.000000	90.000000	0.000000	0.0000
50%	9900.000000	60.000000	63634.000000	110.000000	1.000000	0.0000
75%	11950.000000	70.000000	87000.000000	110.000000	1.000000	0.0000
max	32500.000000	80.000000	243000.000000	192.000000	1.000000	1.0000

```
1 df.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   object
 9   Weight     1436 non-null   int64
dtypes: float64(4), int64(4), object(2)
memory usage: 123.4+ KB
```

```
1 print(df.isnull().sum())
```

```
Price          0
Age          100
KM            15
FuelType     100
HP             6
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

```
1 df['Age'].fillna(df['Age'].mean(), inplace=True)
2 df.head(50)
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors
0	13500	23.000000	46986.0	Diesel	90.0	1.0	0	2000	three
1	13750	23.000000	72937.0	Diesel	90.0	1.0	0	2000	3
2	13950	24.000000	41711.0	Diesel	90.0	NaN	0	2000	3
3	14950	26.000000	48000.0	Diesel	90.0	0.0	0	2000	3
4	13750	30.000000	38500.0	Diesel	90.0	0.0	0	2000	3
5	12950	32.000000	61000.0	Diesel	90.0	0.0	0	2000	3
6	16900	27.000000	NaN	Diesel	NaN	NaN	0	2000	3
7	18600	30.000000	75889.0	NaN	90.0	1.0	0	2000	3
8	21500	27.000000	19700.0	Petrol	192.0	0.0	0	1800	3
9	12950	23.000000	71138.0	Diesel	NaN	NaN	0	1900	3
10	20950	25.000000	31461.0	Petrol	192.0	0.0	0	1800	3
11	19950	22.000000	43610.0	Petrol	192.0	0.0	0	1800	3
12	19600	25.000000	32189.0	Petrol	192.0	0.0	0	1800	3
13	21500	31.000000	23000.0	Petrol	192.0	1.0	0	1800	3
14	22500	32.000000	34131.0	Petrol	192.0	1.0	0	1800	3
15	22000	28.000000	18739.0	Petrol	NaN	0.0	0	1800	3
16	22750	30.000000	34000.0	Petrol	192.0	1.0	0	1800	3
17	17950	24.000000	21716.0	Petrol	110.0	1.0	0	1600	3
18	16750	24.000000	25563.0	Petrol	110.0	0.0	0	1600	3
19	16950	30.000000	64359.0	Petrol	110.0	1.0	0	1600	3
20	15950	30.000000	67660.0	Petrol	110.0	1.0	0	1600	3
21	16950	29.000000	43905.0	NaN	110.0	0.0	1	1600	3
22	15950	28.000000	56349.0	Petrol	110.0	1.0	0	1600	3
23	16950	28.000000	32220.0	Petrol	110.0	1.0	0	1600	3
24	16250	29.000000	25813.0	Petrol	110.0	1.0	0	1600	3
25	15950	25.000000	28450.0	Petrol	110.0	1.0	0	1600	3
26	17495	27.000000	34545.0	NaN	110.0	1.0	0	1600	3
27	15750	29.000000	41415.0	Petrol	110.0	1.0	0	1600	3
28	16950	28.000000	44142.0	Petrol	110.0	0.0	0	1600	3
29	17950	30.000000	11090.0	NaN	110.0	NaN	0	1600	3
30	12950	29.000000	9750.0	Petrol	97.0	1.0	0	1400	3
31	15750	22.000000	35199.0	Petrol	97.0	1.0	0	1400	3
32	15950	27.000000	29510.0	Petrol	97.0	1.0	0	1400	3

33	14950	55.672156	32692.0	Petrol	97.0	1.0	0	1400	3
34	15500	22.000000	41000.0	Petrol	97.0	1.0	0	1400	3
35	15750	26.000000	43000.0	Petrol	97.0	0.0	0	1400	3
36	15950	25.000000	25000.0	Petrol	97.0	0.0	0	1400	3
37	14950	23.000000	10000.0	Petrol	97.0	1.0	0	1400	3
38	15750	32.000000	25329.0	Petrol	97.0	1.0	0	1400	3

```
df['KM'].fillna(df['KM'].median(), inplace=True)
df.head(50)
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors
0	13500	23.000000	46986.0	Diesel	90.0	1.0	0	2000	three
1	13750	23.000000	72937.0	Diesel	90.0	1.0	0	2000	3
2	13950	24.000000	41711.0	Diesel	90.0	NaN	0	2000	3
3	14950	26.000000	48000.0	Diesel	90.0	0.0	0	2000	3
4	13750	30.000000	38500.0	Diesel	90.0	0.0	0	2000	3
5	12950	32.000000	61000.0	Diesel	90.0	0.0	0	2000	3
6	16900	27.000000	63634.0	Diesel	NaN	NaN	0	2000	3
7	18600	30.000000	75889.0	NaN	90.0	1.0	0	2000	3
8	21500	27.000000	19700.0	Petrol	192.0	0.0	0	1800	3
9	12950	23.000000	71138.0	Diesel	NaN	NaN	0	1900	3
10	20950	25.000000	31461.0	Petrol	192.0	0.0	0	1800	3
11	19950	22.000000	43610.0	Petrol	192.0	0.0	0	1800	3
12	19600	25.000000	32189.0	Petrol	192.0	0.0	0	1800	3
13	21500	31.000000	23000.0	Petrol	192.0	1.0	0	1800	3
14	22500	32.000000	34131.0	Petrol	192.0	1.0	0	1800	3
15	22000	28.000000	18739.0	Petrol	NaN	0.0	0	1800	3
16	22750	30.000000	34000.0	Petrol	192.0	1.0	0	1800	3
17	17950	24.000000	21716.0	Petrol	110.0	1.0	0	1600	3
18	16750	24.000000	25563.0	Petrol	110.0	0.0	0	1600	3
19	16950	30.000000	64359.0	Petrol	110.0	1.0	0	1600	3
20	15950	30.000000	67660.0	Petrol	110.0	1.0	0	1600	3
21	16950	29.000000	43905.0	NaN	110.0	0.0	1	1600	3
22	15950	28.000000	56349.0	Petrol	110.0	1.0	0	1600	3
23	16950	28.000000	32220.0	Petrol	110.0	1.0	0	1600	3
24	16250	29.000000	25813.0	Petrol	110.0	1.0	0	1600	3

```
df['HP'].fillna(df['HP'].mean(),inplace=True)
df.head(50)
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Do
0	13500	23.000000	46986.0	Diesel	90.000000	1.0	0	2000	[illegible]
1	13750	23.000000	72937.0	Diesel	90.000000	1.0	0	2000	
2	13950	24.000000	41711.0	Diesel	90.000000	NaN	0	2000	
3	14950	26.000000	48000.0	Diesel	90.000000	0.0	0	2000	
4	13750	30.000000	38500.0	Diesel	90.000000	0.0	0	2000	
5	12950	32.000000	61000.0	Diesel	90.000000	0.0	0	2000	
6	16900	27.000000	63634.0	Diesel	101.478322	NaN	0	2000	
7	18600	30.000000	75889.0	NaN	90.000000	1.0	0	2000	
8	21500	27.000000	19700.0	Petrol	192.000000	0.0	0	1800	
9	12950	23.000000	71138.0	Diesel	101.478322	NaN	0	1900	
10	20950	25.000000	31461.0	Petrol	192.000000	0.0	0	1800	
11	19950	22.000000	43610.0	Petrol	192.000000	0.0	0	1800	
12	19600	25.000000	32189.0	Petrol	192.000000	0.0	0	1800	
13	21500	31.000000	23000.0	Petrol	192.000000	1.0	0	1800	
14	22500	32.000000	34131.0	Petrol	192.000000	1.0	0	1800	
15	22000	28.000000	18739.0	Petrol	101.478322	0.0	0	1800	
16	22750	30.000000	34000.0	Petrol	192.000000	1.0	0	1800	
17	17950	24.000000	21716.0	Petrol	110.000000	1.0	0	1600	
18	16750	24.000000	25563.0	Petrol	110.000000	0.0	0	1600	
19	16950	30.000000	64359.0	Petrol	110.000000	1.0	0	1600	
20	15950	30.000000	67660.0	Petrol	110.000000	1.0	0	1600	
21	16950	29.000000	43905.0	NaN	110.000000	0.0	1	1600	
22	15950	28.000000	56349.0	Petrol	110.000000	1.0	0	1600	
23	16950	28.000000	32220.0	Petrol	110.000000	1.0	0	1600	
24	16250	29.000000	25813.0	Petrol	110.000000	1.0	0	1600	
25	15950	25.000000	28450.0	Petrol	110.000000	1.0	0	1600	
26	17495	27.000000	34545.0	NaN	110.000000	1.0	0	1600	
27	15750	29.000000	41415.0	Petrol	110.000000	1.0	0	1600	
28	16950	28.000000	44142.0	Petrol	110.000000	0.0	0	1600	
29	17950	30.000000	11090.0	NaN	110.000000	NaN	0	1600	
30	12950	29.000000	9750.0	Petrol	97.000000	1.0	0	1400	
31	15750	22.000000	35199.0	Petrol	97.000000	1.0	0	1400	
32	15950	27.000000	29510.0	Petrol	97.000000	1.0	0	1400	

33	14950	55.672156	32692.0	Petrol	97.000000	1.0	0	1400
34	15500	22.000000	41000.0	Petrol	97.000000	1.0	0	1400
35	15750	26.000000	43000.0	Petrol	97.000000	0.0	0	1400
36	15950	25.000000	25000.0	Petrol	97.000000	0.0	0	1400
37	14950	23.000000	10000.0	Petrol	97.000000	1.0	0	1400
38	15750	32.000000	25329.0	Petrol	97.000000	1.0	0	1400
39	14750	27.000000	27500.0	Petrol	97.000000	0.0	0	1400
40	13950	22.000000	49059.0	Petrol	97.000000	0.0	0	1400
41	16750	27.000000	44068.0	Petrol	97.000000	1.0	0	1400
42	13950	22.000000	46961.0	Petrol	97.000000	0.0	0	1400
43	16950	27.000000	110404.0	Diesel	90.000000	NaN	0	2000
44	16950	22.000000	100250.0	NaN	90.000000	0.0	0	2000
45	19000	23.000000	84000.0	Diesel	90.000000	NaN	0	2000

```
print(df.isnull().sum())
```

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

```
df['FuelType'].mode()
```

```
0    Petrol
dtype: object
```

```
df['FuelType'].value_counts().index[0]
```

```
'Petrol'
```

```
df['FuelType'].fillna(df['FuelType'].mode()[0], inplace=True)
df.head(50)
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Dc
0	13500	23.000000	46986.0	Diesel	90.000000	1.0	0	2000	
1	13750	23.000000	72937.0	Diesel	90.000000	1.0	0	2000	
2	13950	24.000000	41711.0	Diesel	90.000000	NaN	0	2000	
3	14950	26.000000	48000.0	Diesel	90.000000	0.0	0	2000	
4	13750	30.000000	38500.0	Diesel	90.000000	0.0	0	2000	
5	12950	32.000000	61000.0	Diesel	90.000000	0.0	0	2000	
6	16900	27.000000	63634.0	Diesel	101.478322	NaN	0	2000	
7	18600	30.000000	75889.0	Petrol	90.000000	1.0	0	2000	
8	21500	27.000000	19700.0	Petrol	192.000000	0.0	0	1800	
9	12950	23.000000	71138.0	Diesel	101.478322	NaN	0	1900	
10	20950	25.000000	31461.0	Petrol	192.000000	0.0	0	1800	
11	19950	22.000000	43610.0	Petrol	192.000000	0.0	0	1800	
12	19600	25.000000	32189.0	Petrol	192.000000	0.0	0	1800	
13	21500	31.000000	23000.0	Petrol	192.000000	1.0	0	1800	
14	22500	32.000000	34131.0	Petrol	192.000000	1.0	0	1800	
15	22000	28.000000	18739.0	Petrol	101.478322	0.0	0	1800	
16	22750	30.000000	34000.0	Petrol	192.000000	1.0	0	1800	
17	17950	24.000000	21716.0	Petrol	110.000000	1.0	0	1600	
18	16750	24.000000	25563.0	Petrol	110.000000	0.0	0	1600	
19	16950	30.000000	64359.0	Petrol	110.000000	1.0	0	1600	
20	15950	30.000000	67660.0	Petrol	110.000000	1.0	0	1600	
21	16950	29.000000	43905.0	Petrol	110.000000	0.0	1	1600	
22	15950	28.000000	56349.0	Petrol	110.000000	1.0	0	1600	
23	16950	28.000000	32220.0	Petrol	110.000000	1.0	0	1600	
24	16250	29.000000	25813.0	Petrol	110.000000	1.0	0	1600	
25	15950	25.000000	28450.0	Petrol	110.000000	1.0	0	1600	
26	17495	27.000000	34545.0	Petrol	110.000000	1.0	0	1600	
27	15750	29.000000	41415.0	Petrol	110.000000	1.0	0	1600	
28	16950	28.000000	44142.0	Petrol	110.000000	0.0	0	1600	
29	17950	30.000000	11090.0	Petrol	110.000000	NaN	0	1600	
30	12950	29.000000	9750.0	Petrol	97.000000	1.0	0	1400	
31	15750	22.000000	35199.0	Petrol	97.000000	1.0	0	1400	
32	15950	27.000000	29510.0	Petrol	97.000000	1.0	0	1400	

33	14950	55.672156	32692.0	Petrol	97.000000	1.0	0	1400
34	15500	22.000000	41000.0	Petrol	97.000000	1.0	0	1400
35	15750	26.000000	43000.0	Petrol	97.000000	0.0	0	1400
36	15950	25.000000	25000.0	Petrol	97.000000	0.0	0	1400
37	14950	23.000000	10000.0	Petrol	97.000000	1.0	0	1400
38	15750	32.000000	25329.0	Petrol	97.000000	1.0	0	1400
39	14750	27.000000	27500.0	Petrol	97.000000	0.0	0	1400
40	13950	22.000000	49059.0	Petrol	97.000000	0.0	0	1400
41	16750	27.000000	44068.0	Petrol	97.000000	1.0	0	1400
42	13950	22.000000	46961.0	Petrol	97.000000	0.0	0	1400
43	16950	27.000000	110404.0	Diesel	90.000000	NaN	0	2000
44	16950	22.000000	100250.0	Petrol	90.000000	0.0	0	2000
45	19000	23.000000	84000.0	Diesel	90.000000	NaN	0	2000
46	17950	27.000000	79375.0	Diesel	90.000000	1.0	0	2000

```
1 df['MetColor'].mode()
```

```
0    1.0
dtype: float64
```

```
1 df['MetColor'].fillna(df['MetColor'].mode()[0],inplace=True)
2 df.head(50)
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Do
0	13500	23.000000	46986.0	Diesel	90.000000	1.0	0	2000	[illegible]
1	13750	23.000000	72937.0	Diesel	90.000000	1.0	0	2000	
2	13950	24.000000	41711.0	Diesel	90.000000	1.0	0	2000	
3	14950	26.000000	48000.0	Diesel	90.000000	0.0	0	2000	
4	13750	30.000000	38500.0	Diesel	90.000000	0.0	0	2000	
5	12950	32.000000	61000.0	Diesel	90.000000	0.0	0	2000	
6	16900	27.000000	63634.0	Diesel	101.478322	1.0	0	2000	
7	18600	30.000000	75889.0	Petrol	90.000000	1.0	0	2000	
8	21500	27.000000	19700.0	Petrol	192.000000	0.0	0	1800	
9	12950	23.000000	71138.0	Diesel	101.478322	1.0	0	1900	
10	20950	25.000000	31461.0	Petrol	192.000000	0.0	0	1800	
11	19950	22.000000	43610.0	Petrol	192.000000	0.0	0	1800	
12	19600	25.000000	32189.0	Petrol	192.000000	0.0	0	1800	
13	21500	31.000000	23000.0	Petrol	192.000000	1.0	0	1800	
14	22500	32.000000	34131.0	Petrol	192.000000	1.0	0	1800	
15	22000	28.000000	18739.0	Petrol	101.478322	0.0	0	1800	
16	22750	30.000000	34000.0	Petrol	192.000000	1.0	0	1800	
17	17950	24.000000	21716.0	Petrol	110.000000	1.0	0	1600	
18	16750	24.000000	25563.0	Petrol	110.000000	0.0	0	1600	
19	16950	30.000000	64359.0	Petrol	110.000000	1.0	0	1600	
20	15950	30.000000	67660.0	Petrol	110.000000	1.0	0	1600	
21	16950	29.000000	43905.0	Petrol	110.000000	0.0	1	1600	
22	15950	28.000000	56349.0	Petrol	110.000000	1.0	0	1600	
23	16950	28.000000	32220.0	Petrol	110.000000	1.0	0	1600	
24	16250	29.000000	25813.0	Petrol	110.000000	1.0	0	1600	
25	15950	25.000000	28450.0	Petrol	110.000000	1.0	0	1600	
26	17495	27.000000	34545.0	Petrol	110.000000	1.0	0	1600	
27	15750	29.000000	41415.0	Petrol	110.000000	1.0	0	1600	
28	16950	28.000000	44142.0	Petrol	110.000000	0.0	0	1600	
29	17950	30.000000	11090.0	Petrol	110.000000	1.0	0	1600	
30	12950	29.000000	9750.0	Petrol	97.000000	1.0	0	1400	
31	15750	22.000000	35199.0	Petrol	97.000000	1.0	0	1400	
32	15950	27.000000	29510.0	Petrol	97.000000	1.0	0	1400	

33	14950	55.672156	32692.0	Petrol	97.000000	1.0	0	1400
34	15500	22.000000	41000.0	Petrol	97.000000	1.0	0	1400
35	15750	26.000000	43000.0	Petrol	97.000000	0.0	0	1400
36	15950	25.000000	25000.0	Petrol	97.000000	0.0	0	1400

```
print(df.isnull().sum())
```

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

```
quartile_one = df['Age'].quantile(0.25)
quartile_three = df['Age'].quantile(0.75)
iqr = quartile_three - quartile_one
```

```
hp = df['HP']
price = df['Price']
print(hp.corr(price))
```

```
0.3084140566307208
```

```
df_min_max_scaled = df.copy()
```

Applying Normalization Techniques

For the column in `df_min_max_scaled.columns`:

```
# Apply normalization techniques
# For column in df_min_max_scaled.columns:
df_min_max_scaled['Price']= (df_min_max_scaled['Price'] - df_min_max_

# view normalized data
print(df_min_max_scaled)
```

```
      Price        Age       KM FuelType  ...  Automatic    CC  Doors  Wei
0  0.325044  23.000000  46986.0   Diesel  ...          0  2000  three    1
1  0.333925  23.000000  72937.0   Diesel  ...          0  2000      3    1
2  0.341030  24.000000  41711.0   Diesel  ...          0  2000      3    1
3  0.376554  26.000000  48000.0   Diesel  ...          0  2000      3    1
4  0.333925  30.000000  38500.0   Diesel  ...          0  2000      3    1
```

```
...        ...        ...       ...      ...  ...         ...   ...    ...
1431  0.111901  55.672156  20544.0   Petrol  ...           0  1300      3    1(
1432  0.230728  72.000000  63634.0   Petrol  ...           0  1300      3    1(
1433  0.147425  55.672156  17016.0   Petrol  ...           0  1300      3    1(
1434  0.103020  70.000000  63634.0   Petrol  ...           0  1300      3    1(
1435  0.092362  76.000000      1.0   Petrol  ...           0  1600      5    1'

[1436 rows x 10 columns]
```

```python
df_z_scaled = df.copy()

# Apply normalization techniques
# For column in df_z_scaled.columns:
df_z_scaled['Price'] = (df_z_scaled['Price']-df_z_scaled['Price'].mea

# View normalized data
display(df_z_scaled)
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doo
0	0.763497	23.000000	46986.0	Diesel	90.0	1.0	0	2000	th
1	0.832425	23.000000	72937.0	Diesel	90.0	1.0	0	2000	
2	0.887567	24.000000	41711.0	Diesel	90.0	1.0	0	2000	
3	1.163280	26.000000	48000.0	Diesel	90.0	0.0	0	2000	
4	0.832425	30.000000	38500.0	Diesel	90.0	0.0	0	2000	
...	...	...	...	...	...	...	...	...	
1431	-0.890779	55.672156	20544.0	Petrol	86.0	1.0	0	1300	
1432	0.031480	72.000000	63634.0	Petrol	86.0	0.0	0	1300	
1433	-0.615067	55.672156	17016.0	Petrol	86.0	0.0	0	1300	
1434	-0.959707	70.000000	63634.0	Petrol	86.0	1.0	0	1300	
1435	-1.042421	76.000000	1.0	Petrol	110.0	0.0	0	1600	

1436 rows × 10 columns

```python
df.dtypes
df_cars=df.copy()
df_cars=df_cars.drop([0])
df_cars=df_cars.drop(columns=['FuelType','Doors'], axis=1)
df_cars
```

	Price	Age	KM	HP	MetColor	Automatic	CC	Weight
1	13750	23.000000	72937.0	90.0	1.0	0	2000	1165
2	13950	24.000000	41711.0	90.0	1.0	0	2000	1165
3	14950	26.000000	48000.0	90.0	0.0	0	2000	1165
4	13750	30.000000	38500.0	90.0	0.0	0	2000	1170
5	12950	32.000000	61000.0	90.0	0.0	0	2000	1170
...	...	...	...	...	...	...	...	...
1431	7500	55.672156	20544.0	86.0	1.0	0	1300	1025
1432	10845	72.000000	63634.0	86.0	0.0	0	1300	1015
1433	8500	55.672156	17016.0	86.0	0.0	0	1300	1015

```
from sklearn.preprocessing import MinMaxScaler

# create a scaler object
scaler = MinMaxScaler()
# fit and transform the data
df_norm = pd.DataFrame(scaler.fit_transform(df_cars), columns=df_cars

df_norm
```

	Price	Age	KM	HP	MetColor	Automatic	CC	Weight
0	0.333925	0.278481	0.300149	0.170732	1.0	0.0	1.000000	0.268293
1	0.341030	0.291139	0.171647	0.170732	1.0	0.0	1.000000	0.268293
2	0.376554	0.316456	0.197528	0.170732	0.0	0.0	1.000000	0.268293
3	0.333925	0.367089	0.158433	0.170732	0.0	0.0	1.000000	0.276423
4	0.305506	0.392405	0.251026	0.170732	0.0	0.0	1.000000	0.276423
...	...	...	...	...	...	...	...	...
1430	0.111901	0.692053	0.084539	0.138211	1.0	0.0	0.000000	0.040650
1431	0.230728	0.898734	0.261865	0.138211	0.0	0.0	0.000000	0.024390
1432	0.147425	0.692053	0.070021	0.138211	0.0	0.0	0.000000	0.024390
1433	0.103020	0.873418	0.261865	0.138211	1.0	0.0	0.000000	0.024390
1434	0.092362	0.949367	0.000000	0.333333	0.0	0.0	0.428571	0.185366

1435 rows × 8 columns

0s completed at 22:19