Steps for Data preprocessing

1.Data Cleaning
2.Dta transformation

1. DATA CLEANING

In [150]:

```
import pandas as pd
```

In [151]:

```
weather_data_2010=pd.read_csv('data_clean.csv')
weather_data_2010
```

Out[151]:

	Unnamed: 0	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
0	1	41.0	190.0	7.4	67	5	1	2010	67	S
1	2	36.0	118.0	8.0	72	5	2	2010	72	C
2	3	12.0	149.0	12.6	74	5	3	2010	74	PS
3	4	18.0	313.0	11.5	62	5	4	2010	62	S
4	5	NaN	NaN	14.3	56	5	5	2010	56	S
...	...	...	...	...	...	...	...	...	...	...
153	154	41.0	190.0	7.4	67	5	1	2010	67	C
154	155	30.0	193.0	6.9	70	9	26	2010	70	PS
155	156	NaN	145.0	13.2	77	9	27	2010	77	S
156	157	14.0	191.0	14.3	75	9	28	2010	75	S
157	158	18.0	131.0	8.0	76	9	29	2010	76	C

158 rows × 10 columns

3. Data Understanding

In [152]:

```
weather_data_2010.shape
```

Out[152]:

```
(158, 10)
```

In [153]:

```
weather_data_2010.isna().sum()
```

Out[153]:

```
Unnamed: 0     0
Ozone         38
Solar.R        7
Wind           0
Temp C         0
Month          0
Day            0
Year           0
Temp           0
Weather        3
dtype: int64
```

In [154]:

```
38/158
```

Out[154]:

```
0.24050632911392406
```

In [155]:

```
weather_data_2010.describe(include='all')
```

Out[155]:

	Unnamed: 0	Ozone	Solar.R	Wind	Temp C	Month	Day	Year	
count	158.000000	120.000000	151.000000	158.000000	158	158	158.000000	158.0	158
unique	NaN	NaN	NaN	NaN	41	6	NaN	NaN	
top	NaN	NaN	NaN	NaN	81	9	NaN	NaN	
freq	NaN	NaN	NaN	NaN	11	34	NaN	NaN	
mean	79.500000	41.583333	185.403974	9.957595	NaN	NaN	16.006329	2010.0	77
std	45.754781	32.620709	88.723103	3.511261	NaN	NaN	8.997166	0.0	9
min	1.000000	1.000000	7.000000	1.700000	NaN	NaN	1.000000	2010.0	56
25%	40.250000	18.000000	119.000000	7.400000	NaN	NaN	8.000000	2010.0	72
50%	79.500000	30.500000	197.000000	9.700000	NaN	NaN	16.000000	2010.0	78
75%	118.750000	61.500000	257.000000	11.875000	NaN	NaN	24.000000	2010.0	84
max	158.000000	168.000000	334.000000	20.700000	NaN	NaN	31.000000	2010.0	97

4.Data preprocessing

Client,Mainuddin agreed to drop the ozone feature as they information is not

captured properly

In [156]:

```
del weather_data_2010['Unnamed: 0']
```

In [157]:

```
del weather_data_2010['Ozone']
```

In [158]:

```
weather_data_2010.head(30)
```

Out[158]:

	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
0	190.0	7.4	67	5	1	2010	67	S
1	118.0	8.0	72	5	2	2010	72	C
2	149.0	12.6	74	5	3	2010	74	PS
3	313.0	11.5	62	5	4	2010	62	S
4	NaN	14.3	56	5	5	2010	56	S
5	NaN	14.9	66	5	6	2010	66	C
6	299.0	8.6	65	5	7	2010	65	PS
7	99.0	13.8	59	5	8	2010	59	C
8	19.0	20.1	61	5	9	2010	61	PS
9	194.0	8.6	69	5	10	2010	69	S
10	NaN	6.9	C	5	11	2010	74	C
11	256.0	9.7	69	5	12	2010	69	PS
12	290.0	9.2	66	5	13	2010	66	S
13	274.0	10.9	68	5	14	2010	68	S
14	65.0	13.2	58	5	15	2010	58	C
15	334.0	11.5	64	5	16	2010	64	S
16	307.0	12.0	66	5	17	2010	66	S
17	78.0	18.4	57	5	18	2010	57	C
18	322.0	11.5	68	5	19	2010	68	PS
19	44.0	9.7	62	5	20	2010	62	S
20	8.0	9.7	59	5	21	2010	59	S
21	320.0	16.6	73	5	22	2010	73	C
22	25.0	9.7	61	5	23	2010	61	PS
23	92.0	12.0	61	May	24	2010	61	C
24	66.0	16.6	57	5	25	2010	57	PS
25	266.0	14.9	58	5	26	2010	58	C
26	NaN	8.0	57	5	27	2010	57	PS
27	13.0	12.0	67	5	28	2010	67	S
28	252.0	14.9	81	5	29	2010	81	S
29	223.0	5.7	79	5	30	2010	79	C

In [159]:

```
weather_data_2010.isna().sum()
```

Out[159]:

```
Solar.R    7
Wind       0
Temp C     0
Month      0
Day        0
Year       0
Temp       0
Weather    3
dtype: int64
```

In [160]:

```
7/158
```

Out[160]:

```
0.04430379746835443
```

In [161]:

```
weather_data_2010.describe()
```

Out[161]:

	Solar.R	Wind	Day	Year	Temp
count	151.000000	158.000000	158.000000	158.0	158.000000
mean	185.403974	9.957595	16.006329	2010.0	77.727848
std	88.723103	3.511261	8.997166	0.0	9.377877
min	7.000000	1.700000	1.000000	2010.0	56.000000
25%	119.000000	7.400000	8.000000	2010.0	72.000000
50%	197.000000	9.700000	16.000000	2010.0	78.500000
75%	257.000000	11.875000	24.000000	2010.0	84.000000
max	334.000000	20.700000	31.000000	2010.0	97.000000

Client,Mainuddin agreed to go with Mean Imputation for Solar.R

In [162]:

```
weather_data_2010['Solar.R'].fillna(value=185.40, inplace=True)
```

In [163]:

```
weather_data_2010.head(30)
```

Out[163]:

	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
0	190.0	7.4	67	5	1	2010	67	S
1	118.0	8.0	72	5	2	2010	72	C
2	149.0	12.6	74	5	3	2010	74	PS
3	313.0	11.5	62	5	4	2010	62	S
4	185.4	14.3	56	5	5	2010	56	S
5	185.4	14.9	66	5	6	2010	66	C
6	299.0	8.6	65	5	7	2010	65	PS
7	99.0	13.8	59	5	8	2010	59	C
8	19.0	20.1	61	5	9	2010	61	PS
9	194.0	8.6	69	5	10	2010	69	S
10	185.4	6.9	C	5	11	2010	74	C
11	256.0	9.7	69	5	12	2010	69	PS
12	290.0	9.2	66	5	13	2010	66	S
13	274.0	10.9	68	5	14	2010	68	S
14	65.0	13.2	58	5	15	2010	58	C
15	334.0	11.5	64	5	16	2010	64	S
16	307.0	12.0	66	5	17	2010	66	S
17	78.0	18.4	57	5	18	2010	57	C
18	322.0	11.5	68	5	19	2010	68	PS
19	44.0	9.7	62	5	20	2010	62	S
20	8.0	9.7	59	5	21	2010	59	S
21	320.0	16.6	73	5	22	2010	73	C
22	25.0	9.7	61	5	23	2010	61	PS
23	92.0	12.0	61	May	24	2010	61	C
24	66.0	16.6	57	5	25	2010	57	PS
25	266.0	14.9	58	5	26	2010	58	C
26	185.4	8.0	57	5	27	2010	57	PS
27	13.0	12.0	67	5	28	2010	67	S
28	252.0	14.9	81	5	29	2010	81	S
29	223.0	5.7	79	5	30	2010	79	C

In [164]:

```
weather_data_2010.isna().sum()
```

Out[164]:

```
Solar.R    0
Wind       0
Temp C     0
Month      0
Day        0
Year       0
Temp       0
Weather    3
dtype: int64
```

In [165]:

```
weather_data_2010.describe(include='all')
```

Out[165]:

	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
count	158.000000	158.000000	158	158	158.000000	158.0	158.000000	155
unique	NaN	NaN	41	6	NaN	NaN	NaN	3
top	NaN	NaN	81	9	NaN	NaN	NaN	S
freq	NaN	NaN	11	34	NaN	NaN	NaN	59
mean	185.403797	9.957595	NaN	NaN	16.006329	2010.0	77.727848	NaN
std	86.722647	3.511261	NaN	NaN	8.997166	0.0	9.377877	NaN
min	7.000000	1.700000	NaN	NaN	1.000000	2010.0	56.000000	NaN
25%	127.000000	7.400000	NaN	NaN	8.000000	2010.0	72.000000	NaN
50%	192.500000	9.700000	NaN	NaN	16.000000	2010.0	78.500000	NaN
75%	255.000000	11.875000	NaN	NaN	24.000000	2010.0	84.000000	NaN
max	334.000000	20.700000	NaN	NaN	31.000000	2010.0	97.000000	NaN

In [166]:

```
weather_data_2010['Weather'].unique()
```

Out[166]:

```
array(['S', 'C', 'PS', nan], dtype=object)
```

Drop that 3 observations of weather

In [167]:

```
weather_data_2010.dropna(inplace=True)
```

In [168]:

```
weather_data_2010.isna().sum()
```

Out[168]:

```
Solar.R    0
Wind       0
Temp C     0
Month      0
Day        0
Year       0
Temp       0
Weather    0
dtype: int64
```

In [169]:

```
weather_data_2010.head(40)
```

Out[169]:

	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
0	190.0	7.4	67	5	1	2010	67	S
1	118.0	8.0	72	5	2	2010	72	C
2	149.0	12.6	74	5	3	2010	74	PS
3	313.0	11.5	62	5	4	2010	62	S
4	185.4	14.3	56	5	5	2010	56	S
5	185.4	14.9	66	5	6	2010	66	C
6	299.0	8.6	65	5	7	2010	65	PS
7	99.0	13.8	59	5	8	2010	59	C
8	19.0	20.1	61	5	9	2010	61	PS
9	194.0	8.6	69	5	10	2010	69	S
10	185.4	6.9	C	5	11	2010	74	C
11	256.0	9.7	69	5	12	2010	69	PS
12	290.0	9.2	66	5	13	2010	66	S
13	274.0	10.9	68	5	14	2010	68	S
14	65.0	13.2	58	5	15	2010	58	C
15	334.0	11.5	64	5	16	2010	64	S
16	307.0	12.0	66	5	17	2010	66	S
17	78.0	18.4	57	5	18	2010	57	C
18	322.0	11.5	68	5	19	2010	68	PS
19	44.0	9.7	62	5	20	2010	62	S
20	8.0	9.7	59	5	21	2010	59	S
21	320.0	16.6	73	5	22	2010	73	C
22	25.0	9.7	61	5	23	2010	61	PS
23	92.0	12.0	61	May	24	2010	61	C
24	66.0	16.6	57	5	25	2010	57	PS
25	266.0	14.9	58	5	26	2010	58	C
26	185.4	8.0	57	5	27	2010	57	PS
27	13.0	12.0	67	5	28	2010	67	S
28	252.0	14.9	81	5	29	2010	81	S
29	223.0	5.7	79	5	30	2010	79	C
30	279.0	7.4	76	5	31	2010	76	PS
31	286.0	8.6	78	6	1	2010	78	S
32	287.0	9.7	74	6	2	2010	74	C
33	242.0	16.1	67	6	3	2010	67	PS

	Solar.R	Wind	Temp C	Month	Day	Year	Temp	Weather
34	186.0	9.2	84	6	4	2010	84	C
35	220.0	8.6	85	6	5	2010	85	PS
36	264.0	14.3	79	6	6	2010	79	C
37	127.0	9.7	82	6	7	2010	82	PS
38	273.0	6.9	87	6	8	2010	87	S
39	291.0	13.8	90	6	9	2010	90	S

In [170]:

```
weather_data_2010.dtypes
```

Out[170]:

```
Solar.R    float64
Wind       float64
Temp C      object
Month       object
Day          int64
Year         int64
Temp         int64
Weather     object
dtype: object
```

1.2 DATA TYPE CONVERSION

In [179]:

```
weather_data_2010['Month']=pd.to_numeric(weather_data_2010['Month'],errors='coerce').astype
```

In [180]:

```
weather_data_2010.head(30)
```

Out[180]:

	Solar.R	Wind	Month	Day	Year	Temp	Weather
0	190.0	7.4	5	1	2010	67	S
1	118.0	8.0	5	2	2010	72	C
2	149.0	12.6	5	3	2010	74	PS
3	313.0	11.5	5	4	2010	62	S
4	185.4	14.3	5	5	2010	56	S
5	185.4	14.9	5	6	2010	66	C
6	299.0	8.6	5	7	2010	65	PS
7	99.0	13.8	5	8	2010	59	C
8	19.0	20.1	5	9	2010	61	PS
9	194.0	8.6	5	10	2010	69	S
10	185.4	6.9	5	11	2010	74	C
11	256.0	9.7	5	12	2010	69	PS
12	290.0	9.2	5	13	2010	66	S
13	274.0	10.9	5	14	2010	68	S
14	65.0	13.2	5	15	2010	58	C
15	334.0	11.5	5	16	2010	64	S
16	307.0	12.0	5	17	2010	66	S
17	78.0	18.4	5	18	2010	57	C
18	322.0	11.5	5	19	2010	68	PS
19	44.0	9.7	5	20	2010	62	S
20	8.0	9.7	5	21	2010	59	S
21	320.0	16.6	5	22	2010	73	C
22	25.0	9.7	5	23	2010	61	PS
23	92.0	12.0	5	24	2010	61	C
24	66.0	16.6	5	25	2010	57	PS
25	266.0	14.9	5	26	2010	58	C
26	185.4	8.0	5	27	2010	57	PS
27	13.0	12.0	5	28	2010	67	S
28	252.0	14.9	5	29	2010	81	S
29	223.0	5.7	5	30	2010	79	C

Numpy's where function

In [181]:

```
weather_data_2010['Month'].fillna(5,inplace=True)
```

In [186]:

```
weather_data_2010.head(30)
```

Out[186]:

	Solar.R	Wind	Month	Day	Year	Temp	Weather
0	190.0	7.4	5	1	2010	67	S
1	118.0	8.0	5	2	2010	72	C
2	149.0	12.6	5	3	2010	74	PS
3	313.0	11.5	5	4	2010	62	S
4	185.4	14.3	5	5	2010	56	S
5	185.4	14.9	5	6	2010	66	C
6	299.0	8.6	5	7	2010	65	PS
7	99.0	13.8	5	8	2010	59	C
8	19.0	20.1	5	9	2010	61	PS
9	194.0	8.6	5	10	2010	69	S
10	185.4	6.9	5	11	2010	74	C
11	256.0	9.7	5	12	2010	69	PS
12	290.0	9.2	5	13	2010	66	S
13	274.0	10.9	5	14	2010	68	S
14	65.0	13.2	5	15	2010	58	C
15	334.0	11.5	5	16	2010	64	S
16	307.0	12.0	5	17	2010	66	S
17	78.0	18.4	5	18	2010	57	C
18	322.0	11.5	5	19	2010	68	PS
19	44.0	9.7	5	20	2010	62	S
20	8.0	9.7	5	21	2010	59	S
21	320.0	16.6	5	22	2010	73	C
22	25.0	9.7	5	23	2010	61	PS
23	92.0	12.0	5	24	2010	61	C
24	66.0	16.6	5	25	2010	57	PS
25	266.0	14.9	5	26	2010	58	C
26	185.4	8.0	5	27	2010	57	PS
27	13.0	12.0	5	28	2010	67	S
28	252.0	14.9	5	29	2010	81	S
29	223.0	5.7	5	30	2010	79	C

In [187]:

```
weather_data_2010.dtypes
```

Out[187]:

```
Solar.R    float64
Wind       float64
Month        int32
Day          int64
Year         int64
Temp         int64
Weather     object
dtype: object
```

2. DATA TRANSFORMATION

There are 2 ways to transform the Discrete Data 1.Label Encoder 2.One Hot Encoder

2.1 Label Encoder

In [201]:

```
weather_data_2010_copy_1=weather_data_2010.copy()
weather_data_2010_copy_1
```

Out[201]:

	Solar.R	Wind	Month	Day	Year	Temp	Weather
0	190.0	7.4	5	1	2010	67	S
1	118.0	8.0	5	2	2010	72	C
2	149.0	12.6	5	3	2010	74	PS
3	313.0	11.5	5	4	2010	62	S
4	185.4	14.3	5	5	2010	56	S
...	...	...	...	...	...	...	...
153	190.0	7.4	5	1	2010	67	C
154	193.0	6.9	9	26	2010	70	PS
155	145.0	13.2	9	27	2010	77	S
156	191.0	14.3	9	28	2010	75	S
157	131.0	8.0	9	29	2010	76	C

155 rows × 7 columns

In [202]:

```
from sklearn.preprocessing import LabelEncoder
le =LabelEncoder()
weather_data_2010_copy_1['Weather']=le.fit_transform(weather_data_2010_copy_1['Weather'])
```

In [203]:

```
weather_data_2010_copy_1.head(30)
```

Out[203]:

	Solar.R	Wind	Month	Day	Year	Temp	Weather
0	190.0	7.4	5	1	2010	67	2
1	118.0	8.0	5	2	2010	72	0
2	149.0	12.6	5	3	2010	74	1
3	313.0	11.5	5	4	2010	62	2
4	185.4	14.3	5	5	2010	56	2
5	185.4	14.9	5	6	2010	66	0
6	299.0	8.6	5	7	2010	65	1
7	99.0	13.8	5	8	2010	59	0
8	19.0	20.1	5	9	2010	61	1
9	194.0	8.6	5	10	2010	69	2
10	185.4	6.9	5	11	2010	74	0
11	256.0	9.7	5	12	2010	69	1
12	290.0	9.2	5	13	2010	66	2
13	274.0	10.9	5	14	2010	68	2
14	65.0	13.2	5	15	2010	58	0
15	334.0	11.5	5	16	2010	64	2
16	307.0	12.0	5	17	2010	66	2
17	78.0	18.4	5	18	2010	57	0
18	322.0	11.5	5	19	2010	68	1
19	44.0	9.7	5	20	2010	62	2
20	8.0	9.7	5	21	2010	59	2
21	320.0	16.6	5	22	2010	73	0
22	25.0	9.7	5	23	2010	61	1
23	92.0	12.0	5	24	2010	61	0
24	66.0	16.6	5	25	2010	57	1
25	266.0	14.9	5	26	2010	58	0
26	185.4	8.0	5	27	2010	57	1
27	13.0	12.0	5	28	2010	67	2
28	252.0	14.9	5	29	2010	81	2
29	223.0	5.7	5	30	2010	79	0

In [204]:

```
weather_data_2010_copy_1.dtypes
```

Out[204]:

```
Solar.R    float64
Wind       float64
Month        int32
Day          int64
Year         int64
Temp         int64
Weather      int32
dtype: object
```

2.2 One Hot Encoding

This can be done either of 2 ways: 1.Using Pandas library 2.Using Sklearn library

In [212]:

```
weather_data_2010_copy_2=weather_data_2010.copy()
weather_data_2010_copy_2
```

Out[212]:

	Solar.R	Wind	Month	Day	Year	Temp	Weather
0	190.0	7.4	5	1	2010	67	S
1	118.0	8.0	5	2	2010	72	C
2	149.0	12.6	5	3	2010	74	PS
3	313.0	11.5	5	4	2010	62	S
4	185.4	14.3	5	5	2010	56	S
...	...	...	...	...	...	...	...
153	190.0	7.4	5	1	2010	67	C
154	193.0	6.9	9	26	2010	70	PS
155	145.0	13.2	9	27	2010	77	S
156	191.0	14.3	9	28	2010	75	S
157	131.0	8.0	9	29	2010	76	C

155 rows × 7 columns

In [221]:

```
weather_data_2010_copy_2=pd.get_dummies(data=weather_data_2010_copy_2)
weather_data_2010_copy_2
```

Out[221]:

	Solar.R	Wind	Month	Day	Year	Temp	Weather_C	Weather_PS	Weather_S
0	190.0	7.4	5	1	2010	67	0	0	1
1	118.0	8.0	5	2	2010	72	1	0	0
2	149.0	12.6	5	3	2010	74	0	1	0
3	313.0	11.5	5	4	2010	62	0	0	1
4	185.4	14.3	5	5	2010	56	0	0	1
...	...	...	...	...	...	...	...	...	...
153	190.0	7.4	5	1	2010	67	1	0	0
154	193.0	6.9	9	26	2010	70	0	1	0
155	145.0	13.2	9	27	2010	77	0	0	1
156	191.0	14.3	9	28	2010	75	0	0	1
157	131.0	8.0	9	29	2010	76	1	0	0

155 rows × 9 columns

In [222]:

```
weather_data_2010_copy_2.dtypes
```

Out[222]:

```
Solar.R       float64
Wind          float64
Month           int32
Day             int64
Year            int64
Temp            int64
Weather_C       uint8
Weather_PS      uint8
Weather_S       uint8
dtype: object
```

2. Using sklearn library

In [236]:

```
weather_data_2010_copy_3=weather_data_2010.copy()
weather_data_2010_copy_3
```

Out[236]:

	Solar.R	Wind	Month	Day	Year	Temp	Weather
0	190.0	7.4	5	1	2010	67	S
1	118.0	8.0	5	2	2010	72	C
2	149.0	12.6	5	3	2010	74	PS
3	313.0	11.5	5	4	2010	62	S
4	185.4	14.3	5	5	2010	56	S
...	...	...	...	...	...	...	...
153	190.0	7.4	5	1	2010	67	C
154	193.0	6.9	9	26	2010	70	PS
155	145.0	13.2	9	27	2010	77	S
156	191.0	14.3	9	28	2010	75	S
157	131.0	8.0	9	29	2010	76	C

155 rows × 7 columns

In [237]:

```
from sklearn.preprocessing import OneHotEncoder
ohe=OneHotEncoder()
ohe.fit_transform(weather_data_2010_copy_3['Weather'])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_13232/909129641.py in <module>
      1 from sklearn.preprocessing import OneHotEncoder
      2 ohe=OneHotEncoder()
----> 3 ohe.fit_transform(weather_data_2010_copy_3['Weather'])
      4

~\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py in fit_transform(self, X, y)
    449         """
    450         self._validate_keywords()
--> 451         return super().fit_transform(X, y)
    452
    453     def transform(self, X):

~\anaconda3\lib\site-packages\sklearn\base.py in fit_transform(self, X, y, **fit_params)
    697         if y is None:
    698             # fit method of arity 1 (unsupervised transformation)
--> 699             return self.fit(X, **fit_params).transform(X)
    700         else:
    701             # fit method of arity 2 (supervised transformation)

~\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py in fit(self, X, y)
    421         """
    422         self._validate_keywords()
--> 423         self._fit(X, handle_unknown=self.handle_unknown,
    424                   force_all_finite='allow-nan')
    425         self.drop_idx_ = self._compute_drop_idx()

~\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py in _fit(self, X, handle_unknown, force_all_finite)
     75
     76     def _fit(self, X, handle_unknown='error', force_all_finite=True):
---> 77         X_list, n_samples, n_features = self._check_X(
     78             X, force_all_finite=force_all_finite)
     79

~\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py in _check_X(self, X, force_all_finite)
     42         if not (hasattr(X, 'iloc') and getattr(X, 'ndim', 0) == 2):
     43             # if not a dataframe, do normal check_array validation
---> 44             X_temp = check_array(X, dtype=None,
     45                                  force_all_finite=force_all_finite)
     46             if (not hasattr(X, 'dtype')
```

```
~\anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*arg
s, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(a
rray, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_fi
nite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estima
tor)
    692             # If input is 1D raise error
    693             if array.ndim == 1:
--> 694                 raise ValueError(
    695                     "Expected 2D array, got 1D array instead:\narr
ay={}.\n"
    696                     "Reshape your data either using array.reshape
(-1, 1) if "

ValueError: Expected 2D array, got 1D array instead:
array=['S' 'C' 'PS' 'S' 'S' 'C' 'PS' 'C' 'PS' 'S' 'C' 'PS' 'S' 'S' 'C' 'S'
'S'
 'C' 'PS' 'S' 'S' 'C' 'PS' 'C' 'PS' 'C' 'PS' 'S' 'S' 'C' 'PS' 'S' 'C' 'PS'
 'C' 'PS' 'C' 'PS' 'S' 'S' 'S' 'C' 'PS' 'S' 'S' 'C' 'PS' 'C' 'PS' 'S' 'S'
 'S' 'C' 'PS' 'S' 'C' 'PS' 'C' 'PS' 'S' 'S' 'S' 'C' 'PS' 'S' 'S' 'C' 'C'
 'PS' 'C' 'PS' 'S' 'S' 'S' 'C' 'PS' 'S' 'S' 'C' 'PS' 'C' 'S' 'S' 'C' 'PS'
 'PS' 'C' 'S' 'C' 'PS' 'C' 'PS' 'PS' 'S' 'C' 'C' 'PS' 'C' 'PS' 'S' 'S' 'S'
 'C' 'C' 'C' 'PS' 'C' 'PS' 'S' 'S' 'C' 'PS' 'C' 'PS' 'S' 'S' 'S' 'S' 'S'
 'C' 'C' 'PS' 'C' 'PS' 'S' 'S' 'S' 'C' 'PS' 'C' 'PS' 'S' 'S' 'PS' 'PS' 'S'
 'PS' 'S' 'C' 'PS' 'PS' 'S' 'S' 'C' 'PS' 'C' 'PS' 'S' 'PS' 'S' 'C' 'PS'
 'S' 'S' 'C'].
Reshape your data either using array.reshape(-1, 1) if your data has a sin
gle feature or array.reshape(1, -1) if it contains a single sample.
```

In [ ]:

```
```

On what basis we have to choose Label Encoder and One Hot Encoder

Point to remember

- Always use Label encoder because we cant take more than one dependent variable in machine learning problems.

Input Features

for parametric models ,we go with *one Hot Encoding**

for Non-Parametric models,we go with *Label Encoder**

In [ ]:

```
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

```

In [ ]:

```

```