Handling Categorical data

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df = pd.read_csv("cars.csv")
df
```

Out[3]:

	Car_Name	mpg	cyl	disp	hp	drat	wt	qsec
0	Valiant	21.0	6	160.0	110	3.90	2.620	16.46
1	Valiant	21.0	6	160.0	110	3.90	2.875	17.02
2	Valiant	22.8	4	108.0	93	3.85	2.320	18.61
3	Valiant	21.4	6	258.0	110	3.08	3.215	19.44
4	Valiant	18.7	8	360.0	175	3.15	3.440	17.02
5	Valiant	18.1	6	225.0	105	2.76	3.460	20.22
6	Duster	14.3	8	360.0	245	3.21	3.570	15.84
7	Duster	24.4	4	146.7	62	3.69	3.190	20.00
8	Duster	22.8	4	140.8	95	3.92	3.150	22.90
9	Duster	19.2	6	167.6	123	3.92	3.440	18.30
10	Duster	17.8	6	167.6	123	3.92	3.440	18.90
11	Duster	16.4	8	275.8	180	3.07	4.070	17.40
12	Duster	17.3	8	275.8	180	3.07	3.730	17.60
13	Toyota Corona	15.2	8	275.8	180	3.07	3.780	18.00
14	Toyota Corona	10.4	8	472.0	205	2.93	5.250	17.98
15	Toyota Corona	10.4	8	460.0	215	3.00	5.424	17.82
16	Toyota Corona	14.7	8	440.0	230	3.23	5.345	17.42
17	Toyota Corona	32.4	4	78.7	66	4.08	2.200	19.47
18	Toyota Corona	30.4	4	75.7	52	4.93	1.615	18.52
19	Toyota Corona	33.9	4	71.1	65	4.22	1.835	19.90
20	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01
21	Volvo 142E	15.5	8	318.0	150	2.76	3.520	16.87
22	Volvo 142E	15.2	8	304.0	150	3.15	3.435	17.30
23	Volvo 142E	13.3	8	350.0	245	3.73	3.840	15.41
24	Volvo 142E	19.2	8	400.0	175	3.08	3.845	17.05
25	Volvo 142E	27.3	4	79.0	66	4.08	1.935	18.90
26	Volvo 142E	26.0	4	120.3	91	4.43	2.140	16.70
27	Volvo 142E	30.4	4	95.1	113	3.77	1.513	16.90
28	Volvo 142E	15.8	8	351.0	264	4.22	3.170	14.50
29	Volvo 142E	19.7	6	145.0	175	3.62	2.770	15.50
30	Volvo 142E	15.0	8	301.0	335	3.54	3.570	14.60
31	Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60

1. One hot encoding

In [4]:
```
dummies = pd.get_dummies(df.Car_Name)
dummies
```

Out[4]:

	Duster	Toyota Corona	Valiant	Volvo 142E
0	0	0	1	0
1	0	0	1	0
2	0	0	1	0

3	0	0	1	0
4	0	0	1	0
5	0	0	1	0
6	1	0	0	0
7	1	0	0	0
8	1	0	0	0
9	1	0	0	0
10	1	0	0	0
11	1	0	0	0
12	1	0	0	0
13	0	1	0	0
14	0	1	0	0
15	0	1	0	0
16	0	1	0	0
17	0	1	0	0
18	0	1	0	0
19	0	1	0	0
20	0	1	0	0
21	0	0	0	1
22	0	0	0	1
23	0	0	0	1
24	0	0	0	1
25	0	0	0	1
26	0	0	0	1
27	0	0	0	1
28	0	0	0	1
29	0	0	0	1
30	0	0	0	1
31	0	0	0	1

In [5]:

```
merged = pd.concat([df,dummies],axis='columns')
merged
```

Out[5]:

	Car_Name	mpg	cyl	disp	hp	drat	wt	qsec	Duster	Toyota Corona	Valiant	Volvo 142E
0	Valiant	21.0	6	160.0	110	3.90	2.620	16.46	0	0	1	0
1	Valiant	21.0	6	160.0	110	3.90	2.875	17.02	0	0	1	0
2	Valiant	22.8	4	108.0	93	3.85	2.320	18.61	0	0	1	0
3	Valiant	21.4	6	258.0	110	3.08	3.215	19.44	0	0	1	0
4	Valiant	18.7	8	360.0	175	3.15	3.440	17.02	0	0	1	0
5	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	0	0	1	0
6	Duster	14.3	8	360.0	245	3.21	3.570	15.84	1	0	0	0
7	Duster	24.4	4	146.7	62	3.69	3.190	20.00	1	0	0	0
8	Duster	22.8	4	140.8	95	3.92	3.150	22.90	1	0	0	0
9	Duster	19.2	6	167.6	123	3.92	3.440	18.30	1	0	0	0
10	Duster	17.8	6	167.6	123	3.92	3.440	18.90	1	0	0	0
11	Duster	16.4	8	275.8	180	3.07	4.070	17.40	1	0	0	0
12	Duster	17.3	8	275.8	180	3.07	3.730	17.60	1	0	0	0
13	Toyota Corona	15.2	8	275.8	180	3.07	3.780	18.00	0	1	0	0
14	Toyota Corona	10.4	8	472.0	205	2.93	5.250	17.98	0	1	0	0
15	Toyota Corona	10.4	8	460.0	215	3.00	5.424	17.82	0	1	0	0
16	Toyota Corona	14.7	8	440.0	230	3.23	5.345	17.42	0	1	0	0
17	Toyota Corona	32.4	4	78.7	66	4.08	2.200	19.47	0	1	0	0
18	Toyota Corona	30.4	4	75.7	52	4.93	1.615	18.52	0	1	0	0
19	Toyota Corona	33.9	4	71.1	65	4.22	1.835	19.90	0	1	0	0
20	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	0	1	0	0

21	Volvo 142E	15.5	8	318.0	150	2.76	3.520	16.87	0	0	0	1
22	Volvo 142E	15.2	8	304.0	150	3.15	3.435	17.30	0	0	0	1
23	Volvo 142E	13.3	8	350.0	245	3.73	3.840	15.41	0	0	0	1
24	Volvo 142E	19.2	8	400.0	175	3.08	3.845	17.05	0	0	0	1
25	Volvo 142E	27.3	4	79.0	66	4.08	1.935	18.90	0	0	0	1
26	Volvo 142E	26.0	4	120.3	91	4.43	2.140	16.70	0	0	0	1
27	Volvo 142E	30.4	4	95.1	113	3.77	1.513	16.90	0	0	0	1
28	Volvo 142E	15.8	8	351.0	264	4.22	3.170	14.50	0	0	0	1
29	Volvo 142E	19.7	6	145.0	175	3.62	2.770	15.50	0	0	0	1
30	Volvo 142E	15.0	8	301.0	335	3.54	3.570	14.60	0	0	0	1
31	Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	0	0	0	1

2. Label Encoding

In [7]:
```
df1 = pd.read_csv("Tips.csv")
df1
```

Out[7]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [8]:
```
from sklearn.preprocessing import LabelEncoder
```

In [16]:
```
le = LabelEncoder()
label = le.fit_transform(df1.sex)
```

In [17]:
```
label
```

Out[17]:
```
array([0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 0])
```

In [18]:
```
le.classes_
```

Out[18]:
```
array(['Female', 'Male'], dtype=object)
```

In [19]:
```
Data = df1.drop("sex", axis ='columns')
```

In [20]:
```
Data
```

Out[20]:

	total_bill	tip	smoker	day	time	size
0	16.99	1.01	No	Sun	Dinner	2
1	10.34	1.66	No	Sun	Dinner	3
2	21.01	3.50	No	Sun	Dinner	3
3	23.68	3.31	No	Sun	Dinner	2
4	24.59	3.61	No	Sun	Dinner	4
...	...	...	...	...	...	...
239	29.03	5.92	No	Sat	Dinner	3
240	27.18	2.00	Yes	Sat	Dinner	2
241	22.67	2.00	Yes	Sat	Dinner	2
242	17.82	1.75	No	Sat	Dinner	2
243	18.78	3.00	No	Thur	Dinner	2

244 rows × 6 columns

In [21]:
```
Data["Sex"] = label
```

In [22]:
```
Data
```

Out[22]:

	total_bill	tip	smoker	day	time	size	Sex
0	16.99	1.01	No	Sun	Dinner	2	0
1	10.34	1.66	No	Sun	Dinner	3	1
2	21.01	3.50	No	Sun	Dinner	3	1
3	23.68	3.31	No	Sun	Dinner	2	1
4	24.59	3.61	No	Sun	Dinner	4	0
...	...	...	...	...	...	...	...
239	29.03	5.92	No	Sat	Dinner	3	1
240	27.18	2.00	Yes	Sat	Dinner	2	0
241	22.67	2.00	Yes	Sat	Dinner	2	1
242	17.82	1.75	No	Sat	Dinner	2	1
243	18.78	3.00	No	Thur	Dinner	2	0

244 rows × 7 columns

3. ordinal encoding:

In [23]:
```
from sklearn.preprocessing import OrdinalEncoder
```

In [43]:
```
Grades = ['K', 'I', 'G', 'L', 'H', 'G', 'H', 'I', 'K'] # mixed to see effect
SRs = ['SR1', 'SR9', 'SR2', 'SR8', 'SR3', 'SR7', 'SR4', 'SR6', 'SR5'] # mixed to see effect
reating = ['Low', 'High', 'Low', 'High', 'Low', 'Med', 'Low', 'High', 'Med']
df2 = pd.DataFrame({'grades':Grades,
                    'ranks':SRs,
                    'Reating':reating
                   })
```

In [44]:
```
df2
```

Out[44]:

	grades	ranks	Reating
0	K	SR1	Low

1	I	SR9	High
2	G	SR2	Low
3	L	SR8	High
4	H	SR3	Low
5	G	SR7	Med
6	H	SR4	Low
7	I	SR6	High
8	K	SR5	Med

In [35]:
```
df2['Reating'].unique()
```

Out[35]:
```
array(['Low', 'High', 'Med'], dtype=object)
```

In [36]:
```
enc = OrdinalEncoder()
```

In [38]:
```
enc.fit_transform(df2[['Reating']])
```

Out[38]:
```
array([[1.],
       [0.],
       [1.],
       [0.],
       [1.],
       [2.],
       [1.],
       [0.],
       [2.]])
```

In [39]:
```
df2[['Reating']] = enc.fit_transform(df2[['Reating']])
```

In [40]:
```
df2
```

Out[40]:

	grades	ranks	Reating
0	K	SR1	1.0
1	I	SR9	0.0
2	G	SR2	1.0
3	L	SR8	0.0
4	H	SR3	1.0
5	G	SR7	2.0
6	H	SR4	1.0
7	I	SR6	0.0
8	K	SR5	2.0

OR

In [41]:
```
Reating = ['Low', 'High', 'Med']
```

In [42]:
```
enc = OrdinalEncoder(categories=[Reating])
```

In [49]:
```
df2[['Reating']] = enc.fit_transform(df2[['Reating']])
```

In [50]:
```
df2
```

Out[50]:

	grades	ranks	Reating
0	K	SR1	0.0

1	I	SR9	1.0
2	G	SR2	0.0
3	L	SR8	1.0
4	H	SR3	0.0
5	G	SR7	2.0
6	H	SR4	0.0
7	I	SR6	1.0
8	K	SR5	2.0

4. Mean encoding

In [51]:
```
data={'SubjectName':['s1','s2','s3','s1','s4','s3','s2','s1','s2','s4','s1'],
      'Target':[1,0,1,1,1,0,0,1,1,1,0]}

df3 = pd.DataFrame(data)

print(df3)
```

```
   SubjectName  Target
0           s1       1
1           s2       0
2           s3       1
3           s1       1
4           s4       1
5           s3       0
6           s2       0
7           s1       1
8           s2       1
9           s4       1
10          s1       0
```

In [53]:
```
df3.groupby(['SubjectName'])['Target'].count()
```

Out[53]:
```
SubjectName
s1    4
s2    3
s3    2
s4    2
Name: Target, dtype: int64
```

In [54]:
```
df3.groupby(['SubjectName'])['Target'].mean()
```

Out[54]:
```
SubjectName
s1    0.750000
s2    0.333333
s3    0.500000
s4    1.000000
Name: Target, dtype: float64
```

In [58]:
```
Mean_encoded_subject = df3.groupby(['SubjectName'])['Target'].mean().to_dict()
```

In [59]:
```
Mean_encoded_subject
```

Out[59]:
```
{0.3333333333333333: 0.3333333333333333, 0.5: 0.5, 0.75: 0.75, 1.0: 1.0}
```

In [60]:
```
df3['SubjectName'] = df3['SubjectName'].map(Mean_encoded_subject)

print(df3)
```

```
    SubjectName  Target
0      0.750000       1
```

```
1      0.333333       0
2      0.500000       1
3      0.750000       1
4      1.000000       1
5      0.500000       0
6      0.333333       0
7      0.750000       1
8      0.333333       1
9      1.000000       1
10     0.750000       0
```

5. Probability ratio encoding

In [66]:
```
data={'SubjectName':['s1','s2','s3','s1','s4','s3','s2','s1','s2','s4','s1'],
      'Target':[1,0,1,1,1,0,0,1,1,1,0]}

df4 = pd.DataFrame(data)

print(df4)
```

```
   SubjectName  Target
0           s1       1
1           s2       0
2           s3       1
3           s1       1
4           s4       1
5           s3       0
6           s2       0
7           s1       1
8           s2       1
9           s4       1
10          s1       0
```

In [73]:
```
df4=df4.groupby(['SubjectName'])['Target'].mean()
```

In [74]:
```
df4=pd.DataFrame(df4)
```

In [80]:
```
df4
```

Out[80]:

	Target
SubjectName	
s1	0.750000
s2	0.333333
s3	0.500000
s4	1.000000

In [81]:
```
df4['Non_Target'] = 1 - df4['Target']
```

In [90]:
```
df4.reset_index()
```

Out[90]:

	SubjectName	Target	Non_Target	ratio
0	s1	0.750000	0.250000	3.0
1	s2	0.333333	0.666667	0.5
2	s3	0.500000	0.500000	1.0
3	s4	1.000000	0.000000	inf

In [91]:
```
df4['ratio'] = df4['Target'] / df4['Non_Target']
```

In [92]:
```
ratio_mapping = df4['ratio'].to_dict()
```

In [93]:
```
ratio_mapping
```

Out[93]:
```
{'s1': 3.0, 's2': 0.49999999999999994, 's3': 1.0, 's4': inf}
```

In [100…
```
df4['SubjectName'] = df4['SubjectName'].map(ratio_mapping)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3079             try:
-> 3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'SubjectName'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
<ipython-input-100-5c8396d65861> in <module>
----> 1 df4['SubjectName'] = df4['SubjectName'].map(ratio_mapping)

C:\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   3022             if self.columns.nlevels > 1:
   3023                 return self._getitem_multilevel(key)
-> 3024             indexer = self.columns.get_loc(key)
   3025             if is_integer(indexer):
   3026                 indexer = [indexer]

C:\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:
-> 3082                 raise KeyError(key) from err
   3083
   3084         if tolerance is not None:

KeyError: 'SubjectName'
```

In [ ]:
```
```