Types of Encoding : Categorical variable

1.Nominal Encoding

2.Ordinal Encoding(Rank)

Example-One-Hot Encoding

In [1]:

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
df_country = pd.DataFrame({'country': ['russia', 'germany', 'australia','korea','germany','
```

In [2]:

```
len(df_country['country'].unique())
```

Out[2]:

```
11
```

In [3]:

```
pd.get_dummies(df_country["country"],drop_first=True)
```

Out[3]:

	Croatia	Denmark	France	Luxembourg	Netherlands	Switzerland	australia	germany	kore
0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	1	
2	0	0	0	0	0	0	1	0	
3	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	1	
5	0	0	0	1	0	0	0	0	
6	0	0	0	0	1	0	0	0	
7	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	1	0	
9	0	0	0	1	0	0	0	0	
10	0	1	0	0	0	0	0	0	
11	0	0	0	0	0	1	0	0	
12	0	0	0	0	1	0	0	0	
13	0	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	1	0	
15	0	0	0	1	0	0	0	0	
16	0	1	0	0	0	0	0	0	
17	0	0	0	0	0	1	0	0	
18	1	0	0	0	0	0	0	0	
19	0	0	1	0	0	0	0	0	
20	1	0	0	0	0	0	0	0	

In [4]:

```
df_titanic=pd.read_csv('titanic.csv')
```

In [5]:

```
df_titanic['Embarked'].unique()
```

Out[5]:

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

In [6]:

```
df_titanic=df_titanic.dropna()
```

In [7]:

```
df_titanic['Embarked'].unique()
```

Out[7]:

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

In [8]:

```
pd.get_dummies(df_titanic['Embarked'],prefix='Embarked',drop_first=True).head()
```

Out[8]:

	Embarked_Q	Embarked_S
1	0	0
3	0	1
6	0	1
10	0	1
11	0	1

One-Hot Encoding with many categorical-Ensemble Selection

In [9]:

```
df_mercedes=pd.read_csv('mercedes.csv',usecols=["X0","X1","X2","X3","X4","X5","X6"])
```

In [10]:

```
df_mercedes
```

Out[10]:

	X0	X1	X2	X3	X4	X5	X6
0	k	v	at	a	d	u	j
1	k	t	av	e	d	y	l
2	az	w	n	c	d	x	j
3	az	t	n	f	d	x	l
4	az	v	n	f	d	h	d
...	...	...	...	...	...	...	...
4204	ak	s	as	c	d	aa	d
4205	j	o	t	d	d	aa	h
4206	ak	v	r	a	d	aa	g
4207	al	r	e	f	d	aa	l
4208	z	r	ae	c	d	aa	g

4209 rows × 7 columns

In [11]:

```
for i in df_mercedes.columns:
    print(len(df_mercedes[i].unique()))
```

```
47
27
44
7
4
29
12
```

In [12]:

```
df_mercedes.X1.value_counts().sort_values(ascending=False).head(10)
```

Out[12]:

```
aa    833
s     598
b     592
l     590
v     408
r     251
i     203
a     143
c     121
o      82
Name: X1, dtype: int64
```

In [13]:

```
lst_10=df_mercedes.X1.value_counts().sort_values(ascending=False).head(10).index
lst_10=list(lst_10)
```

In [14]:

```
lst_10
```

Out[14]:

```
['aa', 's', 'b', 'l', 'v', 'r', 'i', 'a', 'c', 'o']
```

In [15]:

```
for categories in lst_10:
    df_mercedes[categories]=np.where(df_mercedes['X1']==categories,1,0)
```

In [16]:

```
lst_10.append('X1')
```

In [17]:

```
df_mercedes[lst_10]
```

Out[17]:

	aa	s	b	l	v	r	i	a	c	o	X1
0	0	0	0	0	1	0	0	0	0	0	v
1	0	0	0	0	0	0	0	0	0	0	t
2	0	0	0	0	0	0	0	0	0	0	w
3	0	0	0	0	0	0	0	0	0	0	t
4	0	0	0	0	1	0	0	0	0	0	v
...	...	...	...	...	...	...	...	...	...	...	...
4204	0	1	0	0	0	0	0	0	0	0	s
4205	0	0	0	0	0	0	0	0	0	1	o
4206	0	0	0	0	1	0	0	0	0	0	v
4207	0	0	0	0	0	1	0	0	0	0	r
4208	0	0	0	0	0	1	0	0	0	0	r

4209 rows × 11 columns

Target/Mean Encoding

In [18]:

```
#! pip install category-encoders
from category_encoders import TargetEncoder
```

In [19]:

```
df1_titanic=pd.read_csv('titanic.csv')
```

In [20]:

```
df1_titanic=df1_titanic.dropna()
```

In [21]:

```
cols=['Sex','Embarked']
target='Survived'
for i in cols:
    te=TargetEncoder()
    te.fit(X=df1_titanic[i],y=df1_titanic[target])
    new_col=te.transform(df1_titanic[i])
    df1_titanic=pd.concat([df1_titanic,new_col],axis=1)
```

In [22]:

```
df1_titanic.head(10)
```

Out[22]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000
11	12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.0000
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000
52	53	1	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	female	49.0	1	0	PC 17572	76.7292
54	55	0	1	Ostby, Mr. Engelhart Cornelius	male	65.0	0	1	113509	61.9792

In [23]:

```
df1_titanic.tail()
```

Out[23]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000

Effect Encoding

In [24]:

```
import category_encoders as ce
```

In [25]:

```
df_state=pd.DataFrame({'state':['West Bengal','Assam','Telangana','Bihar','Punjab','Madhya
```

In [26]:

```
df_state
```

Out[26]:

	state
0	West Bengal
1	Assam
2	Telangana
3	Bihar
4	Punjab
5	Madhya Pradesh

In [27]:

```
encoder=ce.sum_coding.SumEncoder(cols='state')
```

In [28]:

```
encoder.fit_transform(df_state)

##For Further redaing oogle.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwicvY_soe
## Categorical Variables in Regression Analysis:A Comparison of Dummy and Effect Coding uss
```

Out[28]:

	intercept	state_0	state_1	state_2	state_3	state_4
0	1	1.0	0.0	0.0	0.0	0.0
1	1	0.0	1.0	0.0	0.0	0.0
2	1	0.0	0.0	1.0	0.0	0.0
3	1	0.0	0.0	0.0	1.0	0.0
4	1	0.0	0.0	0.0	0.0	1.0
5	1	-1.0	-1.0	-1.0	-1.0	-1.0

Advantages and disadvantages of Effect Encoding

- Effect coding is appropriate when each group is compared with the entire set of groups rather than with a reference group. In other words, effect coding is useful in testing the effect of a treatment defined as the deviation between the treatment mean and the grand mean. However, to determine which means differ significantly from each other, one of the methods for multiple comparisons of means has to be applied

Frequency Encoding

In [29]:

```
df2_titanic=pd.read_csv('titanic.csv')
```

In [30]:

```
df2_titanic=df2_titanic.dropna()
```

In [31]:

```
col_Embarked = (df2_titanic.groupby('Embarked').size()) / len(df2_titanic)
col_Embarked
```

Out[31]:

```
Embarked
C    0.355191
Q    0.010929
S    0.633880
dtype: float64
```

In [32]:

```
df2_titanic['col_Embarked'] = df2_titanic['Embarked'].apply(lambda x : col_Embarked[x])
```

In [33]:

```
df2_titanic.tail()
```

Out[33]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542
872	873	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000

Hash Encoding

In [34]:

```
import seaborn as sns
df_mpg = sns.load_dataset('mpg')
```

In [35]:

```
df_mpg
```

Out[35]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	n
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chev che m
1	15.0	8	350.0	165.0	3693	11.5	70	usa	t sk
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plym sat
3	16.0	8	304.0	150.0	3433	12.0	70	usa	rebe
4	17.0	8	302.0	140.0	3449	10.5	70	usa	t
...	...	...	...	...	...	...	...	...	
393	27.0	4	140.0	86.0	2790	15.6	82	usa	mus
394	44.0	4	97.0	52.0	2130	24.6	82	europe	pi
395	32.0	4	135.0	84.0	2295	11.6	82	usa	d ramp
396	28.0	4	120.0	79.0	2625	18.6	82	usa	ra
397	31.0	4	119.0	82.0	2720	19.4	82	usa	che

398 rows × 9 columns

In [36]:

```
hash_encoder=ce.HashingEncoder(cols='model_year',n_components=6)
hash_res = hash_encoder.fit_transform(df_mpg)
hash_res.head(10)
```

Out[36]:

	col_0	col_1	col_2	col_3	col_4	col_5	mpg	cylinders	displacement	horsepower	weight
0	0	0	0	0	0	1	18.0	8	307.0	130.0	3504
1	0	0	0	0	0	1	15.0	8	350.0	165.0	3693
2	0	0	0	0	0	1	18.0	8	318.0	150.0	3436
3	0	0	0	0	0	1	16.0	8	304.0	150.0	3433
4	0	0	0	0	0	1	17.0	8	302.0	140.0	3449
5	0	0	0	0	0	1	15.0	8	429.0	198.0	4341
6	0	0	0	0	0	1	14.0	8	454.0	220.0	4354
7	0	0	0	0	0	1	14.0	8	440.0	215.0	4312
8	0	0	0	0	0	1	14.0	8	455.0	225.0	4425
9	0	0	0	0	0	1	15.0	8	390.0	190.0	3850

In [37]:

```
pd.concat([hash_encoder.fit_transform(df_mpg['model_year']), df_mpg], axis =1).sample(5)
```

Out[37]:

	col_0	col_1	col_2	col_3	col_4	col_5	mpg	cylinders	displacement	horsepower	weig
89	0	0	0	1	0	0	15.0	8	318.0	150.0	377
48	1	0	0	0	0	0	18.0	6	250.0	88.0	31:
78	0	0	0	1	0	0	21.0	4	120.0	87.0	297
111	0	0	0	1	0	0	18.0	3	70.0	90.0	21:
381	0	0	0	0	1	0	36.0	4	107.0	75.0	22(

One-Hot Encoding's major weakness is the features it produced are equivalent to the categorical cardinal, which causes dimensionality issues when the cardinality is too high.One way to alleviate this problem is to represent the categorical data into a lesser number of columns, and that is what Hash Encoding did.

- Hash Encoding represents the categorical data into numerical value by the hashing function.the Hashing encoder uses the md5 hashing algorithm but a user can pass any algorithm of his choice.

Advantage:

- The main advantage of using Hash Encoding is that you can control the number of numerical columns produced by the process. You could represent categorical data with 25 or 50 values with five columns (or any number you want)

Disadvantage:

- Since Hashing transforms the data in lesser dimensions, it may lead to loss of information. Another issue faced by hashing encoder is the collision. Since here, a large number of features are depicted into lesser dimensions, hence multiple values can be represented by the same hash value, this is known as a collision.

Leave One Out Encoding (LOOE)

Leave One Out Encoding is similar to Target Encoding, but it adds one more step to handle overfitting. Leave One Out Encoding has an exact approach with the Target Encoding except that LOOE excludes the current row's target in the calculation to ease the outlier effect. This means the calculation result between the True and False classes of the target could be different for each class. LOOE function could also introduce the Gaussian noise distribution to decrease overfitting.

In [38]:

```
titanic = sns.load_dataset('titanic')
#Passing value 0.1 at sigma parameter to introduce noise
loo_encoder=ce.LeaveOneOutEncoder(cols='pclass', sigma = 0.1)
loo_res = loo_encoder.fit_transform(titanic['pclass'], titanic['survived']).rename(columns
```

In [39]:

```
pd.concat([loo_res,titanic], axis =1).sample(5)
```

Out[39]:

	loo_pclass	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	wh
81	0.198501	1	3	male	29.0	0	0	9.5000	S	Third	ma
166	0.667729	1	1	female	NaN	0	1	55.0000	S	First	woma
500	0.226742	0	3	male	17.0	0	0	8.6625	S	Third	ma
57	0.212872	0	3	male	28.5	0	0	7.2292	C	Third	ma
746	0.243138	0	3	male	16.0	1	1	20.2500	S	Third	ma

Weight of Evidence Encoding

In [40]:

```
from category_encoders import WOEEncoder
```

In [41]:

```
WOEE = WOEEncoder(cols=['embarked'],regularization=0.5)
titanic['Type_Embarked']=WOEE.fit_transform(titanic['embarked'],titanic['survived'])
titanic[['embarked','Type_Embarked','survived']]
```

Out[41]:

	embarked	Type_Embarked	survived
0	S	-0.203568	0
1	C	0.686017	1
2	S	-0.203568	1
3	S	-0.203568	1
4	S	-0.203568	0
...	...	...	...
886	S	-0.203568	0
887	S	-0.203568	1
888	S	-0.203568	0
889	C	0.686017	1
890	Q	0.029185	0

891 rows × 3 columns

Weight of Evidence Encoding (WoE) is a measure of how much the evidence supports or undermines a hypothesis.

Advantage:

- Work well with logistic regression since WoE transformation has the same logistic scale. Can use WoE to compare across feature since their values are standardized.

Disadvantages:

- May lose information due to some category may have the same WoE,Does not take into account features correlation Overfitting

Label Encoding

In [42]:

```
label_df = pd.DataFrame({'Business unit': ['A', 'A', 'B', 'B', 'B', 'B', 'C', 'C'],
                   'Experience': [5,6,7,5,5,6,8,9]})


print(label_df)
```

```
  Business unit  Experience
0             A           5
1             A           6
2             B           7
3             B           5
4             B           5
5             B           6
6             C           8
7             C           9
```

In [43]:

```
from sklearn.preprocessing import LabelEncoder
```

In [44]:

```
lab = LabelEncoder()
```

In [45]:

```
label_df['Business unit'] = lab.fit_transform(label_df['Business unit'])
```

In [46]:

```
print(label_df)
```

```
   Business unit  Experience
0              0           5
1              0           6
2              1           7
3              1           5
4              1           5
5              1           6
6              2           8
7              2           9
```

In [ ]: