Ordinal Number Encoding

In [1]:

```
import datetime
```

In [2]:

```
today_date=datetime.datetime.today()
```

In [3]:

```
today_date
```

Out[3]:

```
datetime.datetime(2020, 7, 16, 19, 13, 59, 571679)
```

In [6]:

```
today_date-datetime.timedelta(3)
```

Out[6]:

```
datetime.datetime(2020, 7, 13, 19, 13, 59, 571679)
```

In [9]:

```
#### List Comprehension
days=[today_date-datetime.timedelta(x) for x in range(0,15)]
```

In [11]:

```
import pandas as pd
data=pd.DataFrame(days)
data.columns=["Day"]
```

In [12]:

```
data.head()
```

Out[12]:

	Day
0	2020-07-16 19:13:59.571679
1	2020-07-15 19:13:59.571679
2	2020-07-14 19:13:59.571679
3	2020-07-13 19:13:59.571679
4	2020-07-12 19:13:59.571679

In [22]:

```
data['weekday']=data['Day'].dt.weekday_name
data.head()
```

Out[22]:

	Day	weekday
0	2020-07-16 19:13:59.571679	Thursday
1	2020-07-15 19:13:59.571679	Wednesday
2	2020-07-14 19:13:59.571679	Tuesday
3	2020-07-13 19:13:59.571679	Monday
4	2020-07-12 19:13:59.571679	Sunday

In [23]:

```
dictionary={'Monday':1,'Tuesday':2,'Wednesday':3,'Thursday':4,'Friday':5,'Saturday':6,'Sund


}
```

In [24]:

```
dictionary
```

Out[24]:

```
{'Monday': 1,
 'Tuesday': 2,
 'Wednesday': 3,
 'Thursday': 4,
 'Friday': 5,
 'Saturday': 6,
 'Sunday': 7}
```

In [26]:

```
data['weekday_ordinal']=data['weekday'].map(dictionary)
```

In [27]:

```
data
```

Out[27]:

	Day	weekday	weekday_ordinal
0	2020-07-16 19:13:59.571679	Thursday	4
1	2020-07-15 19:13:59.571679	Wednesday	3
2	2020-07-14 19:13:59.571679	Tuesday	2
3	2020-07-13 19:13:59.571679	Monday	1
4	2020-07-12 19:13:59.571679	Sunday	7
5	2020-07-11 19:13:59.571679	Saturday	6
6	2020-07-10 19:13:59.571679	Friday	5
7	2020-07-09 19:13:59.571679	Thursday	4
8	2020-07-08 19:13:59.571679	Wednesday	3
9	2020-07-07 19:13:59.571679	Tuesday	2
10	2020-07-06 19:13:59.571679	Monday	1
11	2020-07-05 19:13:59.571679	Sunday	7
12	2020-07-04 19:13:59.571679	Saturday	6
13	2020-07-03 19:13:59.571679	Friday	5
14	2020-07-02 19:13:59.571679	Thursday	4

Count Or Frequency Encoding

In [46]:

```
train_set = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adu
train_set.head()
```

Out[46]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0

In [47]:

```
columns=[1,3,5,6,7,8,9,13]
```

In [48]:

```
train_set=train_set[columns]
```

In [49]:

```
train_set.columns=['Employment','Degree','Status','Designation','family_job','Race','Sex','
```

In [50]:

```
train_set.head()
```

Out[50]:

	Employment	Degree	Status	Designation	family_job	Race	Sex	Country
0	State-gov	Bachelors	Never-married	Adm-clerical	Not-in-family	White	Male	United-States
1	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	United-States
2	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	United-States
3	Private	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	United-States
4	Private	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	Cuba

In [51]:

```
for feature in train_set.columns[:]:
    print(feature,":",len(train_set[feature].unique()),'labels')
```

```
Employment : 9 labels
Degree : 16 labels
Status : 7 labels
Designation : 15 labels
family_job : 6 labels
Race : 5 labels
Sex : 2 labels
Country : 42 labels
```

In [52]:

```
country_map=train_set['Country'].value_counts().to_dict()
```

In [53]:

```
train_set['Country']=train_set['Country'].map(country_map)
train_set.head(20)
```

Out[53]:

	Employment	Degree	Status	Designation	family_job	Race	Sex	Country
0	State-gov	Bachelors	Never-married	Adm-clerical	Not-in-family	White	Male	29170
1	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	29170
2	Private	HS-grad	Divorced	Handlers-cleaners	Not-in-family	White	Male	29170
3	Private	11th	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	29170
4	Private	Bachelors	Married-civ-spouse	Prof-specialty	Wife	Black	Female	95
5	Private	Masters	Married-civ-spouse	Exec-managerial	Wife	White	Female	29170
6	Private	9th	Married-spouse-absent	Other-service	Not-in-family	Black	Female	81
7	Self-emp-not-inc	HS-grad	Married-civ-spouse	Exec-managerial	Husband	White	Male	29170
8	Private	Masters	Never-married	Prof-specialty	Not-in-family	White	Female	29170
9	Private	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	29170
10	Private	Some-college	Married-civ-spouse	Exec-managerial	Husband	Black	Male	29170
11	State-gov	Bachelors	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	100
12	Private	Bachelors	Never-married	Adm-clerical	Own-child	White	Female	29170
13	Private	Assoc-acdm	Never-married	Sales	Not-in-family	Black	Male	29170
14	Private	Assoc-voc	Married-civ-spouse	Craft-repair	Husband	Asian-Pac-Islander	Male	583
15	Private	7th-8th	Married-civ-spouse	Transport-moving	Husband	Amer-Indian-Eskimo	Male	643
16	Self-emp-not-inc	HS-grad	Never-married	Farming-fishing	Own-child	White	Male	29170
17	Private	HS-grad	Never-married	Machine-op-inspct	Unmarried	White	Male	29170
18	Private	11th	Married-civ-spouse	Sales	Husband	White	Male	29170
19	Self-emp-not-inc	Masters	Divorced	Exec-managerial	Unmarried	White	Female	29170

Advantages

1. Easy To Use
2. Not increasing feature space

   Disadvantages
3. It will provide same weight if the frequencies are same

Target Guided Ordinal Encoding

1. Ordering the labels according to the target
2. Replace the labels by the joint probability of being 1 or 0

In [55]:

```
import pandas as pd
df=pd.read_csv('titanic.csv', usecols=['Cabin','Survived'])
df.head()
```

Out[55]:

	Survived	Cabin
0	0	NaN
1	1	C85
2	1	NaN
3	1	C123
4	0	NaN

In [56]:

```
df['Cabin'].fillna('Missing',inplace=True)
```

In [61]:

```
df['Cabin']=df['Cabin'].astype(str).str[0]
```

In [62]:

```
df.head()
```

Out[62]:

	Survived	Cabin
0	0	M
1	1	C
2	1	M
3	1	C
4	0	M

In [63]:

```
df.Cabin.unique()
```

Out[63]:

```
array(['M', 'C', 'E', 'G', 'D', 'A', 'B', 'F', 'T'], dtype=object)
```

In [64]:

```
df.groupby(['Cabin'])['Survived'].mean()
```

Out[64]:

```
Cabin
A    0.466667
B    0.744681
C    0.593220
D    0.757576
E    0.750000
F    0.615385
G    0.500000
M    0.299854
T    0.000000
Name: Survived, dtype: float64
```

In [68]:

```
df.groupby(['Cabin'])['Survived'].mean().sort_values().index
```

Out[68]:

```
Index(['T', 'M', 'A', 'G', 'C', 'F', 'B', 'E', 'D'], dtype='object', name='C
abin')
```

In [69]:

```
ordinal_labels=df.groupby(['Cabin'])['Survived'].mean().sort_values().index
ordinal_labels
```

Out[69]:

```
Index(['T', 'M', 'A', 'G', 'C', 'F', 'B', 'E', 'D'], dtype='object', name='C
abin')
```

In [70]:

```
enumerate(ordinal_labels,0)
```

Out[70]:

```
<enumerate at 0x24e376166c0>
```

In [71]:

```
ordinal_labels2={k:i for i,k in enumerate(ordinal_labels,0)}
ordinal_labels2
```

Out[71]:

```
{'T': 0, 'M': 1, 'A': 2, 'G': 3, 'C': 4, 'F': 5, 'B': 6, 'E': 7, 'D': 8}
```

In [72]:

```
df['Cabin_ordinal_labels']=df['Cabin'].map(ordinal_labels2)
df.head()
```

Out[72]:

	Survived	Cabin	Cabin_ordinal_labels
0	0	M	1
1	1	C	4
2	1	M	1
3	1	C	4
4	0	M	1

Mean Encoding

In [75]:

```
mean_ordinal=df.groupby(['Cabin'])['Survived'].mean().to_dict()
```

In [76]:

```
mean_ordinal
```

Out[76]:

```
{'A': 0.4666666666666667,
 'B': 0.7446808510638298,
 'C': 0.5932203389830508,
 'D': 0.7575757575757576,
 'E': 0.75,
 'F': 0.6153846153846154,
 'G': 0.5,
 'M': 0.29985443959243085,
 'T': 0.0}
```

In [77]:

```
df['mean_ordinal_encode']=df['Cabin'].map(mean_ordinal)
df.head()
```

Out[77]:

	Survived	Cabin	Cabin_ordinal_labels	mean_ordinal_encode
0	0	M	1	0.299854
1	1	C	4	0.593220
2	1	M	1	0.299854
3	1	C	4	0.593220
4	0	M	1	0.299854

In [1]:

```
pwd
```

Out[1]:

```
'C:\\Users\\91920\\Machine Learning\\EDA'
```

In [ ]:

```

```