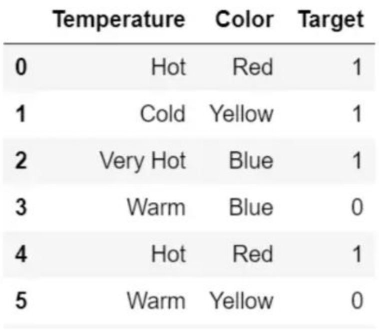
One hot encoder:

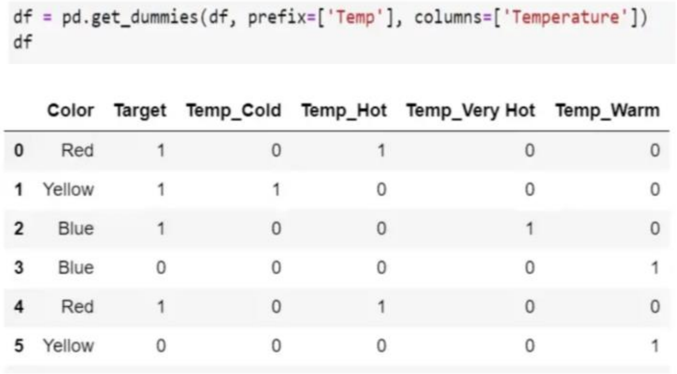
It maps each category to a vector that contains 1 and 0, denoting the presence or absence of the feature.

each category value is converted into a new column and assigned a 1 or 0 (notation for true/false) value to the column.

Example:



	Temperature	Color	Target
0	Hot	Red	1
1	Cold	Yellow	1
2	Very Hot	Blue	1
3	Warm	Blue	0
4	Hot	Red	1
5	Warm	Yellow	0



```
df = pd.get_dummies(df, prefix=['Temp'], columns=['Temperature'])
df
```

	Color	Target	Temp_Cold	Temp_Hot	Temp_Very Hot	Temp_Warm
0	Red	1	0	1	0	0
1	Yellow	1	1	0	0	0
2	Blue	1	0	0	1	0
3	Blue	0	0	0	0	1
4	Red	1	0	1	0	0
5	Yellow	0	0	0	0	1

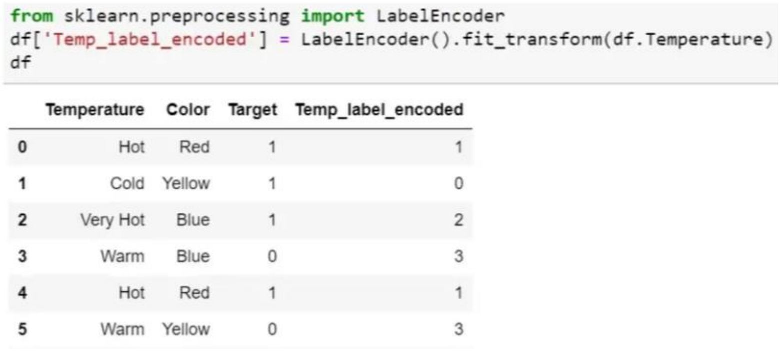
Limitations:

For a feature having a large number of unique feature values or categories, one-hot encoding is not a great choice. For example, time-based features such as day of month,day of weak , etc have a cyclic nature and have many feature values. One-hot encoding day of month feature results in 30 dimensionality vector, day of year results in 366 dimension vector.

Label encoder

In this method, each category is assigned a value from 1 through N where N is the number of categories for the feature.

Example:



```
from sklearn.preprocessing import LabelEncoder
df['Temp_label_encoded'] = LabelEncoder().fit_transform(df.Temperature)
df
```

	Temperature	Color	Target	Temp_label_encoded
0	Hot	Red	1	1
1	Cold	Yellow	1	0
2	Very Hot	Blue	1	2
3	Warm	Blue	0	3
4	Hot	Red	1	1
5	Warm	Yellow	0	3

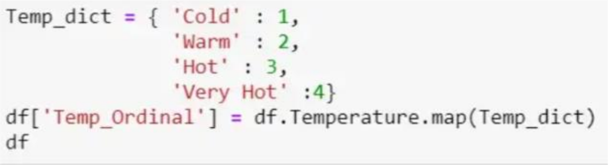
Limitations :

Label encoding assigns a unique number(starting from 0) to each class of data. This may lead to the generation of priority issues in the training of data sets. A label with a high value may be considered to have high priority than a label having a lower value.

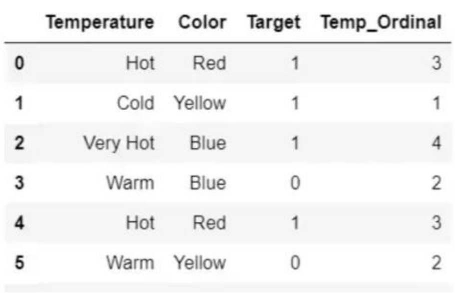
Ordinal encoding:

Ordinal encoding ensures that the encoding of variables retains the ordinal nature of the variable.

Example :



```
Temp_dict = { 'Cold' : 1,
              'Warm' : 2,
              'Hot' : 3,
              'Very Hot' :4}
df['Temp_Ordinal'] = df.Temperature.map(Temp_dict)
df
```



	Temperature	Color	Target	Temp_Ordinal
0	Hot	Red	1	3
1	Cold	Yellow	1	1
2	Very Hot	Blue	1	4
3	Warm	Blue	0	2
4	Hot	Red	1	3
5	Warm	Yellow	0	2

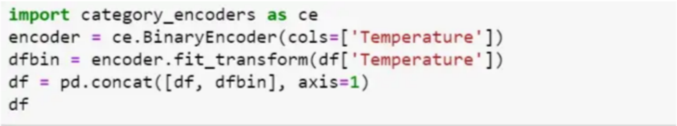
Limitations :

Ordinal encoder should not be used if your data has no meaningful order.

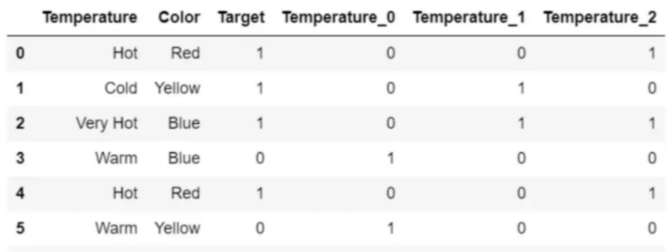
Binary encoder:

Binary encoding is an encoding technique to transform an original categorical variable to a numerical variable by encoding the categories as Integer and then converted into binary code.

Example:



```
import category_encoders as ce
encoder = ce.BinaryEncoder(cols=['Temperature'])
dfbin = encoder.fit_transform(df['Temperature'])
df = pd.concat([df, dfbin], axis=1)
df
```

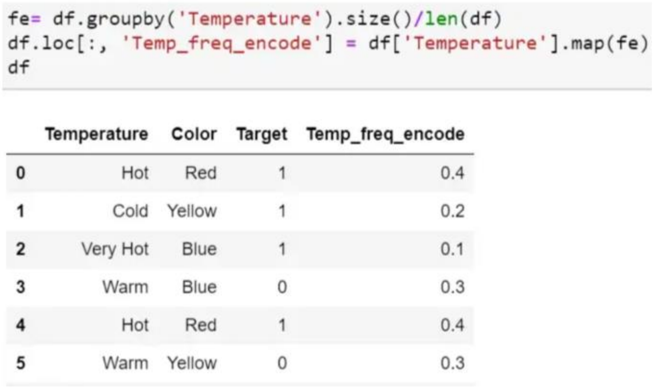


	Temperature	Color	Target	Temperature_0	Temperature_1	Temperature_2
0	Hot	Red	1	0	0	1
1	Cold	Yellow	1	0	1	0
2	Very Hot	Blue	1	0	1	1
3	Warm	Blue	0	1	0	0
4	Hot	Red	1	0	0	1
5	Warm	Yellow	0	1	0	0

Frequency encoding:

This method utilizes the frequency of the categories as labels.

Example:



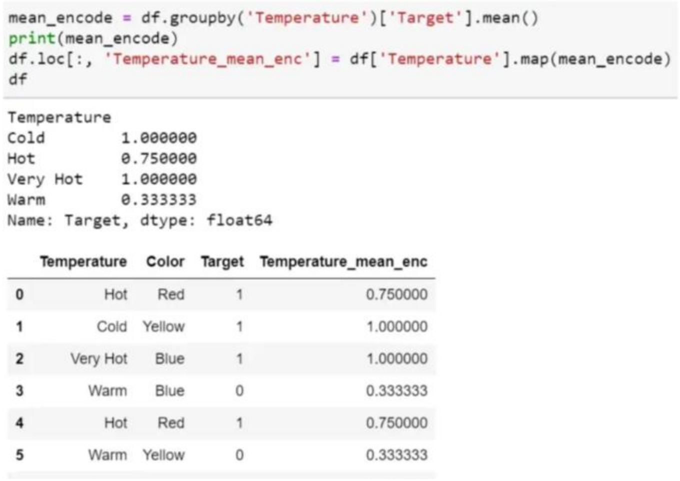
```
fe= df.groupby('Temperature').size()/len(df)
df.loc[:, 'Temp_freq_encode'] = df['Temperature'].map(fe)
df
```

	Temperature	Color	Target	Temp_freq_encode
0	Hot	Red	1	0.4
1	Cold	Yellow	1	0.2
2	Very Hot	Blue	1	0.1
3	Warm	Blue	0	0.3
4	Hot	Red	1	0.4
5	Warm	Yellow	0	0.3

Mean Encoding

in mean target encoding for each category in the feature label is decided with the mean value of the target variable on training data. It does not affect the volume of the data and helps in faster learning.

Example:



```
mean_encode = df.groupby('Temperature')['Target'].mean()
print(mean_encode)
df.loc[:, 'Temperature_mean_enc'] = df['Temperature'].map(mean_encode)
df
```

```
Temperature
Cold        1.000000
Hot         0.750000
Very Hot    1.000000
Warm        0.333333
Name: Target, dtype: float64
```

	Temperature	Color	Target	Temperature_mean_enc
0	Hot	Red	1	0.750000
1	Cold	Yellow	1	1.000000
2	Very Hot	Blue	1	1.000000
3	Warm	Blue	0	0.333333
4	Hot	Red	1	0.750000
5	Warm	Yellow	0	0.333333

Limitations:

One important effect is *Target Leakage.*