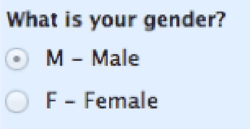
There can be two kinds of categorical data:

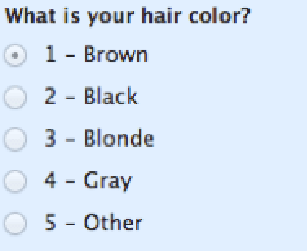
- **Nominal data**
- **Ordinal data**

Nominal data: Consists of the name variable without any numerical values, and order does not matter. For example,


What is your gender?

◉ M – Male

○ F – Female


What is your hair color?

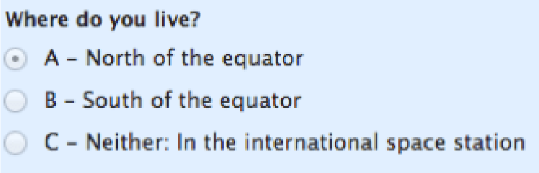
◉ 1 – Brown

○ 2 – Black

○ 3 – Blonde

○ 4 – Gray

○ 5 – Other

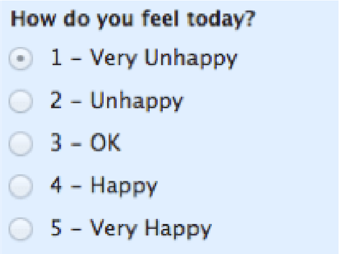

Where do you live?

◉ A – North of the equator

○ B – South of the equator

○ C – Neither: In the international space station

Ordinal data: Consists of a set of orders or scales. For example,


How do you feel today?

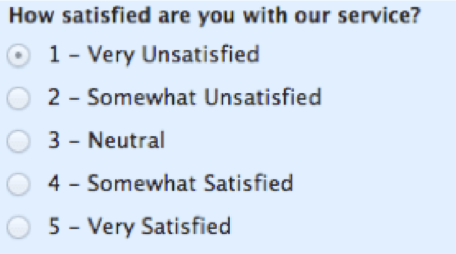
◉ 1 – Very Unhappy

○ 2 – Unhappy

○ 3 - OK

○ 4 – Happy

○ 5 – Very Happy


How satisfied are you with our service?

◉ 1 – Very Unsatisfied

○ 2 – Somewhat Unsatisfied

○ 3 – Neutral

○ 4 – Somewhat Satisfied

○ 5 – Very Satisfied

- **Label Encoding:** It converts labels into a numeric form (starting from 0 to n_categories - 1) where order doesn't matter.

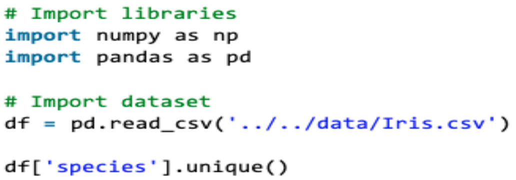
Example:

Suppose we have a column *(Height)* in some dataset.

Height
Tall
Medium
Short

Height
0
1
2

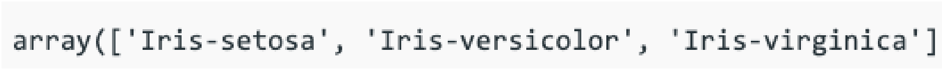
Applying *(Label Encoding)* on iris dataset, the target column is (Species.) which contains three species (/*Iris-setosa, Iris-versicolor, Iris-virginica)*.



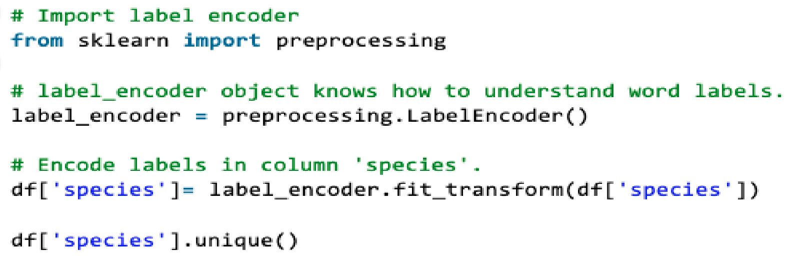
```
# Import libraries
import numpy as np
import pandas as pd

# Import dataset
df = pd.read_csv('../../data/Iris.csv')

df['species'].unique()
```



```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
```

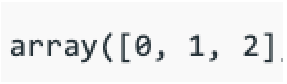


```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
df['species']= label_encoder.fit_transform(df['species'])

df['species'].unique()
```



```
array([0, 1, 2]
```

- **Ordinal Encoding (used with Ordinal data)**: It is converting labels into a numeric form where order matters.

Example:

Applying *(Ordinal Encoding)* on:

```
df = pd.DataFrame({"shirts": ['small', 'medium', 'large',
'small'], "costs": [10, 20, 30 , 40]})
```

```
      Shirt

0     small
1     medium
2     large
3     small
```

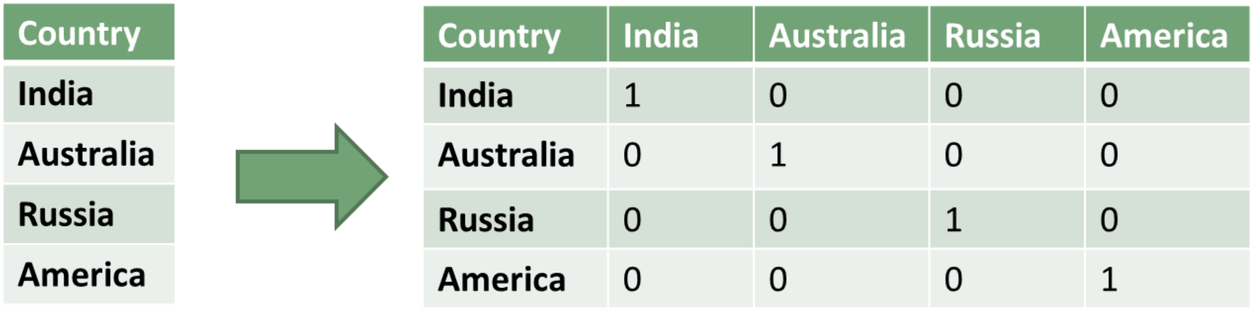
```
import category_encoders
mapping = [
    {'col': 'shirts', 'mapping':{'small': 0,'medium': 1,
'large': 2,
        }}]
encoder = category_encoders.OrdinalEncoder(cols = ['shirts'],
    return_df = True, mapping = mapping)
encoder.fit_transform(df['shirts'])
```

```
    Shirt

0     0
1     1
2     2
3     0
```

- **One-Hot Encoding (used with Nominal data)**: Each category is mapped with a binary variable containing either 0 or 1. 0 represents the absence, and 1 represents the presence of that category.

Example:



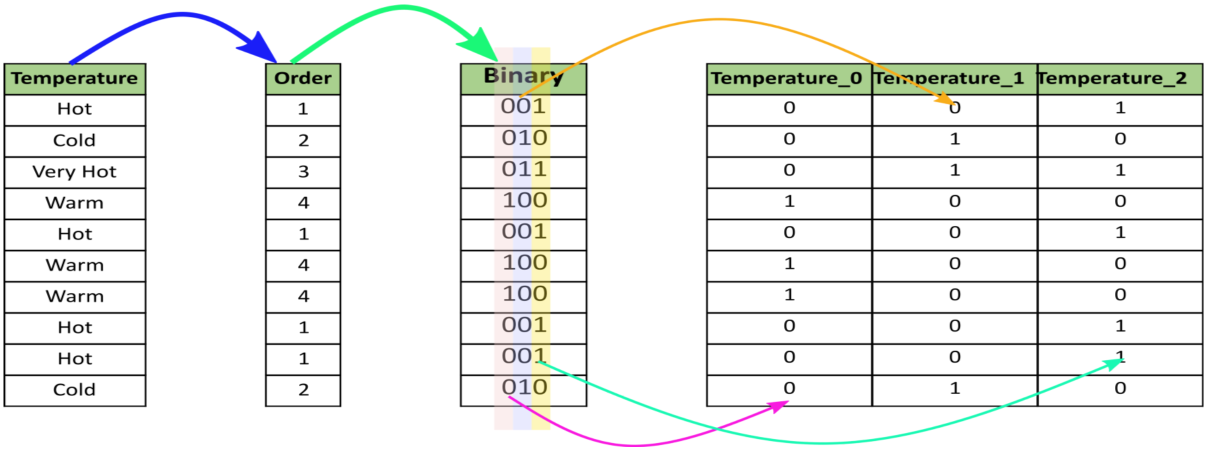
Country
India
Australia
Russia
America

Country	India	Australia	Russia	America
India	1	0	0	0
Australia	0	1	0	0
Russia	0	0	1	0
America	0	0	0	1

Drawbacks of One-Hot:

- A large number of levels are present in data. For example, a column with 30 different values will require 30 new variables for coding.
- Can be used only for nominal data.

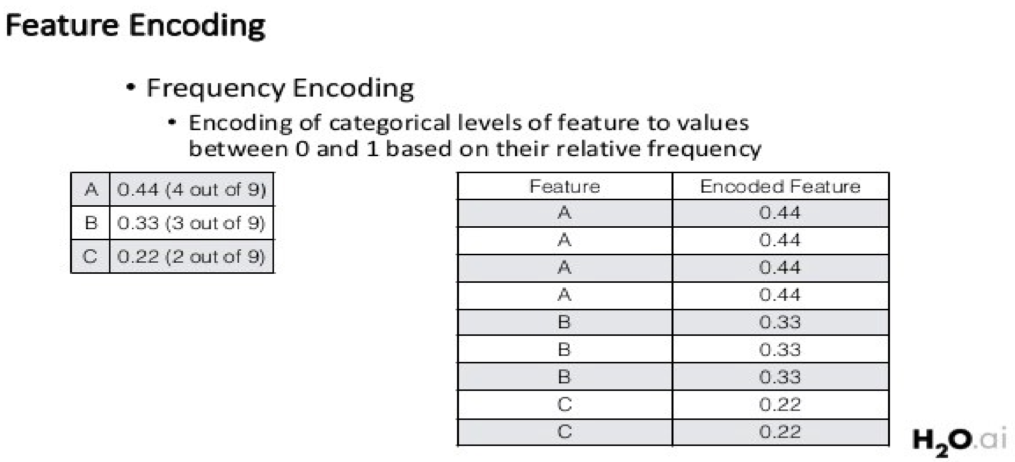
- **Binary Encoding**: It converts the categorical data into binary digits (1s and 0s) and each binary digit creates one feature column



Temperature	Order	Binary	Temperature_0	Temperature_1	Temperature_2
Hot	1	001	0	0	1
Cold	2	010	0	1	0
Very Hot	3	011	0	1	1
Warm	4	100	1	0	0
Hot	1	001	0	0	1
Warm	4	100	1	0	0
Warm	4	100	1	0	0
Hot	1	001	0	0	1
Hot	1	001	0	0	1
Cold	2	010	0	1	0

- **Frequency Encoding:**

  Encodes each categorical value based on how many times it is shown up.



Feature Encoding

- Frequency Encoding
  - Encoding of categorical levels of feature to values between 0 and 1 based on their relative frequency

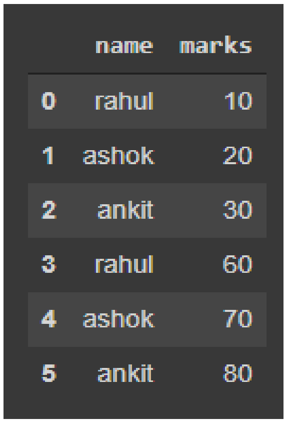
A	0.44 (4 out of 9)
B	0.33 (3 out of 9)
C	0.22 (2 out of 9)

Feature	Encoded Feature
A	0.44
A	0.44
A	0.44
A	0.44
B	0.33
B	0.33
B	0.33
C	0.22
C	0.22

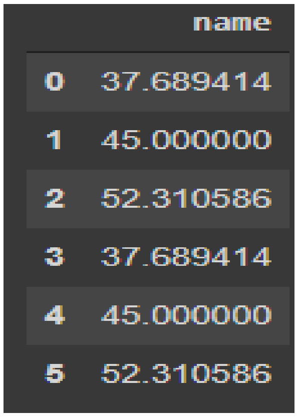
- **Target mean encoding:** It converts a categorical value into the mean of the target variable.

Example:

```
df=pd.DataFrame({'name':['rahul','ashok','ankit','rahul','ashok','ankit'],'marks' :[10,20,30,60,70,80,]})
encoder=ce.TargetEncoder(cols='name')
encoder.fit_transform(df['name'],df['marks'])
```



	name	marks
0	rahul	10
1	ashok	20
2	ankit	30
3	rahul	60
4	ashok	70
5	ankit	80



	name
0	37.689414
1	45.000000
2	52.310586
3	37.689414
4	45.000000
5	52.310586

References:
https://www.mygreatlearning.com/blog/label-encoding-in-python/
projectpro.io/recipes/encode-ordinal-categorical-features-in-python
https://analyticsindiamag.com/when-to-use-one-hot-encoding-in-deep-learning/
https://www.kdnuggets.com/2021/05/deal-with-categorical-data-machine-learning.html
https://medium.com/analytics-vidhya/different-type-of-feature-engineering-encoding-techniques-for-categorical-variable-encoding-214363a016fb