## LabelEncoder and OneHotEncoder

One hot encoding is the `technique to convert categorical values into a 1-dimensional numerical vector`.

The resulting vector will have only one element equal to 1 and the rest will be 0.

```
The 1 is called Hot and the 0’s are Cold. This is where its name of one hot encoding comes from.
```

### X = [Dog, Cat, Bird]

After one hot encoding each element of our vector X, we end up with the following:

`Dog = [1 0 0]

Cat = [0 1 0]

Bird = [0 0 1]`

### Why one hot encode numerical categorical variables?

Our machine learning algorithm can only read numerical values.

It is essential to encoding categorical features into numerical values.

## One Hot Encoding with Pandas

### Data Set

The data set we use here is from UCI Machine Learning Repository. It is used to predict whether a patient has kidney disease using various blood indicators as features. We use pandas to read the data in.

In [2]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
```

In [3]:

```python
tips = sns.load_dataset("tips")
tips.head()
```

Out[3]:

| | 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 |

In [4]:

```python
tips.dtypes
```

Out[4]:

```
total_bill     float64
tip            float64
sex           category
smoker        category
day           category
time          category
size             int64
dtype: object
```

Data has various categorical features, such as `‘sex’(Male and Female)`, `‘smoker’(yes or No)`, `‘day’(sun, mon..)` and so on.

In [5]:

```python
# Categorical boolean mask
categorical_feature_mask = tips.dtypes=='category'  #for Qualitative data dtype can be of type "object" also
categorical_feature_mask
```

Out[5]:

```
total_bill    False
tip           False
sex            True
smoker         True
day            True
time           True
size          False
dtype: bool
```

In [6]:

```python
# filter categorical columns using mask and turn it into a list
categorical_cols = tips.columns[categorical_feature_mask].tolist()
categorical_cols
```

Out[6]:

```
['sex', 'smoker', 'day', 'time']
```

```
LabelEncoder converts each class under specified feature to a numerical value.
```

In [7]:

```python
tips
```

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]:

```python
# import labelencoder
import sklearn
from sklearn.preprocessing import LabelEncoder
# instantiate labelencoder object
le = LabelEncoder()
```

In [9]:

```python
# apply le on categorical feature columns
tips[categorical_cols] = tips[categorical_cols].apply(lambda col: le.fit_transform(col))
tips[categorical_cols]
```

Out[9]:

| | sex | smoker | day | time |
|---|---|---|---|---|
| **0** | 0 | 0 | 2 | 0 |
| **1** | 1 | 0 | 2 | 0 |
| **2** | 1 | 0 | 2 | 0 |
| **3** | 1 | 0 | 2 | 0 |
| **4** | 0 | 0 | 2 | 0 |
| **...** | ... | ... | ... | ... |
| **239** | 1 | 0 | 1 | 0 |
| **240** | 0 | 1 | 1 | 0 |
| **241** | 1 | 1 | 1 | 0 |
| **242** | 1 | 0 | 1 | 0 |
| **243** | 0 | 0 | 3 | 0 |

244 rows × 4 columns

### All the categorical feature columns are binary class.

### But if the categorical `feature is multi class`, like : `day` column in the tips then

LabelEncoder will `return different values for different classes`.

## One-HotEncoding

In [10]:

```python
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

ct = ColumnTransformer([("encoder", OneHotEncoder(), list(categorical_feature_mask))], remainder = 'passthrough

one_hot_enc = ct.fit_transform(tips)
```

In [11]:

```python
pd.DataFrame(one_hot_enc)
```

Out[11]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 16.99 | 1.01 | 2.0 |
| **1** | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 10.34 | 1.66 | 3.0 |
| **2** | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 21.01 | 3.50 | 3.0 |
| **3** | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 23.68 | 3.31 | 2.0 |
| **4** | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 24.59 | 3.61 | 4.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **239** | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 29.03 | 5.92 | 3.0 |
| **240** | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 27.18 | 2.00 | 2.0 |
| **241** | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 22.67 | 2.00 | 2.0 |
| **242** | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 17.82 | 1.75 | 2.0 |
| **243** | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 18.78 | 3.00 | 2.0 |

244 rows × 13 columns

In [12]:

```python
one_hot_enc.shape
```

Out[12]:

```
(244, 13)
```

In [13]:

```python
tips.shape
```

Out[13]:

```
(244, 7)
```

# HAPPY LEARNING **