Categorical Encoding

In ML models we are often required to convert the categorical i.e text features to its numeric representation. The two most common ways to do this is to use Label Encoder or OneHot Encoder.

However, most of the ML beginners are not familiar with the impact of the choice of encoding has on their model, the accuracy of the model may shift by large numbers by using the right encoding at the right scenario.

Imports

In [1]:
```
import numpy as np
import pandas as pd
```

Read CSV File

In [2]:
```
df = pd.read_csv('Salaries_Cleaned.csv', index_col=0)
```

In [3]:
```
df.head()
```

Out[3]:

Id	BasePay	OvertimePay	OtherPay	Benefits	Year
2	155966.02	245131.88	137811.38	0.0	2011
3	212739.13	106088.18	16452.60	0.0	2011
4	77916.00	56120.71	198306.90	0.0	2011
5	134401.60	9737.00	182234.59	0.0	2011
6	118602.00	8601.00	189082.74	0.0	2011

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101716 entries, 2 to 148621
Data columns (total 5 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   BasePay      101716 non-null  float64
 1   OvertimePay  101716 non-null  float64
 2   OtherPay     101716 non-null  float64
 3   Benefits     101716 non-null  float64
 4   Year         101716 non-null  int64  
dtypes: float64(4), int64(1)
memory usage: 4.7 MB
```

In [5]:
```
df.describe()
```

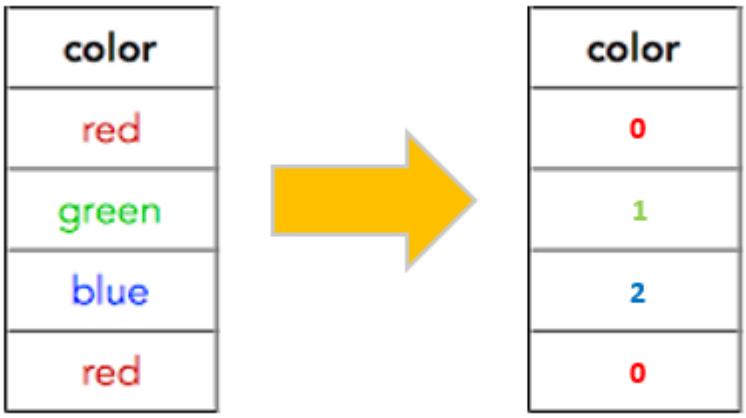
Out[5]:

	BasePay	OvertimePay	OtherPay	Benefits	Year
count	101716.000000	101716.000000	101716.000000	101716.000000	101716.000000

	BasePay	OvertimePay	OtherPay	Benefits	Year
mean	66039.414888	4367.986505	3269.793674	17784.721382	2012.439665
std	42944.744270	10652.098794	7692.445042	17294.099478	1.141663
min	6.040000	-0.010000	-7058.590000	0.000000	2011.000000
25%	32171.617500	0.000000	0.000000	0.000000	2011.000000
50%	64436.695000	0.000000	624.000000	18236.795000	2012.000000
75%	94864.125000	3393.670000	3486.000000	32730.600000	2013.000000
max	319275.010000	245131.880000	342802.630000	96570.660000	2014.000000

Label Encoder

Label Encoding in Python can be achieved using Sklearn Library. Sklearn provides a very efficient tool for encoding the levels of categorical features into numeric values. LabelEncoder encode labels with a value between 0 and n-1 where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier.



The problem here is since there are different numbers in the same column, the model will misunderstand the data to be in some kind of order, 0 < 1 < 2. To overcome this problem, we use One Hot Encoder.

In [6]:
```
from sklearn.preprocessing import LabelEncoder
```

In [7]:
```
df['Year'].unique()
```

Out[7]:
```
array([2011, 2012, 2013, 2014], dtype=int64)
```

In [8]:
```
label = LabelEncoder()
```

In [9]:
```
df['YearLabel'] = label.fit_transform(df['Year'])
```

In [10]:
```
df['YearLabel'].unique()
```

Out[10]:
```
array([0, 1, 2, 3], dtype=int64)
```

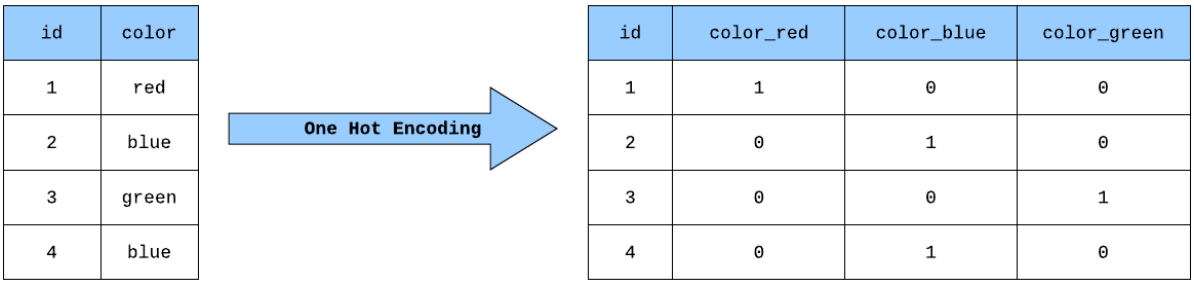
In [11]:
```
df.head()
```

Out[11]:

Id	BasePay	OvertimePay	OtherPay	Benefits	Year	YearLabel
2	155966.02	245131.88	137811.38	0.0	2011	0
3	212739.13	106088.18	16452.60	0.0	2011	0
4	77916.00	56120.71	198306.90	0.0	2011	0
5	134401.60	9737.00	182234.59	0.0	2011	0
6	118602.00	8601.00	189082.74	0.0	2011	0

One Hot Encoder

What one hot encoding does is, it takes a column which has categorical data, which has been label encoded and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value.



You can perform One Hot Encoding using the One Hot Encoder function from sklearn. Or you may use the column transformer. However, I would suggest that the simplest method is to use pd.get_dummies().

In [12]:
```
df = pd.get_dummies(df, columns=['Year'], drop_first=True)
```

In [13]:
```
df.head()
```

Out[13]:

Id	BasePay	OvertimePay	OtherPay	Benefits	YearLabel	Year_2012	Year_2013	Year_2014
2	155966.02	245131.88	137811.38	0.0	0	0	0	0
3	212739.13	106088.18	16452.60	0.0	0	0	0	0
4	77916.00	56120.71	198306.90	0.0	0	0	0	0
5	134401.60	9737.00	182234.59	0.0	0	0	0	0
6	118602.00	8601.00	189082.74	0.0	0	0	0	0

In [14]:
```
df.drop('YearLabel', axis=1, inplace=True)
```

In [15]:
```
df.head()
```

Out[15]:

Id	BasePay	OvertimePay	OtherPay	Benefits	Year_2012	Year_2013	Year_2014
2	155966.02	245131.88	137811.38	0.0	0	0	0
3	212739.13	106088.18	16452.60	0.0	0	0	0
4	77916.00	56120.71	198306.90	0.0	0	0	0
5	134401.60	9737.00	182234.59	0.0	0	0	0
6	118602.00	8601.00	189082.74	0.0	0	0	0

Now, let's save the encoded DataFrame to a csv file so that we can use it in the next file.

In [16]:
```
df.to_csv('Salaries_Encoded.csv')
```