# One-Hot-Encoding, Multicollinearity and the Dummy Variable Trap

### One-Hot-Encoding

Dummy variables are used for classifying categorical variables into a binary vector of 0 and 1 so that the computer can understand them. All elements of the vector are 0 except the element that matches the actual category which takes the value of The length of the vector is equal to the number of categories.

OneHotEncorder is a sklearn implementation of dummy variables. Dummies are executed outside sklearn by use of pandas.

### Dummy Variable Trap

Because dummy variable vectors take values of either 0 or 1 and because their length has to match that of the cateogry, it will create extra columns on the dataset with each column representing one category. This can create a problem of dummy variable trap in our model by causing our features to be multicollinear.

### Multicollinearity

Multicollinearity is the ability of one or more features to predict or be predicted by other features. Multicollinearity can distort our model. We should not have any feature that can be predicted by the help of another feature/s. To solve this problem, we need to drop/delete one column from those columns created by dummy variables. This is however not mandatory when working with sklearn as the library is confiqured to detect and disregard the excess/base column automatically.

# What is Nominal and Ordinal Variables

### Nominal Variables

These are variables whose order does not matter. This means one variable cannot then be ranked or infered as supperior/better/worse than another. Examples are Male/Female, Sales/Marketing, English/French, NewYork/London, etc

### Ordinal Variables

These are variables whose order matter. Therefore, an element of one category in an ordinal variable can be deemed to be superior/better/worse than another element in a different category. Examples are College degree rankings (Bachelor/Masters/PHD), Marathon winners(Gold/Bronze/Silver), Survey results(Low/Medium/High), etc

In [17]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score as acs

salary_data = pd.read_csv('salary.csv')

print(salary_data.shape)
salary_data.head()
```

```
(52, 6)
```

Out[17]:

| | sx | rk | yr | dg | yd | sl |
|---|---|---|---|---|---|---|
| **0** | male | full | 5 | doctorate | 18 | 25400 |
| **1** | male | full | 6 | masters | 21 | 24450 |
| **2** | male | full | 7 | doctorate | 13 | 27959 |
| **3** | male | full | 7 | doctorate | 15 | 29342 |
| **4** | male | full | 9 | doctorate | 17 | 28200 |

In [43]:
```
#Split dataset into features and labels
X = salary_data.iloc[:, :-1]
y = salary_data.iloc[:, -1]
```

## Dummy Encoding

### Encoding the norminal variable

In [45]:
```
#Create dummy variables
sx_dummies = pd.get_dummies(X.sx)

sx_dummies.head()
```

Out[45]:

| | female | male |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 0 | 1 |
| **2** | 0 | 1 |
| **3** | 0 | 1 |
| **4** | 0 | 1 |

In [46]:
```
#merge the dummies data with the rest of features
merged_X = pd.concat([X, sx_dummies], axis=1)

print(merged_X.head())
```

```
     sx    rk  yr         dg  yd  female  male
0  male  full   5  doctorate  18       0     1
1  male  full   6    masters  21       0     1
2  male  full   7  doctorate  13       0     1
3  male  full   7  doctorate  15       0     1
4  male  full   9  doctorate  17       0     1
```

In [58]:
```
#drop the original sex feature then drop one dummy variable column
new_X = merged_X.drop(['sx', 'female'], axis=1)

print(final_X.shape)
new_X.head()
```

```
(52, 5)
```

Out[58]:

| | rk | yr | dg | yd | male |
|---|---|---|---|---|---|
| **0** | full | 5 | doctorate | 18 | 1 |
| **1** | full | 6 | masters | 21 | 1 |
| **2** | full | 7 | doctorate | 13 | 1 |
| **3** | full | 7 | doctorate | 15 | 1 |
| **4** | full | 9 | doctorate | 17 | 1 |

## Label Encoding

### Encoding ordinal variables using sklearn

In [124]:
```
#Import and define label encoder
from sklearn.preprocessing import LabelEncoder

#fit the rank column into the encoder object then add the encoded column into the
lb_encode = LabelEncoder()
new_X['rank'] = lb_encode.fit_transform(new_X[['rk']])

#fit the degree column into the encoder object then add the encoded column into t
lb_encode1 = LabelEncoder()
new_X['degree'] = lb_encode1.fit_transform(new_X[['dg']])

new_X.head()
```

Out[124]:

| | rk | yr | dg | yd | male | rank | degree |
|---|---|---|---|---|---|---|---|
| **0** | full | 5 | doctorate | 18 | 1 | 2 | 0 |
| **1** | full | 6 | masters | 21 | 1 | 2 | 1 |
| **2** | full | 7 | doctorate | 13 | 1 | 2 | 0 |
| **3** | full | 7 | doctorate | 15 | 1 | 2 | 0 |
| **4** | full | 9 | doctorate | 17 | 1 | 2 | 0 |

In [105]:
```
#drop the old rank and degree featrures
final_X = new_X.drop(['rk', 'dg'], axis=1)

final_X.head()
```

Out[105]:

| | yr | yd | male | rank | degree |
|---|---|---|---|---|---|
| **0** | 5 | 18 | 1 | 2 | 0 |
| **1** | 6 | 21 | 1 | 2 | 1 |
| **2** | 7 | 13 | 1 | 2 | 0 |
| **3** | 7 | 15 | 1 | 2 | 0 |
| **4** | 9 | 17 | 1 | 2 | 0 |

## Train a Logistic Regression model to predict professor salary

In [106]:
```
from sklearn.linear_model import LogisticRegression

#create and fit the model
logreg = LogisticRegression(max_iter = 5000).fit(final_X,y)
y_pred = logreg.predict(final_X)


# Evaluate the model
from sklearn.metrics import accuracy_score as acs
print(f'Model Accuracy is {round(acs(y, y_pred)*100, 2)}')
```

```
Model Accuracy is 88.46
```

In [117]:
```
#Predict the salary of someone with the below description
#6years of experience
#15years from graduation
#full doctorate female professor with

new_prof = [[6, 15, 0, 2, 0]]
pred_salary = logreg.predict(new_prof)

print(f' Predicted salary for the new professor is {pred_salary} Dollars')
```

```
 Predicted salary for the new professor is [25500] Dollars
```