What is Multicollinearity

- Multicollinearity occurs when two or more independent variables are highly correlated with one another in a regression model.
- There are two basic kinds of multicollinearity:
    - 1. Structural multicollinearity : This type occurs when we create a model term using other terms.
    - 1. Data multicollinearity : This type of multicollinearity is present in the data itself rather than being an artifact of our model.

What Problems Do Multicollinearity Cause?

- Multicollinearity reduces the precision of the estimate coefficients, which weakens the statistical power of your regression model.
- The coefficient becomes very sensitive to small changes in the model. It's a disconcerting feeling when slightly different models lead to very different conclusions.

Detecting Multicollinearity

- Here i use two metthod to Detecting Multicollinearity
    - 1. Using OLS - Ordinary Least Squares regression
    - 1. Using VIF - Variable Inflation Factor

Data set where NO Multicollinearity

1. Detecting Multicollinearity using OLS Method

In [1]:

```
import pandas as pd
import statsmodels.api as sm
```

In [2]:

```
df = pd.read_csv('data/Advertising.csv', index_col=0)
```

In [3]:

```
df.head()
```

Out[3]:

	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

In [4]:

```
X = df.drop(['sales'],axis=True)
y = df['sales']
```

- Now See is there any multicolinearity between TV,Radio,Newspaper using OLS
- formula y = b0 + b1(TV)+b2(Radio)+b3(newspaper)
- ##### In Ols we need to compute b0 value but in data set no b0 value so now i am create that b0 in data set. so i add_constant as b0 value

In [5]:

```
X = sm.add_constant(X)
X.head()
```

Out[5]:

	const	TV	radio	newspaper
1	1.0	230.1	37.8	69.2
2	1.0	44.5	39.3	45.1
3	1.0	17.2	45.9	69.3
4	1.0	151.5	41.3	58.5
5	1.0	180.8	10.8	58.4

- Now i am created a const column which measure the b0 value

In [6]:

```
model= sm.OLS(y, X).fit()
```

In [7]:

```
model.summary()
```

Out[7]:

OLS Regression Results

Dep. Variable:	sales	**R-squared:**	0.897
Model:	OLS	**Adj. R-squared:**	0.896
Method:	Least Squares	**F-statistic:**	570.3
Date:	Thu, 17 Sep 2020	**Prob (F-statistic):**	1.58e-96
Time:	12:24:52	**Log-Likelihood:**	-386.18
No. Observations:	200	**AIC:**	780.4
Df Residuals:	196	**BIC:**	793.6
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.9389	0.312	9.422	0.000	2.324	3.554
TV	0.0458	0.001	32.809	0.000	0.043	0.049
radio	0.1885	0.009	21.893	0.000	0.172	0.206
newspaper	-0.0010	0.006	-0.177	0.860	-0.013	0.011

Omnibus:	60.414	**Durbin-Watson:**	2.084
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	151.241
Skew:	-1.327	**Prob(JB):**	1.44e-33
Kurtosis:	6.332	**Cond. No.**	454.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

- In model Summary we look only 4 things

- 1. coef 2. std err 3. p value 4. r-squared
  - mainly which have high std err these are Multicollinearity each other
  - As you see here std-err is low so here no Multicollinearity

2. Detecting Multicollinearity using VIF Method

In [8]:

```
# Import library for VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):

    # Calculating VIF
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

    return(vif)
```

In [9]:

```
X = df.iloc[:,:-1]
calc_vif(X)
```

Out[9]:

	variables	VIF
0	TV	2.486772
1	radio	3.285462
2	newspaper	3.055245

- VIF starts at 1 and has no upper limit
- VIF = 1, no correlation between the independent variable and the other variables
- VIF exceeding 5 or 10 indicates high multicollinearity between this independent variable and the others
- Here all variable has low VIF Value so no Multicollinearity
- In the above i am use OLS and VIF method to find any Multicollinearity between Variables as a result i found there is no 1. Multicollinearity between any variables

Data set where Multicollinearity exist and How to Overcome

- Here also i use this two method to find Multicollinearity
- but in project you can use any of these which is suitable for you .

1. Detecting Multicollinearity using OLS Method

In [10]:

```
df_salary = pd.read_csv('data/Salary_Data.csv')
df_salary.head()
```

Out[10]:

	YearsExperience	Age	Salary
0	1.1	21.0	39343
1	1.3	21.5	46205
2	1.5	21.7	37731
3	2.0	22.0	43525
4	2.2	22.2	39891

In [11]:

```
X = df_salary.drop(['Salary'],axis=True)
y = df_salary['Salary']
```

In [12]:

```
X = sm.add_constant(X)
X.head()
```

Out[12]:

	const	YearsExperience	Age
0	1.0	1.1	21.0
1	1.0	1.3	21.5
2	1.0	1.5	21.7
3	1.0	2.0	22.0
4	1.0	2.2	22.2

In [13]:

```
model= sm.OLS(y, X).fit()
```

In [14]:

```
model.summary()
```

Out[14]:

OLS Regression Results

Dep. Variable:	Salary	**R-squared:**	0.960
Model:	OLS	**Adj. R-squared:**	0.957
Method:	Least Squares	**F-statistic:**	323.9
Date:	Thu, 17 Sep 2020	**Prob (F-statistic):**	1.35e-19
Time:	12:24:52	**Log-Likelihood:**	-300.35
No. Observations:	30	**AIC:**	606.7
Df Residuals:	27	**BIC:**	610.9
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-6661.9872	2.28e+04	-0.292	0.773	-5.35e+04	4.02e+04
YearsExperience	6153.3533	2337.092	2.633	0.014	1358.037	1.09e+04
Age	1836.0136	1285.034	1.429	0.165	-800.659	4472.686

Omnibus:	2.695	**Durbin-Watson:**	1.711
Prob(Omnibus):	0.260	**Jarque-Bera (JB):**	1.975
Skew:	0.456	**Prob(JB):**	0.372
Kurtosis:	2.135	**Cond. No.**	626.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

- As You see YearsExperience and Age Has High std err so here Multicollinearity occure

- so to fix it we have to remove one variable which has high p value
    - Here Age Has High p value so just drop the age feature

2. Detecting Multicollinearity using VIF Method

In [15]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):

    # Calculating VIF
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

    return(vif)
```

In [16]:

```
X = df_salary.iloc[:,:-1]
calc_vif(X)
```

Out[16]:

	variables	VIF
0	YearsExperience	11.24047
1	Age	11.24047

- We can see here that the ‘Age’ and ‘ YearsExperience’ have a high VIF value, meaning they can be predicted by other independent variables in the dataset.

Fixing Multicollinearity

- 1. Dropping one of the correlated features will help in bringing down the multicollinearity between correlated features.
- 1. combine the correlated variables into one and drop the others. This will reduce the multicollinearity:

Another Method To Know multicollinearity

- Use Correlation matrix

In [17]:

```
df_salary.corr()
```

Out[17]:

	YearsExperience	Age	Salary
YearsExperience	1.000000	0.987258	0.978242
Age	0.987258	1.000000	0.974530
Salary	0.978242	0.974530	1.000000

In [18]:

```
df.corr()
```

Out[18]:

	TV	radio	newspaper	sales
TV	1.000000	0.054809	0.056648	0.782224
radio	0.054809	1.000000	0.354104	0.576223
newspaper	0.056648	0.354104	1.000000	0.228299
sales	0.782224	0.576223	0.228299	1.000000

- Here You Can Use Seaborn To plot this matrix

--- This All About multicollinearity If It Helpful to you Please Like Me ---