In [157]:
```
# 50 start ups
# trying to find out which variable is imp for me to increace my profit
```

In [158]:
```
import numpy as np
import pandas as pd
```

In [184]:
```
# importing the salary data set
startups_df = pd.read_csv(r'F:\Prthon Programming\Udemy\Part 2 - Regression\Section 5 - Multiple Linear Regression\50_Startups.csv', header=0)
```

In [185]:
```
startups_df.head() # by default first 5 obv it shows
```

Out[185]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349	136898	471784	New York	192262
1	162598	151378	443899	California	191792
2	153442	101146	407935	Florida	191050
3	144372	118672	383200	New York	182902
4	142107	91392	366168	Florida	166188

In [186]:
```
startups_df.tail() # by default last 5 obv it shows
```

Out[186]:

	R&D Spend	Administration	Marketing Spend	State	Profit
45	1000	124153	1904	New York	64926
46	1315	115816	297114	Florida	49491
47	0	135427	0	California	42560
48	542	51743	0	New York	35673
49	0	116984	45173	California	14681

In [187]:
```
startups_df.shape # try to see number of obv + number of columns
```

Out[187]:
```
(50, 5)
```

In [188]:
```
startups_df.isnull().sum() # checking if there are any missing values in data set
```

Out[188]:
```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

In [189]:
```
# Converting From categorical to Numerical:

colname=[] # trying to find out categorial in data set
for X in startups_df.columns[:]:
    if startups_df[X].dtype=='object':
        colname.append(X)
colname
```

Out[189]:
```
['State']
```

In [190]:
```
from sklearn import preprocessing  # Handling Categorical Data

labelencoder_X = preprocessing.LabelEncoder()

for X in colname:
    startups_df[X]=labelencoder_X.fit_transform(startups_df[X])

startups_df.head()
```

Out[190]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349	136898	471784	2	192262
1	162598	151378	443899	0	191792
2	153442	101146	407935	1	191050
3	144372	118672	383200	2	182902
4	142107	91392	366168	1	166188

In [191]:
```
startups_df.dtypes
```

Out[191]:
```
R&D Spend          int64
Administration     int64
Marketing Spend    int64
State              int32
Profit             int64
dtype: object
```

In [192]:
```
X = startups_df.values[:, :-1] # -1 means that i m not taking the salary colum
n in (X variable) is IV
Y = startups_df.values[:,-1] # (Y variable) salary is my dependent variable
```

In [193]:
```
#from sklearn.cross_validation import train_test_split

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, ra
ndom_state = 0)

# I have split the data set in 80:20 ratio with the help of model_selection
```

In [169]:
```
#====================END OF THE DATA PREPROCESSING=========================
=======
```

In [194]:
```
from sklearn.linear_model import LinearRegression # taken class as LinearRegression

regressor = LinearRegression()
regressor.fit(X_train, Y_train) # fitting X_train and Y_train by using fit function
```

Out[194]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [171]:
```
#====================WE HAVE TRAIN THE MACHINE=============================
```

In [195]:
```
Y_pred = regressor.predict(X_test) # predicting test set results
```

In [173]:
```
#====================BACKWARD ELIMINATION PREPARATION=======================
```

In [196]:
```
X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis = 1)
X.shape
```

Out[196]:
```
(50, 5)
```

In [176]:
```
import statsmodels.formula.api as sm
```

In [200]:
```
# 1st Result from B E P

X_optimal = X[:,[0,1,2,3]]
regressor_OLS = sm.OLS(endog = Y, exog = X_optimal).fit()
regressor_OLS.summary()
```

Out[200]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.951
Model:	OLS	**Adj. R-squared:**	0.948
Method:	Least Squares	**F-statistic:**	296.0
Date:	Sat, 21 Sep 2019	**Prob (F-statistic):**	4.53e-30
Time:	16:37:36	**Log-Likelihood:**	-525.39
No. Observations:	50	**AIC:**	1059.
Df Residuals:	46	**BIC:**	1066.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.012e+04	6572.339	7.626	0.000	3.69e+04	6.34e+04
x1	0.8057	0.045	17.846	0.000	0.715	0.897
x2	-0.0268	0.051	-0.525	0.602	-0.130	0.076
x3	0.0272	0.016	1.655	0.105	-0.006	0.060

Omnibus:	14.839	**Durbin-Watson:**	1.282
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	21.444
Skew:	-0.949	**Prob(JB):**	2.20e-05
Kurtosis:	5.587	**Cond. No.**	1.40e+06

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.4e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [201]:
```
X_optimal = X[:,[0,1,3]]
regressor_OLS = sm.OLS(endog = Y, exog = X_optimal).fit()
regressor_OLS.summary()
```

Out[201]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.950
Model:	OLS	**Adj. R-squared:**	0.948
Method:	Least Squares	**F-statistic:**	450.8
Date:	Sat, 21 Sep 2019	**Prob (F-statistic):**	2.15e-31
Time:	16:39:08	**Log-Likelihood:**	-525.54
No. Observations:	50	**AIC:**	1057.
Df Residuals:	47	**BIC:**	1063.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.698e+04	2689.930	17.464	0.000	4.16e+04	5.24e+04
x1	0.7966	0.041	19.266	0.000	0.713	0.880
x2	0.0299	0.016	1.927	0.060	-0.001	0.061

Omnibus:	14.678	**Durbin-Watson:**	1.257
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	21.163
Skew:	-0.939	**Prob(JB):**	2.54e-05
Kurtosis:	5.576	**Cond. No.**	5.32e+05

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.32e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:
```
#=====================END OF WITH GOOD RESULTS ==============================
===
```