Multiple Linear Regression Chapter

In [ ]:
```
Import the Libraries:

1. Numpy - Math functions
2. MatplotLib - Visualization
3. SciKitLearn

    a) Build Models
    b) Preprocessing (Encoding,One-Hot Encoding)

Pandas Package

    a) Read Files into Dataframes
    b) Create, Manipulate Dataframes
    c) Create Feature Matrix
    d) Create Output Vector

SciKit Learn Details:

Preprocess Data

    a) Imputer - Missing values using strategies like Mean
    b) Convert Categorical data to Numbers - LabelEncoder
    c) Convert Numbers into One Hot Encoding - OneHotEncoder
    d) Model Selection - Split into Training and Test Set Data
    e) Scaling Data - Standaradization / Normalization

Build Linear Regression Models

    a) linear_model library, LinearModel object
```

Pre-processing for Multiple Linear Regression Model

The key item to notice in this is the Dummy Variable Trap When there are n levels of dumy variables created, we will need to remove one vector. For example in this example, state is encoded as follows New York, Florida, California. When we encode this using one-hot encoding, When we have the value of the two states, we can infer the third - there is Collienearity and values of the other dummy variables determines the last dummy variable. So one has to be removed.

Handle Dummy Variable Trap

In [14]:
```
# Simple Linear Regression

#-----------------Preprocessing Section -------------------------------------
-----
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 27 21:24:44 2019
Multiple Linear Regression
@author: Anand
"""
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values

# Encoding categorical data
# Encoding the Independent Variable
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:, 3] = labelencoder_X.fit_transform(X[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
X = onehotencoder.fit_transform(X).toarray()

# Removing first column to avoid the Dummy Variable Trap
X= X[:, 1:]

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .2
, random_state = 0)

#---------------End of Pre-processing Section ---------------------------
-------
```

```
/Users/aaa/anaconda3/lib/python3.7/site-packages/sklearn/preprocessing/
_encoders.py:368: FutureWarning: The handling of integer data will chan
ge in version 0.22. Currently, the categories are determined based on t
he range [0, max(values)], while in the future they will be determined
based on the unique values.
If you want the future behaviour and silence this warning, you can spec
ify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert th
e categories to integers, then you can now use the OneHotEncoder direct
ly.
  warnings.warn(msg, FutureWarning)
/Users/aaa/anaconda3/lib/python3.7/site-packages/sklearn/preprocessing/
_encoders.py:390: DeprecationWarning: The 'categorical_features' keywor
d is deprecated in version 0.20 and will be removed in 0.22. You can us
e the ColumnTransformer instead.
  "use the ColumnTransformer instead.", DeprecationWarning)
```

Fitting the Model (to Training set) and Predicting (on Test set)

Use the LinearRegression Object from the linear_model library Use the Fit method of the Regressor object to the training Set Use the Predict methd of the Regressor object on the Test Set

In [15]:
```
#Fitting Linear Regression Model to the Taining Set - Ordinary least squ
ares Linear Regression.

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train)

# Predcting the test set observations
y_pred=regressor.predict(X_test)
```

Building an Optimal Model usinf Back Validation

Create a linear regrssion model using the Ordinary Least Squares (OLS) with all the independant variables. Set a Significance level say 5% (SL = .05) Check the P value of the Model for each of the independent variables Based upon this P value and Significance Level decide to keep/remove the Variables with high P Values. Successively Do this and inspect the Model using the Summary function. Use a Line to predict the training inputs, and predicted outputs on training set

In [17]:
```
# Building an optimal model using back validation
# As the stats model does not take care of intercept, we need to incldud
e a ones array

import statsmodels.formula.api as sm
X=np.append(arr=np.ones((50,1)).astype(int),values=X,axis=1)
```

In [18]:
```
X_opt=X[:, [0,1,2,3,4,5]]
```

In [19]:
```
regressor_OLS=sm.OLS(endog=y,exog=X_opt).fit()
regressor_OLS.summary()
```

Out[19]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.948
Model:	OLS	**Adj. R-squared:**	0.943
Method:	Least Squares	**F-statistic:**	205.0
Date:	Sun, 31 Mar 2019	**Prob (F-statistic):**	2.90e-28
Time:	17:34:04	**Log-Likelihood:**	-526.75
No. Observations:	50	**AIC:**	1064.
Df Residuals:	45	**BIC:**	1073.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	2.73e+04	3185.530	8.571	0.000	2.09e+04	3.37e+04
x1	2.73e+04	3185.530	8.571	0.000	2.09e+04	3.37e+04
x2	1091.1075	3377.087	0.323	0.748	-5710.695	7892.910
x3	-39.3434	3309.047	-0.012	0.991	-6704.106	6625.420
x4	0.8609	0.031	27.665	0.000	0.798	0.924
x5	-0.0527	0.050	-1.045	0.301	-0.154	0.049

Omnibus:	14.275	**Durbin-Watson:**	1.197
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	19.260
Skew:	-0.953	**Prob(JB):**	6.57e-05
Kurtosis:	5.369	**Cond. No.**	7.08e+17

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.15e-24. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Inspecting the P values above, we can remove x3 above

x3 above corrsponds to the state variable of the data set, that will be removed as part of the back validation approach

In [6]:
```
X_opt=X[:, [0,1,3,4,5]]
regressor_OLS=sm.OLS(endog=y,exog=X_opt).fit()
regressor_OLS.summary()
```

Out[6]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.948
Model:	OLS	**Adj. R-squared:**	0.944
Method:	Least Squares	**F-statistic:**	278.7
Date:	Sun, 31 Mar 2019	**Prob (F-statistic):**	1.68e-29
Time:	16:47:32	**Log-Likelihood:**	-526.81
No. Observations:	50	**AIC:**	1062.
Df Residuals:	46	**BIC:**	1069.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.753e+04	3072.973	8.960	0.000	2.13e+04	3.37e+04
x1	2.753e+04	3072.973	8.960	0.000	2.13e+04	3.37e+04
x2	-573.7029	2838.043	-0.202	0.841	-6286.386	5138.981
x3	0.8624	0.030	28.282	0.000	0.801	0.924
x4	-0.0530	0.050	-1.063	0.294	-0.154	0.047

Omnibus:	14.902	**Durbin-Watson:**	1.199
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	21.212
Skew:	-0.964	**Prob(JB):**	2.48e-05
Kurtosis:	5.543	**Cond. No.**	1.37e+17

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 5.78e-23. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Now the Variable that corresponds to x2 above has to be removed.

In [7]:
```
X_opt=X[:, [0,3,4,5]]
regressor_OLS=sm.OLS(endog=y,exog=X_opt).fit()
regressor_OLS.summary()
```

Out[7]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.948
Model:	OLS	**Adj. R-squared:**	0.944
Method:	Least Squares	**F-statistic:**	278.7
Date:	Sun, 31 Mar 2019	**Prob (F-statistic):**	1.68e-29
Time:	16:49:05	**Log-Likelihood:**	-526.81
No. Observations:	50	**AIC:**	1062.
Df Residuals:	46	**BIC:**	1069.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	5.507e+04	6145.947	8.960	0.000	4.27e+04	6.74e+04
x1	-573.7029	2838.043	-0.202	0.841	-6286.386	5138.981
x2	0.8624	0.030	28.282	0.000	0.801	0.924
x3	-0.0530	0.050	-1.063	0.294	-0.154	0.047

Omnibus:	14.902	**Durbin-Watson:**	1.199
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	21.212
Skew:	-0.964	**Prob(JB):**	2.48e-05
Kurtosis:	5.543	**Cond. No.**	6.74e+05

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.74e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:
```
Now the Variable that corresponds to x1 above has to be removed.
```

In [8]:
```
X_opt=X[:, [0,3,5]]
regressor_OLS=sm.OLS(endog=y,exog=X_opt).fit()
regressor_OLS.summary()
```

Out[8]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.041
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	1.010
Date:	Sun, 31 Mar 2019	**Prob (F-statistic):**	0.372
Time:	16:49:22	**Log-Likelihood:**	-599.60
No. Observations:	50	**AIC:**	1205.
Df Residuals:	47	**BIC:**	1211.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	7.613e+04	2.59e+04	2.942	0.005	2.41e+04	1.28e+05
x1	2555.2116	1.2e+04	0.212	0.833	-2.16e+04	2.68e+04
x2	0.2885	0.205	1.404	0.167	-0.125	0.702

Omnibus:	0.119	**Durbin-Watson:**	0.097
Prob(Omnibus):	0.942	**Jarque-Bera (JB):**	0.139
Skew:	0.099	**Prob(JB):**	0.933
Kurtosis:	2.835	**Cond. No.**	5.67e+05

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.67e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

While the answers are not matching the course output and my own output on Spyder, I am leaving it as it is.