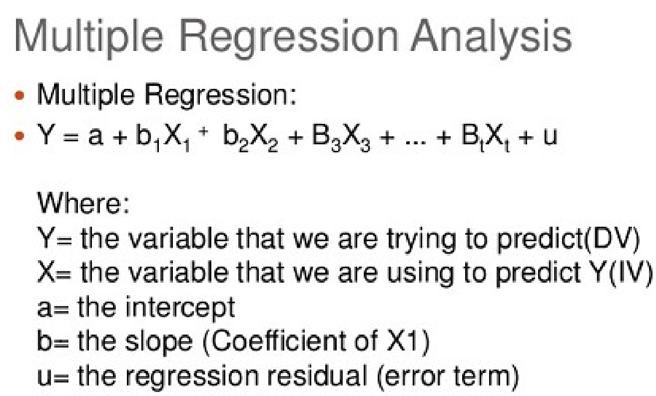
Multi Variable Regression :

Multi variable regression is merely the extension of simple linear regression. A simple linear regression looks something like y= mx+b where x is the only independent varaibe. But in a realistic situation, a target or dependant variable might depend on more than one independent variable. In that case, the linear regression equation will look some thing like



For example, For sales predictions, independent variables might include a company’s advertising spend on radio, TV, and newspapers. For that case the equation will look like

Sales= c1*Radio+c2*TV+c3*newspapers+e

Where Radio,TV and newspapers represent spend in Radio TV and newpapers respectively

Dataset:

In [2]:

```
import pandas as pd
data = pd.read_csv('data.csv', index_col=0)
data.head()
```

Out[2]:

	TV	Radio	Newspaper	Sales
Index				
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

Model-building and Variable Selection:

In [10]:

```
from sklearn.linear_model import LinearRegression #import Linear Re
Feature_columns=['TV','Radio','Newspaper'] #Segregating the indeper
X = data[Feature_columns]
y = data['Sales']# Target Variable

# instantiate and fit
lm = LinearRegression()
lm.fit(X, y)

# print the coefficients
print(lm.intercept_)
print(lm.coef_)
```

```
2.59802287597
[ 0.06500596  0.23956879 -0.06174037]
```

In [4]:

```
import statsmodels.formula.api as smf
lm1 = smf.ols(formula='Sales ~ TV + Radio + Newspaper', data=data).

# print the coefficients
lm1.params
lm1.summary() # Print summary to display the p value for all the va
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:133
  "anyway, n=%i" % int(n))
```

Out[4]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.973
Model:	OLS	**Adj. R-squared:**	0.958
Method:	Least Squares	**F-statistic:**	61.19
Date:	Wed, 23 Oct 2019	**Prob (F-statistic):**	0.000231
Time:	18:28:23	**Log-Likelihood:**	-12.745
No. Observations:	9	**AIC:**	33.49
Df Residuals:	5	**BIC:**	34.28
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	2.5980	3.001	0.866	0.426	-5.116	10.312
TV	0.0650	0.005	12.002	0.000	0.051	0.079
Radio	0.2396	0.032	7.524	0.001	0.158	0.321
Newspaper	-0.0617	0.042	-1.454	0.206	-0.171	0.047

Omnibus:	0.063	**Durbin-Watson:**	1.005
Prob(Omnibus):	0.969	**Jarque-Bera (JB):**	0.284
Skew:	-0.084	**Prob(JB):**	0.868
Kurtosis:	2.146	**Cond. No.**	1.09e+03

From the regression results we can see P>|t| which is the significance for each variable or feature. If we use a cutoff value of 0.05, TV and Radio seems to be the significant variable because for Newspaper p value is greater than 0.05 making it insignificant.So we select the variables TV and Radio , and run the model again.

In [5]:

```
# instantiate and fit model with new set of variables
lm2 = smf.ols(formula='Sales ~ TV + Radio', data=data).fit()

# calculate r-square
lm2.rsquared
```

Out[5]:

```
0.96227137335894464
```

The r squared= 0.96 shows overfitting - as the dataset sample number is low and model is built on the whole dataset, the model is overfitted. R-squared will always increase as you add more features to the model.

Prediction :

In [7]:

```
y_pred = lm2.predict(X)

y_pred
```

Out[7]:

```
Index
1    22.662979
2    11.600730
3    11.610174
4    18.713153
5    12.695629
6    10.279623
7     5.978122
8    25.119854
9    19.339736
dtype: float64
```

crossvalidation Using Train test split and determination of RMSE :

In [ ]:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import metrics

X = data[Feature_columns]
y = data.Sales

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, random_st

# Instantiate model
lm3 = LinearRegression()

# Fit Model
lm3.fit(X_train, y_train)


# Predict
y_pred = lm3.predict(X_test)
y_test
y_pred
# RMSE
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

From cross validation we can validate the dataset and reduce the overfitting problem. In this proccess we are developing the model on a part of dataset(training) and testing the model on a different part of the dataset. This reduces the overfitting problem and helps validating the data more efficiently.

In [17]:

```
print(y_test,y_pred)
```

```
Index
9    21.1
3     9.3
7     7.3
Name: Sales, dtype: float64 [ 16.47027162  11.99717276   5.9245908
```

Questionarrie:

1. What is the role of Cross- validation in linear regression?
2. On basis of which summary attributes feature is selected in multi linear regression?
3. What is overfitting problem?
4. What is adjusted R square? how it helps in overcoming the limitations of R square?

In [ ]:

```

```

In [ ]:

```

```