Multilinear Regression(Based on back propagation) is used for optimization. As there are five diffrent modules, we are using Multilinear Regression algorithm to rank them based on there importance.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
dataset = pd.read_csv('Results_optimization.csv')
X = dataset.iloc[:, 1:6].values
y = dataset.iloc[:, 6].values
```

In [4]:
```
print(dataset)
print(X)
print(y)
```

```
                  Name  Verbal     Logic    Emotion  Object    Social  Result
0           jay_jain_4       5  5.833333   8.333333     6.0  6.666667  8.1695
1         John Abraham       5  3.333333   7.500000     4.5  4.166667  5.6966
2       Albert_Jonas_6       7  2.500000   9.166667     6.5  5.833333  7.2655
3          joy_marsh_5       6  5.000000   1.666667     3.5  6.666667  5.2935
4           smith_john       4  3.333333   9.166667     2.5  5.833333  5.5496
5              nothing       7  3.333333   6.666667     6.5  5.000000  6.5179
6        Anand_Gupta_6       7  8.333333   6.666667     7.5  5.833333  8.2216
7          Neha_Shah_8       6  7.500000   9.166667     6.0  3.333333  8.2216
8        Sheela_Modi_8       2  3.333333   9.166667     3.5  5.000000  5.1047
9        Aarti_Patel_6       7  7.166667   5.833333     4.5  9.166667  7.5695
10     Priya_Panchal_8       7  2.500000   7.500000     5.0  0.833333  5.0000
11  Karishma_Agrawal_7       7  6.666667  10.000000     6.0  4.166667  6.6927
12   Elliot_Anderson_8       7  5.833333   5.000000     3.0  5.833333  6.7640
13     Saurabh_Tiwari_7      7  6.666667   9.166667     4.0  5.000000  6.6927
14      Brian_Thomas_6       9  8.333333  10.000000     5.0  4.166667  8.1695
15     Sheetal_Patil_5       3  5.833333   5.000000     3.0  5.000000  5.1053
16     Rushabh_Mutha_6       2  5.000000   5.000000     2.5  5.833333  5.0000
17       Shahid_Khan_5       6  6.666667   5.833333     5.5  6.666667  8.1695
[[  5.           5.83333333   8.33333333   6.           6.66666667]
 [  5.           3.33333333   7.5          4.5          4.16666667]
 [  7.           2.5          9.16666667   6.5          5.83333333]
 [  6.           5.           1.66666667   3.5          6.66666667]
 [  4.           3.33333333   9.16666667   2.5          5.83333333]
 [  7.           3.33333333   6.66666667   6.5          5.        ]
 [  7.           8.33333333   6.66666667   7.5          5.83333333]
 [  6.           7.5          9.16666667   6.           3.33333333]
 [  2.           3.33333333   9.16666667   3.5          5.        ]
 [  7.           7.16666667   5.83333333   4.5          9.16666667]
 [  7.           2.5          7.5          5.           0.83333333]
 [  7.           6.66666667  10.           6.           4.16666667]
 [  7.           5.83333333   5.           3.           5.83333333]
 [  7.           6.66666667   9.16666667   4.           5.        ]
 [  9.           8.33333333  10.           5.           4.16666667]
 [  3.           5.83333333   5.           3.           5.        ]
 [  2.           5.           5.           2.5          5.83333333]
 [  6.           6.66666667   5.83333333   5.5          6.66666667]]
[ 8.1695  5.6966  7.2655  5.2935  5.5496  6.5179  8.2216  8.2216  5.1047
  7.5695  5.      6.6927  6.764   6.6927  8.1695  5.1053  5.      8.1695]
```

In [5]:
```
#Splitting the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_s
tate=0)
```

In [6]:
```
#Fiting multiple linear regression to training set
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)
```

Out[6]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [7]:
```
#Predicting the value
y_pred = reg.predict(X_test)
```

In [8]:
```
#Building the optimal model
import statsmodels.formula.api as sms
X_opt = X[: , [0,1,2,3,4]]
reg_ols  = sms.OLS(endog = y, exog = X_opt).fit()
reg_ols.summary()
```

```
C:\Users\expert\Anaconda3\lib\site-packages\scipy\stats\stats.py:1334: UserWar
ning: kurtosistest only valid for n>=20 ... continuing anyway, n=18
  "anyway, n=%i" % int(n))
```

Out[8]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.995
Model:	OLS	**Adj. R-squared:**	0.994
Method:	Least Squares	**F-statistic:**	559.1
Date:	Sun, 01 Apr 2018	**Prob (F-statistic):**	1.08e-14
Time:	09:14:46	**Log-Likelihood:**	-11.495
No. Observations:	18	**AIC:**	32.99
Df Residuals:	13	**BIC:**	37.44
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
x1	0.1345	0.085	1.585	0.137	-0.049	0.318
x2	0.2699	0.076	3.554	0.004	0.106	0.434
x3	0.1602	0.053	3.032	0.010	0.046	0.274
x4	0.3726	0.109	3.407	0.005	0.136	0.609
x5	0.2760	0.064	4.308	0.001	0.138	0.414

Omnibus:	0.200	**Durbin-Watson:**	2.598
Prob(Omnibus):	0.905	**Jarque-Bera (JB):**	0.034
Skew:	-0.063	**Prob(JB):**	0.983
Kurtosis:	2.829	**Cond. No.**	12.9

In [9]:

```
#2nd iteration
X_opt = X[: , [1,2,3,4]]
reg_ols  = sms.OLS(endog = y, exog = X_opt).fit()
reg_ols.summary()
```

```
C:\Users\expert\Anaconda3\lib\site-packages\scipy\stats\stats.py:1334: UserWar
ning: kurtosistest only valid for n>=20 ... continuing anyway, n=18
  "anyway, n=%i" % int(n))
```

Out[9]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.994
Model:	OLS	**Adj. R-squared:**	0.993
Method:	Least Squares	**F-statistic:**	630.2
Date:	Sun, 01 Apr 2018	**Prob (F-statistic):**	1.25e-15
Time:	09:15:02	**Log-Likelihood:**	-13.085
No. Observations:	18	**AIC:**	34.17
Df Residuals:	14	**BIC:**	37.73
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	0.3113	0.075	4.147	0.001	0.150	0.472
x2	0.1707	0.055	3.093	0.008	0.052	0.289
x3	0.4693	0.096	4.910	0.000	0.264	0.674
x4	0.2785	0.067	4.131	0.001	0.134	0.423

Omnibus:	0.215	**Durbin-Watson:**	3.027
Prob(Omnibus):	0.898	**Jarque-Bera (JB):**	0.030
Skew:	-0.062	**Prob(JB):**	0.985
Kurtosis:	2.843	**Cond. No.**	9.21

In [10]:

```
#3rd iteration
X_opt = X[: , [1,3,4]]
reg_ols  = sms.OLS(endog = y, exog = X_opt).fit()
reg_ols.summary()
```

```
C:\Users\expert\Anaconda3\lib\site-packages\scipy\stats\stats.py:1334: UserWar
ning: kurtosistest only valid for n>=20 ... continuing anyway, n=18
  "anyway, n=%i" % int(n))
```

Out[10]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.991
Model:	OLS	**Adj. R-squared:**	0.989
Method:	Least Squares	**F-statistic:**	532.8
Date:	Sun, 01 Apr 2018	**Prob (F-statistic):**	1.87e-15
Time:	09:15:15	**Log-Likelihood:**	-17.773
No. Observations:	18	**AIC:**	41.55
Df Residuals:	15	**BIC:**	44.22
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	0.3521	0.093	3.801	0.002	0.155	0.550
x2	0.6564	0.093	7.079	0.000	0.459	0.854
x3	0.2948	0.084	3.499	0.003	0.115	0.474

Omnibus:	0.947	**Durbin-Watson:**	2.802
Prob(Omnibus):	0.623	**Jarque-Bera (JB):**	0.877
Skew:	-0.442	**Prob(JB):**	0.645
Kurtosis:	2.377	**Cond. No.**	6.37

In [11]:
```
#4th iteration
X_opt = X[: , [1,3]]
reg_ols  = sms.OLS(endog = y, exog = X_opt).fit()
reg_ols.summary()
```

```
C:\Users\expert\Anaconda3\lib\site-packages\scipy\stats\stats.py:1334: UserWar
ning: kurtosistest only valid for n>=20 ... continuing anyway, n=18
  "anyway, n=%i" % int(n))
```

Out[11]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.983
Model:	OLS	**Adj. R-squared:**	0.981
Method:	Least Squares	**F-statistic:**	465.8
Date:	Sun, 01 Apr 2018	**Prob (F-statistic):**	6.60e-15
Time:	09:15:38	**Log-Likelihood:**	-23.143
No. Observations:	18	**AIC:**	50.29
Df Residuals:	16	**BIC:**	52.07
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
x1	0.5360	0.100	5.387	0.000	0.325	0.747
x2	0.7568	0.115	6.577	0.000	0.513	1.001

Omnibus:	2.692	**Durbin-Watson:**	2.011
Prob(Omnibus):	0.260	**Jarque-Bera (JB):**	1.008
Skew:	-0.474	**Prob(JB):**	0.604
Kurtosis:	3.668	**Cond. No.**	5.06

After each iteration the most insigificant module or 'test' was eliminated. So, Here are the ranking:

1. Logic
2. Object
3. Social
4. Emotion
5. Verbal

Note: All this calculation are based on Dummy data, just for testing the correctness of model. Actual value may changed when data used in algorithm is based on test performed by kids in real world.