GROUP 4

Predict portion of time (%) that CPUs run in user mode

Members:

1. Neeraj Bhukania (13MA20025)
2. Priyank Yadav (13MA20031)
3. Harshit Khandelwal (13MA20051)
4. Ayush Bhargava (13MA20013)
5. Divesh Hura (13ME10020)
6. Satyam Jha (13MA20036)
7. Arpan Agrawal (13MA20012)

Under the guidance of Prof Somesh Kumar

ABOUT THE DATASET:

- The database consists of a collection of a computer systems activity measures. The data was collected from a Sun Sparcstation 20/712 with 128 Mbytes of memory running in a multi-user university department.
- Users would typically be doing a large variety of tasks ranging from accessing the internet, editing files or running very CPU-bound programs.
- The data was collected continuously on two separate occasions. On both occasions, system activity was gathered every 5 seconds. The final dataset is taken from both occasions with equal numbers of observations coming from each collection epoch in random order.

The dataset consists of 22 attributes all of which are numerical.
- **lread** - Reads (transfers per second) between system memory and user memory.
- **lwrite** - Writes (transfers per second) between system memory and user memory.
- **scall** - Number of system calls of all types per second.
- **sread** - Number of system read calls per second.
- **swrite** - Number of system write calls per second.
- **fork** - Number of system fork calls per second.
- **exec** - Number of system exec calls per second.
- **rchar** - Number of characters transferred per second by system read calls.
- **wchar** - Number of characters transferred per second by system write calls.
- **pgout** - Number of page out requests per second.
- **ppgout** - Number of pages, paged out per second.
- **pgfree** - Number of pages per second placed on the free list.
- **pgscan** - Number of pages checked if they can be freed per second.
- **atch** - Number of page attaches (satisfying a page fault by reclaiming a page in memory) per second.
- **pgin** - Number of page-in requests per second.
- **ppgin** - Number of pages paged in per second.
- **pflt** - Number of page faults caused by protection errors (copy-on-writes).
- **vflt** - Number of page faults caused by address translation.
- **runqsz** - Process run queue size.
- **freemem** - Number of memory pages available to user processes.
- **freeswap** - Number of disk blocks available for page swapping.
- **usr** - Portion of time (%) that CPUs run in user mode

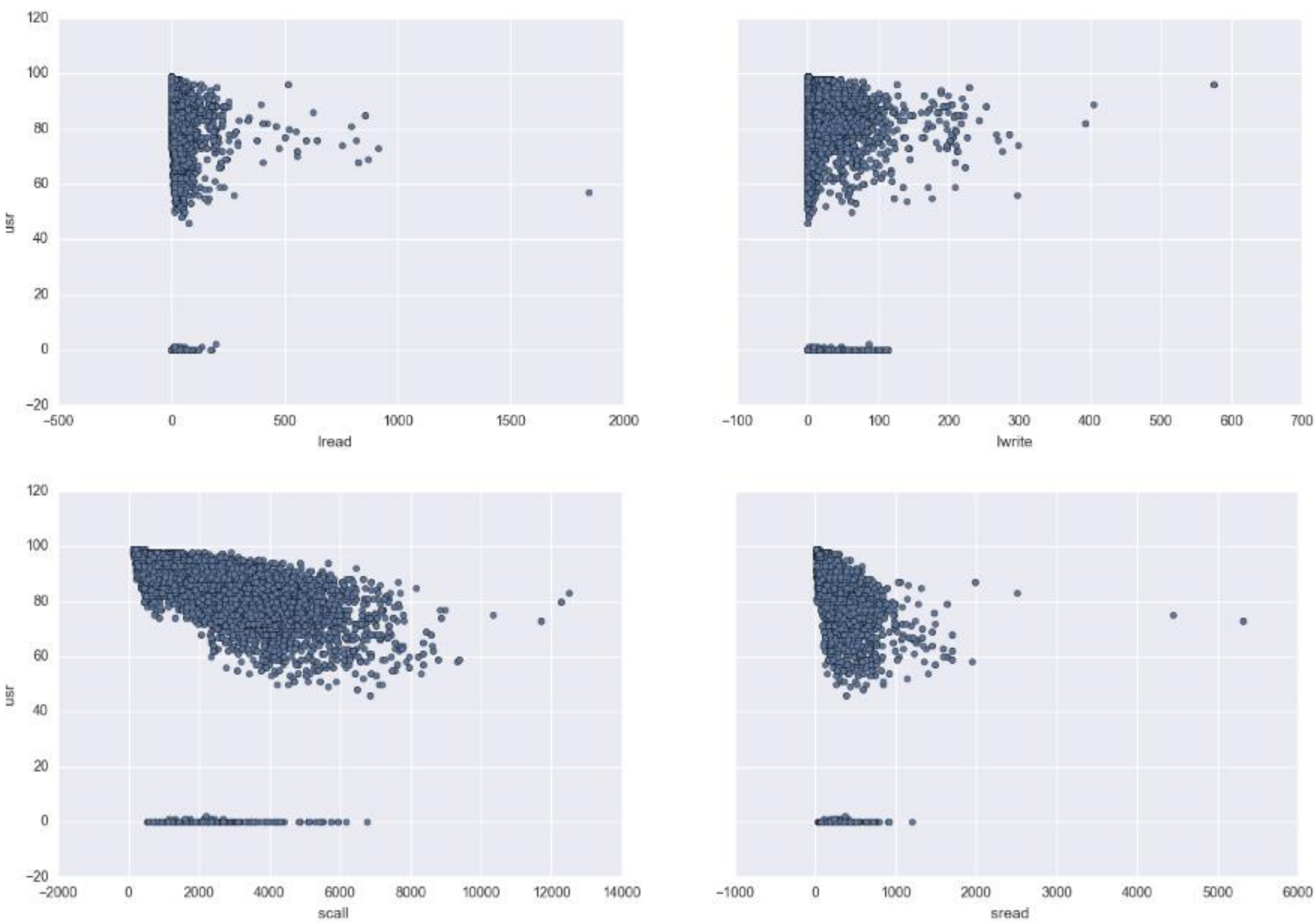
OBJECTIVE:

To predict **usr**- the portion of time (%) that CPUs run in user mode

About the Model:

- Since we did not have a test set, we split the entire dataset into training and test with (60:40) ratio.
- First we trained our Ordinary Least Squares Regression on the training dataset and found out that we have high condition number due to multicollinearity. We also had a high value of R^2 due to said multicollinearity
- Then we diagnosed multicollinearity and removed the features contributing to it.
- The model was again trained on the new training set and the values for the test set were predicted.
- The residuals were calculated and analyzed.

Scatter plots of features vs the dependent variable



Initial Regression Summary

Dep. Variable:	usr	**R-squared:**	0.967
Model:	OLS	**Adj. R-squared:**	0.967
Method:	Least Squares	**F-statistic:**	1.542e+04
Date:	Wed, 02 Nov 2016	**Prob (F-statistic):**	0.00
Time:	08:13:02	**Log-Likelihood:**	-47735.
No. Observations:	11468	**AIC:**	9.551e+04
Df Residuals:	11446	**BIC:**	9.568e+04
Df Model:	22		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
lread	-0.0335	0.003	-10.145	0.000	-0.040 -0.027
lwrite	0.0455	0.006	7.910	0.000	0.034 0.057
scall	0.0028	0.000	20.921	0.000	0.003 0.003
sread	0.0073	0.002	3.856	0.000	0.004 0.011
swrite	-0.0031	0.002	-1.526	0.127	-0.007 0.001
fork	-5.5780	0.244	-22.873	0.000	-6.056 -5.100
exec	0.3819	0.048	7.952	0.000	0.288 0.476
rchar	-4.64e-06	8.6e-07	-5.395	0.000	-6.33e-06 -2.95e-06
wchar	1.155e-06	1.31e-06	0.881	0.378	-1.41e-06 3.72e-06
pgout	0.5544	0.063	8.829	0.000	0.431 0.678
ppgout	-0.1380	0.036	-3.800	0.000	-0.209 -0.067
pgfree	-0.1026	0.019	-5.462	0.000	-0.139 -0.066
pgscan	0.0193	0.006	3.467	0.001	0.008 0.030
atch	-0.0073	0.028	-0.265	0.791	-0.062 0.047
pgin	0.1261	0.030	4.227	0.000	0.068 0.185
ppgin	-0.0195	0.019	-1.002	0.317	-0.058 0.019
pflt	-0.0622	0.004	-14.518	0.000	-0.071 -0.054
vflt	0.0829	0.003	25.730	0.000	0.077 0.089
runqsz	-0.0050	0.002	-2.936	0.003	-0.008 -0.002
runocc	-6.493e-06	1.24e-06	-5.238	0.000	-8.92e-06 -4.06e-06
freemem	-0.0020	7.48e-05	-26.756	0.000	-0.002 -0.002
freeswap	5.963e-05	2.27e-07	262.627	0.000	5.92e-05 6.01e-05

Omnibus:	70.267	**Durbin-Watson:**	1.990
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	51.908
Skew:	0.058	**Prob(JB):**	5.35e-12
Kurtosis:	2.691	**Cond. No.**	2.38e+06

Coefficients of Regression:

	0	1
0	lread	-9.766547e-03
1	lwrite	-2.560977e-03
2	scall	-1.429981e-03
3	sread	7.535274e-04
4	swrite	-3.687619e-03
5	fork	1.330016e-01
6	exec	-2.994976e-01
7	rchar	-1.404658e-06
8	wchar	-4.352555e-06
9	pgout	-7.467340e-02
10	ppgout	5.305402e-03
11	pgfree	-9.631219e-03
12	pgscan	2.320223e-03
13	atch	-3.667315e-03
14	pgin	-1.940026e-02
15	ppgin	-3.649881e-02
16	pflt	-1.698672e-02
17	vflt	-1.475070e-02
18	runqsz	2.358057e-03
19	runocc	-9.186837e-05
20	freemem	1.854078e-04
21	freeswap	-5.885929e-07

R^2= 0.97706391181526542

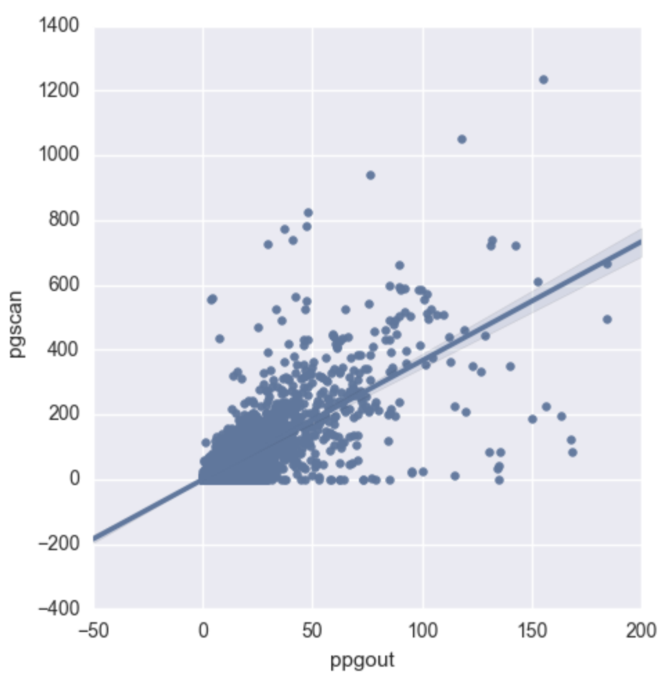
MULTICOLLINEARITY DIAGONOSIS:

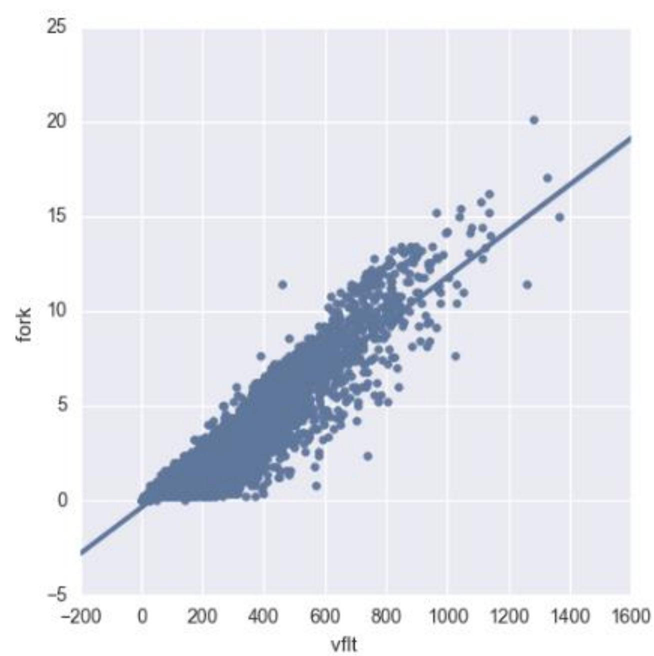
- Examination of the correlation matrix
- Variance Inflation Factor calculation
- Eigensystem analysis of X'X

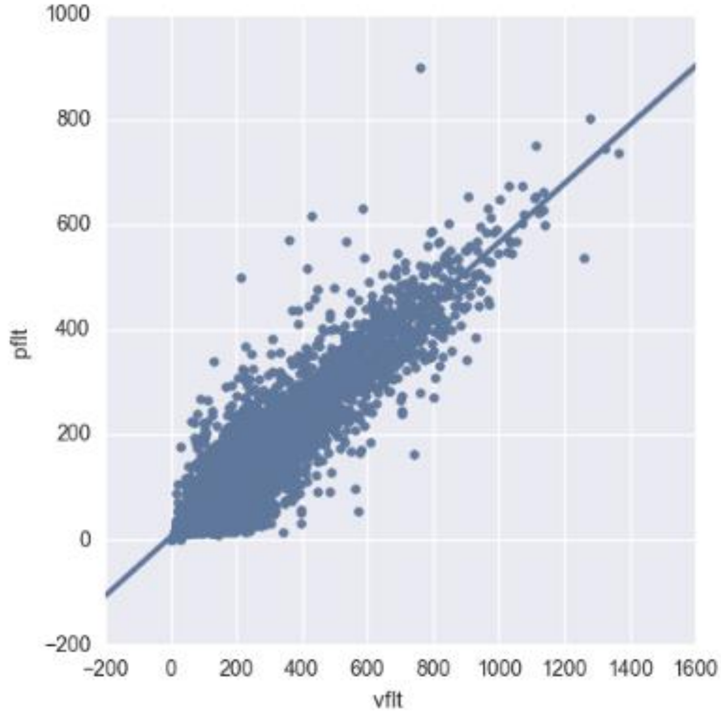
Examination of the Correlation Matrix

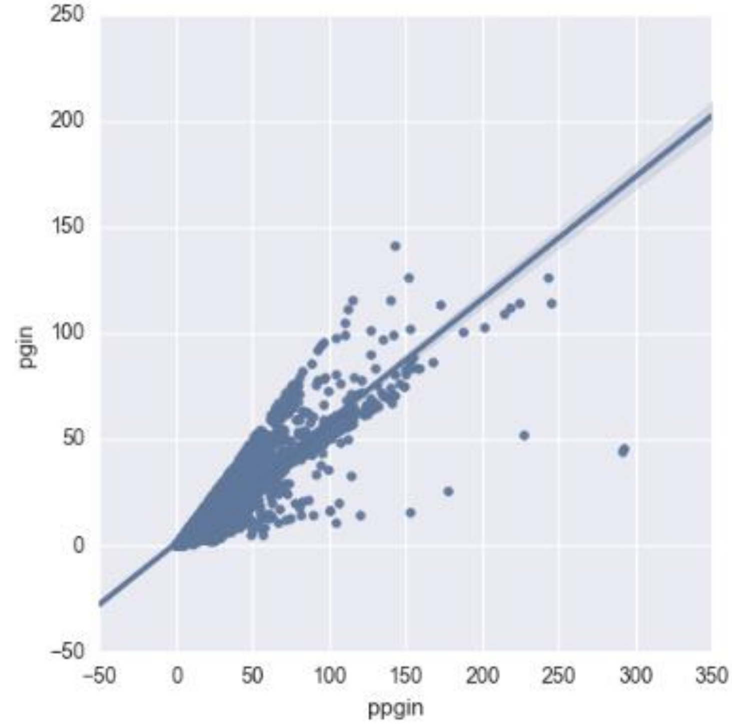
- If regressors x_i and x_j are nearly linearly dependent, then $|r_{ij}|$ will be near unity
- Examining the simple correlations r_{ij} between the regressors is helpful in detecting near-linear dependence between pairs of regressors only

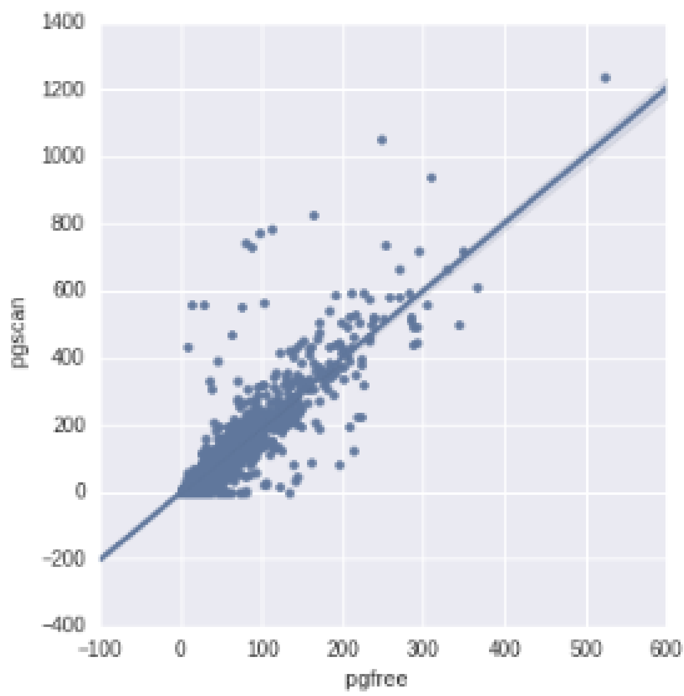
```
swrite   sread    0.882400420819
exec     fork     0.762592480586
pflt     fork     0.932327925446
vflt     fork     0.939741683726
ppgout   pgout    0.869241931534
pgfree   pgout    0.721936319599
pgfree   ppgout   0.916330717524
pgscan   ppgout   0.780116397143
pgscan   pgfree   0.914827178376
ppgin    pgin     0.92836881544
vflt     pflt     0.936623701651
runocc   runqsz   0.718176687589
```

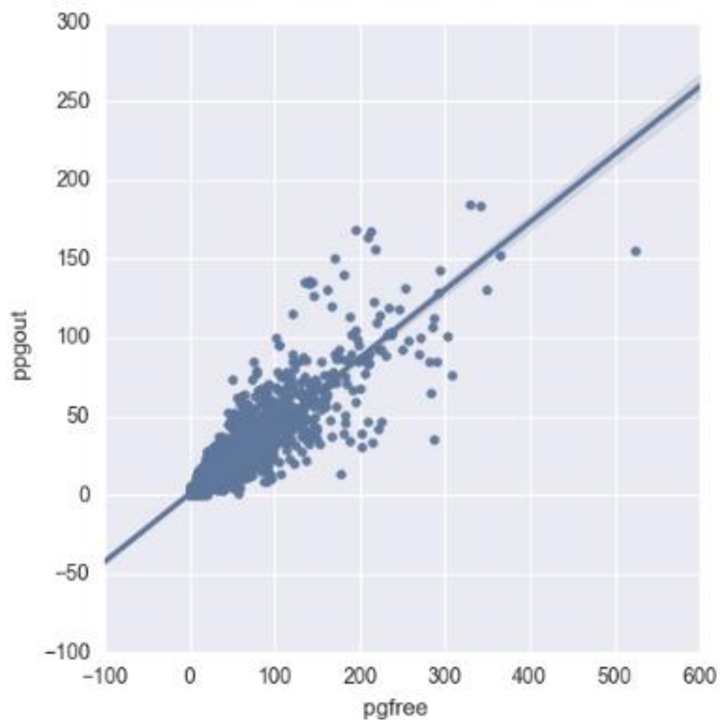





1000
800
600
400
200
0
-200
pflt
-200 0 200 400 600 800 1000 1200 1400 1600
vflt


250
200
150
100
50
0
-50
pgin
-50 0 50 100 150 200 250 300 350
ppgin


1400
1200
1000
800
600
400
200
0
-200
-400
pgscan
-100 0 100 200 300 400 500 600
pgfree


300
250
200
150
100
50
0
-50
-100
ppgout
-100 0 100 200 300 400 500 600
pgfree

Correlation Matrix

	lread	lwrite	scall	sread	swrite	fork	exec	rchar	wchar	pgout	...	pgscan	atch
lread	1.000000	0.499844	0.186200	0.120380	0.113440	0.132223	0.105046	0.099622	0.075593	0.078487	...	0.074096	0.021262
lwrite	0.499844	1.000000	0.146741	0.129631	0.105951	0.041909	0.034465	0.118197	0.093389	0.071242	...	0.052094	0.031432
scall	0.186200	0.146741	1.000000	0.700398	0.625059	0.442940	0.307162	0.356368	0.273215	0.196399	...	0.186445	0.078301
sread	0.120380	0.129631	0.700398	1.000000	0.873570	0.419047	0.161758	0.522610	0.412021	0.184068	...	0.205606	0.080848
swrite	0.113440	0.105951	0.625059	0.873570	1.000000	0.384504	0.102439	0.352578	0.406076	0.143721	...	0.124157	0.057862
fork	0.132223	0.041909	0.442940	0.419047	0.384504	1.000000	0.763886	0.286032	0.065020	0.121880	...	0.155465	0.040120
exec	0.105046	0.034465	0.307162	0.161758	0.102439	0.763886	1.000000	0.166577	0.003451	0.110812	...	0.148349	0.052763
rchar	0.099622	0.118197	0.356368	0.522610	0.352578	0.286032	0.166577	1.000000	0.511981	0.208586	...	0.259357	0.171488
wchar	0.075593	0.093389	0.273215	0.412021	0.406076	0.065020	0.003451	0.511981	1.000000	0.196293	...	0.123419	0.179827
pgout	0.078487	0.071242	0.196399	0.184068	0.143721	0.121880	0.110812	0.208586	0.196293	1.000000	...	0.553282	0.158095
ppgout	0.115285	0.084193	0.211087	0.220172	0.152984	0.158859	0.151586	0.262598	0.191540	0.877947	...	0.775254	0.101647
pgfree	0.098525	0.073524	0.207777	0.216242	0.146691	0.161992	0.150459	0.276686	0.165754	0.741927	...	0.908353	0.076162
pgscan	0.074096	0.052094	0.186445	0.205606	0.124157	0.155465	0.148349	0.259357	0.123419	0.553282	...	1.000000	0.041355
atch	0.021262	0.031432	0.078301	0.080848	0.057862	0.040120	0.052763	0.171488	0.179827	0.158095	...	0.041355	1.000000
pgin	0.184929	0.088303	0.244461	0.205474	0.147474	0.165117	0.189487	0.292248	0.169523	0.379031	...	0.496114	0.060896
ppgin	0.156250	0.088376	0.224441	0.214415	0.147490	0.134270	0.153824	0.339194	0.196099	0.408805	...	0.572292	0.059314
pflt	0.127878	0.057747	0.476753	0.454318	0.404338	0.931814	0.644817	0.315157	0.088977	0.145334	...	0.173057	0.042893
vflt	0.152978	0.085074	0.530620	0.495024	0.426615	0.939722	0.692647	0.365414	0.114861	0.218224	...	0.268579	0.089576
runqsz	0.020741	0.040907	-0.000360	0.050551	0.017594	-0.018566	-0.005874	0.142271	0.210907	-0.012479	...	-0.020665	0.247183

Now with the help of the correlation matrix we have determined that the columns that we might have to remove are:

- sread
- fork
- pgout
- pgscan
- pgfree
- pgin
- pfit runocc

Variance Inflation Factor Calculation

Since the variance of the jth regression coefficients is $C_{jj}\sigma^2$, we can view C_{jj} as the factor by which the variance of is increased due to near linear dependences among the regressors

$$VIF_j = C_{jj} = (1-R_j^2)^{-1}$$

```
VIF1 1.70688996782
VIF2 1.71140040889
VIF3 6.87277930799
VIF4 14.4861014344
VIF5 9.99473935188
VIF6 27.7600297193
VIF7 3.85210100251
VIF8 3.30511662728
VIF9 2.33117017385
VIF10 6.40814772778
VIF11 17.2696266413
VIF12 20.3900727394
VIF13 8.27692917546
VIF14 1.16976359731
VIF15 11.06555766
VIF16 11.5083085167
VIF17 21.9897042466
VIF18 35.0293359152
VIF19 2.27035078797
VIF20 2.29344951172
VIF21 2.43883512914
VIF22 4.75203350263
```

The VIF for each term in the model measures the combined effect of the dependences among the regressors on the variance of that term.

Practical experience indicates that if any of the VIFs exceeds 5 or 10, it is an indication that the associated regression coefficients are poorly estimated because of multicollinearity.

The above Variance inflation Factor Analysis indicates that we've got some features due to which multicollinearity arises

Eigensystem Analysis

- The characteristic roots or eigenvalues of X'X, say λ_1 , λ_2 ,..., λ_p , can be used to measure the extent of multicollinearity in the data.
- If there are one or more near-linear dependences in the data, then one or more of the characteristic roots will be small.
- One or more small eigenvalues imply that there are near-linear dependencies among the columns of X.

Eigenvalues:

```
array([ 9.51951377,  2.72118944,  1.82682823,  1.35897483,  1.18784139,
        1.00018536,  0.91274753,  0.81301255,  0.49169641,  0.42546776,
        0.39439462,  0.29619037,  0.26284072,  0.24297053,  0.19591756,
        0.11421002,  0.06383148,  0.0189818 ,  0.0498973 ,  0.04446706,
        0.02999262,  0.02884864])
```

Condition number = 501.507488272

Generally, if the condition number is less than 100, there is no serious problem with multicollinearity.

Condition numbers between 100 and 1000 imply moderate to strong multicollinearity, and if κ exceeds 1000, severe multicollinearity is indicated.

$$\kappa_j = \frac{\lambda_{max}}{\lambda_j}, j = 1,2,3, \dots , p$$

Condition indices:

```
[1.0,
 3.4982914563232801,
 5.2109517554621139,
 7.0049227975963486,
 8.0141286968812455,
 9.5177495545356745,
 10.429514704295135,
 11.708938329034442,
 19.360551646402026,
 22.37423044757265,
 24.137027739612545,
 32.139849508988142,
 36.217804192126792,
 39.179704986158441,
 48.589384217280539,
 83.350949883250806,
 149.13508338504792,
 501.50748827228932,
 190.78212506585302,
 214.0801098669553,
 317.39524677739007,
 329.98135101740991]
```

There are 6 features whose kappa value is greater than 100,which means there are 6 features contribute to multicollinearity

The above found Kaapa values indicate that we have 6 features that lead to multicollinearity

SVD DECOMPOSITION:

We have divided X in 3 parts

- X = UDT' is called the singular-value decomposition of X.
- U is a matrix whose columns are the eigenvectors associated with the p nonzero eigenvalues of XX'
- T is a p × p orthogonal matrix whose columns are the eigenvectors of X'X

(D is a p × p diagonal matrix)

The singular-value decomposition is closely related to the concepts of eigenvalues and eigenvectors, since

$$\mathbf{X'X = (UDT')'UDT' = TD^2T' = T\wedge T}$$

The covariance matrix of $\hat{\beta}$ is

$$\text{Var}(\widehat{\boldsymbol{\beta}}) = \sigma^2(X'X)^{-1} = \sigma^2\mathbf{T}\wedge^{-\mathbf{1}}\mathbf{T'}$$

And the variance of the jth regression coefficient is the jth diagonal element of this matrix, or

$$\text{Var}(\hat{\beta}_j) = \sigma^2 \sum_{i=1}^{p} \frac{t_{ji}^2}{\mu_i^2} = \sigma^2 \sum_{i=1}^{p} \frac{t_{ji}^2}{\lambda_i}$$

Note also that apart from σ^2, the jth diagonal element of $\boldsymbol{T}\wedge^{-\mathbf{1}}\boldsymbol{T}'$ is the jth VIF, so

$$VIF_j = \sum_{i=1}^{p} \frac{t_{ji}^2}{\mu_i^2} = \sigma^2 \sum_{i=1}^{p} \frac{t_{ji}^2}{\lambda_i}$$

Clearly, one or more small singular values (or small eigenvalues) can dramatically inflate the variance of $\widehat{\beta_j}$ using the **variance decomposition proportions**, defines as

$$\pi_{ij} = \frac{t_{ji}^2/\mu_i^2}{VIF_j} \quad j = 1,2,\dots,p$$

if the variance decomposition proportions of a particular coefficient are explained by more than one features which can be observed from the (pie$_{ij}$) matrix (p_{ij}>0.5),then it is a strong indication that these features contribute to multi-collinearity of the coefficient

	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	...	B12	B13
lread	6.036906e-01	5.165274e-01	9.415045e-02	7.728108e-02	0.016470	0.477889	1.277402e-02	2.593920e-05	2.983275e-05	0.595455	...	0.054957	2.3087 03
lwrite	4.094230e-02	6.043840e-01	8.196935e-04	1.433635e-02	0.003499	0.281627	5.348902e-03	3.133938e-03	2.110888e-04	0.009468	...	0.025227	1.0113 03
scall	2.020416e-04	2.219172e-05	5.618933e-01	1.563486e-01	0.000824	0.188362	1.578843e-02	8.219201e-04	2.716425e-03	0.009008	...	0.000021	1.3399 04
sread	2.085749e-05	4.881446e-05	1.966367e-02	6.354115e-01	0.324032	0.009811	1.765260e-04	3.323268e-02	1.226922e-03	0.003737	...	0.005595	2.7804 04
swrite	5.260344e-06	1.409877e-05	1.226225e-04	3.834494e-01	0.466625	0.051559	6.226210e-03	2.131527e-02	1.027661e-02	0.001960	...	0.003750	4.1308 05
fork	1.934767e-05	1.438459e-04	3.553670e-03	1.471686e-03	0.006536	0.638024	2.794634e-02	5.137820e-06	5.327051e-05	0.000182	...	0.000025	6.0121 04
exec	9.719488e-06	5.134540e-05	5.598046e-03	4.976634e-04	0.014833	0.525218	2.889047e-01	2.481254e-04	4.315761e-04	0.000726	...	0.000526	3.1004 04

The pie matrix basically shows us the variance of a particular regression coefficient explained by each feature, so if more than one feature is explaining the variance for a coefficient that is a sure indication of multicollinearity.

```
{'B1': ['lread   ', 'lwrite  '],
 'B10': ['lread   ', 'pgout   ', 'ppgout  '],
 'B11': ['pgfree  ', 'pgscan  '],
 'B14': ['lread   ', 'pgin    '],
 'B15': ['lread   ', 'pgin    ', 'ppgin   '],
 'B16': ['fork    ', 'exec    ', 'pflt    '],
 'B17': ['lread   ',
  'lwrite  ',
  'scall   ',
  'fork    ',
  'rchar   ',
  'wchar   ',
  'atch    ',
  'vflt    '],
 'B19': ['runqsz  ', 'runocc  '],
 'B20': ['scall   ', 'freemem '],
 'B21': ['lwrite  ', 'scall   ', 'freemem ', 'freeswap'],
 'B5': ['lread   ', 'fork    ', 'exec    ']}
```

The above dict which we have obtained from pie matrix shows us the features that are contributing to multicollinearity and thereby causing errors in the calculation of our coefficients

Coefficients of Regression after removing features causing multicollinearity

	0	1
0	lread	-4.613051e-03
1	lwrite	-2.974563e-03
2	scall	-7.417510e-07
3	sread	-5.544192e-06
4	swrite	-9.250530e-02
5	fork	-1.067728e-02
6	exec	1.119657e-02
7	rchar	9.812504e-03
8	wchar	-5.110514e-02
9	pgout	-3.283252e-02
10	ppgout	2.107388e-03
11	pgfree	-8.810184e-05
12	pgscan	1.427864e-06

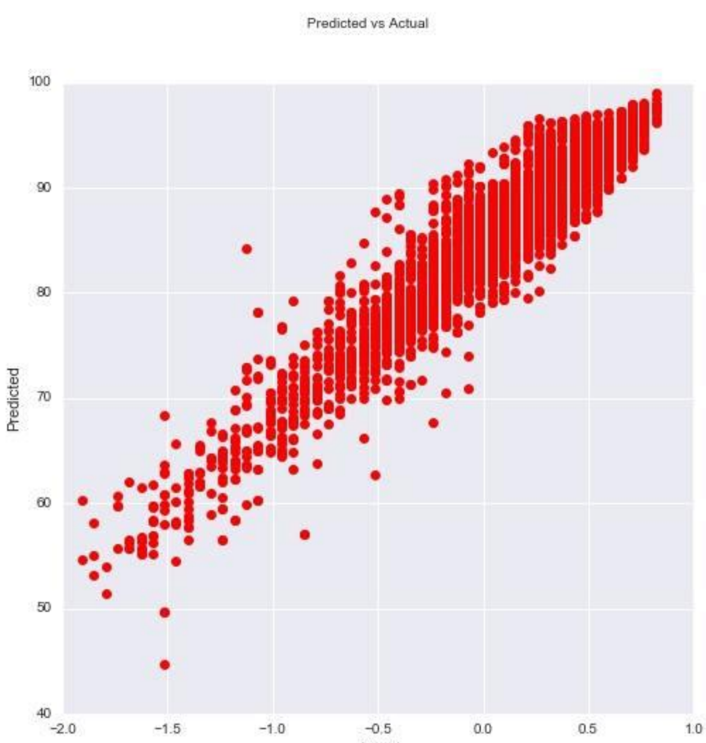
R^2= 0.96195994549056463

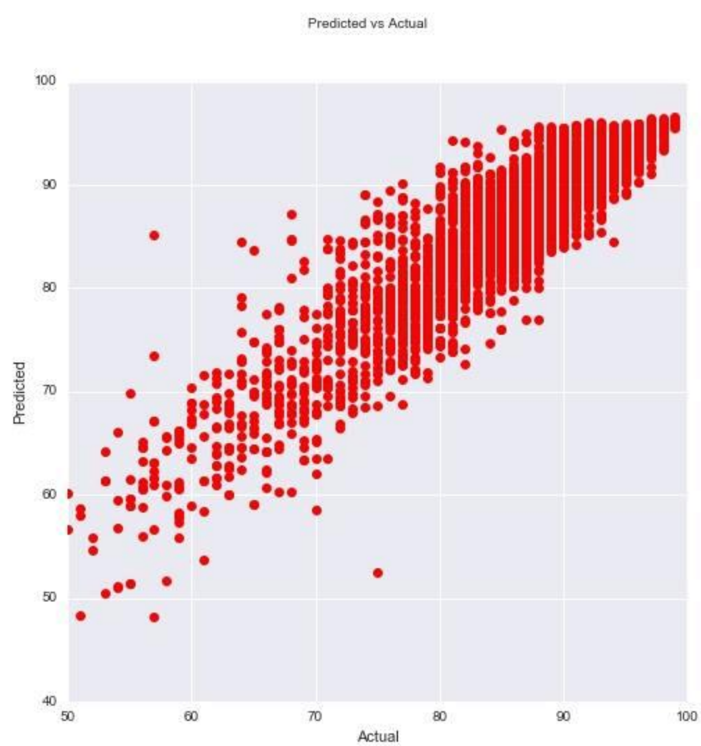
We can see here that even though our R squared value has reduced our accuracy has improved after removing multicollinearity

We can see that after treating multicollinearity, we are getting better predictions.

	residual	usr_predicted	usr
0	1.497382	84.502618	86.0
1	-1.188186	84.188186	83.0
2	2.062270	81.937730	84.0
3	0.282981	-0.282981	0.0
4	2.100502	74.899498	77.0
5	-6.228642	77.228642	71.0
6	-9.427723	59.427723	50.0
7	-3.356548	90.356548	87.0
8	2.772019	92.227981	95.0
9	2.751543	92.248457	95.0
10	-5.402214	91.402214	86.0
11	1.518545	95.481455	97.0
12	-0.857177	90.857177	90.0
13	-3.798974	72.798974	69.0
14	4.896606	91.103394	96.0
15	1.353746	84.646254	86.0
16	0.851687	90.148313	91.0
17	2.656261	77.343739	80.0
18	2.886036	94.113964	97.0
19	-5.538483	95.538483	90.0

	residual	usr_predicted	usr
0	1.795213	84.204787	86.0
1	0.341177	82.658823	83.0
2	0.832499	83.167501	84.0
3	-2.377485	2.377485	0.0
4	-0.013084	77.013084	77.0
5	-2.915931	73.915931	71.0
6	-9.756106	59.756106	50.0
7	-4.036442	91.036442	87.0
8	2.098114	92.901886	95.0
9	2.398001	92.601999	95.0
10	-2.517448	88.517448	86.0
11	0.400643	96.599357	97.0
12	-0.013905	90.013905	90.0
13	-1.546153	70.546153	69.0
14	2.076790	93.923210	96.0
15	0.432793	85.567207	86.0
16	1.266798	89.733202	91.0
17	1.307363	78.692637	80.0
18	2.845918	94.154082	97.0
19	-6.239545	96.239545	90.0





Coefficients of Regression after scaling

	0	1
0	lread	-0.047938
1	lwrite	-0.028880
2	scall	-0.011916
3	sread	-0.040630
4	swrite	-0.023317
5	fork	-0.018666
6	exec	0.024663
7	rchar	0.003312
8	wchar	-0.059672
9	pgout	-0.343712
10	ppgout	0.015695
11	pgfree	-0.847327
12	pgscan	0.034794

R^2= 0.96180443682301775

After treating multicollinearity and performing Regression Analysis again, the final model is:

Dep. Variable:	y	**R-squared:**	0.961
Model:	OLS	**Adj. R-squared:**	0.961
Method:	Least Squares	**F-statistic:**	2.155e+04
Date:	Wed, 02 Nov 2016	**Prob (F-statistic):**	0.00
Time:	21:37:15	**Log-Likelihood:**	2289.5
No. Observations:	11468	**AIC:**	-4553.
Df Residuals:	11455	**BIC:**	-4458.
Df Model:	13		
Covariance Type:	nonrobust		

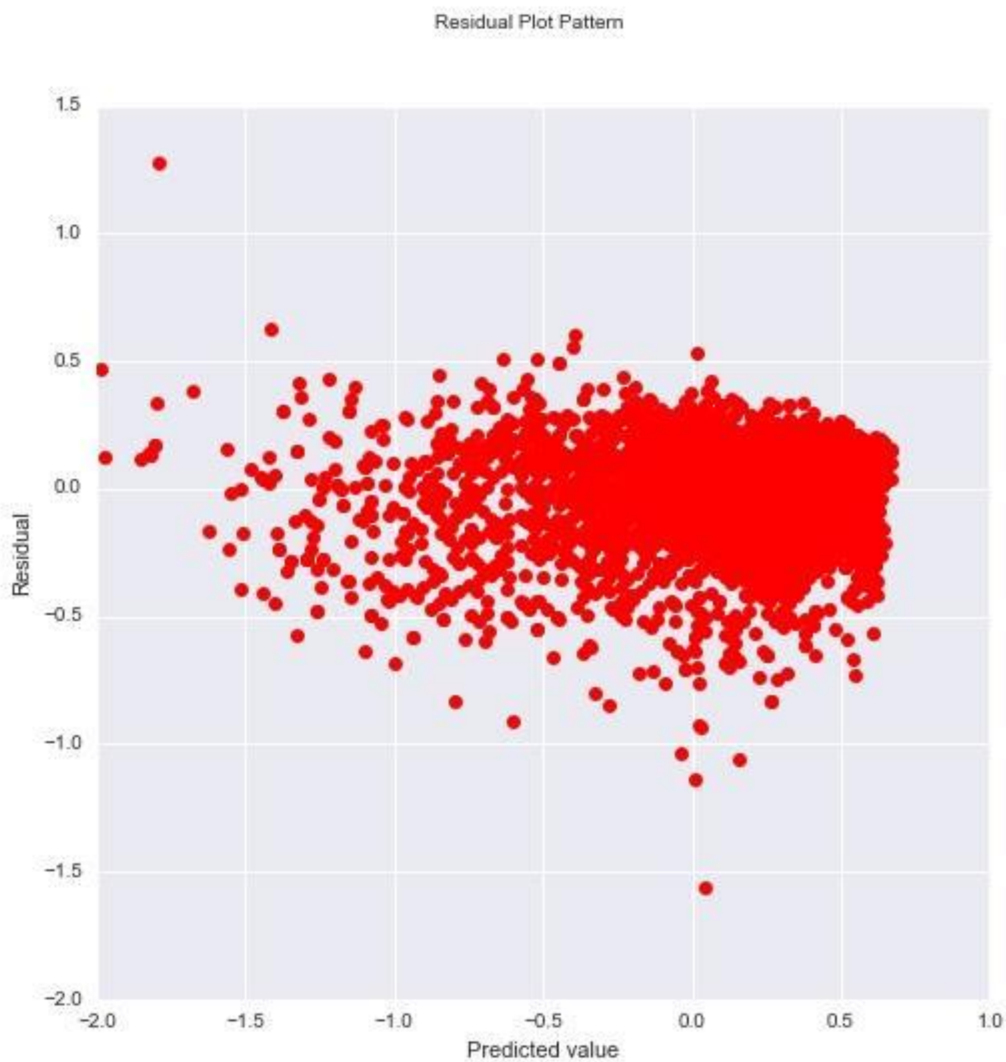
	coef	std err	t	P>|t|	[95.0% Conf. Int.]
x1	-0.0460	0.004	-10.283	0.000	-0.055 -0.037
x2	-0.0321	0.004	-7.880	0.000	-0.040 -0.024
x3	-0.0119	0.003	-4.577	0.000	-0.017 -0.007
x4	-0.0393	0.002	-16.600	0.000	-0.044 -0.035
x5	-0.0308	0.004	-7.268	0.000	-0.039 -0.022
x6	-0.0117	0.007	-1.676	0.094	-0.025 0.002
x7	0.0258	0.005	4.995	0.000	0.016 0.036
x8	0.0024	0.002	1.220	0.223	-0.001 0.006
x9	-0.0602	0.002	-24.711	0.000	-0.065 -0.055
x10	-0.3390	0.002	-149.692	0.000	-0.343 -0.335
x11	0.0133	0.003	4.758	0.000	0.008 0.019
x12	-0.8471	0.003	-272.688	0.000	-0.853 -0.841
x13	0.0339	0.003	13.055	0.000	0.029 0.039

Omnibus:	1532.638	**Durbin-Watson:**	1.993
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	13301.242
Skew:	-0.354	**Prob(JB):**	0.00
Kurtosis:	8.228	**Cond. No.**	9.69

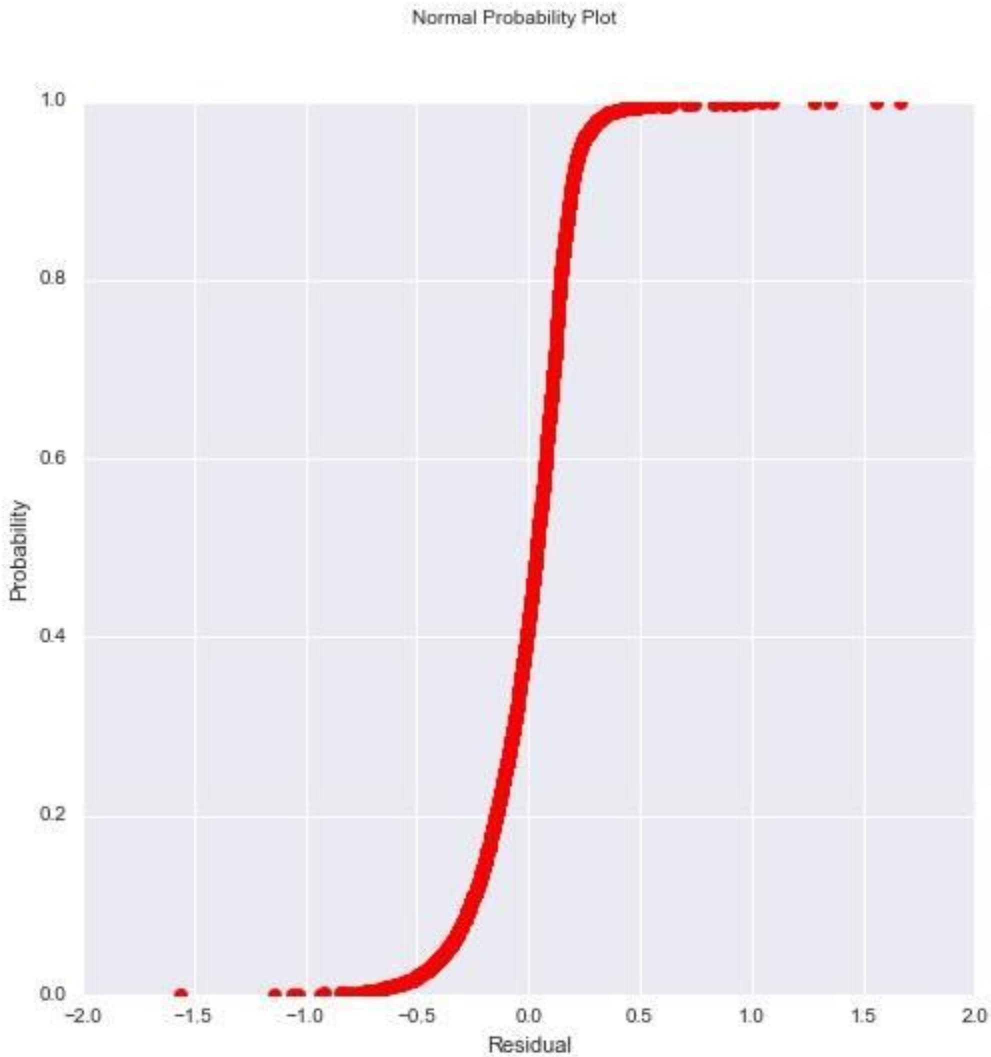
RESIDUAL ANALYSIS:

We carried out the graphical analysis of residuals as it is a very effective way to test adequacy of fit of a regression model and to check the underlying assumptions. We constructed three basic residual plots which are generated by matplotlib and seaborn.

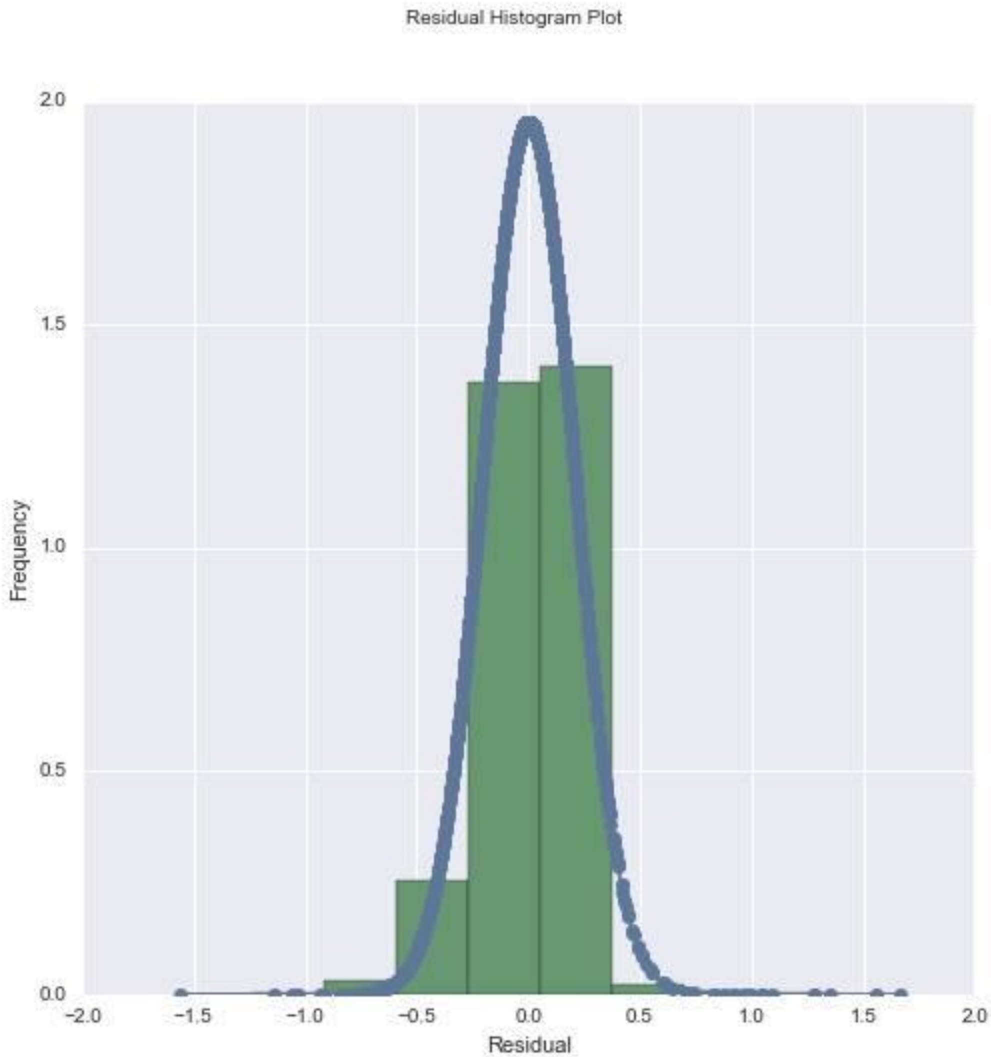
First Plot - Plot of residuals against fitted value. Since residuals are contained in a horizontal band there are no obvious model defects



Second Plot- Normal Probability Plot: Graph should lie on a straight line. We see it's a light-tailed distribution



Last Plot- We plot the residual histogram and see that it is following normal distribution which is what the plot should have been



Residual Mean = $1.0593*10^{-16}$

CONCLUSION:

- With stabilized Condition index and R^2 nearly 96% of regression is explained from the given factors after removing multicollinearity
- From our analysis of the given dataset, we may conclude that factors like Number of system buffer reads & write per sec, no of system calls per sec, pages free and scan per sec played a significant role in determining the percentage of CPU usage by the user.
- **Application** :
  a) Suppose we are in a data center where we want to allocate different tasks to different machines in an efficient and fast manner. If we can predict the user percentage before allocating the process to a machine, we will be able to decide whether allocating a specific task to a particular machine is a good decision or not.
  b) The model can also be used during a software development cycle where we have a goal of making the software lightweight and efficient. If we know the relationship between different types of calls/tasks with user CPU time then we'll be able to model our software accordingly.

Suggestions for improvement of model:

The above analysis has been done without transforming all the variables Involved in the linear regression model. Better results can be obtained by applying transformations to each variable to make the entire dataset closer to the definition of normality.

References:

http://www.cs.toronto.edu/~delve/data/comp-activ/desc.html