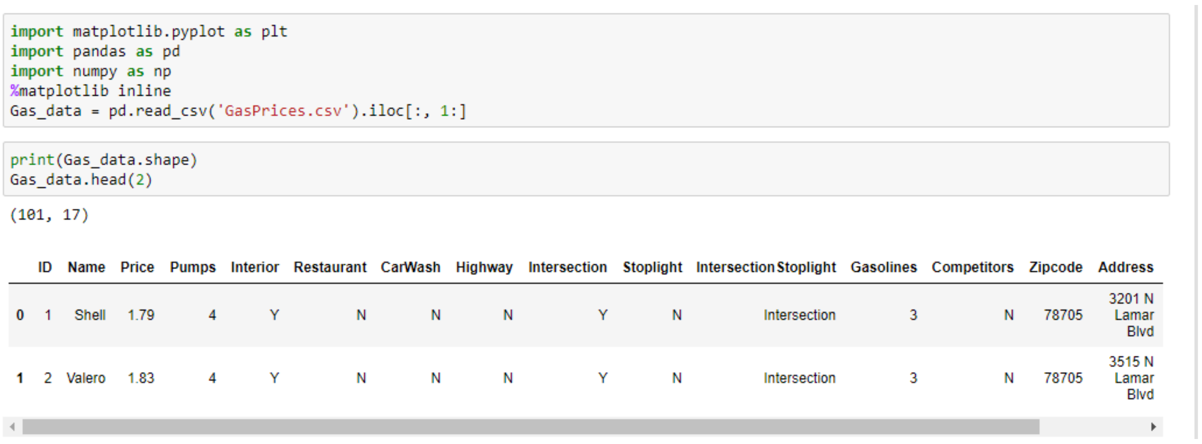
Lab Course Machine Learning Kalaiselvan Panneerselvam

Exercise 2 278920

1.1 Pandas

– Load the data using pandas



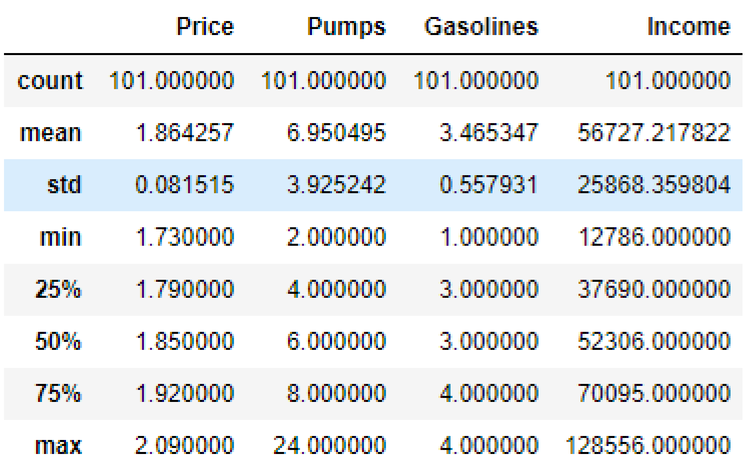
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
%matplotlib inline
Gas_data = pd.read_csv('GasPrices.csv').iloc[:, 1:]
```

```
print(Gas_data.shape)
Gas_data.head(2)
```

```
(101, 17)
```

	ID	Name	Price	Pumps	Interior	Restaurant	CarWash	Highway	Intersection	Stoplight	IntersectionStoplight	Gasolines	Competitors	Zipcode	Address
0	1	Shell	1.79	4	Y	N	N	N	Y	N	Intersection	3	N	78705	3201 N Lamar Blvd
1	2	Valero	1.83	4	Y	N	N	N	Y	N	Intersection	3	N	78705	3515 N Lamar Blvd

- **Summarize each NUMERIC field in the data, i.e. mean, average etc.**



	Price	Pumps	Gasolines	Income
count	101.000000	101.000000	101.000000	101.000000
mean	1.864257	6.950495	3.465347	56727.217822
std	0.081515	3.925242	0.557931	25868.359804
min	1.730000	2.000000	1.000000	12786.000000
25%	1.790000	4.000000	3.000000	37690.000000
50%	1.850000	6.000000	3.000000	52306.000000
75%	1.920000	8.000000	4.000000	70095.000000
max	2.090000	24.000000	4.000000	128556.000000

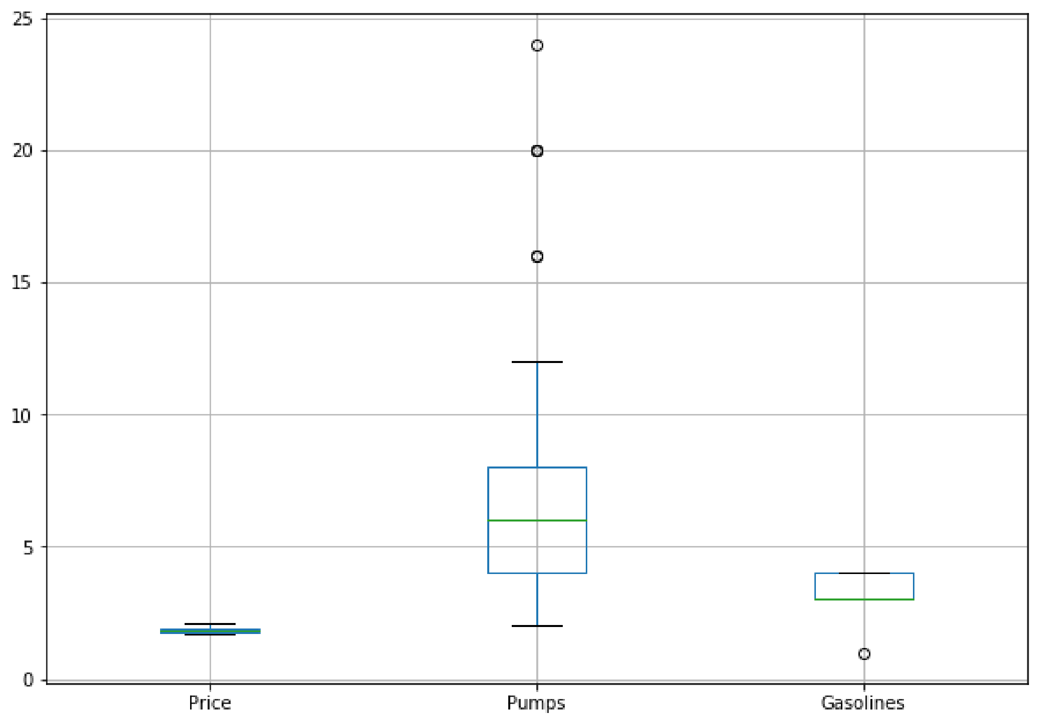
– Group data by the field 'Name'.

∗ Find the average price,average income and average number of pumps for each group.

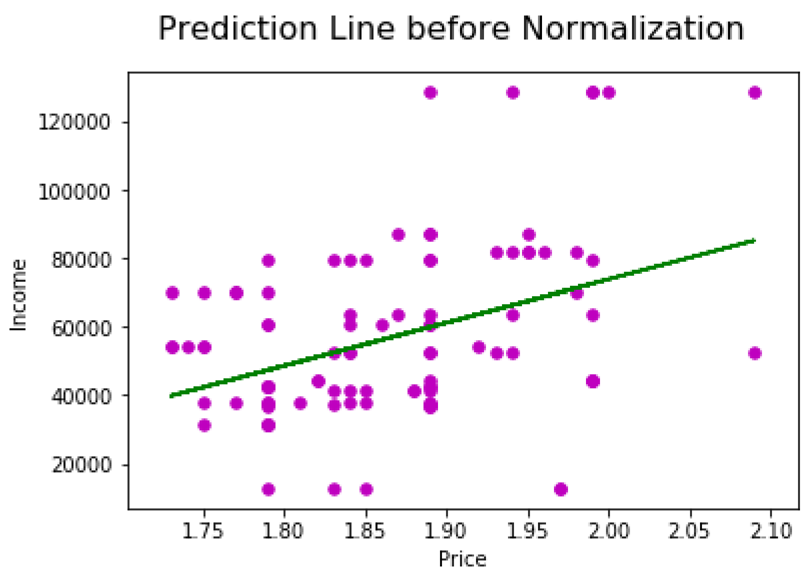
	Price	Pumps	Income
Name			
7-Eleven	1.887778	4.666667	53432.333333
Around the Corner Store	1.940000	2.000000	63750.000000
Chevron	1.871818	8.727273	61754.636364
Citgo	1.835000	4.000000	49387.000000
Conoco	1.890000	4.000000	43545.500000
Costco	1.730000	12.000000	70095.000000
Double R Grocery	1.790000	4.000000	37690.000000
East 1st Grocery	1.770000	4.000000	37690.000000
Exxon	1.855000	11.500000	52344.333333
Gulf	1.788571	5.714286	50084.142857
HEB Fuel	1.790000	11.000000	36903.500000
Kool Corner	1.790000	4.000000	42615.000000
Lamar Corner Store	1.890000	2.000000	37396.000000

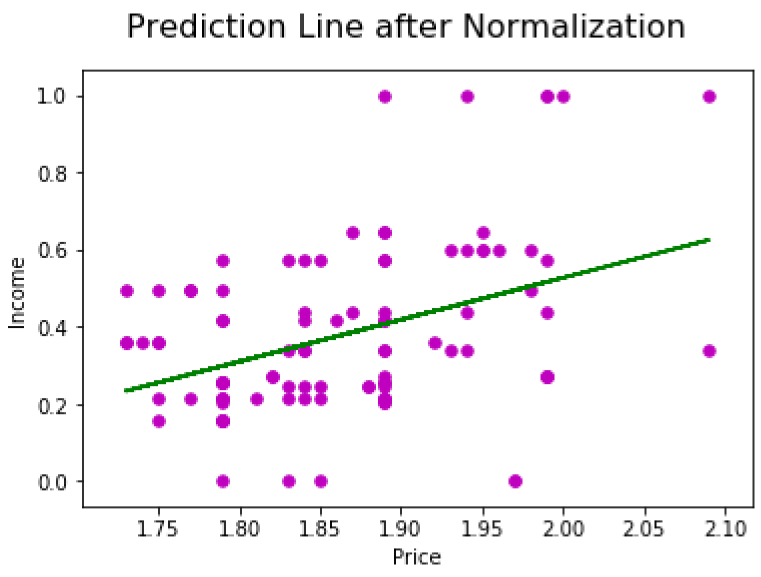
∗ **Use a boxplot that visualizes the statistical information about (price, pumps, gasoline).**

```
Gas_data[['Price','Pumps','Gasolines']].boxplot(figsize=(10,7))
plt.show()
```



***Use the Price and Income features in order to plot a prediction line similar to the first exercise. Normalize the Income (implement this yourself) and plot the line again. Comment on the different of the two plots.**





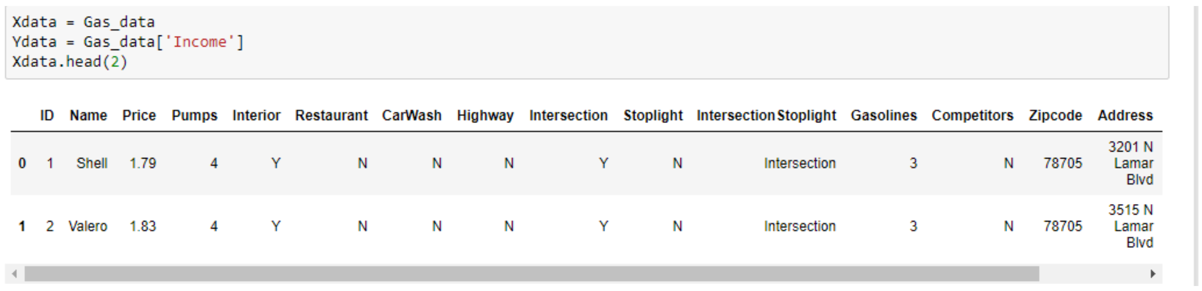
Normalization makes training less sensitive to the scale of features, so we can better solve for coefficients.

We can see that there are some odd behaviours with both features Income and Price as well as massive outliers and binning issues. We also have a clustering of income over 120,000 so the dataset probably puts anyone over that bracket into that bin. It's going to be hard to equate both these features as they are right now.

But after Normalization all the values are all now between 0 and 1, and the outliers are gone, but still remain visible within our normalized data. However, our features are now more consistent with each other, which will allow us to evaluate the output of our future models better.

1.2 Linear Regression via Normal Equations

***Reuse dataset from Excercise 1. Load it as Xdata, [Hint:] from loaded data you need to separate ydata i.e. Income, which is your target.**

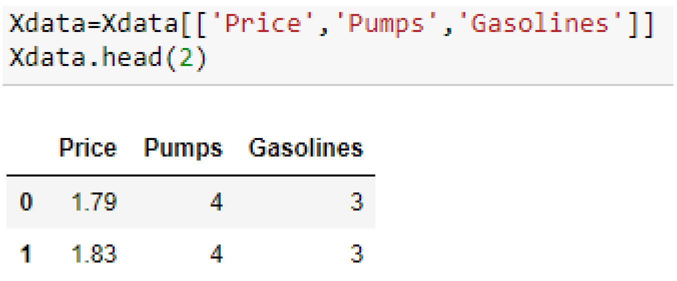


```
Xdata = Gas_data
Ydata = Gas_data['Income']
Xdata.head(2)
```

	ID	Name	Price	Pumps	Interior	Restaurant	CarWash	Highway	Intersection	Stoplight	IntersectionStoplight	Gasolines	Competitors	Zipcode	Address
0	1	Shell	1.79	4	Y	N	N	N	Y	N	Intersection	3	N	78705	3201 N Lamar Blvd
1	2	Valero	1.83	4	Y	N	N	N	Y	N	Intersection	3	N	78705	3515 N Lamar Blvd

• Choose those columns, which can help you in prediction i.e. contain some useful information. You can drop irrelevant columns. Give reason for choosing or dropping any column.

I have choosen only the numeric fields like Price, Pumps , Gasolines which affects the target variable Income. Some columns like Interior, Restaurant, CarWash also affects the income but since it is string variable I have not taken it into account for problem simplicity.



```
Xdata=Xdata[['Price','Pumps','Gasolines']]
Xdata.head(2)
```

	Price	Pumps	Gasolines
0	1.79	4	3
1	1.83	4	3

• Split your dataset Xdata, Ydata into Xtrain, YtrainandXtest, Ytest i.e. you can randomly assign 80% of the data to a Xtrain, Ytrain set and remaining 20% to a Xtest, ytest set.

```
msk = np.random.rand(len(Xdata))<0.8
Xtrain = Xdata[msk]
Ytrain = Ydata[msk]
Ytrain
```

```
0       12786
1       12786
2       41279
3       41279
4       41279
5       37396
6       37396
7       37396
8       37396
9       41279
10      12786
11      12786
```

***Implement learn-linreg-NormEq algorithm and learn a parameter vector β using Xtrain set. You have to learn a model to predict sales price of houses i.e. , ytest.**

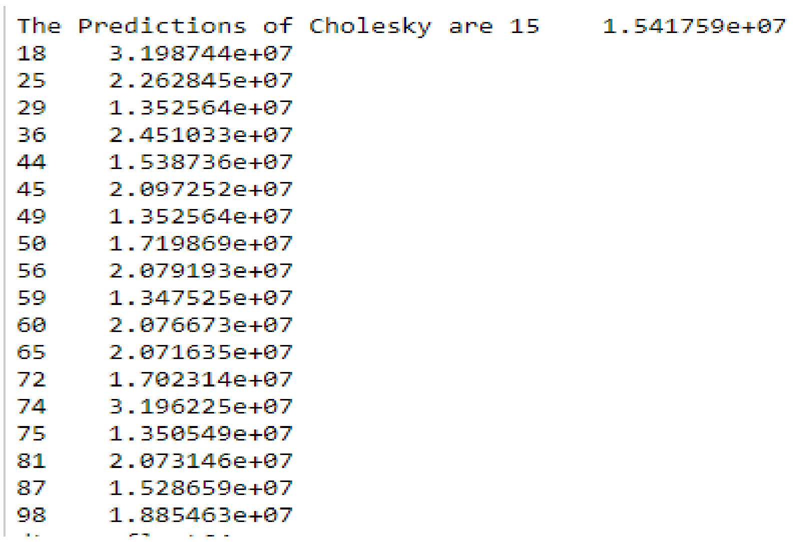
```
from numpy.linalg import inv
def learn_linreg_NormEq(X, y):
    #print("inside")
    X_transpose = X.T
    best_params = inv(X_transpose.dot(X)).dot(X_transpose).dot(y)
    # normal equation
    # theta_best = (X.T * X)^(-1) * X.T * y

    return best_params # returns a list
```

```
params=learn_linreg_NormEq(Xtrain,Ytrain)
print(params)
# test prediction
y_prediction = Xtest.dot(params)
# y = h_Theta_X(Theta) = Theta.T * X
#print(y_prediction.shape)
residual=np.abs(y_prediction-Ytest)
#print(residual.shape)
```

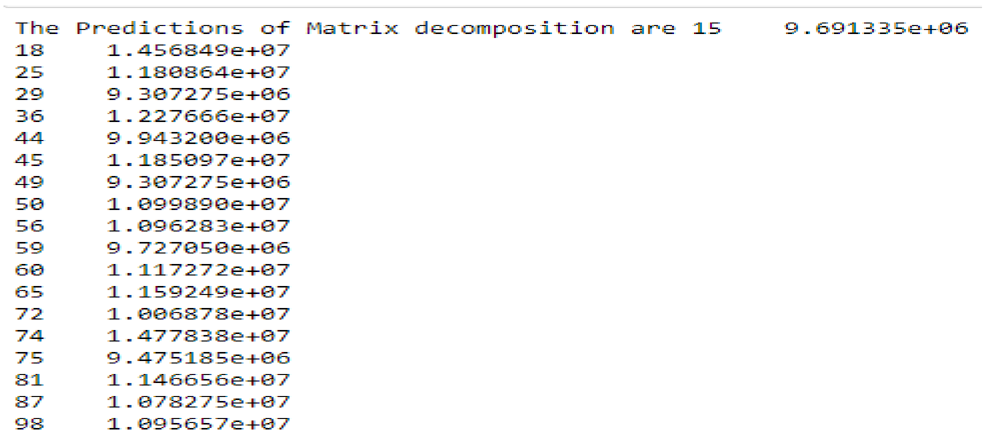
```
[26692.47095013   721.8004919    872.82788988]
```

***Perform prediction ¯y on test dataset i.e. Xtest using the set of parameters learned in steps 5**

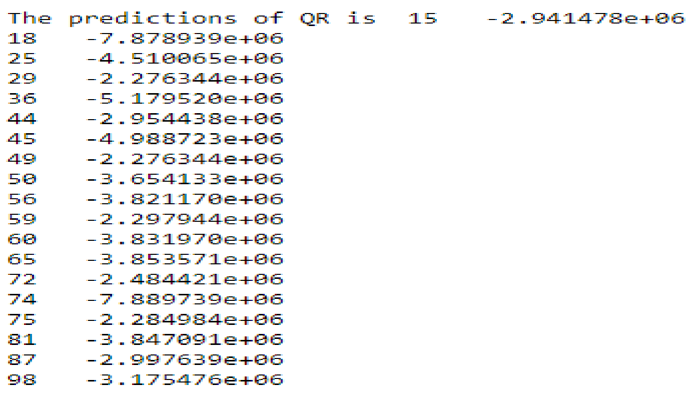
and 6



```
The Predictions of Cholesky are 15     1.541759e+07
18     3.198744e+07
25     2.262845e+07
29     1.352564e+07
36     2.451033e+07
44     1.538736e+07
45     2.097252e+07
49     1.352564e+07
50     1.719869e+07
56     2.079193e+07
59     1.347525e+07
60     2.076673e+07
65     2.071635e+07
72     1.702314e+07
74     3.196225e+07
75     1.350549e+07
81     2.073146e+07
87     1.528659e+07
98     1.885463e+07
```



```
The Predictions of Matrix decomposition are 15     9.691335e+06
18     1.456849e+07
25     1.180864e+07
29     9.307275e+06
36     1.227666e+07
44     9.943200e+06
45     1.185097e+07
49     9.307275e+06
50     1.099890e+07
56     1.096283e+07
59     9.727050e+06
60     1.117272e+07
65     1.159249e+07
72     1.006878e+07
74     1.477838e+07
75     9.475185e+06
81     1.146656e+07
87     1.078275e+07
98     1.095657e+07
```

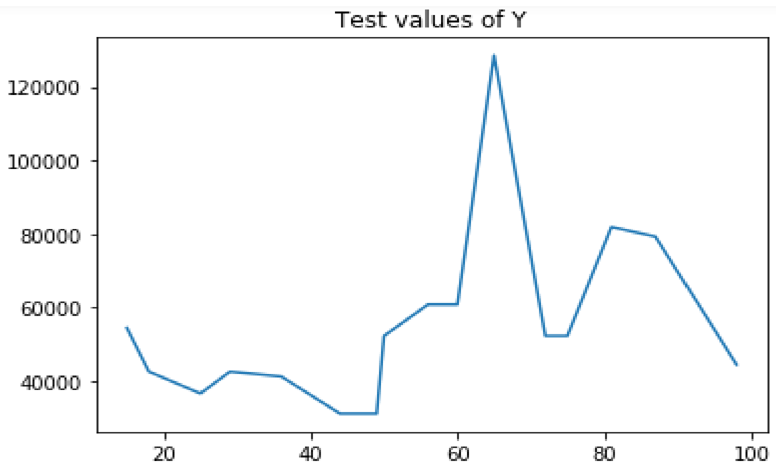


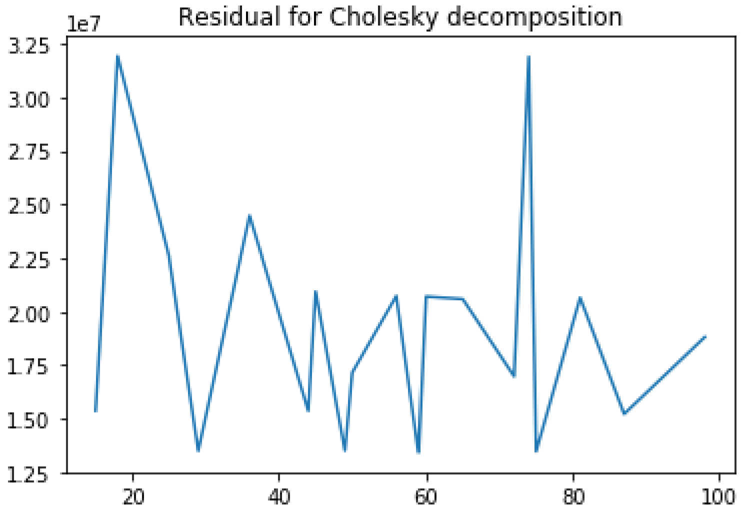
```
The predictions of QR is  15    -2.941478e+06
18    -7.878939e+06
25    -4.510065e+06
29    -2.276344e+06
36    -5.179520e+06
44    -2.954438e+06
45    -4.988723e+06
49    -2.276344e+06
50    -3.654133e+06
56    -3.821170e+06
59    -2.297944e+06
60    -3.831970e+06
65    -3.853571e+06
72    -2.484421e+06
74    -7.889739e+06
75    -2.284984e+06
81    -3.847091e+06
87    -2.997639e+06
98    -3.175476e+06
```

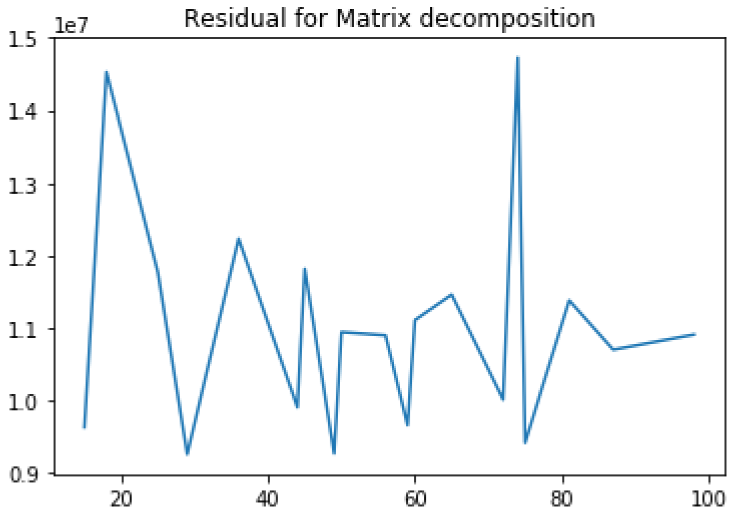
- **Final step is to find how close these three models are to the original values.**

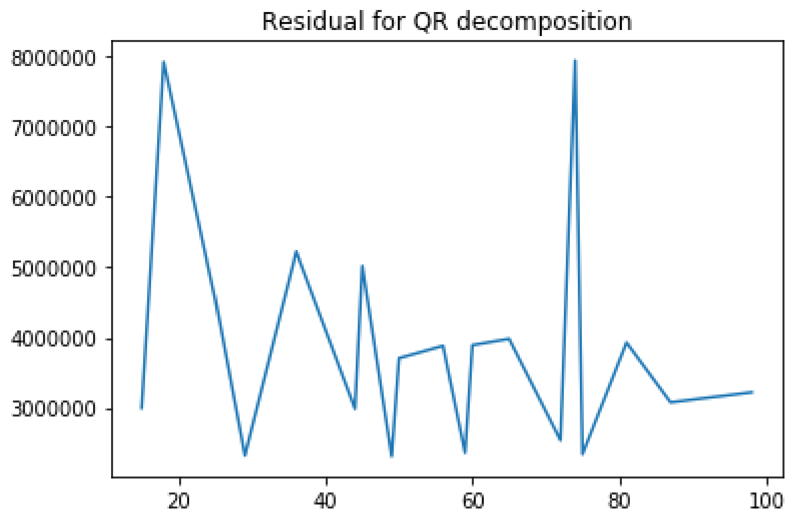
– plot residual E = |ytest – y¯| vs true value of ytest for each model.

Below is the plot of true values (test values) of Y ie Income vs the residual mod(Ytest-y^).


Test values of Y
120000
100000
80000
60000
40000
20
40
60
80
100


Residual for Cholesky decomposition
1e7
3.25
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
20
40
60
80
100


Residual for Matrix decomposition
1e7
1.5
1.4
1.3
1.2
1.1
1.0
0.9
20
40
60
80
100



– Find the average residual ⍰ = |ytest – y¯| of each model.

```
The average error of Cholesky is  19327901.28860306

The average error of Matrix decomposition is  11036774.090821216

The average error of QR is  3904281.001148893
```

– Find the root-mean-square error (RMSE) = qPN n=1(ytest(n)−y¯(n))2) N N of each model.

```
The RMSE of Cholesky model is  20077352.96129315
The RMSE of Gaussian Matrix Decomposition 11140783.175918583
The RMSE of QR Decomposition 4230855.170320108
```