Arkar Nyan Hein

Submitted Partially to Fulfil the Requirement To Complete Master of Engineering (Mechnical)

Unit Coordinator-Simon Denman

Name- Ar kar Nyan Hein

ID Number-n8709041

Table of Contents

List Of Figures

List Of Tables

1.0 Introduction

This report aims to analyse and predict the data using different methods under four sections.

The first section of this report predicts the data using the linear regression method along with training data. The second section includes ridge and lasso regression to fit the data into a better fit model, comparing the different data.

As for the third section, the k-means clustering method is used to predict the power meter usage under season and timeline. The final section involves using hierarchical clustering (HAC)and density-based spatial clustering of applications with noise (DBSCAN) method to generate the more precise cluster data.

Details of the each of the section, will be as described further in the following section.

2.0 Section 1-Linear Regression Modelling

In this section, linear regression is analysed to understand the variation of residuary resistance per weight of displacement of the sailing yacht to evaluate ship performance based on different input data. These test input data include the longitudinal position of the centre of buoyancy, prismatic coefficient, length to displacement ratio, beam draught ratio and Froude number.

2.1 Method of Analysis and Results

2.1.1 Analysis of Section 1 Part 1

To successfully carry out the linear regression analysis, the following steps were followed.

Step: 1 Inputting the table data

To predict the linear regression, firstly, a data file containing input variables and output variables are uploaded using the **table_variable_name**=**readable** ***(table name. file type)*** command. There are many ways to input the data "***import data***" in Matlab navigation pane. After the data is uploaded, the table is then converted into a double matrix either using "***table2array (tablename.filetype)***"

Step: 2 Defining each of the columns from the tables

After that, each of the column matrices of the table is defined to do further analysis using the following command. **Column_Variable_Name=table_variable_name (: column_ number_ of _matrix_ to_ be assigned).** These include all the column variables of the given matrix.

Step3 Defining input values and out Output values for regression Analysis

Input and output were defined using the row matrix. In our case, there are six input variables (i.e. longitudinal position of the centre of buoyancy, prismatic coefficient, length to displacement ratio, beam draught ratio and Froude number). Meanwhile, the output variable contains residual resistance column. By using the following pseudo-Matlab command, input and output were defined in Matlab.

X= [input_name1, input_name2, input_name3, input_name4_input_name5, input_name6]

The above equations creates the matrix containing input variables in Matlab.

Subsequently, the output is defined using the following command.

Y= [Residual Resistance]

This command generates the output variable containing a single column matrix.

Step 4: using fitlm command to predict a linear regression

Using the fitlm command, as shown in below the regression following results are obtained, as shown in Figure 1.

Model=fitlm (x,y)

```
Estimated Coefficients:
                                Estimate      SE        tStat       pValue
                                ________    _______    ________    __________

    (Intercept)                  -19.237     27.113    -0.70949       0.47857
    Longitudinal_Position        0.19384    0.33807     0.57338       0.56681
    Prismatic_Coefficient        -6.4194     44.159    -0.14537       0.88452
    Length_Displacement_Ratio      4.233     14.165     0.29883       0.76527
    Beam_Draught_Ratio           -1.7657     5.5212     -0.3198       0.74934
    Length_Beam_Ratio            -4.5164       14.2    -0.31806       0.75066
    Froude_Number                 121.67     5.0658      24.018    6.2077e-72


Number of observations: 308, Error degrees of freedom: 301
Root Mean Squared Error: 8.96
R-squared: 0.658,  Adjusted R-Squared 0.651
F-statistic vs. constant model: 96.3, p-value = 4.53e-67
```

Figure 1 Linear Regression Result Using Fitlm command in Matlab

According to figure 1, linear regression can be interpreted as:

Residual Resistance=-19.237+0.19384 Longitudinal Position -6.4194*Prismatic_Coefficient+4.233 Length_ Displacement Ratio+121.67 Froude Number.

From this graph, 308 rows of the input matrix were analysed. Root Mean square error of 8.96, which is the average estimate of the error with variation R^2 of 65 per cent. The hypothesis test parameter p-value is $4.53*10^{-72}$ which could explain one or more of variables could be effecting the variation of output significantly, which makes the overall regression model valid.

Step 5: Testing if all variables significantly affect the overall regression

However, to understand if all variables are important for the whole model, a further test is required. There are two methods to approach this issue. One of the simple ways is to carry out the regression analysis of each variable against output using the following general pseudo-Matlab command.

Variable_for_analysis =fitlm (variablex$_1$,variable y)

Variable_for_analysis.plot

xlabel('x_variable_name')

Ylabel ('y_variable_name').

By applying this pseudo command, Figure 2 is obtained. In Figure (2) blue crosses represent the residual points while confidence bounds with red dash line represent the confidence of interval where data are most likely be fit. The fit line represents the line where data mostly concentrated. By examining all the graphs, it is observed that figure 2: A to figure 2: E does not affect the output variable, while in figure 2: F the input creates the quadratic trend. To prove the variation of y based on each input variable further, P-value and fit equation for figure 2: A to figure 2:F areas outlined in Table 1

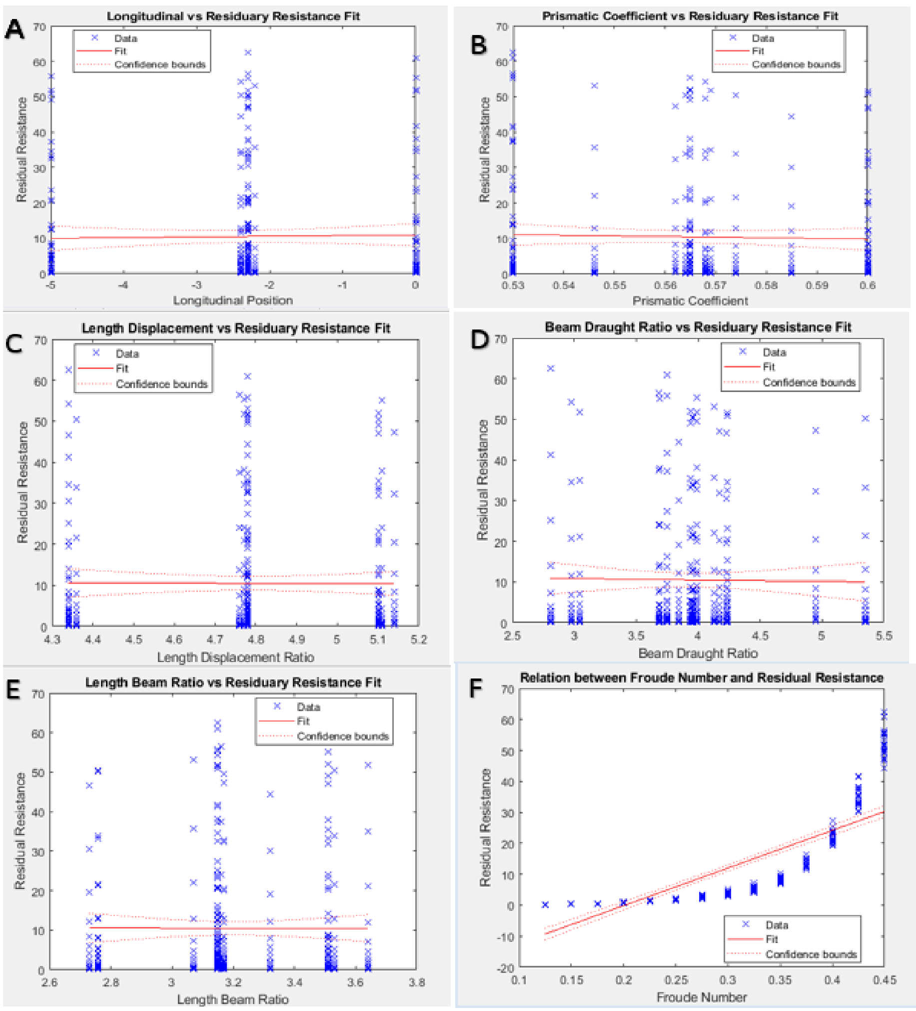


Figure 2 Residual Fit-model plot of each X Variables Against Y(A: Longitudinal Position vs Residuary Resistance Relation, B: Prismatic Coefficient vs Residuary Resistance, C:Length Displacement Ratio vs Residuary Resistance, D: Beam Draught Ratio vs Residuary Resistance, E: Length Beam Ratio vs Residuary Resistance F:Froude Number vs Residuary Resistance Plot)

Table 1 Prediction of Residuary Resistance Using Each Input Test Variables

Regression Description	Fit Equation	P-value
Longitudinal Displacement vs Residuary	Residuary=10.95+0.193*Longitudinal	0.735
Prismatic vs Residuary	Residuary=20.987-18.59*Prismatic	0.617
Length Displacement vs Residuary	Residuary=11.347-0.177*Length Displacement	0.95864
Beam Draught vs Residuary	Residuary=11.848-0.343*Beam Draught	0.828
Length Beam Ratio vs Residuary	Residuary=10.696-0.062Length Beam Ratio	0.985
Froude Number vs Residuary	Residuary=-24.484+121.67 Froude Number	$6.2331*10^{-73}$

By observing Table 1, it can be seen that the Froude number has great significance in causing the variation of y output since the regression factor is largest possessing the smallest p-value.

Other input variables are not important for the regression model since their respective p values are greater than 0.05.

P-Value means probability value which is applied in a hypothesis test. Small p-value (<0.05) indicates rejecting null hypothesis while p-value of (>0.05) shows the weak evidence against null hypothesis test (Ramsey, 2019).

Step 6: Using Stepwise command to eliminate unnecessary data

The above method is used to explain how other variables are rejected.

However, this method is time consuming if there are significant numbers of input variables input. In the case of hundreds or thousands of input variables, Matlab function stepwiselm command can be entered to reject the unnecessary variables as shown in the following pseudo-Matlab example.

[

Variables_ for_ stepwise=dataset (table. Column variables)

Stepwise_variables=stepwiselm (variables_for_stepwise,’interactions’);

Plot (Stepwise_variables)

]

By using the above pseudo command, Figure 3 &

Figure 4 are generated in Matlab as described below.

```
>> ds=dataset(yacht.Longitudinal_Position,yacht.Prismatic_Coefficient,yacht.Length_Displacement_Ratio,yacht.Beam_Draught_Ratio,y
mdl = stepwiselm(ds,'interactions');
plot(mdl)
1. Removing Longitudinal_Position:Length_Beam_Ratio, FStat = 1.1435e-06, pValue = 0.99915
2. Removing Length_Displacement_Ratio:Beam_Draught_Ratio, FStat = 6.9887e-05, pValue = 0.99334
3. Removing Longitudinal_Position:Length_Displacement_Ratio, FStat = 0.0010085, pValue = 0.97469
4. Removing Prismatic_Coefficient:Length_Beam_Ratio, FStat = 0.0074036, pValue = 0.93149
5. Removing Longitudinal_Position:Beam_Draught_Ratio, FStat = 0.0071843, pValue = 0.93251
6. Removing Longitudinal_Position:Prismatic_Coefficient, FStat = 0.0063998, pValue = 0.93629
7. Removing Length_Displacement_Ratio:Length_Beam_Ratio, FStat = 0.03553, pValue = 0.85062
8. Removing Length_Displacement_Ratio:Froude_Number, FStat = 0.077746, pValue = 0.78057
9. Removing Length_Beam_Ratio:Froude_Number, FStat = 0.071131, pValue = 0.78988
10. Removing Prismatic_Coefficient:Beam_Draught_Ratio, FStat = 0.079793, pValue = 0.77778
11. Removing Beam_Draught_Ratio:Length_Beam_Ratio, FStat = 0.10945, pValue = 0.74101
12. Removing Prismatic_Coefficient:Length_Displacement_Ratio, FStat = 0.1069, pValue = 0.74393
13. Removing Length_Displacement_Ratio, FStat = 0.089149, pValue = 0.76547
14. Removing Length_Beam_Ratio, FStat = 0.02133, pValue = 0.88398
15. Removing Beam_Draught_Ratio:Froude_Number, FStat = 0.17522, pValue = 0.67581
16. Removing Beam_Draught_Ratio, FStat = 0.0080416, pValue = 0.92861
17. Removing Longitudinal_Position:Froude_Number, FStat = 0.36038, pValue = 0.54874
18. Removing Longitudinal_Position, FStat = 0.32346, pValue = 0.56996
19. Removing Prismatic_Coefficient:Froude_Number, FStat = 1.9924, pValue = 0.15911
20. Removing Prismatic_Coefficient, FStat = 0.72591, pValue = 0.39488
```

Figure 3Removal of Unnecessary variables

```
>> mdl

mdl =


Linear regression model:
    Residuary_Resistance ~ 1 + Froude_Number

Estimated Coefficients:
                      Estimate       SE        tStat         pValue
```

Figure 4 Leftover accepted variable

Figure 3 remove the unnecessary input variables, while Figure 4 represents the leftover relevant input variable.

2.1.2 Analysis of Linear Regression fitting part 2 –Improvement of model

Step 1: Separating the training Data and Testing Data

Similarly to question one part 1, training data and testing data are loaded into Matlab using the following pseudo Matlab command. Before initiating the data, the yacht data is then split into testing and training data, where training data is 80% of the yacht data and 20% testing data. According to yacht data, there is 308 row. Therefore 80 % of data for training becomes .8*308=246 rows while the rest of the data become the training data. The training can be done using the following Matlab pseudo command.

Training_data_variable= data (1:246, :);

Testing_data_variable=yacht_data(247:end,:);

where these 2 pseudo matlab line break down the row values of the data.

Then column datas are extracted from the main matrix into column representing input value which are column two to column 7 and output value which is column 1 using the following matlab command.

```
y_train=data_train(:,1);
x_train=data_train(:,2:7);
x_test=data_test(:,2:7);
y_test=data_test(:,1);
```

Step 2: Construction of Fitted model of all the line equation

Models are trained using fitlm command using the pseudo-Matlab command.

Training_Model_variable_name=fitlm(xtrainingcolumns,ytrainingcolumns,'modeltype');

The trained model is then used to predict the model using the following pseudo-Matlab command.

Prediction_Model_variable_name=Training_Model_Variable_Name.Predict (xtrainingcolumns,ytrainingcolumns,'modeltype');

As for the models, we have used an interaction model where different input variables interact as the multiplier, quadratic model where variables multiple on itself, pure-quadratic model and polynomial model.

Step 3: Graphing the predicted model

The models are then graphed to see if they fit the actual data using the following command to generate Figure 5, showing how training models fit the actual data for future predictions.

```
figure
set(gcf,'Position',[0 0 500 600])
plot(ytestingcolumns,'g-')
hold on
plot(prediction_interactions,'b-')
plot(Purequadratic_testing,'y-')
plot(QuadraticLine, 'r-.')
plot(prediction_polynomial,'m.')
legend('Actual data','Interaction model','Pure Quadratic model','Quadratic model','polynomial')
```

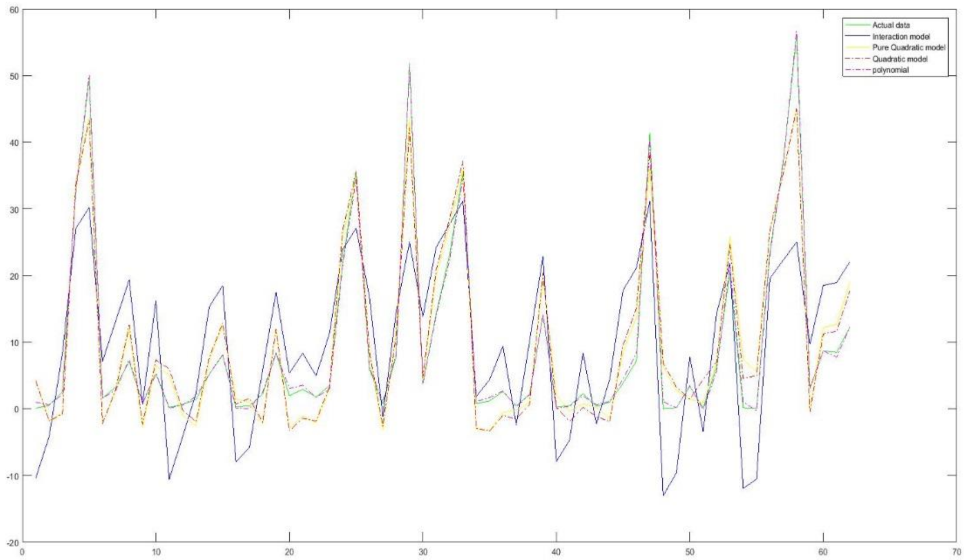


Figure 5 Fitted model of actual data, interaction, and pure quadratic, quadratic and polynomial

Step 4: Interpretation of the graph

From the graph data as shown in Figure 5, it can be seen that quadratic and the pure quadratic fits more than interaction model due to the shape of the actual data. The actual data has the quadratic shape, and therefore, it can be observed that pure quadratic and quadratic fit much better than the interaction model. With polynomial data, the line got fitted excellently with the actual data. In our case the polynomial fit order is order of 4.

Step 5: Checking the optimal fit of the graph with RMSE function

The fitted line is then checked with RMSE function of mat lab to see the errors generated by the training fit model. As described in figure 6, the polynomial order of 4 has the RMSE value of 0.49, which give the best fit value. The best fit value is checked against polynomial order of 3 and 5. When we use polynomial order of 3 we have RMSE value of 1.444 which shows the under fitness of the graph while polynomial of order six gives an RMSE value of 0.6 which shows the slight overfit.

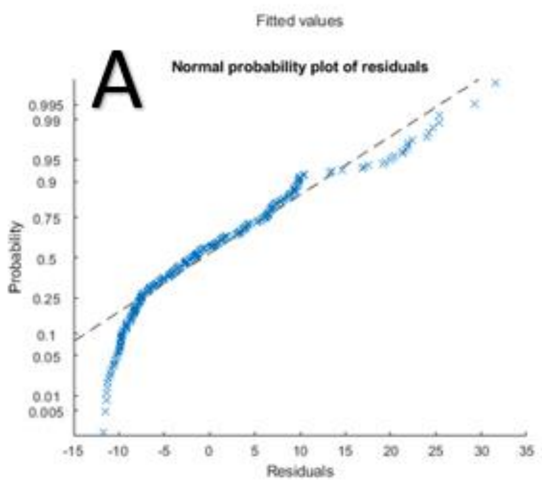
```
rootmeansquareerrorinteraction = 9.7086
rootmeansquareerrorofpurequadratic = 4.0444
rootmeansquareerrorquadratic = 4.1507
rootmeansquarerrorpolynomial = 0.4923
```

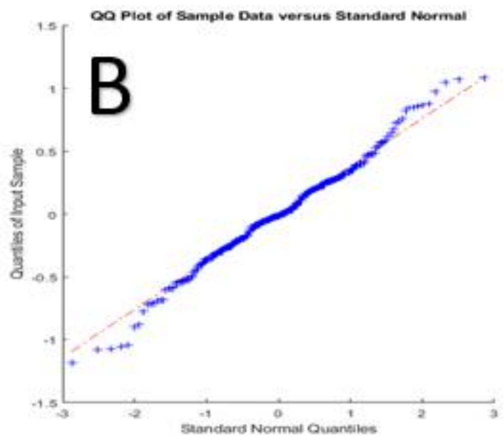
.

Figure 6 Root Mean Square Error Output Generate By Matlab

Step 6: Further Error Check Using Residual Plots histogram of residual

The residuals errors are then compared based on original data without unnecessary data removal and after unnecessary data removal. According to the graph, it was found out that the variation of the error normally distributed after the unnecessary data was removed. This clearly shows that the remained input variable is effectively impacting the output variable.





3.0 Section 2- Regularising Regression

3.1 Fitting Linear Regression Ridge and Lasso Regression Using NoBow Data

In this excerise regularise regression and ridge and lasso models are used to compare with normal regression pattern by fitting them together with no bow data.

Step 1: Loading the Table

The following sections will explain the steps necessary to complete the exercise. These steps involves testing and training of data which include loading the training data and testing data as follow. Meanwhile, data containing high p values are taken off since they are not predicting the model data.

```
Xtrainingdata=x2fx(data_train_matrix(:,17:28));
Xtrainingdata(:,1)=[];%%the first column all 1, delete it.

ytraindata=data_train_matrix(:,29);

Xtestdata = x2fx(data_test_matrix(:,17:28));

Xtestdata(:,1)=[];

ytestdata=data_test_matrix(:,29);

deletingirrelevance=stepwiselm(Xtrainingdata,ytraindata)
```

After uncessary data were kicked new test data were defined.

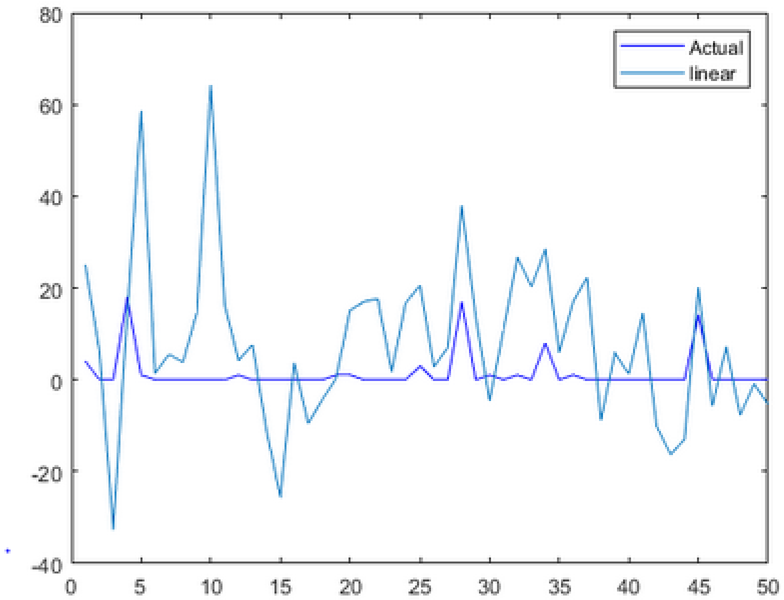
```
X_trainingdata=x2fx(data_train_matrix(:,19:28));
X_trainingdata(:,1)=[];
ytraindata=data_train_matrix(:,29);
Xtestdata = x2fx(data_test_matrix(:,19:28));
Xtestdata(:,1)=[];
ytestdata=data_test_matrix(:,29);
```

Step 2:Plotting linear regression

Then linear regression was plotted using the linear model and compared with the actual data.

```
model_linear=fitlm(X_trainingdata,ytraindata,'linear')
predictions_linear=model_linear.predict(Xtestdata);
mse_linear=sqrt(mean((predictions_linear-ytestdata).^2))
```

which gives the graph as shown below in Figure 7. According to the linear model the errors contain 685 which is quite large.




```
M = 27.4294
idx = 1
minval = 752.3699
mixindex = 1
best_Lambda = 0
best_error = 685.7835
best_index = 93
best_Lambda = 0.0920
```

Figure 7 Actual vs Linear Model

Step 2:Fitting the Linear Regression Model

As for fitting the model using the linear regression fitlm model is used

```
model_linear=fitlm(X_trainingdata,ytraindata,'linear')
predictions_linear=model_linear.predict(Xtestdata);

mse_linear=sqrt(mean((predictions_linear-ytestdata).^2))
```

which gives the factorial values of input value and and p values associated with the training model as shown in Figure 8.

```
model_linear =
Linear regression model:
    y ~ 1 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10

Estimated Coefficients:
                    Estimate        SE         tStat       pValue
                   ___________    ________    ________    __________

    (Intercept)         21.357      7.3072      2.9228      0.003622
    x1                -0.15865    0.089367     -1.7753      0.076447
    x2                 0.22707    0.062024      3.6609    0.00027735
    x3                 -6.1399      18.094    -0.33934       0.73449
    x4                  12.412      8.3606      1.4846       0.13826
    x5                  5.3587      7.9912     0.67058       0.50279
    x6                  1.9869      20.055    0.099068       0.92112
    x7                -0.49782      0.1698     -2.9317     0.0035213
    x8             -0.00024874    0.000921    -0.27008       0.78721
    x9                 -3.8298      4.5308    -0.84529       0.39835
    x10               0.022286     0.14233     0.15658       0.87564

Number of observations: 524, Error degrees of freedom: 513
Root Mean Squared Error: 75.9
R-squared: 0.0767,  Adjusted R-Squared 0.0587
F-statistic vs. constant model: 4.26, p-value = 1.01e-05
```

Figure 8 Fit Linear Model of Training Data

Step 4 Construction of Ridge Model to determine best lambda value

The logic isnan is used to delete uncessary data for the ridge moel construction. As for the regularization penalty selection regularization penalty values of 1000 points are selected to find the best lambda coefficient between penalty value of 0 to 1.Then the biases are calculated based on the training data and the regularization penalty value.Then lambda value corresponding to the minimum error is found. In this case penalty parameter corresponding to the minimum error is 1. Therefore, the selected penalty parameter is used to plot the ridge model and compared it against actual data as shown in Figure 9. The curve is not fitting perfecting. This could be improved by chaging the regularization parameter or using higher model but this is beyond the scope of the assignment. As for ridge regression minimum error to be achieved is 752.39

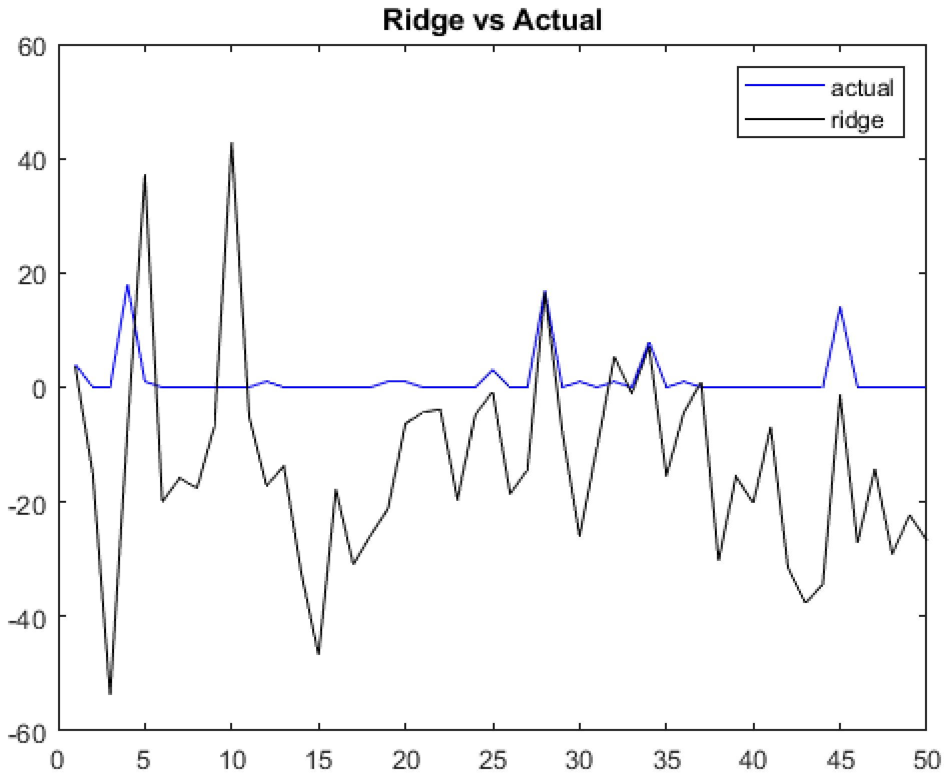


Figure 9 Ridge Regression vs Actual Data

Step 3: Further check for best lambda value using the error check function

Similarly Lasso model is plotted using the following command. Again the lambda values of 100 are abitarily selected to calculate the best lambda value where best lambda value is the lambda value which gives the lowest fit error of lasso regression. In our case, the best error achieved for lasso regression is 685.8 while best lambda achieve is 0.092 which correspond to the error. The lasso regression is then plotted as shown in Figure 10 . In contrast to Figure 9, lasso regression in Figure 10 fits data fits much better.

```
[b,fitinfo] = lasso(X_trainingdata,ytraindata,'CV',5);
[best_error, best_index] = min(fitinfo.MSE);
errors=zeros(100,1);
for i=1:100
    yhat=fitinfo.Intercept(i)+Xtestdata*b(:,i);
    errors(i)=mean((yhat-ytestdata).^2);
end
[best_error, best_index]=min(errors)
lasso_model=b(:,best_index);
lasso_intercept=fitinfo.Intercept(best_index);
predictions_lasso=lasso_intercept+Xtestdata*lasso_model;
best_Lambda=Lambda(best_index)
hold off
plot(ytestdata(1:50),'b-')
hold on
plot(predictions_lasso(1:50),'r-')
legend('actual','lasso')
title('lasso vs Actual')
hold off
```

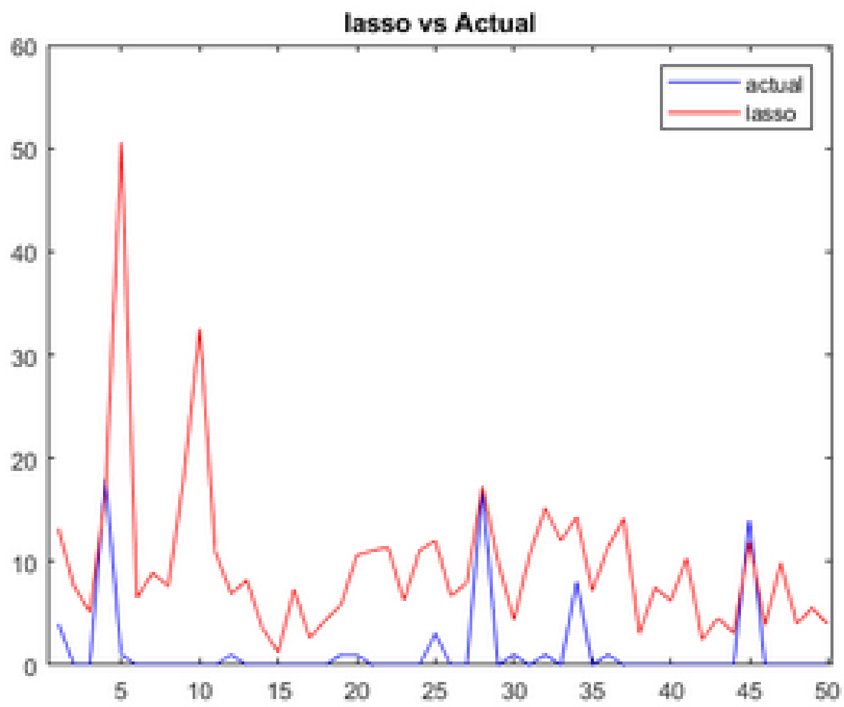


Figure 10 Lasso Regression vs Actual Data

Step 4 Comparing three models

After regression graphs are analysed all the regression lines are put together to comprare the regression using the following code resulting in Figure 11. By looking into graph it hard to interpret which regression methods. Therefore MSE values were checked using matlab command. The resulting Table 2 shows all the MSE for 3 regression model relative to the actual data. According to Table 2, it is found out that linear model fits the best, while lasso fits better than ridge model. The accuracy could be improved better by looksing further into the regularization parameters. However, at this stage it could not be achieved due to the limitation of time to finish the task.

```
figure
plot(ytestdata,'b-')
hold on
plot(predictions_linear,'r-')
plot(predictions_ridge,'k-')
plot(predictions_lasso,'g-')
xlim([0 50])
ylim([-100 100])
legend('Actual','pure quadratic','Ridge','Lasso')
```

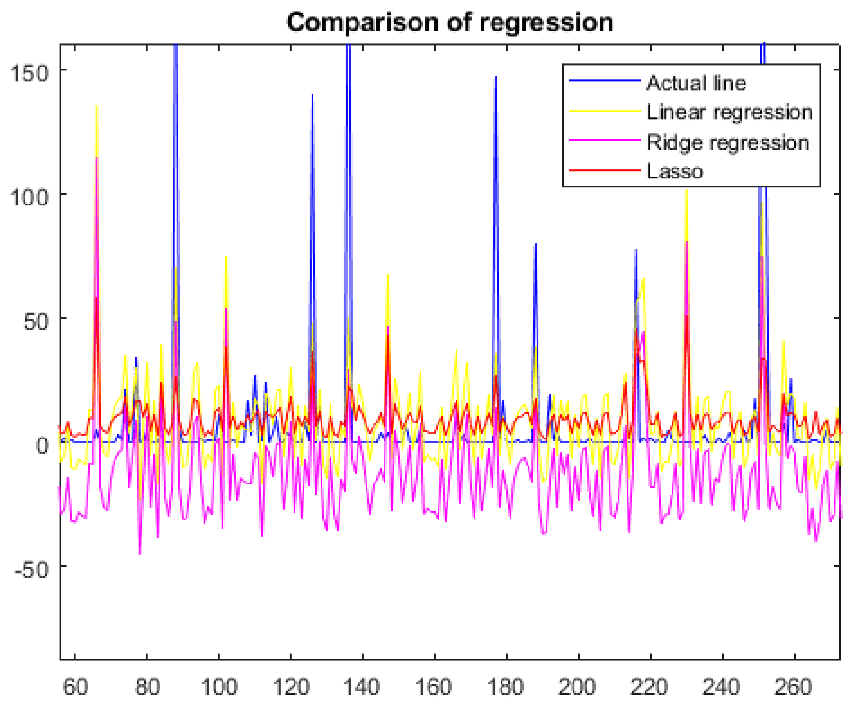


Figure 11 3 Regression Models

Table 2 Mean square erorr for each regression models

Type of Regression Model	MSE obtained
Linear Model	27
Lasso Model	685

Ridge Model	1077

3.2 Fitting Regression using Bag of Word Model

Step 1: Uploading Data

Since the regression is bag of word model, bag of word model is used which contains bag of word model data column. The models are then trained and tested. In this case column 1 to 228 from blog data was choosen as bag of word model by using following code

```
lear all;
clc;
traindata=readtable('blogData_train.csv');
testdata=readtable('blogData_test.csv');
trainingmatrix=traindata{:,:};
testingmatrix=testdata{:,:};

X_train=x2fx(trainingmatrix(:,[17:228]));
X_train(:,1)=[];
y_train=trainingmatrix(:,229);
X_test = x2fx(testingmatrix(:,[17:228]));
X_test(:,1)=[];
y_test=testingmatrix(:,229);
```

Step 2: Linear Fitting

Firstly, the lnear regression is fitted using the fitlm model which creates the output as shown in Figure 12. According to Figure 12 the model is not very relevant due to the high p value. However, since there is the time limit to carry on this task, we may have continue using this fit linear model although the model can be deducted using stepwise method.

```
%%stepwiselm function in this step costs a lot of time, so I do not use it
model_linear=fitlm(X_train,y_train,'linear','CategoricalVars',[11:210])

Warning: Regression design matrix is rank deficient to within machine precision.
    x196_1         4.9784      12.694       0.39218      0.69515
    x197_1         37.108      88.587       0.41889      0.67554
    x199_1        -31.962      59.281      -0.53915       0.5901
    x202_1         4.6793      30.602       0.15291      0.87856
    x203_1        -55.987       60.73      -0.92191      0.35717
    x205_1         13.798      27.016       0.51071      0.60985
    x206_1         58.197      116.58       0.49919      0.61794
    x207_1        -54.649      63.291      -0.86345      0.38844
    x208_1         104.54      93.578        1.1171      0.26468
    x209_1         6.5647      76.632      0.085665      0.93178
    x211          -4.5607      6.0053      -0.75944      0.44807
    x212        -0.012614     0.17004     -0.074181      0.94091

Number of observations: 524, Error degrees of freedom: 375
Root Mean Squared Error: 82.8
R-squared: 0.198,  Adjusted R-Squared -0.119
F-statistic vs. constant model: 0.624, p-value = 0.999
```

Figure 12 Linear Regression Output

Step 3 Construction of Ridge Regression Using Bag of Word features

Similarly to part one ridge model is constructed with best lambda value of 1 where best lambda value is deduced from minimum value obtained from ridge error test. Using the index and lambda value which gives minimum error ,follow ridge graph in contrast to actual data is constructed as shown in Figure 13. The plotted ridge graph has RMSE value 1900 which is very high. This RMSE could be reduced further by optimizing the penalty and index data further. However, due to the scope of the report, ridge regression with RMSE of 1900 was accepted.

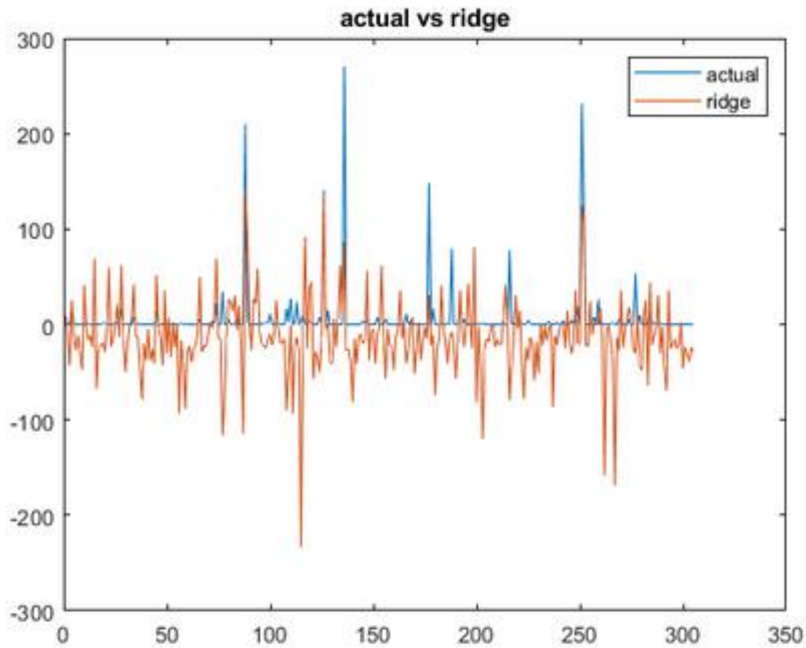


Figure 13 Ridge Regression Vs Actual Data

<u>Step: 4 Construction of Lasso Model using Bag of Word Data using Bag of Word Features</u>

Similarly lasso model was also contstructed using bag of word features similar to the process from part one but with different data and following graph as shown in Figure 14 was obtained. Again, the model does not fit perfectly even though it is following the trend of actual data.

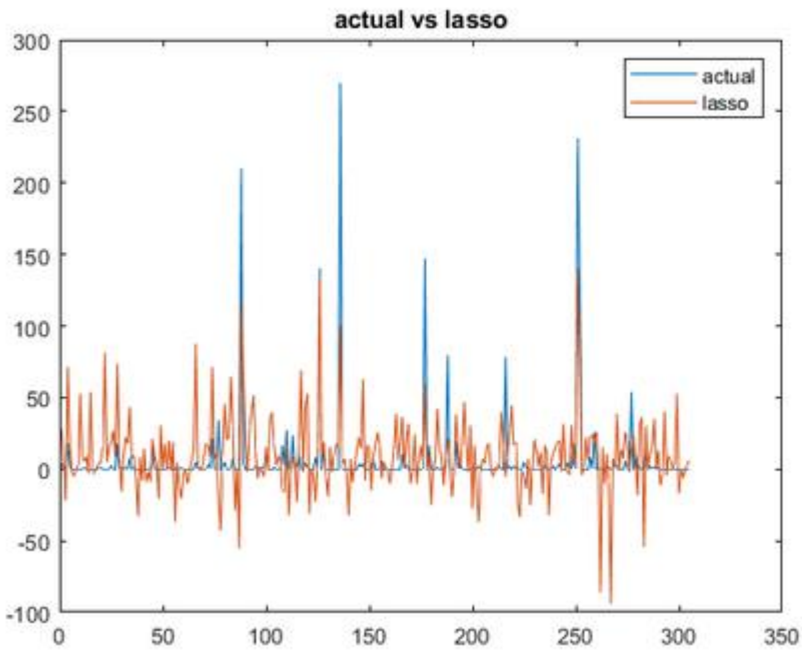


Figure 14 Lasso Graph In Comparison to Actual Model

Step 5: Putting Linear Ridge and Lasso together.

Similarly to Q2 part 1, the models are then put together compare the actual data against ridge, linear graph and lasso graph fit the actual data more than ridge graph. The means squares error are then checked again to see the fitness of our model. Base on the MSE neither the lasso nor ridge fit the model while linear fit the best. This fitness error may have something to do with wrong regularization parameter being picked.

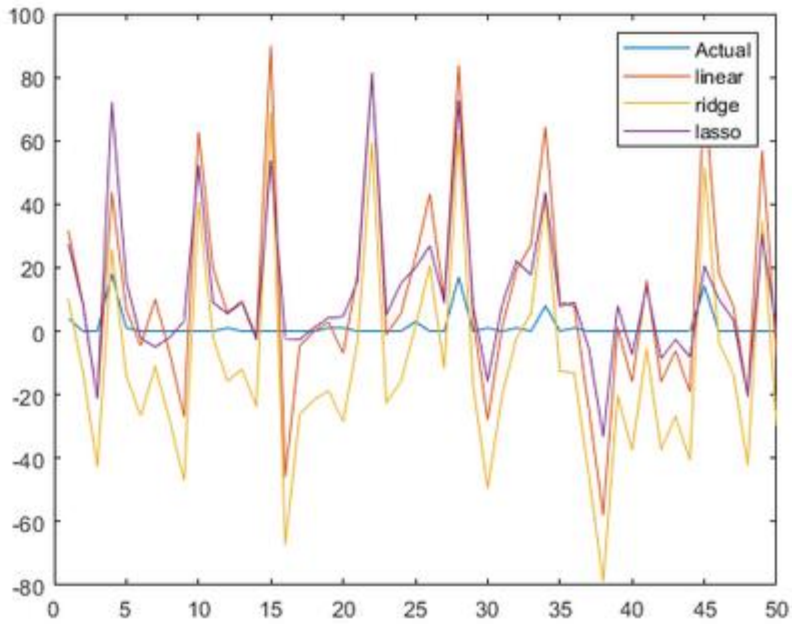


Figure 15 Actual, Linear, Ridge and Lasso Plot based bag of word feature

4.0 Section: 3-Clustering Using K-means algorithm and Gaussian Mixture model

This section focuses on clustering training (unsupervised) training for a large dataset, mainly using two methods which include K-means clustering method and Gaussian Mixture Model (GMM) method. In this section, dataset.csv file is given describing the date, time, global active power, global reactive power, voltage, global intensity and three metering type corresponding to the different type of usage.

The following section will explain how the task is solved, including the methodology and results we have obtained.

4.1 Section 3-Part 1 clustering three-meter modes using different cluster method (Methods and Result Detail)

This section aims to explain the detail steps and results prediction we have obtained for clustering three-meter values.

Steps 1: Defining table and columns

In this step table and columns are defined from table value to double value for easier calculation using the following Matlab command.

```
tablematrix=readtable('household_power_consumption_2007.csv');
DateTime=table2array(tablematrix(:,1));
gobalactive=table2array(tablematrix(:,2));
globalreactive=table2array(tablematrix(:,3));
voltage=table2array(tablematrix(:,4));
global_intensity=table2array(tablematrix(:,5));
powermeter1=table2array(tablematrix(:,6));
powermeter2=table2array(tablematrix(:,7));
powermeter3=table2array(tablematrix(:,8))
```

Step 2: Conversion of Categorical Variable into integer value

The first column of the table date is defined in dd/mm/yyyy format.

To cluster the months or weekday or weekend, the format has to be converted from the original format to nominal values. In the case of converting "DateTime" to the nominal value of day following mat lab code is used.

Days=week_day(DateTime). This code converts the days into nominal values, as shown in Table 3.

Table 3Conversion of Catergorical Name into Nominal Value

Respective Nominal Value	Original Day Description
1	Sunday
2	Monday
3	Tuesday
4	Wedesday

5	Thursday
6	Friday
7	Saturday

As for this task, the command m=month(DateTime) command is used to convert the string data type to nominal data type for further analysis with output as shown in Table 4

Table 4 Conversion of the string value of month name to respective nominal Value

Description	Nominal Value
January	1
February	2
March	3
April	4
May	5
June	6
July	7
August	8
September	9
October	10
November	11
December	12

Step 3: Defining a new matrix which puts three power meters together

After date/time format is converted into an appropriate format for further clustering analysis, the matrix for three power meter is then defined using the following Matlab command.

```
First_data_to_cluster=[powermeter1,powermeter2,powermeter3];
data=first_data_to_cluster;
```

Step 4: Initiation of Kmeans cluster

After the matrix for three power meters is defined in Matlab, K-means cluster is used using the K-means command. Initially, cluster value often is chosen arbitrarily. The optimum K value can be selected using different algorithms for cost K, which produce an elbow curve method. However, the details of algorithm will be explained in the following steps.

```
[idx,C] = kmeans(first_data_to_cluster, 10)
PlotClustersWithColours(first_data_to_cluster, idx, 'Cluster of 3 Submetering');
xlabel('MeteMrvalues')
ylabel('Metervalues')
```

The code described above produces a cluster group diagram of 3 power meter, as shown in Figure 16. Where each colour of a cluster represents different type of power usage.

Code ^

Figure 16 Cluster of 3 Submeter Value

Step 5: Describing Cluster Using Brewer's Map

The cluster plot is then further described using the Brewer's Map and gscatterplot function. The functions snippets are gathered from the blackboard where snippets are as shown in the zip file being mailed Using the code snippet, as written below, Figure 17 is plotted.

Figure 16 can be used to explain the type of cluster, while Figure 17 can be used to locate the clusters.

```
gscatter3(first_data_to_cluster(:,1),first_data_to_cluster(:,2),first_data_to_
cluster(:,3),idx)
xlabel('power meter one')
ylabel('powermeter two')
zlabel('power meter three ')
title('three power meters')
```

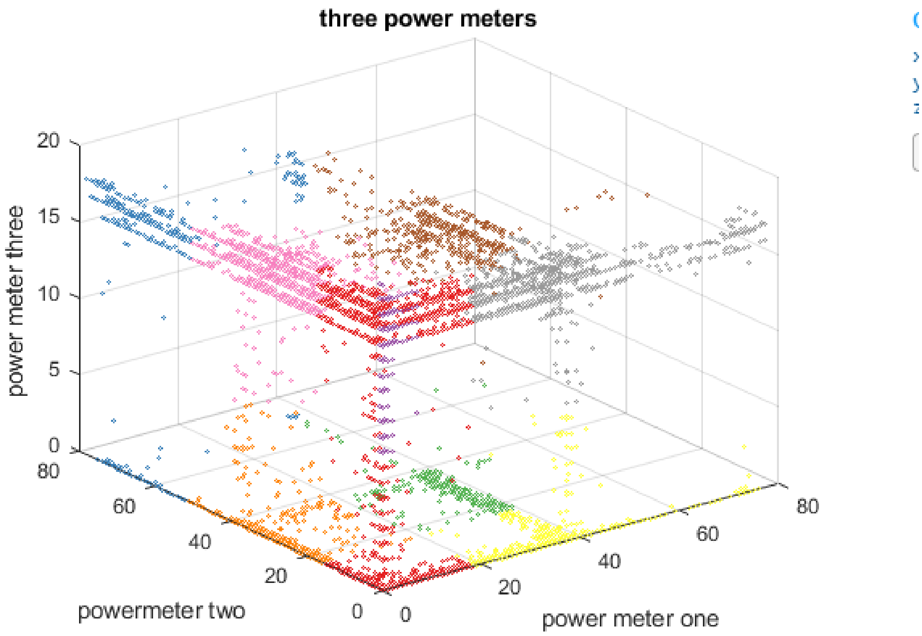


Figure 17 3D Cluster Plot for Three Power Meters

Step 6: Creation of Cluster Distribution for Summer and Winter

The cluster spread for summer and winter is made using the following logic commands in Matlab as shown below. Summer represents months of December or January or February. Meanwhile, winter represents June, July or August. Using the logic, the following code was written.

```
summer=idx((m==12)|(m==1)|(m==2),:)
winter=idx((m==6)|(m==7)|(m==8),:)
```

After the logic for summer and winter periods are specified, histograms, as shown in, are plots using the following Matlab Command.

```
hist(summer,1:10)
title('Number of Cluster Distribution Summer of Kmeans Model')
xlabel('Name of Cluster Group')
ylabel('Number of Data Points in Cluster Group')

hist(winter,1:10)

title('Number of Cluster Distribution Winter of Kmeans Model')
xlabel('Name of Cluster Group')
ylabel('Value of Datas in Cluster Group')
```

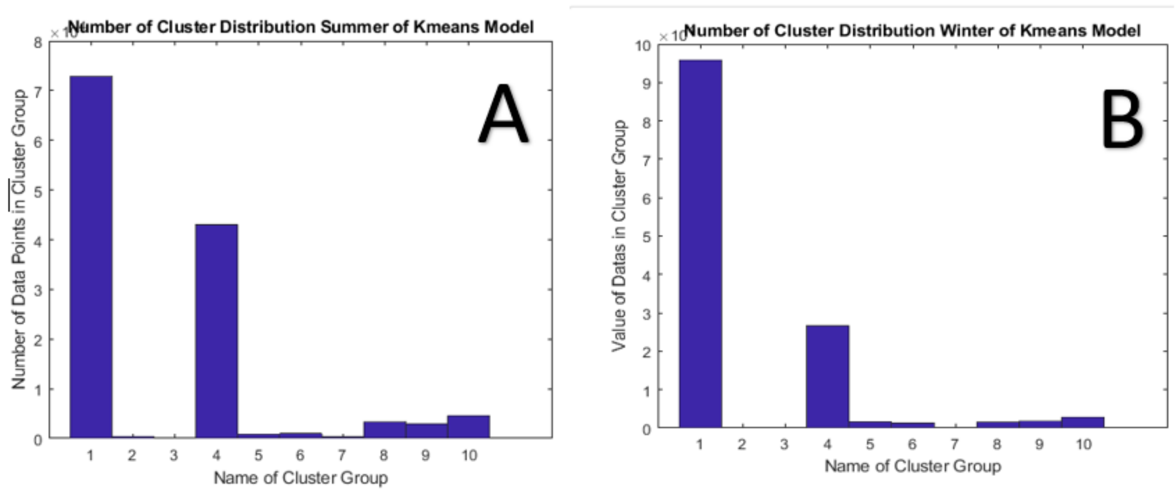


Figure 18 Histogram of Cluster Disbutions of Summer and Winter under Kmeans Model

Step 7: Interpretation of Histograms

Based on the histograms as described in Figure 18, it can be observed that cluster one for winter is significantly higher than summer. Using the references from Figure 16 & Figure 17, cluster 1 represent the metering range value of 0-20 which further serve as the lowest meter value. Meanwhile, clusters representing high meter value, which includes cluster 8 and cluster 9 for summer is higher than winter. Therefore, based on this cluster spread on summer and winter histograms, it was precisely demonstrated that meter usage in summer is higher than meter usage in winter.

Step 8: Justification of K value or Number of Clusters using different algorithms

K-value, number of clusters were arbitrarily selected. However, it is crucial to determine the optimum number of clusters since too many clusters may lose the meaning the group name while few numbers of clusters may categorize the data with error. Therefore, the best possible value to cluster is determined using different algorithms. These algorithms include K-means method, information cost criterion method and Gaussian Mixture Model method. Among these methods K-means method is found to be most simple method with lowest run time while the Information criterion method and Gaussian Mixture Model methods were found to be more complex with higher run time of Matlab. As for this project trial of different trials for 10-35-50 and 100 were ran to check the K value for each model. The values of 35 and above affect the Matlab run time undesirably.As for K means elbow graph as shown in Figure 19, the elbow point intersects at K value of 5. Subsequently, cluster number optimisation values were measured using Gaussian mixture model using trial of 10 and 50. When the model was run with the of ten trials, the misleading optimum K value was obtained. According to the GMM curve as described in Figure 20 A, the elbow curve inside the GMM model was bent in value of 10 and flat out after value of 10. However, this result was found to be misleading when the GMM model was run under the trial of 50 values as shown in Figure 20 B. When ran under the trial of 50, the K value increase as the curve goes further and further. Similarly,large optimum K value to select was observed in information cost criterion method as shown in Figure 21.

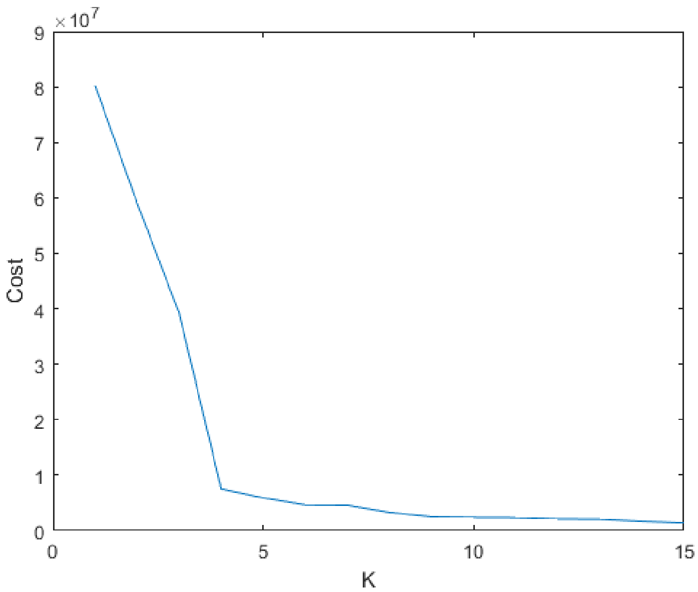


Figure 19 Determining Optimum Value using Kmeans algorithm

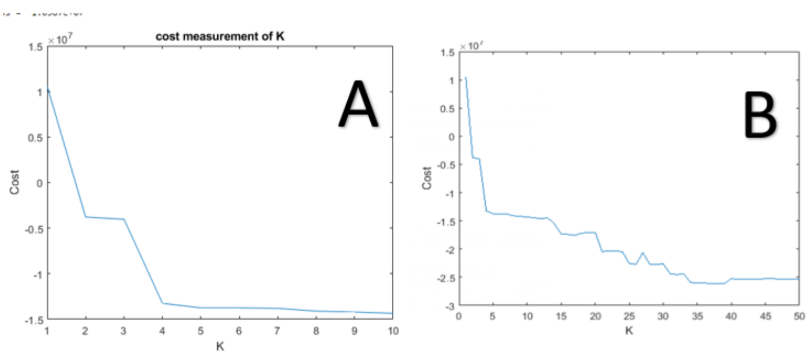


Figure 20 K measurement for Gaussian Mixture Model

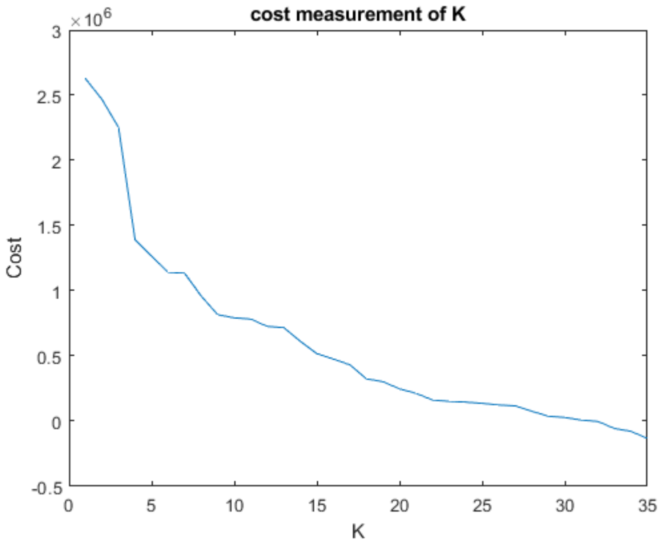


Figure 21 K value in information cost criterion method.

Step 9: Readjustment justification of K means based on elbow curve intersection point

The K value of 4 is then readjusted since the elbow curve stops at the cluster of 4, which gives the minimum criterion of selecting K value. In this case, we are selecting K value with lowest cost since the data is extremely large. After selecting K value, GMM cluster is then re-plotted in 3D and 2D mode, as shown in Figure 22.

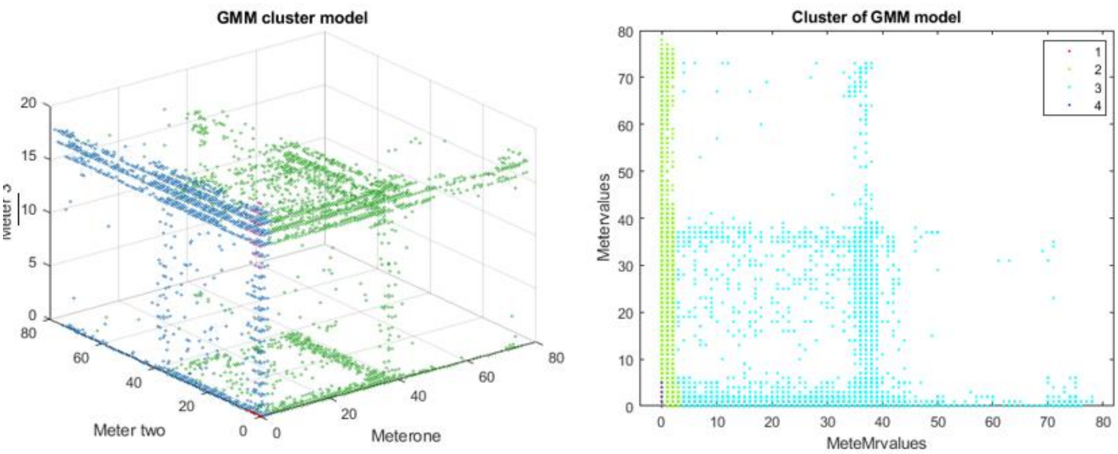


Figure 22 GMM cluster Model

Step 10: Replotting of cluster distribution histogram based on new clusters

According to the new histogram as shown in cluster 2 and cluster 3 for summer is higher than winter where cluster 2 and cluster 3 represent high meter usage, and therefore the new histograms fit with the old histogram data. With new GMM models outliers in the middle between two key clusters are also easily observed, and therefore, it is unnecessary to cluster all of the data.

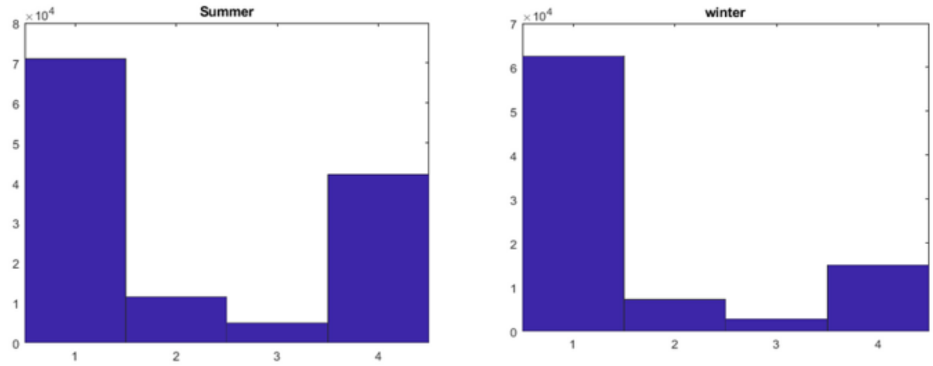


Figure 23 New Cluster Distribution Based on Optimization

Step 11 Abnormal Usage Detection

Abnormal usage was detected using the following command. However, the cluster graph does not seems to look correct due to the numerous amount of abnormal usage cluster.

```
nlls=zeros(length(first_data_to_cluster),1)
for i=1:length(first_data_to_cluster)
    [a,nlls(i)]=GMModel.posterior(first_data_to_cluster(i,:));
end

[nlls,indexes]=sort(nlls);
abnormal_amount=0.01;
abnormal_index=length(first_data_to_cluster)-
int32(abnormal_amount*(length(first_data_to_cluster)))
scatter(first_data_to_cluster(indexes(abnormal_index:end),1),first_data_to_clu
ster(indexes(abnormal_index:end),2),'kx')
```

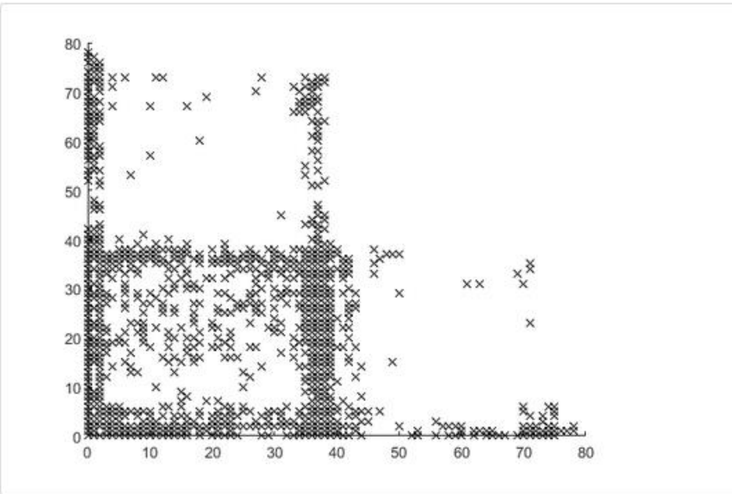


Figure 24 Results for Abnormal Usage Detection

Section 4 Clustering using DB scan and HAC

The author would like the matlab code to be reviewed for this question in the attached appendix section as challenges for this question has been faced significantly. This section has some challenges associated with matlab lab problem.

As for this question dendrogram for ground truth signal data is constructed for ground signal data where dendrogram measure the length between each cluster.

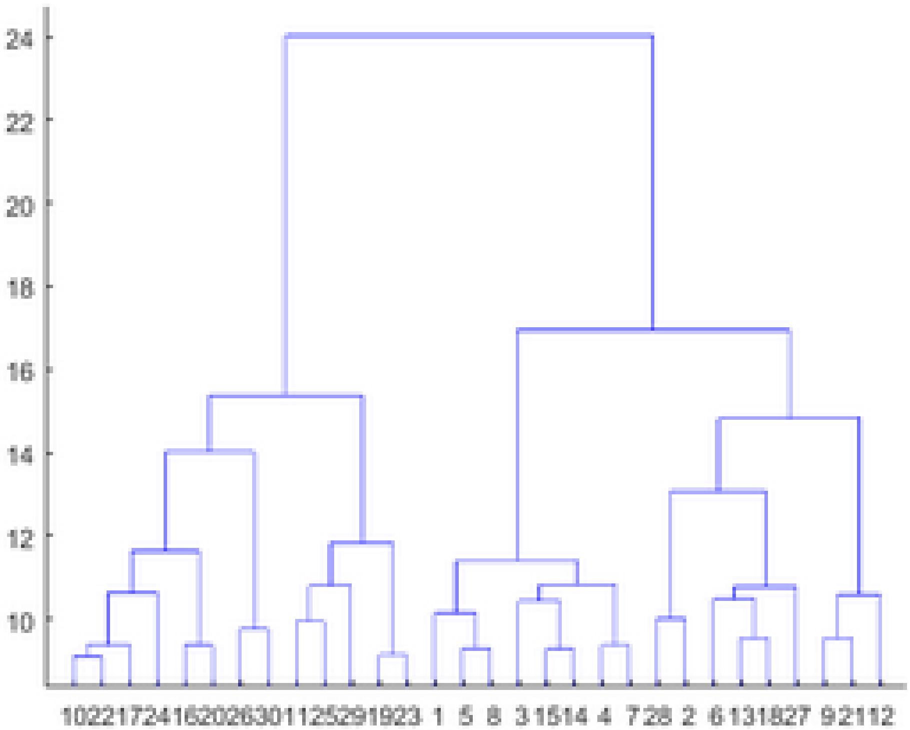


Figure 25 Dendrogram

After the dendrogram is measure the cluster group for signal data is made as shown in

```
idx = cluster(Z,'maxclust',6);
gscatter(A(:,5),A(:,200), idx)%change number here
```

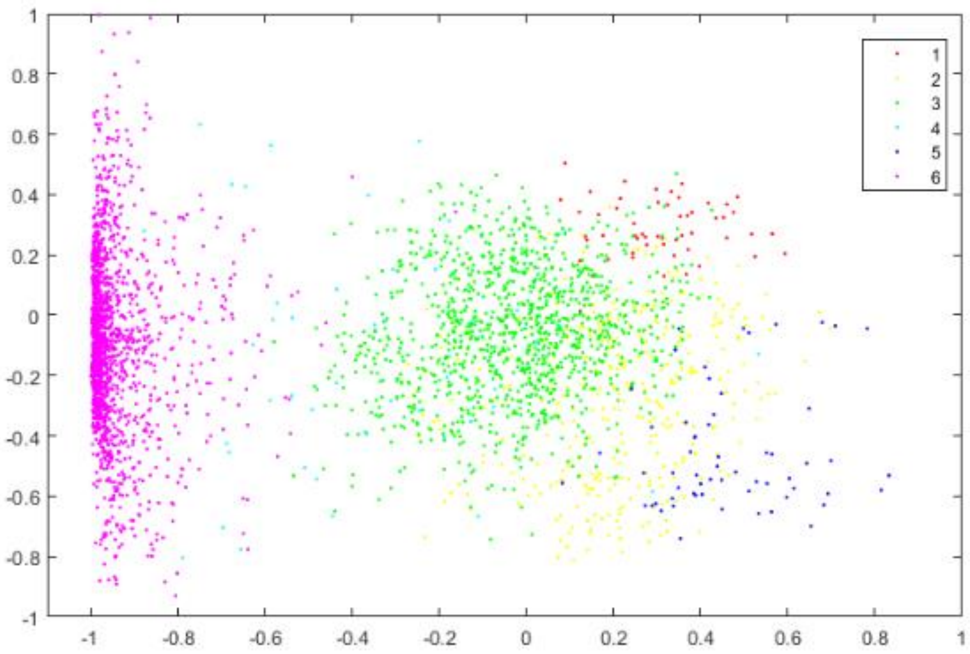
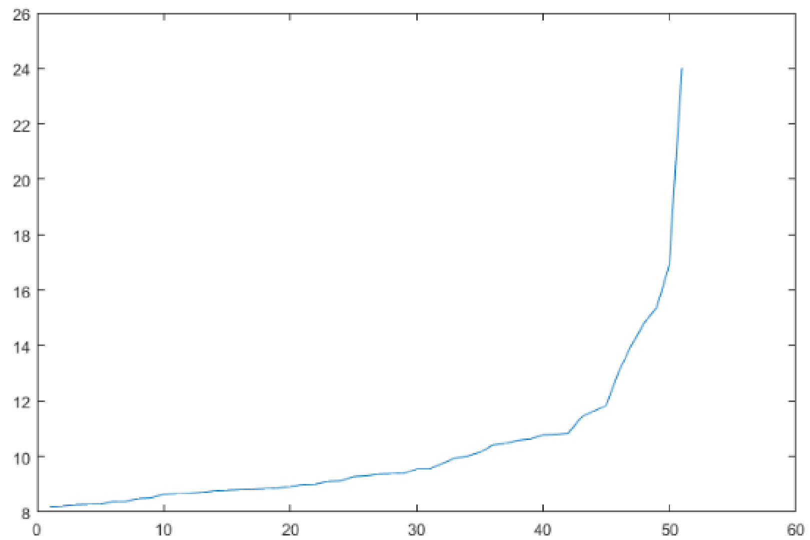


Figure 26 6 Cluster data for ground truth

Then the cluster length of Z was checked using the following command.

```
plot(Z(end-50:end,3))
```

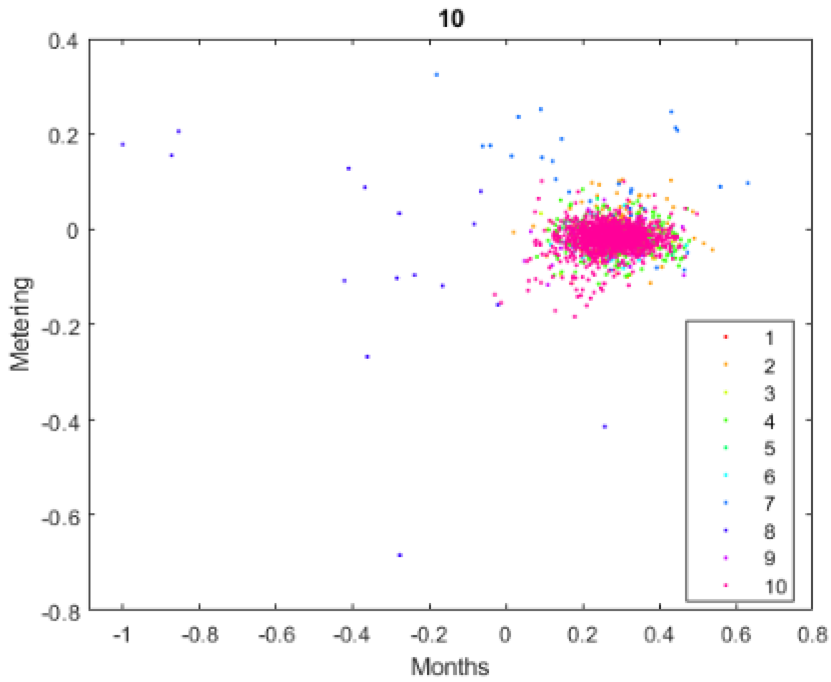


```
length(Z) - min(find(Z(:,3) > 11)) + 2
```

Figure 27 Cluster Length Z

The elbow value 11 is used in this case,to plot the signal cluster using the DB scan function

```
PlotClustersWithColours(A, idx, '10');
```




```
[C, idx, centres] = dbscan(A', 1, 10);

PlotClustersWithColours(A, idx, 'DBScan');
```

Figure 28 DB scan scatter plot to show subject

In order to check the purity and compplete ness of 6 activity classes, the clustered ground data is compared with subject data and activity data. As for subject cluster we have got the purity value of 0.3307 while as for the activity cluster we have the purity value of 0.42. Where purity value of 0 represent lower bound while purity value of 1 repersent maximun purity.

Moreover prediction and actual model are compared for real data and test data.

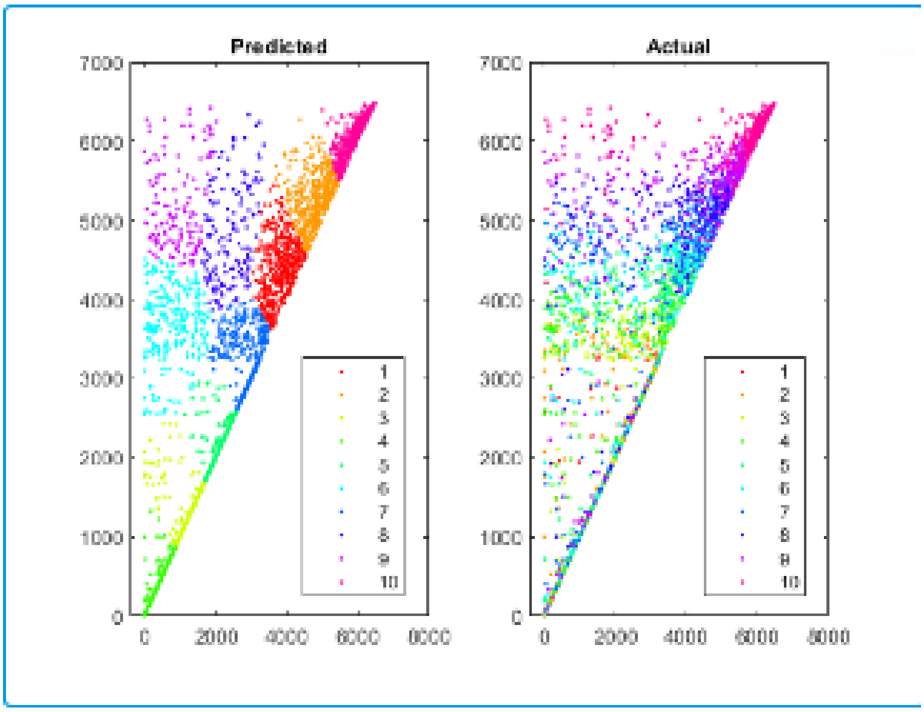


Figure 29 Prediction and Actual Cluster data

Clustering of subject and acitivity was attempted. However, the operation wasn't successful due to the matrix related issues with matlab.

Figure 30 Clustering of subject and activity

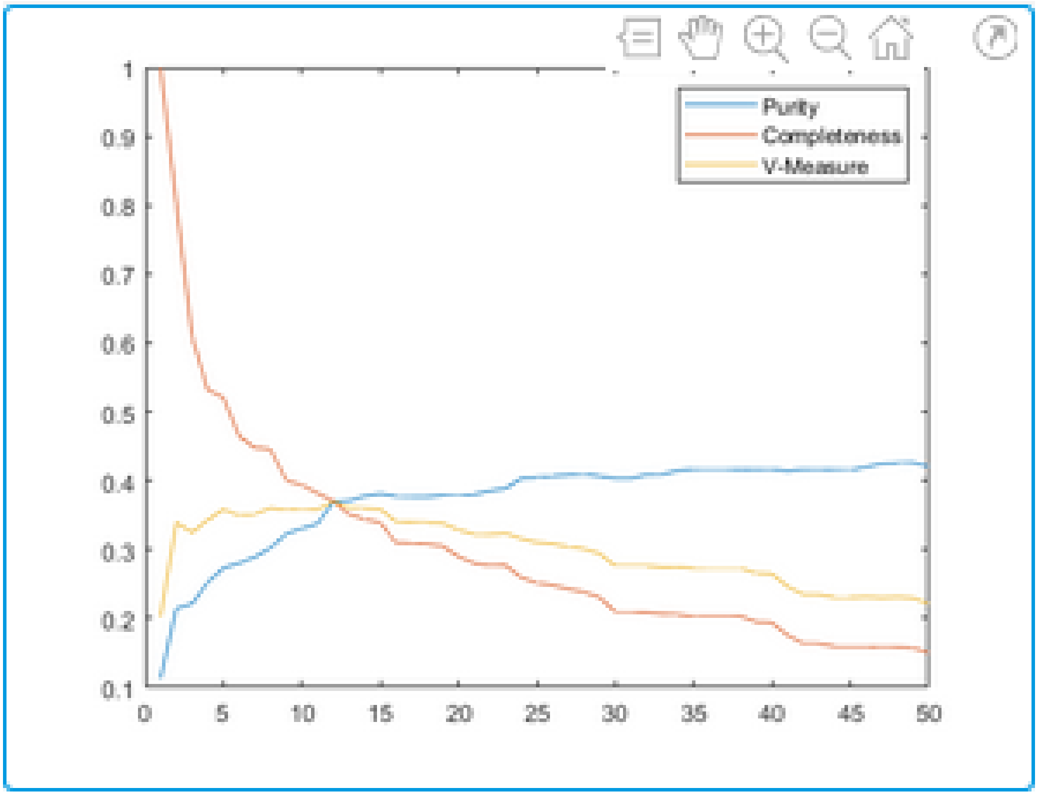


Figure 31 Purity, Completeness and V-measure trade off

Finally, there seems to be trade off between purity completeness and V measure when evaluing the cluster. According to the graph as shown in , as data get purer completeness of data became less and lessBy analysing this graph, it can be predicted that the sweespot between purity, completeness and V measure could be analysed.

Conclusion

In conclusion, problems are solved sequentially. As for section fit model was predicted. Meanwhile, model fit training using different regression fit methods were being applied. As expected, it was found out that polynomial data fits most compare to linear quadratic and pure quadratic regression with lowest mean square error was achieved.

As for section two, ridge lasso and linear models were predicted despite models not being fitting perfectly. As for clustering section, demand for sesonal values were predicted. Meanwhile, GMM model was constructed to remove the outliers while producing the meaningful cluster group. As for section 4 for purity and completeness was checked for activity data and subject data. The clusters were made for activity data (6 clusters) and subject data (10 clusters). However, DBscan reclustering process faces an issue due to the matrix size related errors. Similarly sperating cluster for 60 value with activity and subject data faces a challenge due to the matrix related isuee and matalb skills and capability.