

CLASSIFICATION

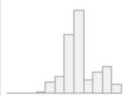

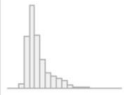

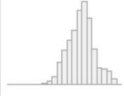

Team TBD

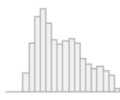
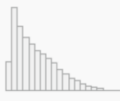
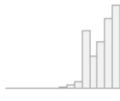

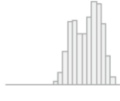
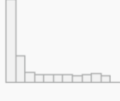
PART I Descriptive Statistics

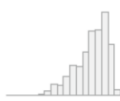
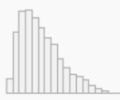
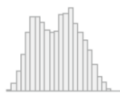

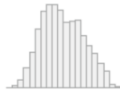

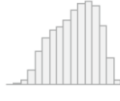
Dataset Source: <https://www.kaggle.com/nicholasjhana/energy-consumption-generation-prices-and-weather> (Same as last time)

As an inherited report of multiple regression, we are going to explore the application of logistic regression, K-NN (k nearest neighbor) and classification tree in classifying the #LOAD column. Besides, we will compare these models' performances in the task respectively and asses their alliance of the outcomes.

Below is a chart of the descriptive statistics of variables in pool.

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
1	generation.biomass [numeric]	Mean (sd) : 383.5 (85.3) min < med < max: 0 < 367 < 592 IQR (CV) : 100 (0.2)	435 distinct values		35064 (100%)	0 (0%)
2	generation.fossil.brown.coal.lignite [numeric]	Mean (sd) : 448.1 (354.6) min < med < max: 0 < 509 < 999 IQR (CV) : 757 (0.8)	964 distinct values		35064 (100%)	0 (0%)
3	generation.fossil.gas [numeric]	Mean (sd) : 5622.7 (2201.5) min < med < max: 0 < 4969.5 < 20034 IQR (CV) : 2303 (0.4)	8310 distinct values		35064 (100%)	0 (0%)
4	generation.fossil.hard.coal [numeric]	Mean (sd) : 4256.5 (1962) min < med < max: 0 < 4475 < 8359 IQR (CV) : 3312 (0.5)	7279 distinct values		35064 (100%)	0 (0%)
5	generation.fossil.oil [numeric]	Mean (sd) : 298.3 (52.5) min < med < max: 0 < 300 < 449 IQR (CV) : 67 (0.2)	334 distinct values		35064 (100%)	0 (0%)
6	generation.hydro.pumped.storage.consumption [numeric]	Mean (sd) : 475.6 (792.3) min < med < max: 0 < 68 < 4523 IQR (CV) : 616 (1.7)	3319 distinct values		35064 (100%)	0 (0%)

7	generation.hydro.run.of.river.and.poundage [numeric]	Mean (sd) : 972.2 (400.7) min < med < max: 0 < 906 < 2000 IQR (CV) : 613 (0.4)	1697 distinct values		35064 (100%)	0 (0%)
8	generation.hydro.water.reservoir [numeric]	Mean (sd) : 2605.5 (1835.2) min < med < max: 0 < 2165 < 9728 IQR (CV) : 2680 (0.7)	7040 distinct values		35064 (100%)	0 (0%)
9	generation.nuclear [numeric]	Mean (sd) : 6263.5 (840.3) min < med < max: 0 < 6564 < 7117 IQR (CV) : 1266 (0.1)	2396 distinct values		35064 (100%)	0 (0%)
10	generation.other [numeric]	Mean (sd) : 60.2 (20.2) min < med < max: 0 < 57 < 106 IQR (CV) : 27 (0.3)	112 distinct values		35064 (100%)	0 (0%)
11	generation.other.renewable [numeric]	Mean (sd) : 85.6 (14.1) min < med < max: 0 < 88 < 119 IQR (CV) : 24 (0.2)	87 distinct values		35064 (100%)	0 (0%)
12	generation.solar [numeric]	Mean (sd) : 1432.8 (1680) min < med < max: 0 < 616 < 5792 IQR (CV) : 2508 (1.2)	5344 distinct values		35064 (100%)	0 (0%)

13	generation.waste [numeric]	Mean (sd) : 269.4 (50.2) min < med < max: 0 < 279 < 357 IQR (CV) : 70 (0.2)	268 distinct values		35064 (100%)	0 (0%)
14	generation.wind.onshore [numeric]	Mean (sd) : 5465 (3213.6) min < med < max: 0 < 4849.5 < 17436 IQR (CV) : 4466.5 (0.6)	11477 distinct values		35064 (100%)	0 (0%)
15	total.load.actual [numeric]	Mean (sd) : 28698.3 (4575.8) min < med < max: 18041 < 28902 < 41015 IQR (CV) : 7387.2 (0.2)	15149 distinct values		35064 (100%)	0 (0%)
16	price.actual [numeric]	Mean (sd) : 57.9 (14.2) min < med < max: 9.3 < 58 < 116.8 IQR (CV) : 18.7 (0.2)	6653 distinct values		35064 (100%)	0 (0%)
17	temp [numeric]	Mean (sd) : 289.7 (7.3) min < med < max: 271.9 < 289 < 309.3 IQR (CV) : 11 (0)	19181 distinct values		35064 (100%)	0 (0%)
18	pressure [numeric]	Mean (sd) : 1016.1 (8.2) min < med < max: 974.6 < 1016.8 < 1039.8 IQR (CV) : 8 (0)	695 distinct values		35064 (100%)	0 (0%)
19	humidity [numeric]	Mean (sd) : 68 (14.8) min < med < max: 22.6 < 69.6 < 100 IQR (CV) : 23.4 (0.2)	364 distinct values		35064 (100%)	0 (0%)

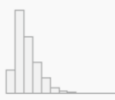
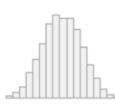



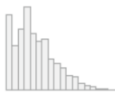
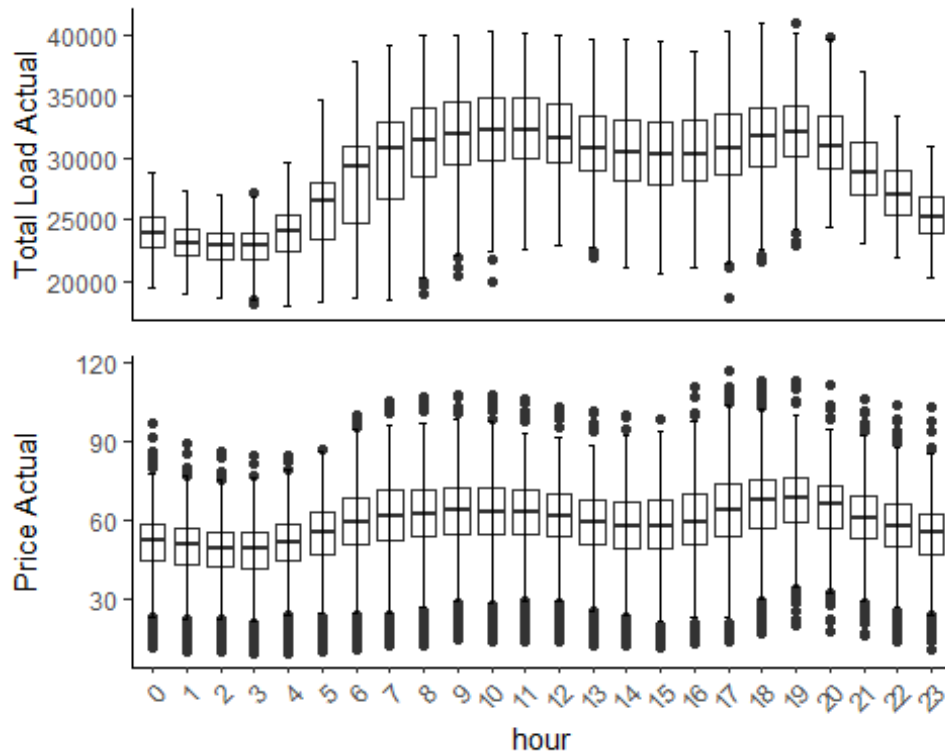
20	wind_speed [numeric]	Mean (sd) : 2.5 (1.3) min < med < max: 0 < 2.2 < 12.8 IQR (CV) : 1.8 (0.5)	60 distinct values		35064 (100%)	0 (0%)
21	wind_deg [numeric]	Mean (sd) : 166.7 (57.2) min < med < max: 0 < 166.2 < 338 IQR (CV) : 80.2 (0.3)	1502 distinct values		35064 (100%)	0 (0%)
22	rain_1h [numeric]	Mean (sd) : 0.1 (0.2) min < med < max: 0 < 0 < 3.2 IQR (CV) : 0.1 (2.8)	40 distinct values		35064 (100%)	0 (0%)
23	rain_3h [numeric]	Mean (sd) : 0 (0) min < med < max: 0 < 0 < 0.5 IQR (CV) : 0 (8.7)	104 distinct values		35064 (100%)	0 (0%)
24	snow_3h [numeric]	Mean (sd) : 0 (0.1) min < med < max: 0 < 0 < 4.3 IQR (CV) : 0 (20.9)	71 distinct values		35064 (100%)	0 (0%)
25	clouds_all [numeric]	Mean (sd) : 24.3 (17) min < med < max: 0 < 22 < 93.6 IQR (CV) : 22.4 (0.7)	434 distinct values		35064 (100%)	0 (0%)

Table 1 *Descriptive Statistics*

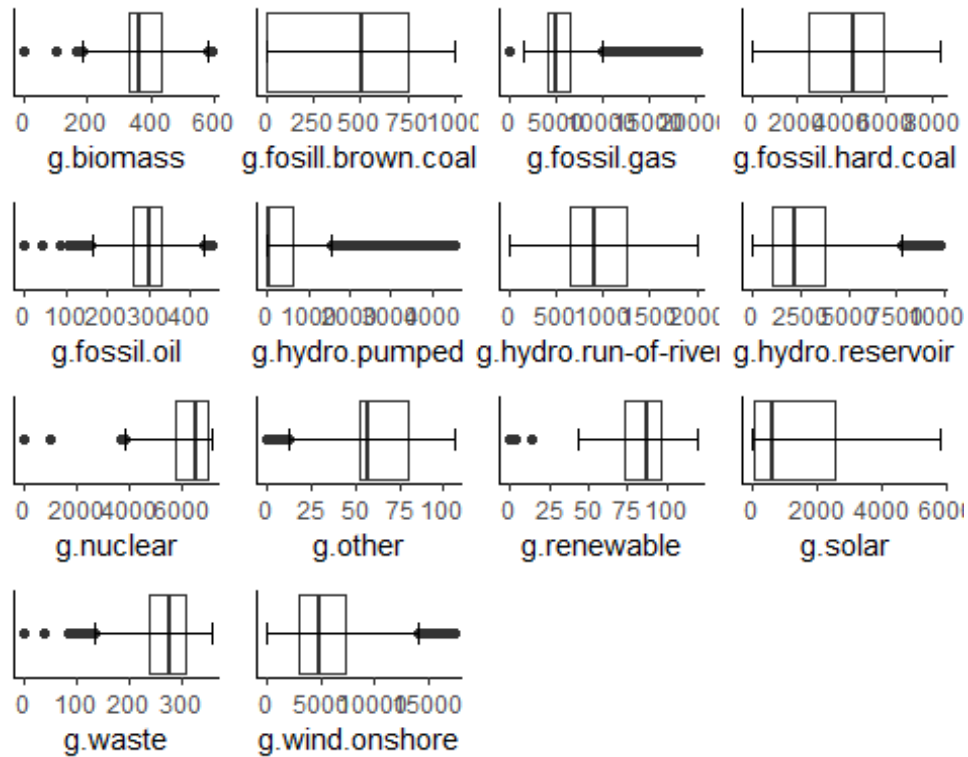
Based on common sense and real-world business, we suspect #LOAD and #PRICE as the target variables. The boxplot shows that #LOAD fluctuates more obviously over time. Meanwhile, power plants wish to know high demand and low demand in the market in order to act accordingly and in time. So, we think #LOAD would be a better choice as a target variable.



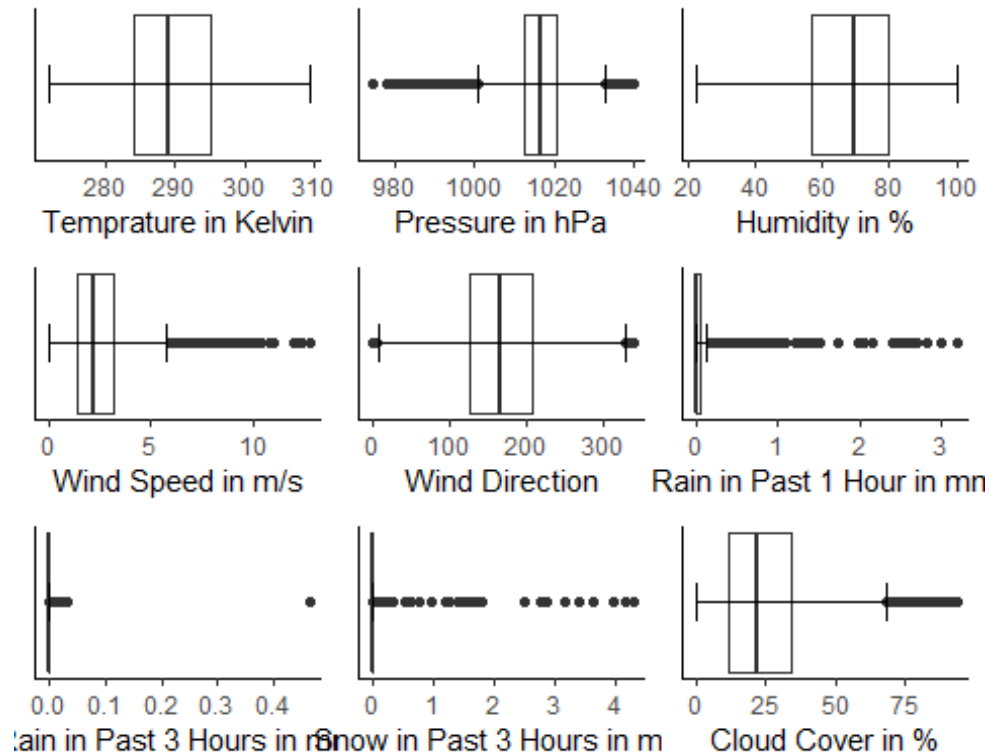
Plot 1 *Demand and Price by Hour*

As last time, we also give the box-and-whisker plots of energy generation methods and weather features. They reflect that these variables are in different scales, suggesting that we need to be cautious when doing the K-NN model. Still, we are not deleting the outliers right now to ensure the integrity of our dataset.

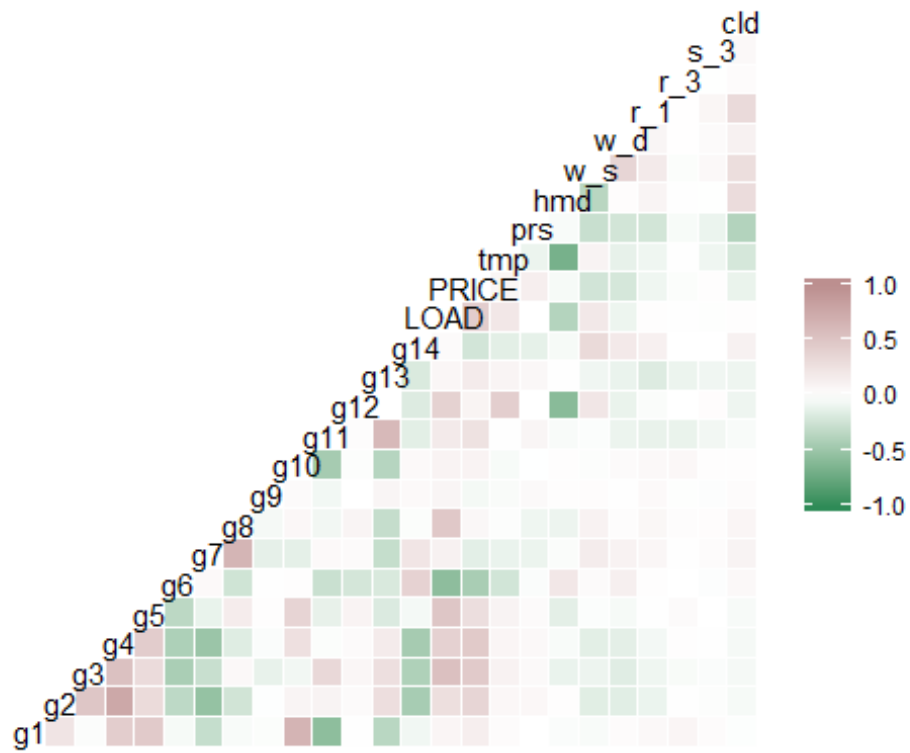
What's more, the correlation matrix and scatterplots are also showed here, which gives us an overall impression of the classification standards. Honestly, if we get familiarized with the distribution and correlation of target variable and potential variables, the model could be judged on an instinctive level of right from wrong, especially true for the classification tree model.



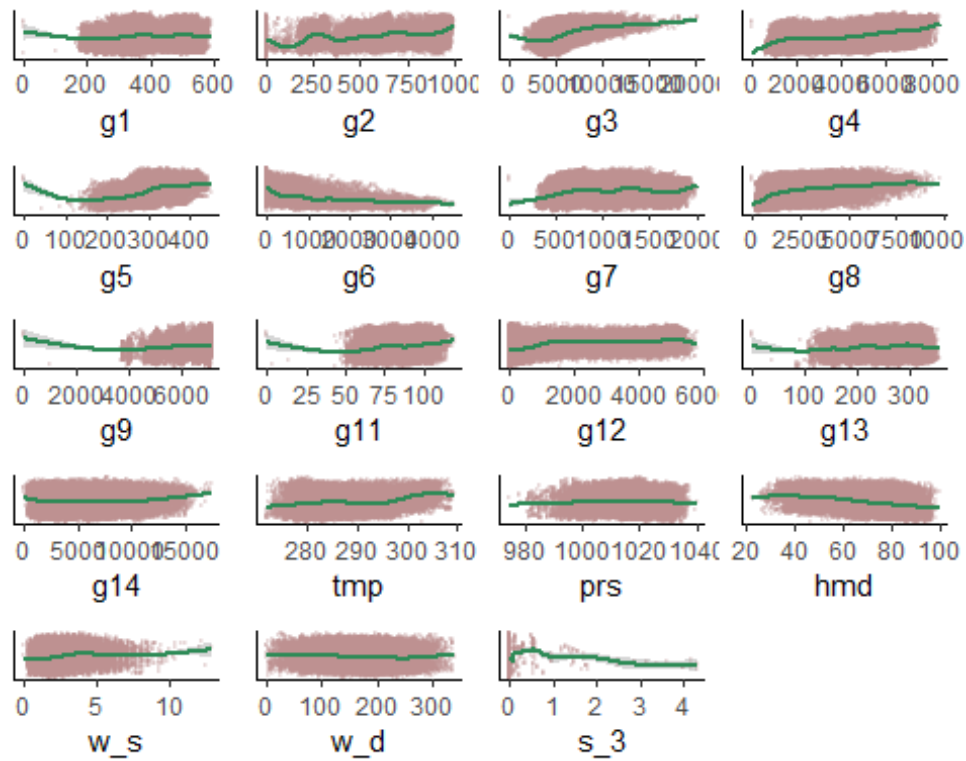
Plot 2 *Energy Generation Boxplot*



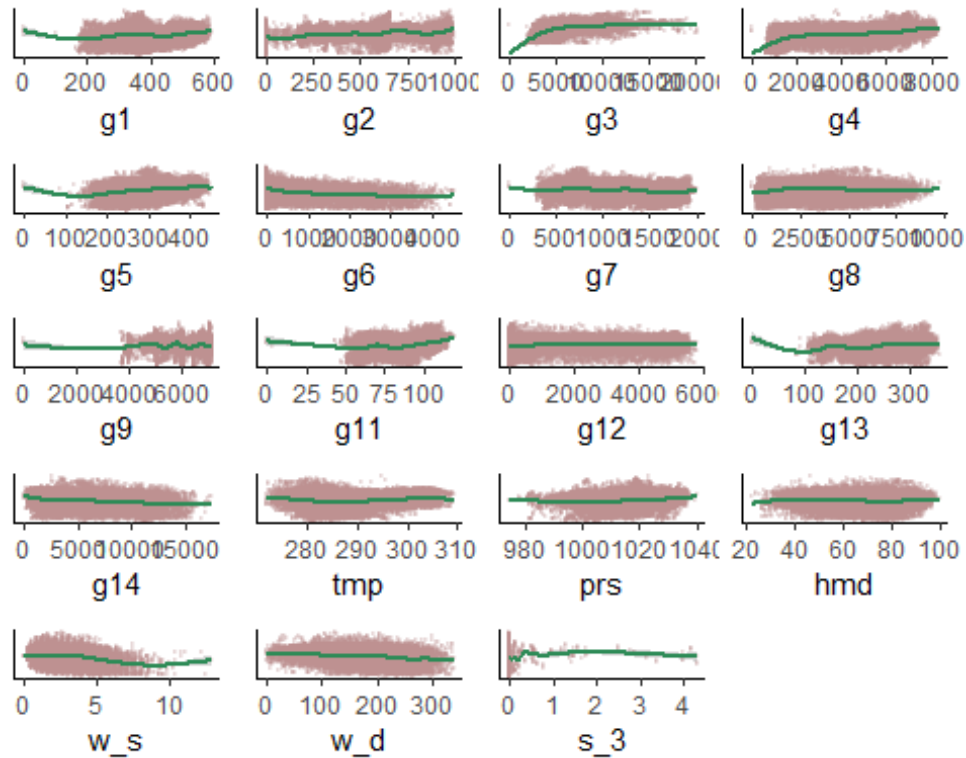
Plot 3 *Weather Feature Boxplot*



Plot 4 *Correlation Matrix*



Plot 5 *Scatterplot of Load vs. Potential Variables*



Plot 6 *Scatterplot of PRICE vs. Potential Variables*

PART II Logistic Regression

To do classification models, we first need to set classes. We divide #LOAD into three grades as beyond upper quartile, below lower quartile and the left over, respectively labeled high, low and median in three columns. And for future usage, we also integrate them into one column called #demand where “-1” stands for low, “0” for median and “1” for high. They are all factors.

We first run a logistic regression of #high versus all other potential variables, except for #PRICE, #LOAD, #med, #low and #demand, to avoid endogeneity and perfect multicollinearity. We can easily get a coefficient table as below.

```
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -93.0976484   5.1648655  -18.03 < 0.0000000000000002 ***
## g1           0.0030719   0.0005800    5.30 0.000000118061001 ***
## g2           0.0014869   0.0001333   11.15 < 0.0000000000000002 ***
## g3           0.0009683   0.0000216   44.80 < 0.0000000000000002 ***
## g4           0.0009107   0.0000309   29.50 < 0.0000000000000002 ***
## g5           0.0095760   0.0007642   12.53 < 0.0000000000000002 ***
## g6          -0.0021182   0.0001198  -17.68 < 0.0000000000000002 ***
## g7           0.0023144   0.0001303   17.77 < 0.0000000000000002 ***
## g8           0.0011485   0.0000295   38.94 < 0.0000000000000002 ***
## g9           0.0011809   0.0000391   30.23 < 0.0000000000000002 ***
## g10          0.0031755   0.0019622    1.62      0.1056
## g11          0.0354759   0.0033157   10.70 < 0.0000000000000002 ***
## g12          0.0009724   0.0000263   36.99 < 0.0000000000000002 ***
## g13          0.0088256   0.0008581   10.28 < 0.0000000000000002 ***
## g14          0.0009280   0.0000187   49.63 < 0.0000000000000002 ***
## tmp          0.0435133   0.0058258    7.47 0.0000000000000081 ***
## prs          0.0358024   0.0040805    8.77 < 0.0000000000000002 ***
## hmd          0.0232895   0.0034958    6.66 0.000000000027004 ***
## w_s          0.0737840   0.0271126    2.72      0.0065 **
## w_d          0.0009852   0.0005596    1.76      0.0783 .
## r_1          -0.0956387   0.1428574   -0.67      0.5032
## r_3         -23.6102585  12.0014739   -1.97      0.0492 *
## s_3          0.8393539   0.2879575    2.91      0.0036 **
## cld          -0.0074061   0.0021270   -3.48      0.0005 ***
## peak_hour     0.4462150   0.0832427    5.36 0.000000083033441 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Table 2 Original Logistic Regression of #high

Then, we remove those statistically insignificant variables and run the final model of #high using train set. And we will get the regression equation and coefficient table as below:¹

$$P(Y = 1 | X = x) = \frac{1}{1 + e^{-(-91.64 + 0.00346*g1 + 0.00149*g2 + 0.00091*g3 + 0.00974*g5 - 0.00211*g6 + 0.00230*g7 + 0.00115*g8 + 0.00118*g9 + 0.03518*g11 + 0.00097*g12 + 0.00876*g13 + 0.00093*g14 + 0.04218*tmp + 0.03492*prs + 0.02320*hmd + 0.0863*w_s - 24.7015*r_3 + 0.81789*s_3 - 0.00784*cld + 0.44875*peak_hour)}}$$

Equation 1 *Logistic Regression Equation of #high*

```
##
## Coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## (Intercept) -91.6406310    5.0561785  -18.12 < 0.0000000000000002 ***
## g1           0.0034611    0.0005404    6.40  0.00000000015109 ***
## g2           0.0014871    0.0001330   11.18 < 0.0000000000000002 ***
## g3           0.0009658    0.0000216   44.77 < 0.0000000000000002 ***
## g4           0.0009107    0.0000307   29.64 < 0.0000000000000002 ***
## g5           0.0097431    0.0007567   12.88 < 0.0000000000000002 ***
## g6          -0.0021086    0.0001195  -17.65 < 0.0000000000000002 ***
## g7           0.0023045    0.0001298   17.75 < 0.0000000000000002 ***
## g8           0.0011511    0.0000293   39.23 < 0.0000000000000002 ***
## g9           0.0011839    0.0000388   30.49 < 0.0000000000000002 ***
## g11          0.0351775    0.0033116   10.62 < 0.0000000000000002 ***
## g12          0.0009690    0.0000262   37.03 < 0.0000000000000002 ***
## g13          0.0087648    0.0008465   10.35 < 0.0000000000000002 ***
## g14          0.0009282    0.0000186   49.78 < 0.0000000000000002 ***
## tmp          0.0421781    0.0057939    7.28  0.000000000000033 ***
## prs          0.0349201    0.0040016    8.73 < 0.0000000000000002 ***
## hmd          0.0232031    0.0034867    6.65  0.000000000002839 ***
## w_s          0.0863110    0.0254835    3.39  0.00071 ***
## r_3         -24.7014659   11.9673444   -2.06  0.03901 *
## s_3          0.8178939    0.2890704    2.83  0.00466 **
## cld         -0.0078436    0.0020970   -3.74  0.00018 ***
## peak_hour    0.4487458    0.0827187    5.42  0.00000005796717 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Table 3 *Final Logistic Regression of #high*

¹ About the outliers: As there is no useful tool to point out the outliers, we use the same method in multiple regression report under the permission of TA.

How do we interpret the coefficients then? Take logistic regression of high as an example, when #tmp increases/decreases by a unit, the logit odd ratio of the demand labeled as high will go up/down by 0.00346. For better understanding the model, or more easily, let's look at the #peak_hour column. When it is a peak hour, the logit odd ratio of demand labeled as high will go up by 0.4487.

Similarly, we generate the model for #med and #low. And the interpretation of coefficients stays the same.

$$P(Y = 1 | X = x) = \frac{1}{1 + e^{-(12.4482 - 0.00094*g1 - 0.00030*g3 - 0.00005*g4 - 0.00109*g5 - 0.00108*g6 - 0.00052*g7 - 0.00003*g8 - 0.00021*g9 - 0.0328*g10 - 0.00005*g12 - 0.00184*g13 - 0.00527*prs - 0.01838*hmd - 0.00153*w_d - 0.28136*s_3 + 0.00440*cld)}}$$

Equation 2 Logistic Regression Equation of #med

```
##
## Coefficients:
##              Estimate Std. Error z value      Pr(>|z|)
## (Intercept) 12.4482776  1.9030676    6.54      0.000000000061 ***
## g1          -0.0009355  0.0002604   -3.59      0.00033 ***
## g3          -0.0002983  0.0000094  -31.74 < 0.0000000000000002 ***
## g4          -0.0000464  0.0000112   -4.13      0.000035776213 ***
## g5          -0.0010879  0.0003427   -3.17      0.00150 **
## g6          -0.0010781  0.0000267  -40.32 < 0.0000000000000002 ***
## g7          -0.0005245  0.0000576   -9.11 < 0.0000000000000002 ***
## g8          -0.0000313  0.0000115   -2.73      0.00638 **
## g9          -0.0002078  0.0000172  -12.07 < 0.0000000000000002 ***
## g10         -0.0032808  0.0009436   -3.48      0.00051 ***
## g12         -0.0000520  0.0000102   -5.07      0.000000397067 ***
## g13         -0.0018383  0.0003559   -5.17      0.000000239266 ***
## prs         -0.0052728  0.0018586   -2.84      0.00455 **
## hmd         -0.0183844  0.0012268  -14.99 < 0.0000000000000002 ***
## w_d         -0.0015260  0.0002521   -6.05      0.000000001428 ***
## s_3         -0.2813593  0.1323890   -2.13      0.03357 *
## cld          0.0043970  0.0009289    4.73      0.000002206015 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Table 4 Final Logistic Regression of #med

$$P(Y = 1 | X = x)$$

$$= \frac{1}{1 + e^{-(59.11422 - 0.00317*g1 - 0.00098*g2 - 0.00132*g3 - 0.00095*g4 - 0.00639*g5 + 0.00210*g6 - 0.00110*g7 - 0.00141*g8 - 0.00108*g9 - 0.02511*g11 - 0.00109*g12 - 0.00753*g13 - 0.00109*g14 - 0.01650*tmp - 0.00214*prs + 0.38822*r_1 + 0.64301*s_3 - 0.00526*cld - 1.14224*peak_hour)}}$$

Equation 3 *Logistic Regression Equation of #low*

```
##
## Coefficients:
##          Estimate Std. Error z value Pr(>|z|)
## (Intercept) 59.1142233  4.7288027  12.50 < 0.0000000000000002 ***
## g1          -0.0031682  0.0005388  -5.88  0.00000000409261845 ***
## g2          -0.0009847  0.0001483  -6.64  0.00000000003092884 ***
## g3          -0.0013188  0.0000461 -28.61 < 0.0000000000000002 ***
## g4          -0.0009518  0.0000340 -27.96 < 0.0000000000000002 ***
## g5          -0.0063889  0.0008511  -7.51  0.00000000000006080 ***
## g6           0.0021007  0.0000561  37.47 < 0.0000000000000002 ***
## g7          -0.0010993  0.0001366  -8.05  0.00000000000000084 ***
## g8          -0.0014073  0.0000403 -34.91 < 0.0000000000000002 ***
## g9          -0.0010790  0.0000416 -25.93 < 0.0000000000000002 ***
## g11         -0.0251111  0.0035256  -7.12  0.00000000000105941 ***
## g12         -0.0010915  0.0000458 -23.85 < 0.0000000000000002 ***
## g13         -0.0075255  0.0008666  -8.68 < 0.0000000000000002 ***
## g14         -0.0010909  0.0000223 -49.01 < 0.0000000000000002 ***
## tmp         -0.0164959  0.0052224  -3.16      0.0016 **
## prs         -0.0213567  0.0040201  -5.31  0.00000010814450427 ***
## r_1          0.3882168  0.1636855   2.37      0.0177 *
## s_3          0.6430062  0.1994498   3.22      0.0013 **
## cld         -0.0052604  0.0019874  -2.65      0.0081 **
## peak_hour   -1.1422417  0.1263068  -9.04 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Table 5 *Final Logistic Regression of #low*

After finishing the training part, how is the prediction outcome in the valid set? Below is the confusion matrix of #high, #med and #low. We set positive as 1 since in real world cases, we would like to know when there is a high/low demand in the market.

Take confusion matrix of #high as an example, there are 7532 observations (True Negative) that are predicted to be low in the valid set and really are low; 2150 observations (True Positive) that are predicted to be high in the valid set and really are high; 460 observations (False Negative) that are predicted to be low in the valid

set but are high actually; and 378 observations (False Positive) that are predicted to be high in the valid set but are low actually.

```
##
##           Reference
## Prediction    0    1
##           0 7532  460
##           1  378 2150
##
```

Table 6 *Confusion Matrix of #high*

```
##
##           Reference
## Prediction    0    1
##           0 3164 1544
##           1 2048 3764
##
```

Table 7 *Confusion Matrix of #med*

```
##
##           Reference
## Prediction    0    1
##           0 7578  349
##           1  340 2253
##
```

Table 8 *Confusion Matrix of #low*

These absolute values are hard to figure out the performance of the models, although we can clearly see logistic model of #med is dragging the leg. Please refer to the table below to compare some important indices.

	#high	#med	#low
Accuracy	.92	.659	.935
Sensitivity	.824	.709	.866
Specificity	.952	.607	.957
Recall	.824	.709	.866
Precision	.850	.648	.869
F1 value	.84	.68	.87

Table 9 *Important Indices of #high, #med and #low*

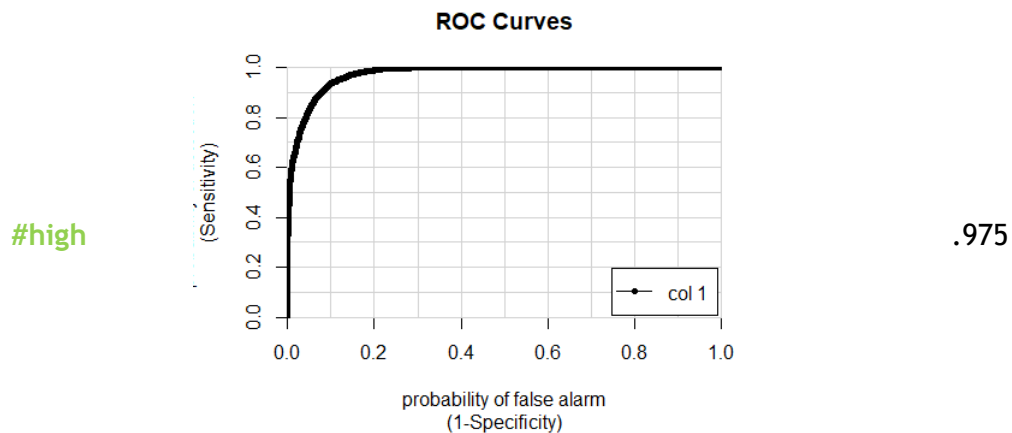
We like the outcome of #high and #low, as their accuracy indices are high, while sensitivity and specificity indices are in a good range and F1 Value are approximate. And there is no sign of overfitting.

We are not so happy with #med as the indices are not so persuasive. However, if we put the whole task in another way of thinking, in real-world cases where we intend to know high/low demand in the market more, we do not quite care about the average stuff.

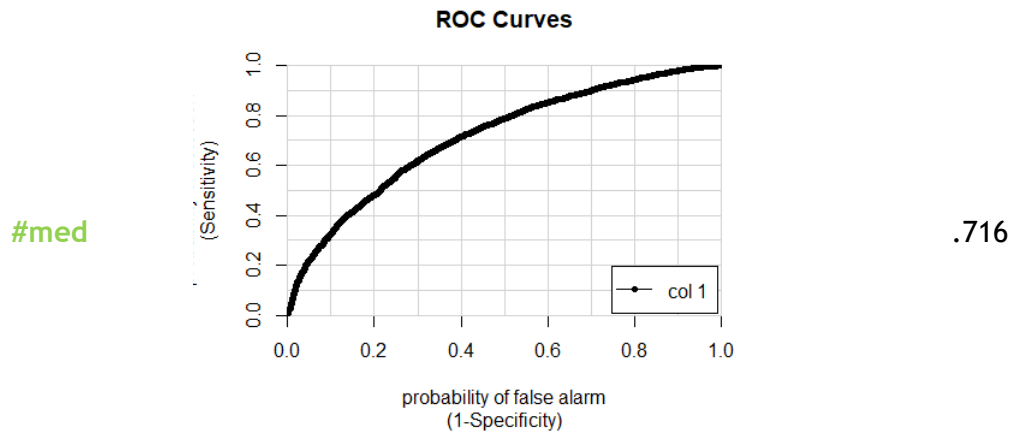
To show it more intuitively, we introduce ROC (Receiving Operating Characteristic), aka Sensitivity Curve to the comparison. It proves the analysis again.

ROC Curve Plots

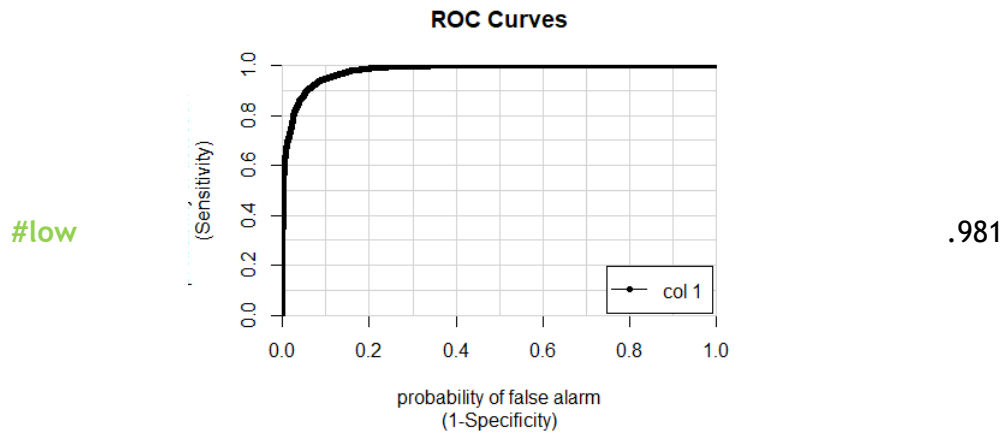
Area under Curve



Plot 7 *ROC Curve of #high*



Plot 8 ROC Curve of #med



Plot 9 ROC Curve of #low

Table 10 ROC Curves

To be true, above is the analysis of a single model, high, low or med. For a single observation in the valid set, each model will be predicting a value, are they in accordance with each other? So, we combine the outcomes together. That is, each observation will be having three prediction values, and we pick the biggest one (max) since we set the positive as 1 to be the reference of label (classification). We generate a confusion matrix as below.

##					
##		x\$max_value			
##	valid\$demand	high	low	median	Row Total
##	-----	-----	-----	-----	-----
##	high	2124	1	485	2610

```
##          | 3595.613 | 627.676 | 558.915 | 0.248 |
##          | 0.814   | 0.000   | 0.186   |        |
##          | 0.843   | 0.000   | 0.089   |        |
##          | 0.202   | 0.000   | 0.046   |        |
## -----|-----|-----|-----|-----|
##          low      | 0        | 2211    | 391     | 2602    |
##          | 623.045 | 3993.178 | 682.353 |        |
##          | 0.000   | 0.850   | 0.150   | 0.247   |
##          | 0.000   | 0.871   | 0.072   |        |
##          | 0.000   | 0.210   | 0.037   |        |
## -----|-----|-----|-----|-----|
##          median   | 395      | 326     | 4587    | 5308    |
##          | 603.752 | 711.571 | 1215.705 |        |
##          | 0.074   | 0.061   | 0.864   | 0.505   |
##          | 0.157   | 0.128   | 0.840   |        |
##          | 0.038   | 0.031   | 0.436   |        |
## -----|-----|-----|-----|-----|
## Column Total      | 2519     | 2538    | 5463    | 10520   |
##          | 0.239   | 0.241   | 0.519   |        |
## -----|-----|-----|-----|-----|
##
##
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##          Reference
```

```
## Prediction high low median
```

```
##    high    2124     1    485
```

```
##    low      0    2211    391
```

```
##    median  395    326   4587
```

```
##
```

Table 11 *Confusion Matrix of Combination of #high, #med and #low*

	logit.pred.h <dbl>	logit.pred.m <dbl>	logit.pred.l <dbl>	max_value <chr>
174	0.000023262723834	0.26652016	0.927286968803313	low
178	0.005115193884697	0.47320527	0.019451951877712	median
180	0.094135001183359	0.53640190	0.000659164180310	median
186	0.043681364223427	0.55619594	0.001154810006772	median
190	0.029277825961098	0.49728471	0.003152908634432	median
195	0.000655713480905	0.59087920	0.234045655237242	median
203	0.883330393027766	0.42063479	0.000003644183314	high
206	0.981118212511818	0.39270122	0.000000473893096	high
209	0.932963042339452	0.43466523	0.000002592758272	high
211	0.865808740778072	0.37099645	0.000007157051508	high

Table 12 *Role Example*

We are not going to repeat the interpretation. Let's just head towards the statistics.

We can see a moderate accuracy level **0.848** this time, neither too high nor too low.

But we are not able to give an overall statistic of other indices as they are calculated in that way.²

```
## Overall Statistics
##
##           Accuracy : 0.848
##           95% CI : (0.841, 0.855)
##      No Information Rate : 0.519
##      P-Value [Acc > NIR] : < 0.0000000000000002
##
##           Kappa : 0.755
##
##  Mcnemar's Test P-Value : 0.00108
##
## Statistics by Class:
##
##           Class: high Class: low Class: median
## Sensitivity           0.843      0.871      0.840
## Specificity           0.939      0.951      0.857
## Pos Pred Value        0.814      0.850      0.864
## Neg Pred Value        0.950      0.959      0.832
## Prevalence            0.239      0.241      0.519
## Detection Rate        0.202      0.210      0.436
## Detection Prevalence  0.248      0.247      0.505
## Balanced Accuracy      0.891      0.911      0.849
```

Table 13 *Statistics of Combination of #high, #med and #low*

We still have some space to talk about the logistic regression. Essentially, what we have gone through is based on the premise where Y is a binary variable. However, we have already classified #demand into three categories as “-1”, “0” and “1” in advance, will it be possible that we do the regression within one try but including all three categories as Y? Below is a try.

The logic still is the same. We generate a logistic regression and do the coefficient test to pick the independent variables. Then we use train set to train the model and valid test to predict the model. Finally, we present confusion matrix and overall statistics as a reference of the performance of the regression.

² ROC is only applicable to binary variables, so part of the logistic model, K-NN and Tree Models will not be using ROC plots to do the comparison.

##				
##		pred_final		
##	valid\$demand	-1	0	1
##				Row Total
##				
##	-1	2234	368	0
##		4004.953	703.078	627.250
##		0.859	0.141	0.000
##		0.868	0.068	0.000

##		0.212	0.035	0.000	
##	-----	-----	-----	-----	-----
##	0	340	4574	394	5308
##		708.224	1247.027	612.890	
##		0.064	0.862	0.074	0.505
##		0.132	0.846	0.155	
##		0.032	0.435	0.037	
##	-----	-----	-----	-----	-----
##	1	1	467	2142	2610
##		636.856	570.481	3637.485	
##		0.000	0.179	0.821	0.248
##		0.000	0.086	0.845	
##		0.000	0.044	0.204	
##	-----	-----	-----	-----	-----
##	Column Total	2575	5409	2536	10520
##		0.245	0.514	0.241	
##	-----	-----	-----	-----	-----

##

Confusion Matrix and Statistics

##				
##		Reference		
##	Prediction	-1	0	1
##	-1	2234	368	0
##	0	340	4574	394
##	1	1	467	2142

Table 14 *Confusion Matrix of Logistic Regression of Three Ctgs.*

```
##
## Overall Statistics
##
##           Accuracy : 0.851
##           95% CI   : (0.844, 0.858)
##       No Information Rate : 0.514
##       P-Value [Acc > NIR] : <0.0000000000000002
##
##           Kappa   : 0.759
##
##  McNemar's Test P-Value : 0.0403
##
## Statistics by Class:
##
##           Class: -1 Class: 0 Class: 1
## Sensitivity           0.868   0.846   0.845
## Specificity           0.954   0.856   0.941
## Pos Pred Value        0.859   0.862   0.821
## Neg Pred Value        0.957   0.840   0.950
## Prevalence            0.245   0.514   0.241
## Detection Rate        0.212   0.435   0.204
```

## Detection Prevalence	0.247	0.505	0.248
## Balanced Accuracy	0.911	0.851	0.893

Table 15 Overall Statistics of Three Ctgs.

We can see that the confusion matrix columns are almost the same and the accuracy .851 is slightly enhanced compared with the combination one .848, which somehow indicates that our outcomes are in accordance with each other.

We also attempt to add some interaction or higher order terms just like what we have done last time. We see a floating-up in Accuracy (0.8548 > 0.851) by doing so. Despite the fact that these newly added terms may not have a practical meaning, we do make the model predict better.

$$P(Y \leq -1 | X = x) = \frac{1}{1 + e^{-\frac{1}{- (5.0365 - 0.3054*g1 - 0.4107*g2 - 2.7578*g3 - 1.7246*g4 - 0.3423*g5 + 1.9583*g6 - 0.6113*g7 - 2.4864*g8 - 0.9369*g9 - 0.3540*g11 - 1.7484*g12 - 0.3758*g13 - 3.2932*g14 - 0.1438*tmp - 0.2517*prs - 0.0934*hmd - 0.0689*ws + 0.0496*r1 - 0.371383*peak_{hour} + 0.1204*(g8)^2 + 0.2332*(g3)^2 - 0.1840*(g6)^2}}}}$$

Equation 7 Logistic Regression Equation with Higher Order Terms of “-1”

$$P(-1 < Y \leq 0 | X = x) = P(Y \leq 0 | X = x) - P(Y \leq -1 | X = x) = \frac{1}{1 + e^{-\frac{1}{- (4.5854 - 0.3054*g1 - 0.4107*g2 - 2.7578*g3 - 1.7246*g4 - 0.3423*g5 + 1.9583*g6 - 0.6113*g7 - 2.4864*g8 - 0.9369*g9 - 0.3540*g11 - 1.7484*g12 - 0.3758*g13 - 3.2932*g14 - 0.1438*tmp - 0.2517*prs - 0.0934*hmd - 0.0689*ws + 0.0496*r1 - 0.371383*peak_{hour} + 0.1204*(g8)^2 + 0.2332*(g3)^2 - 0.1840*(g6)^2}}}} - \frac{1}{1 + e^{-\frac{1}{- (5.0365 - 0.3054*g1 - 0.4107*g2 - 2.7578*g3 - 1.7246*g4 - 0.3423*g5 + 1.9583*g6 - 0.6113*g7 - 2.4864*g8 - 0.9369*g9 - 0.3540*g11 - 1.7484*g12 - 0.3758*g13 - 3.2932*g14 - 0.1438*tmp - 0.2517*prs - 0.0934*hmd - 0.0689*ws + 0.0496*r1 - 0.371383*peak_{hour} + 0.1204*(g8)^2 + 0.2332*(g3)^2 - 0.1840*(g6)^2}}}}$$

Equation 8 Logistic Regression Equation with Higher Order Terms of “0”

$$P(Y \geq 1 | X = x) = 1 - P(-1 < Y \leq 0 | X = x)$$

$$= \frac{1}{1 + e^{-(-4.5854 - 0.3054*g1 - 0.4107*g2 - 2.7578*g3 - 1.7246*g4 - 0.3423*g5 + 1.9583*g6 - 0.6113*g7 - 2.4864*g8 - 0.9369*g9 - 0.3540*g11 - 1.7484*g12 - 0.3758*g13 - 3.2932*g14 - 0.1438*tmp - 0.2517*prs - 0.0934*hmd - 0.0689*ws + 0.0496*r1 - 0.371383*peak_{hour} + 0.1204*(g8)^2 + 0.2332*(g3)^2 - 0.1840*(g6)^2)}}$$

Equation 9 Logistic Regression Equation with Higher Order Terms of “1”

valid\$demand	pred_final			Row Total
	-1	0	1	
-1	2250	352	0	2602
	4048.887	722.555	628.240	
	0.865	0.135	0.000	0.247
	0.869	0.065	0.000	
	0.214	0.033	0.000	
0	337	4587	384	5308
	718.781	1280.393	628.646	
	0.063	0.864	0.072	0.505
	0.130	0.851	0.151	
	0.032	0.436	0.037	
1	1	453	2156	2610
	640.081	585.148	3694.479	
	0.000	0.174	0.826	0.248
	0.000	0.084	0.849	
	0.000	0.043	0.205	
Column Total	2588	5392	2540	10520
	0.246	0.513	0.241	

Confusion Matrix and Statistics

	Reference			
Prediction	-1	0	1	
-1	2250	352	0	
0	337	4587	384	
1	1	453	2156	

Table 16 Confusion Matrix of Logistic Regression with Higher Order Terms of Three Ctg.

Overall Statistics			
Accuracy : 0.8548			
95% CI : (0.848, 0.8615)			
No Information Rate : 0.5125			
P-Value [Acc > NIR] : < 0.0000000000000002			
Kappa : 0.7661			
McNemar's Test P-Value : 0.07143			
Statistics by Class:			
	Class: -1	Class: 0	Class: 1
Sensitivity	0.8694	0.8507	0.8488
Specificity	0.9556	0.8594	0.9431
Pos Pred Value	0.8647	0.8642	0.8261
Neg Pred Value	0.9573	0.8455	0.9515
Prevalence	0.2460	0.5125	0.2414
Detection Rate	0.2139	0.4360	0.2049
Detection Prevalence	0.2473	0.5046	0.2481
Balanced Accuracy	0.9125	0.8551	0.8960

Table 17 Overall Statistics of Logistic Regression with Higher Order Terms of Three Ctg.

Class	Recall	Precision	F1
-1	0.869	0.865	0.867
0	0.851	0.864	0.857
1	0.849	0.826	0.837
Macro F1		0.8539	
MacroPrecision		0.8516	
Macro Recall		0.8563	
Weighted F1		0.8549	
Weighted Precision		0.8551	
Weighted Recall		0.8548	
Accuracy		0.8548 ³	

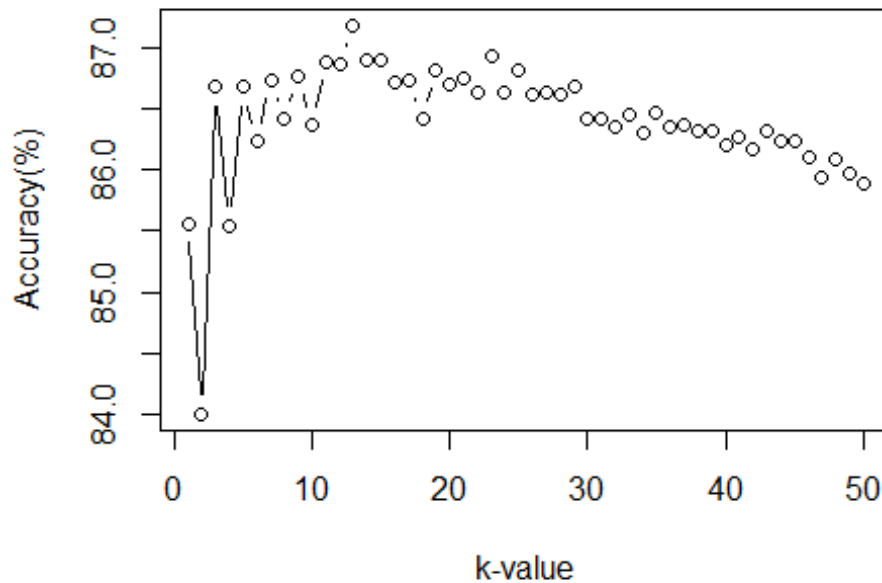
Table 18 Multi Classifier F1 - Logistic Regression

³ Reference: <https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1>

PART III K-NN

After trial with a k value of 3, we run a loop to find the optimized k and we decide to pick that one as our K-NN model referral. The best k turns out to be 13, with an accuracy of 87.17%. We are happy that k is an odd number, which means at least one side will be having 1 more observation than the other. Below is a scatterplot of k-value and accuracy.

And there is not a huge bump between the train set and valid set (0.866 and 0.872) so we do not have to be worried about the overfitting issues.



Plot 10 Scatterplot of k-values versus Accuracy

##				
##				
## valid.labels data_test_pred				
## -----				
## high				
## -----				
## 2206 1 403 2610				
## 4315.509 677.544 647.742				
## 0.845 0.000 0.154 0.248				
## 0.914 0.000 0.075				
## 0.210 0.000 0.038				
## -----				
## low				
## 0 2301 301 2602				
## 596.828 3890.832 793.951				
## 0.000 0.884 0.116 0.247				
## 0.000 0.840 0.056				

##		0.000	0.219	0.029	
##	-----	-----	-----	-----	-----
##	median	207	437	4664	5308
##		838.704	646.181	1411.858	
##		0.039	0.082	0.879	0.505
##		0.086	0.160	0.869	
##		0.020	0.042	0.443	
##	-----	-----	-----	-----	-----
##	Column Total	2413	2739	5368	10520
##		0.229	0.260	0.510	
##	-----	-----	-----	-----	-----
##					
##					

Table 18 *Confusion Matrix of K-NN*

```

##
## Overall Statistics
##
##           Accuracy : 0.872
##           95% CI   : (0.865, 0.878)
##       No Information Rate : 0.505
##       P-Value [Acc > NIR] : <0.0000000000000002
##
##           Kappa   : 0.794
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
## Statistics by Class:
##
##           Class: high Class: low Class: median
## Sensitivity           0.845      0.884      0.879
## Specificity           0.974      0.945      0.865
## Pos Pred Value        0.914      0.840      0.869
## Neg Pred Value        0.950      0.961      0.875
## Prevalence            0.248      0.247      0.505
## Detection Rate        0.210      0.219      0.443
## Detection Prevalence  0.229      0.260      0.510
## Balanced Accuracy     0.910      0.915      0.872

```

Table 19 *Overall Statistics of K-NN*

Class	Recall	Precision	F1
-1	0.840	0.884	0.862
0	0.869	0.879	0.874
1	0.914	0.845	0.878

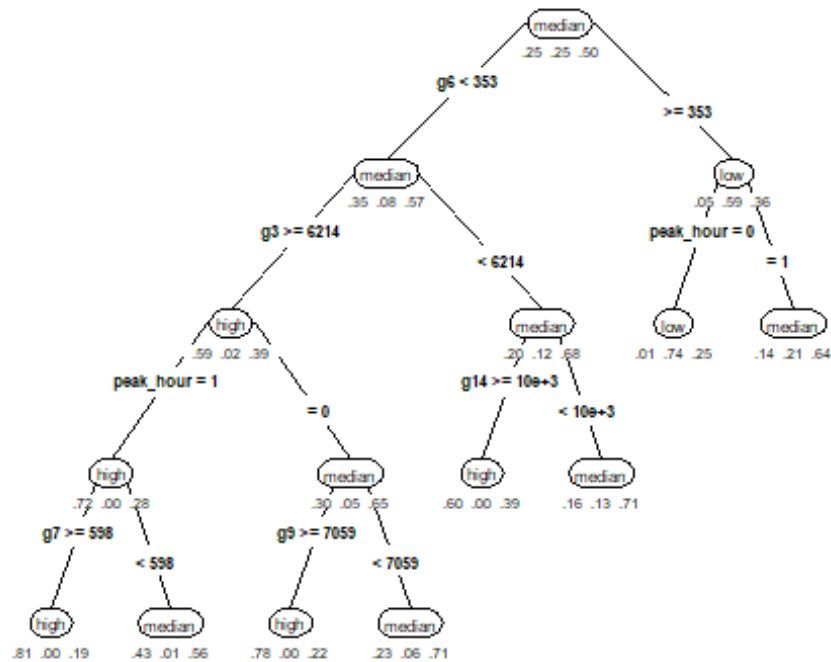
Macro F1	0.8712
----------	--------

MacroPrecision	0.8694
Macro Recall	0.8744
Weighted F1	0.8716
Weighted Precision	0.8725
Weighted Recall	0.8718
Accuracy	0.872

Table 20 *Multi Classifier F1 - K-NN*

PART IV Tree Models

Finally, the Tree Models. Let's begin with Classification Tree. Below is the tree generated with test set, having an accuracy of 0.715.



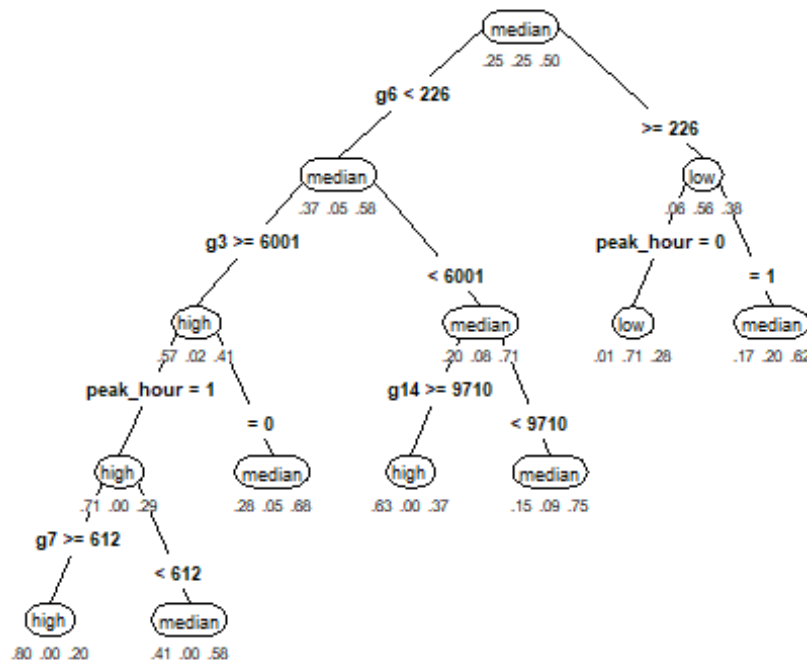
Plot 11 Classification Tree with Train Set

```
##
##           Reference
## Prediction high  low  median
##   high   3484    4   1072
##   low     57 4351   1443
##   median 2619 1810   9704
##
## Overall Statistics
##
##           Accuracy : 0.715
##           95% CI : (0.709, 0.72)
##   No Information Rate : 0.498
##   P-Value [Acc > NIR] : <0.0000000000000002
##
##           Kappa : 0.53
##
##   McNemar's Test P-Value : <0.0000000000000002
##
## Statistics by Class:
```

```
##
##                               Class: high Class: low Class: median
## Sensitivity                   0.566      0.706      0.794
## Specificity                   0.941      0.918      0.641
## Pos Pred Value                0.764      0.744      0.687
## Neg Pred Value                0.866      0.903      0.758
## Prevalence                    0.251      0.251      0.498
## Detection Rate                0.142      0.177      0.395
## Detection Prevalence         0.186      0.238      0.576
## Balanced Accuracy             0.754      0.812      0.717
```

Table 21 Confusion Matrix of *Classification Tree with Train Set*

After that, we will apply it to the valid set. We get an accuracy of 0.715, almost no difference to the train set. And again, due to the multi classifier issue, we are not able to plot the ROC. But we should definitely try to dig it deeper.



Plot 11 *Classification Tree with Valid Set*

```
##
##           Reference
## Prediction high  low  median
##    high   1436    0    443
##    low     27 2002    784
##    median 1147   600   4081
##
```

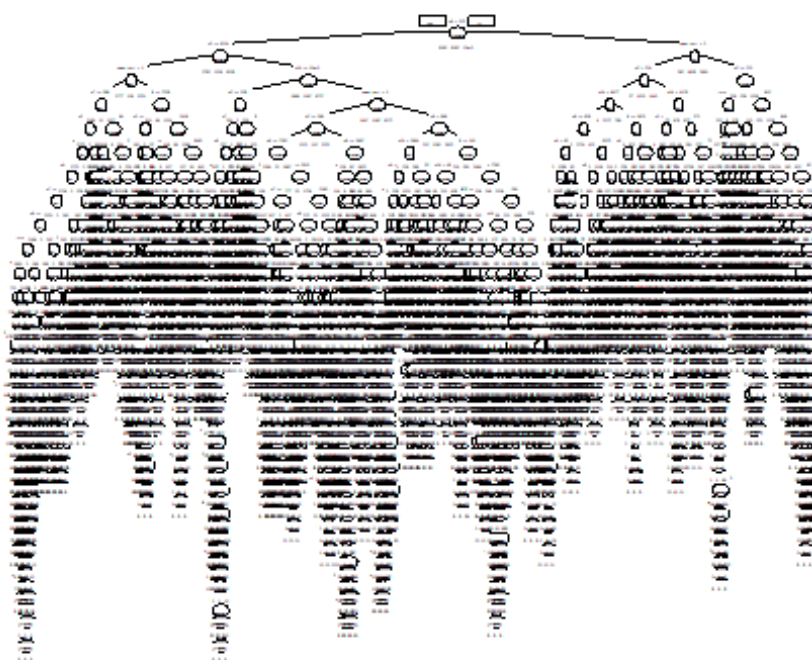
```

## Overall Statistics
##
##           Accuracy : 0.715
##           95% CI   : (0.706, 0.723)
##      No Information Rate : 0.505
##      P-Value [Acc > NIR] : <0.0000000000000002
##
##           Kappa   : 0.532
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
## Statistics by Class:
##
##               Class: high Class: low Class: median
## Sensitivity           0.550      0.769      0.769
## Specificity           0.944      0.898      0.665
## Pos Pred Value        0.764      0.712      0.700
## Neg Pred Value        0.864      0.922      0.738
## Prevalence            0.248      0.247      0.505
## Detection Rate        0.137      0.190      0.388
## Detection Prevalence  0.179      0.267      0.554
## Balanced Accuracy      0.747      0.833      0.717

```

Table 22 Confusion Matrix of *Classification Tree with Valid Set*

Then we move on to deeper tree. Our deeper tree turns out to have 2433 leaves and the plot is as below. Confusion Matrix shows that it has an accuracy of 1, indicating the model is almost surely under the risk of overfitting. Naturally, we have to verify this situation.



Plot 12 *Deeper Tree Model*

Confusion Matrix and Statistics

##

Reference

Prediction high low median

high 6160 0 0

low 0 6165 0

median 0 0 12219

##

Overall Statistics

##

Accuracy : 1

95% CI : (1, 1)

No Information Rate : 0.498

P-Value [Acc > NIR] : <0.0000000000000002

##

Kappa : 1

##

McNemar's Test P-Value : NA

##

Statistics by Class:

##

Class: high Class: low Class: median

Sensitivity 1.000 1.000 1.000

Specificity 1.000 1.000 1.000

Pos Pred Value 1.000 1.000 1.000

## Neg Pred Value	1.000	1.000	1.000
## Prevalence	0.251	0.251	0.498
## Detection Rate	0.251	0.251	0.498
## Detection Prevalence	0.251	0.251	0.498
## Balanced Accuracy	1.000	1.000	1.000

Table 23 Confusion Matrix of *Deeper Tree with Train Set*

After putting the deeper tree model under prediction with valid set, the accuracy dives to 0.824, far lower than 1. We can now draw the conclusion that the model is definitely overfitting.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction high  low median
##      high  2127    5   439
##      low     2 2144   473
##      median 481  453  4396
##
## Overall Statistics
##
##           Accuracy : 0.824
##           95% CI : (0.816, 0.831)
##      No Information Rate : 0.505
##      P-Value [Acc > NIR] : <0.0000000000000002
##
##           Kappa : 0.717
##
##  McNemar's Test P-Value : 0.304
##
## Statistics by Class:
##
##           Class: high Class: low Class: median
## Sensitivity          0.815      0.824      0.828
## Specificity          0.944      0.940      0.821
## Pos Pred Value       0.827      0.819      0.825
## Neg Pred Value       0.939      0.942      0.824
## Prevalence           0.248      0.247      0.505
## Detection Rate       0.202      0.204      0.418
## Detection Prevalence 0.244      0.249      0.507
## Balanced Accuracy    0.879      0.882      0.824
```

Table 24 Confusion Matrix of *Deeper Tree with Valid Set*

So, we have to choose the best CP (complexity parameter). When CP is .0001826, xerror has the minimum of .35294. We allow a standard deviation larger than xerror,

that is $.35294 + .00485 = .35779$. When the first xerror is closest to and less than $.35779$, CP is $.0002898$.

	CP	nsplit	rel error	xerror	xstd
1	0.153995943	0	1.000000	1.00000	0.0063555
2	0.101095335	1	0.846004	0.84682	0.0062841
3	0.081947262	2	0.744909	0.77136	0.0061922
4	0.057849899	3	0.662961	0.65809	0.0059791
5	0.015172414	4	0.605112	0.61047	0.0058606
6	0.011196755	5	0.589939	0.60803	0.0058541
7	0.010385396	6	0.578742	0.59846	0.0058279
8	0.007707911	7	0.568357	0.54856	0.0056787
9	0.007139959	8	0.560649	0.53874	0.0056468
10	0.006267748	9	0.553509	0.53420	0.0056317
11	0.005598377	14	0.517972	0.52503	0.0056007
12	0.005476673	15	0.512373	0.51911	0.0055802
13	0.004868154	18	0.495903	0.50710	0.0055378
14	0.004689655	20	0.486166	0.49874	0.0055074
15	0.004381339	25	0.462718	0.49590	0.0054969
16	0.003813387	26	0.458337	0.48600	0.0054597
17	0.003164300	27	0.454523	0.48057	0.0054389
18	0.003002028	28	0.451359	0.47424	0.0054143
19	0.002839757	30	0.445355	0.47002	0.0053976
20	0.002271805	32	0.439675	0.46450	0.0053756
21	0.002068966	38	0.424016	0.45331	0.0053298
22	0.001703854	41	0.417444	0.44471	0.0052938
23	0.001582150	44	0.412333	0.44138	0.0052796
24	0.001541582	46	0.409168	0.43740	0.0052625

25	0.001379310	48	0.406085	0.43497	0.0052520
26	0.001298174	49	0.404706	0.43221	0.0052399
27	0.001257606	52	0.400811	0.42986	0.0052296
28	0.001217039	56	0.394807	0.42677	0.0052159
29	0.001189993	57	0.393590	0.42475	0.0052069
30	0.001176471	60	0.390020	0.42475	0.0052069
31	0.001135903	62	0.387667	0.42296	0.0051989
32	0.001054767	65	0.384260	0.42118	0.0051909
33	0.001014199	67	0.382150	0.41826	0.0051776
34	0.000973631	69	0.380122	0.41420	0.0051591
35	0.000919540	78	0.371034	0.40990	0.0051393
36	0.000892495	81	0.368276	0.40698	0.0051256
37	0.000851927	85	0.364706	0.40592	0.0051207
38	0.000811359	90	0.359757	0.40349	0.0051092
39	0.000784314	111	0.342394	0.39951	0.0050904
40	0.000770791	119	0.335740	0.39846	0.0050853
41	0.000730223	123	0.332657	0.39448	0.0050662
42	0.000689655	135	0.323813	0.38832	0.0050361
43	0.000676133	142	0.317647	0.38734	0.0050314
44	0.000649087	146	0.314645	0.38734	0.0050314
45	0.000622042	158	0.306288	0.38572	0.0050233
46	0.000608519	161	0.304422	0.38353	0.0050125
47	0.000567951	163	0.303205	0.37931	0.0049914
48	0.000540906	176	0.295740	0.37606	0.0049750
49	0.000527383	180	0.293469	0.37542	0.0049717
50	0.000513861	185	0.290710	0.37509	0.0049700
51	0.000507099	188	0.289168	0.37509	0.0049700

52	0.000486815	192	0.287140	0.36779	0.0049325
53	0.000446247	226	0.270020	0.36787	0.0049329
54	0.000432725	236	0.265558	0.36519	0.0049190
55	0.000405680	239	0.264260	0.36349	0.0049101
56	0.000385396	281	0.246734	0.36154	0.0048998
57	0.000378634	287	0.243895	0.36122	0.0048981
58	0.000365112	290	0.242759	0.36016	0.0048925
59	0.000351589	296	0.240568	0.36016	0.0048925
60	0.000324544	299	0.239513	0.35870	0.0048848
61	0.000297498	365	0.217850	0.35830	0.0048826
62	0.000289771	368	0.216957	0.35554	0.0048679
63	0.000283976	376	0.214442	0.35554	0.0048679
64	0.000270453	395	0.208844	0.35481	0.0048640
65	0.000243408	401	0.207221	0.35538	0.0048670
66	0.000216362	488	0.185639	0.35481	0.0048640
67	0.000202840	496	0.183854	0.35465	0.0048631
68	0.000198332	541	0.174523	0.35310	0.0048548
69	0.000189317	550	0.172738	0.35310	0.0048548
70	0.000182556	576	0.167465	0.35294	0.0048540
71	0.000162272	580	0.166734	0.35465	0.0048631
72	0.000146045	865	0.119351	0.35521	0.0048662
73	0.000141988	870	0.118621	0.35659	0.0048736
74	0.000135227	891	0.115213	0.35651	0.0048731
75	0.000129817	914	0.111886	0.35732	0.0048775
76	0.000121704	919	0.111237	0.35870	0.0048848
77	0.000113590	1045	0.095010	0.35927	0.0048878
78	0.000108181	1050	0.094442	0.35911	0.0048870

79	0.000101420	1089	0.090223	0.35911	0.0048870
80	0.000094659	1105	0.088600	0.36625	0.0049245
81	0.000081136	1111	0.088032	0.36665	0.0049266
82	0.000064909	1389	0.064990	0.36690	0.0049279
83	0.000063106	1394	0.064665	0.36828	0.0049350
84	0.000060852	1403	0.064097	0.36828	0.0049350
85	0.000054091	1411	0.063611	0.36868	0.0049371
86	0.000048682	1441	0.061988	0.36868	0.0049371
87	0.000040568	1453	0.061339	0.36925	0.0049401
88	0.000027045	1487	0.059959	0.37006	0.0049443
89	0.000023182	1499	0.059635	0.37014	0.0049447
90	0.000016227	1506	0.059473	0.37022	0.0049451
91	0.000010000	1511	0.059391	0.37022	0.0049451

Table 25 *CP Table*

We have a new pruned tree model with leaves of 542, much better than last time and no overfitting. Let us see the confusion matrix. Train set has an accuracy of .8911 while valid set .8295. Still, as we are a multi classifier model, we cannot produce ROC Curve. Later, we will use Weighted F1 and Macro F1 to compare the models according to reference 3.

Confusion Matrix and Statistics

Prediction	Reference		
	high	low	median
high	5306	2	503
low	6	5422	574
median	848	741	11142

Overall Statistics

Accuracy : 0.8911
 95% CI : (0.8871, 0.8949)
 No Information Rate : 0.4978
 P-Value [Acc > NIR] : < 0.00000000000000022

Kappa : 0.8245

McNemar's Test P-Value : < 0.00000000000000022

Statistics by Class:

	Class: high	Class: low	Class: median
Sensitivity	0.8614	0.8795	0.9119
Specificity	0.9725	0.9684	0.8711
Pos Pred Value	0.9131	0.9034	0.8752
Neg Pred Value	0.9544	0.9599	0.9088
Prevalence	0.2510	0.2512	0.4978
Detection Rate	0.2162	0.2209	0.4540
Detection Prevalence	0.2368	0.2445	0.5187
Balanced Accuracy	0.9169	0.9240	0.8915

Table 26 *Confusion Matrix of Pruned Tree with Train Set*

Confusion Matrix and Statistics

Prediction	Reference		
	high	low	median
high	2045	0	376
low	0	2117	368
median	565	485	4564

Overall Statistics

Accuracy : 0.8295
 95% CI : (0.8221, 0.8366)
 No Information Rate : 0.5046
 P-Value [Acc > NIR] : < 0.00000000000000022

Kappa : 0.7228

McNemar's Test P-Value : NA

Statistics by Class:

	Class: high	Class: low	Class: median
Sensitivity	0.7835	0.8136	0.8598
Specificity	0.9525	0.9535	0.7985
Pos Pred Value	0.8447	0.8519	0.8130
Neg Pred Value	0.9302	0.9396	0.8483
Prevalence	0.2481	0.2473	0.5046
Detection Rate	0.1944	0.2012	0.4338
Detection Prevalence	0.2301	0.2362	0.5337
Balanced Accuracy	0.8680	0.8836	0.8292

Table 27 *Confusion Matrix of Pruned Tree with Valid Set*

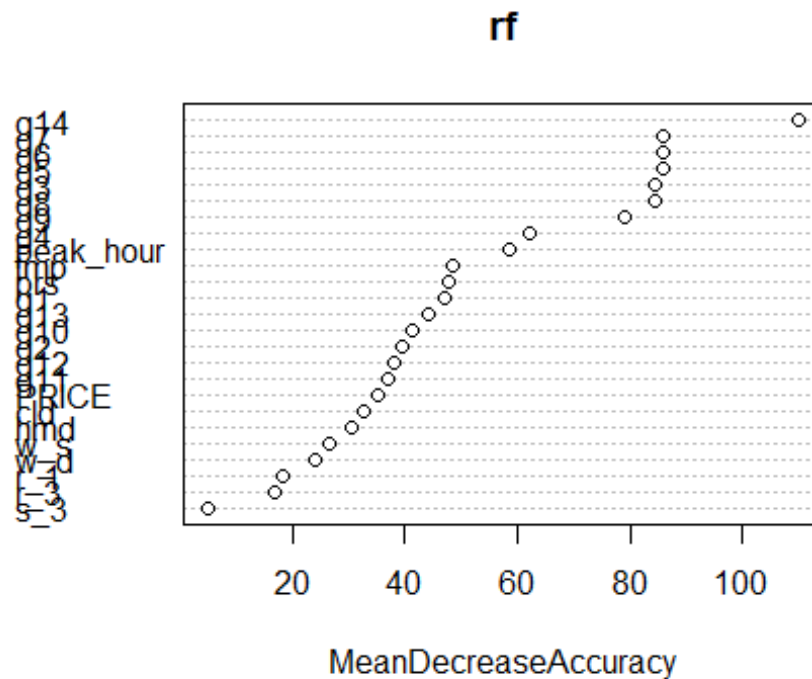
Class	Recall	Precision	F1
-------	--------	-----------	----

-1	0.814	0.852	0.832
0	0.860	0.813	0.836
1	0.784	0.845	0.813

Macro F1	0.8270
MacroPrecision	0.8365
Macro Recall	0.8190
Weighted F1	0.8292
Weighted Precision	0.8305
Weighted Recall	0.8295

Table 28 Multi Classifier F1 - Pruned Classification Tree

The second model of Tree model is Random Forest. We have ntree equal 300, mtry 8 and nodesize 5 to generate the best random forest. Also, we attached a plot showing the relative importance of potential variables. Generally, energy generation variables are more important than weather features.



Plot 13 Relative Importance of Potential Variables

The confusion matrix indicates that random forest has an accuracy of .892, seemingly better than classification tree already.

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction high  low median
##      high    2260    0    207
##      low      0 2279    257
##      median  350  323  4844
##
## Overall Statistics
##
##           Accuracy : 0.892
##           95% CI : (0.886, 0.898)
##      No Information Rate : 0.505
##      P-Value [Acc > NIR] : <0.0000000000000002
##
##           Kappa : 0.825
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##           Class: high Class: low Class: median
## Sensitivity          0.866      0.876      0.913
## Specificity          0.974      0.968      0.871
## Pos Pred Value       0.916      0.899      0.878
## Neg Pred Value       0.957      0.960      0.907
## Prevalence           0.248      0.247      0.505
## Detection Rate       0.215      0.217      0.460
## Detection Prevalence 0.235      0.241      0.524
## Balanced Accuracy     0.920      0.922      0.892

```

Table 29 *Confusion Matrix of Random Forest*

The last model is Boosted Tree. Confusion Matrix is as below. It has an accuracy of .8243, performing relatively bad in the three tree models.

Confusion Matrix and Statistics

Prediction	Reference		
	high	low	median
high	1929	0	304
low	0	2089	350
median	681	513	4654

Overall Statistics

Accuracy : 0.8243
 95% CI : (0.8169, 0.8316)
 No Information Rate : 0.5046
 P-Value [Acc > NIR] : < 0.00000000000000022

Kappa : 0.7118

Mcnemar's Test P-Value : NA

Statistics by Class:

	Class: high	Class: low	Class: median
Sensitivity	0.7391	0.8028	0.8768
Specificity	0.9616	0.9558	0.7709
Pos Pred Value	0.8639	0.8565	0.7958
Neg Pred Value	0.9178	0.9365	0.8600
Prevalence	0.2481	0.2473	0.5046
Detection Rate	0.1834	0.1986	0.4424
Detection Prevalence	0.2123	0.2318	0.5559
Balanced Accuracy	0.8503	0.8793	0.8239

Table 30 *Confusion Matrix of Boosted Tree*

PART V Comparison of Classification Models

We are here to compare Logistic Regression, K-NN (k=13) and Pruned Classification Tree. As is stated above, our project is a multi-classifier work, so ROC Curve will not be applicable. We therefore introduce Weighted F1 and Macro F1 to realize the comparison among three typical classification models. Please refer to [multi classifier.xlsx] for specific calculations.

To be frank, a higher F1-score does not necessarily mean a better classifier, since *in the multi-class case, different prediction errors have different implication. Predicting X as Y is likely to have a different cost than predicting Z as W, as so on.* The standard F1-scores do not take any of the domain knowledge into account. Modifications to the formula of F1, Precision and Recall are necessary. Below is a chart to show the relevant indices.

	Logistic Regression	K-NN	Classification Tree
Accuracy	.8548	.872	.8295
Macro F1	.8539	.8712	.8270
Weighted F1	.8549	.8716	.8292

Table 31 Comparison of Three Models

It is very clear K-NN win the game. It not only has the highest accuracy but the highest F1 indices as well.