Midterm Project Report

Team 4

Wenjin Cao

Jianxing Lu

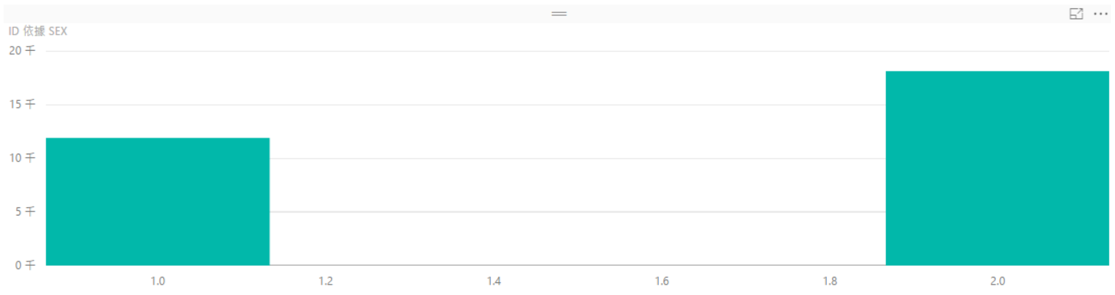
Qiaomin Ling

Problem1:

1. Data Observations using Power BI

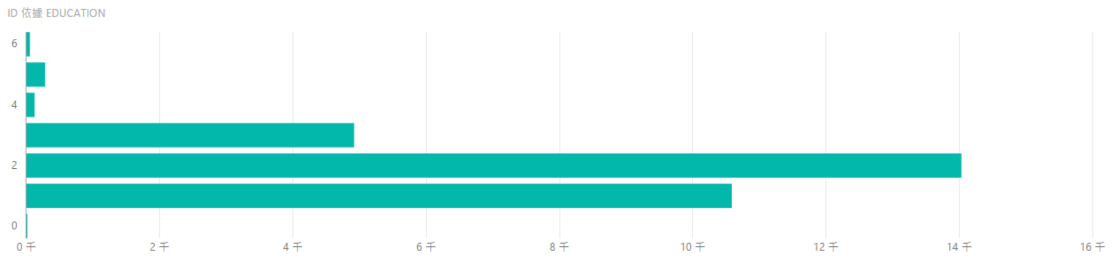
We use Power BI to get the observation of raw data, and the results as follows.

For gender:



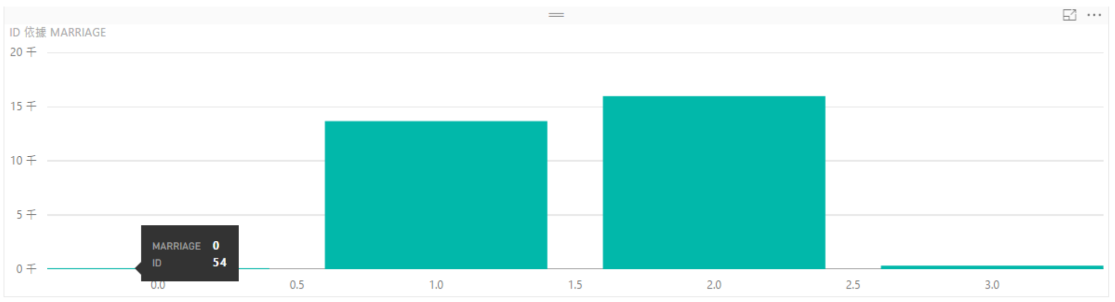
In the data set, 1 means male, 2 means female, there is no other value, which means no abnormal data in this column.

For education:



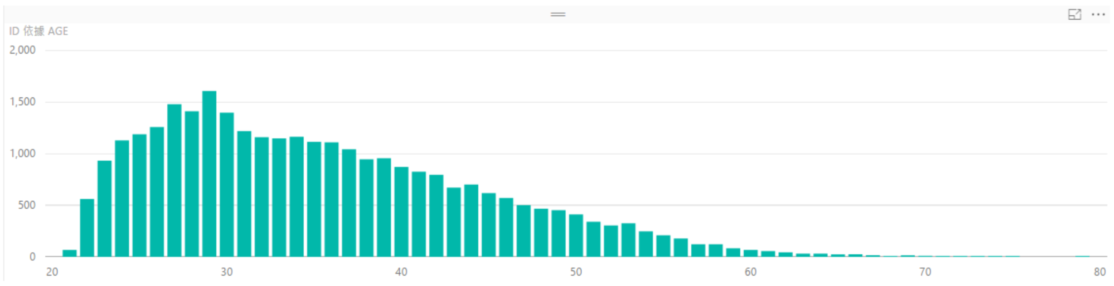
The range of education is 1 to 4, but we notice that there are lots of records is 5 and 6, we need to modify them in data cleansing and merging.

For marriage:



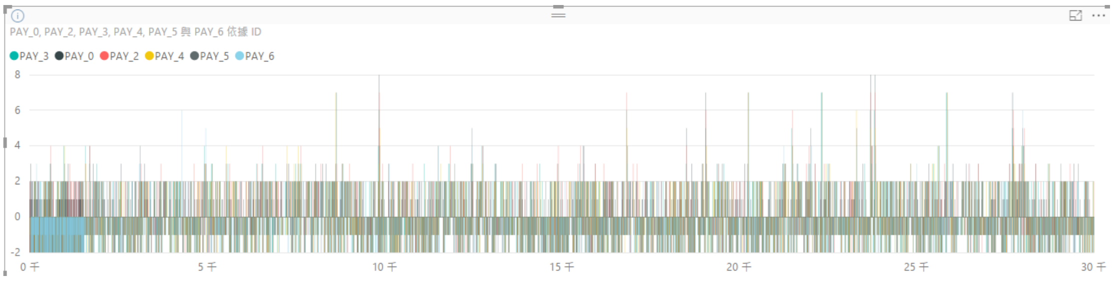
There are 54 records provide the 0 of marriage, they can be treated as abnormal data.

For age:



This graph is pretty normal.

For history of payment:



There are many data values are negative, we think these might mean the users pay their bill before the requirement, we should keep them as normal records;

2. Data Clean and Pre-process

We change the .xls file to the .csv file, then we read the file, change the column name, delete the useless raw.

Save it into p1-ready-data.csv, for the following model training and evaluation.

3. Classification models

We split the data into training dataset (75% of the data) for building the models and testing dataset (25% of the data) for validating the models.

1) Logistic regression

```
lg = glm(Y~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10 + X11 + X12 + X13 + X14 + X15 + X16 + X17 + X18 + X19 + X20 + X21 + X22 + X23,family=binomial(link='logit'),lgTraining)
summary(lg)
```

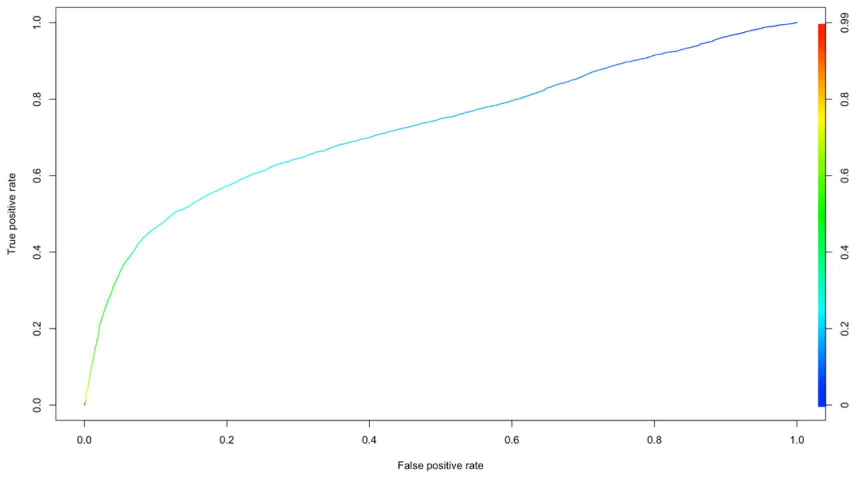
```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1623  -0.6995  -0.5466  -0.2853   3.3497

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.284e-01  1.376e-01  -4.567 4.95e-06 ***
X1          -6.915e-07  1.813e-07  -3.813 0.000137 ***
X2          -1.195e-01  3.540e-02  -3.376 0.000737 ***
X3          -8.975e-02  2.429e-02  -3.696 0.000219 ***
X4          -1.741e-01  3.680e-02  -4.731 2.23e-06 ***
X5           6.568e-03  2.063e-03   3.184 0.001454 **
X6           5.779e-01  2.041e-02  28.308  < 2e-16 ***
X7           1.095e-01  2.335e-02   4.688 2.76e-06 ***
X8           6.436e-02  2.631e-02   2.446 0.014455 *
X9          -8.089e-03  2.926e-02  -0.276 0.782183
X10          4.851e-02  3.126e-02   1.552 0.120653
X11         -6.148e-04  2.550e-02  -0.024 0.980762
X12         -5.378e-06  1.279e-06  -4.205 2.61e-05 ***
X13          2.421e-06  1.736e-06   1.394 0.163242
X14          4.447e-07  1.596e-06   0.279 0.780485
X15          1.280e-07  1.586e-06   0.081 0.935645
X16          1.776e-07  1.766e-06   0.101 0.919922
X17          1.477e-06  1.397e-06   1.057 0.290458
X18         -1.475e-05  2.815e-06  -5.239 1.61e-07 ***
X19         -8.447e-06  2.423e-06  -3.486 0.000491 ***
X20         -6.746e-06  2.358e-06  -2.861 0.004225 **
X21         -5.004e-06  2.156e-06  -2.321 0.020281 *
X22         -4.580e-06  2.153e-06  -2.127 0.033415 *
X23         -1.340e-06  1.480e-06  -0.905 0.365251
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 23779  on 22499  degrees of freedom
Residual deviance: 20885  on 22476  degrees of freedom
AIC: 20933

Number of Fisher Scoring iterations: 6
```

After running the glm() function, we got a warning message: "glm.fit: fitted probabilities numerically 0 or 1 occurr." The reason is perfect separation in logistic regression. A solution to this is to utilize a form of penalized regression.

```
lg2 = glmnet(model.matrix(Y~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10 + X11
+ X12 + X13 + X14 + X15 + X16 + X17 + X18 + X19 + X20 + X21 + X22 +
X23,model.frame(ff,mydata)),Y)
summary(lg2)
```

```
          Length Class     Mode
a0          80   -none-    numeric
beta      1920   dgCMatrix S4
df          80   -none-    numeric
dim          2   -none-    numeric
lambda      80   -none-    numeric
dev.ratio   80   -none-    numeric
nulldev      1   -none-    numeric
npasses      1   -none-    numeric
jerr         1   -none-    numeric
offset       1   -none-    logical
call         3   -none-    call
nobs         1   -none-    numeric
```

Plot ROC curve:

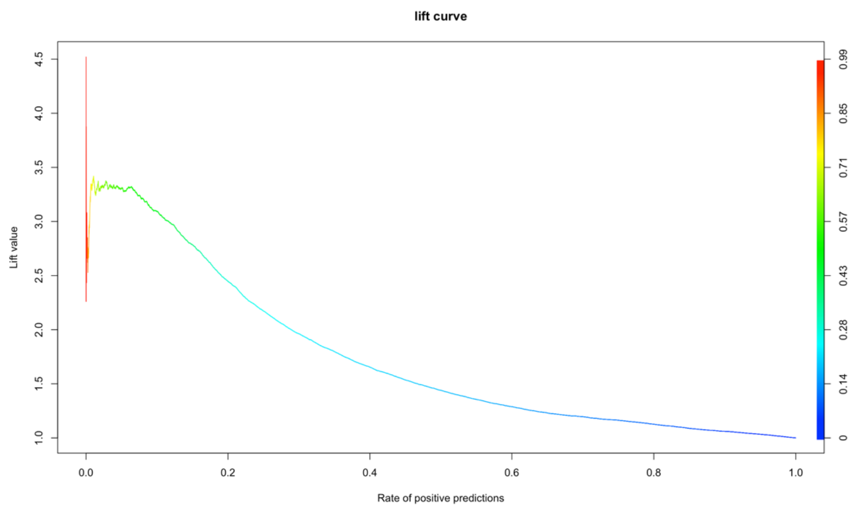


The receiver operating characteristic curve (aka relative operating characteristic curve) is created by plotting the true positive rate (TPR, aka sensitivity or recall) against the false positive rate (FPR, aka the fall-out, equals to 1 - sensitivity) at various threshold settings. The TPR defines how many correct positive results occur among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results occur among all negative samples available during the test.

Each prediction result or instance of a confusion matrix represents one point in the ROC space.

The best possible prediction method under a perfect classification would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). The diagonal line divides the ROC space. Points above the diagonal represent good classification results (better than random), points below the line poor results (worse than random). The bigger the area under the curve(AUC), the better the performance of a binary classifier system. Under the best circumstance, the AUC is approximate to 1. Thus, generally speaking, the curve indicates that our model is good but far from perfect.

Plot Lift charts:



Lift charts also measure the performance of the model against random guessing, indicative of the models ability to accurately predict within a training population. Higher lift value, or steeper curve, shows better performance.

Confusion Metrics:

This is what a general confusion metric is:

	1	0	
1	d, True Positive	c, False Negative	c+d, Actual Positive
0	b, False Positive	a, True Negative	a+b, Actual Negative
	b+d, Predicted Positive	a+c, Predicted Negative	

For this model:

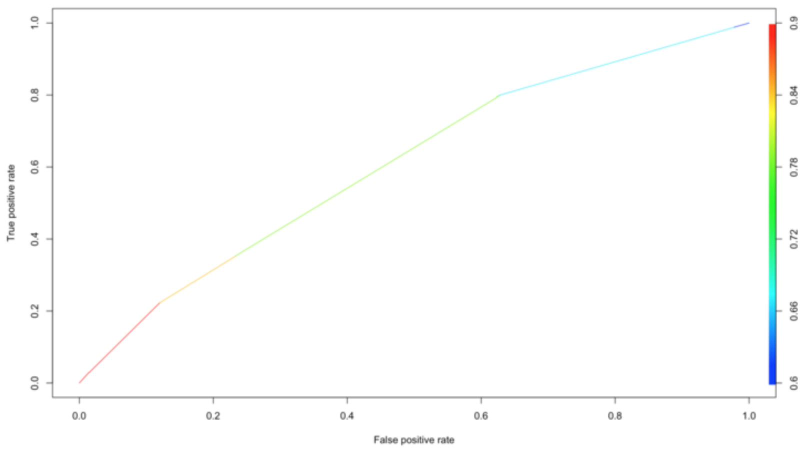
Logisctic Regression		
Overall Error	20.87%	
	Credible	Non-credible
Credible	5788	53
Non-credible	1512	147

2) Neural network

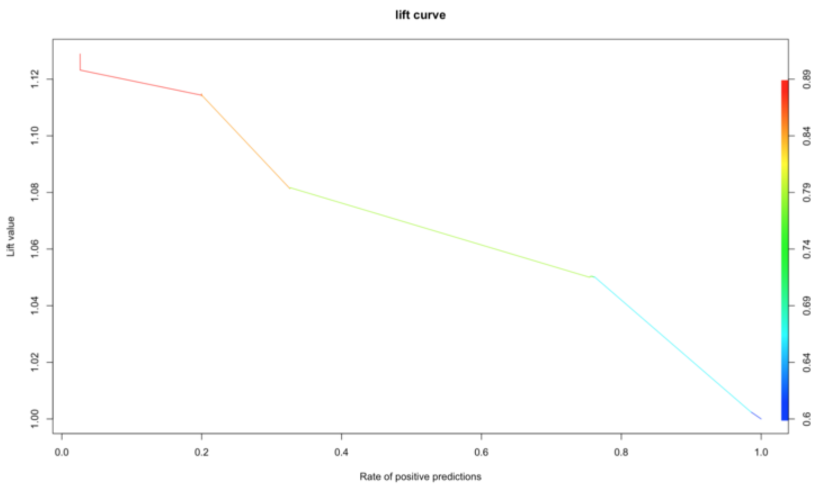
```
trainY_class = ifelse(train$Y>0,"Credible","Non-credible")
trainY_class = as.factor(trainY_class)
nn = nnet(trainY_class~X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 + X9 + X10 + X11 + X12
+ X13 + X14 + X15 + X16 + X17 + X18 + X19 + X20 + X21 + X22 + X23,train,size=4)
summary(nn)
```

```
a 23-4-1 network with 101 weights
options were - entropy fitting
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1 i10->h1 i11->h1 i12->h1 i13->h1 
  -0.32   0.30   0.43   0.68  -0.50   0.00  -0.57   0.61  -0.28  -0.41   0.36   0.68   0.17   0.72 
i14->h1 i15->h1 i16->h1 i17->h1 i18->h1 i19->h1 i20->h1 i21->h1 i22->h1 i23->h1 
  -0.44   0.62  -0.09   0.05  -0.61   0.62   0.36  -0.34  -0.05  -0.24 
  b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i9->h2 i10->h2 i11->h2 i12->h2 i13->h2 
  -0.36  -0.61   0.41   0.23  -0.02   0.39  -0.13   0.59  -0.65   0.59   0.69  -0.64   0.12   0.10 
i14->h2 i15->h2 i16->h2 i17->h2 i18->h2 i19->h2 i20->h2 i21->h2 i22->h2 i23->h2 
   0.25  -0.04   0.48   0.21   0.96   0.46  -0.52   0.46   0.07   0.60 
  b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3  i9->h3 i10->h3 i11->h3 i12->h3 i13->h3 
  -0.57  -0.01   0.11   0.55  -0.38   0.42  -0.10  -0.55  -0.58  -0.13   0.23   0.20   0.63  -0.80 
i14->h3 i15->h3 i16->h3 i17->h3 i18->h3 i19->h3 i20->h3 i21->h3 i22->h3 i23->h3 
   0.10  -0.64   0.53  -0.31   0.54   0.58  -0.34  -0.42   0.35   0.35 
  b->h4  i1->h4  i2->h4  i3->h4  i4->h4  i5->h4  i6->h4  i7->h4  i8->h4  i9->h4 i10->h4 i11->h4 i12->h4 i13->h4 
   0.08   0.21   0.03  -0.13  -0.33  -0.11  -0.05   0.08  -0.57   0.54   0.65  -0.21   0.61  -0.25 
i14->h4 i15->h4 i16->h4 i17->h4 i18->h4 i19->h4 i20->h4 i21->h4 i22->h4 i23->h4 
   0.04  -0.81   0.34  -0.42   0.62   0.79   0.75   0.19  -0.21  -0.34 
 b->o h1->o h2->o h3->o h4->o 
-0.04  0.42  0.31  0.48  0.94 
```

Plot ROC curve:



Plot Lift chart:



The above two curves both indicate a poor performance of the model.

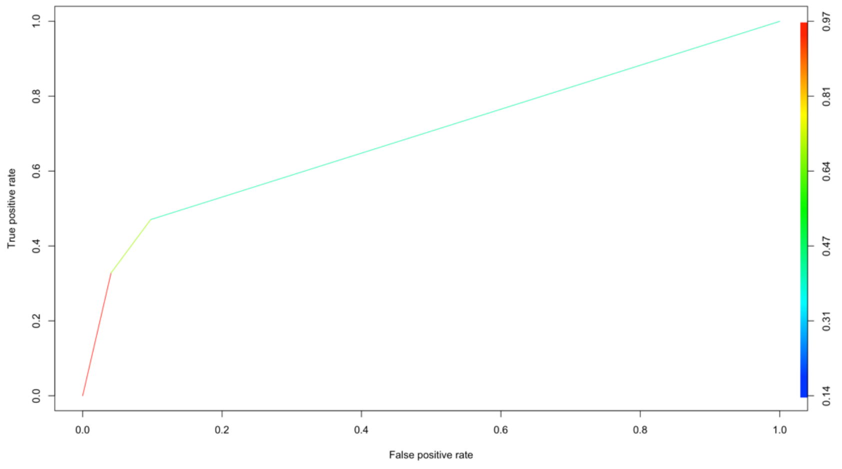
Confusion metrics:

Neural network		
Overall Error	22.27%	
	Credible	Non-credible
Credible	0	0
Non-credible	1670	5830

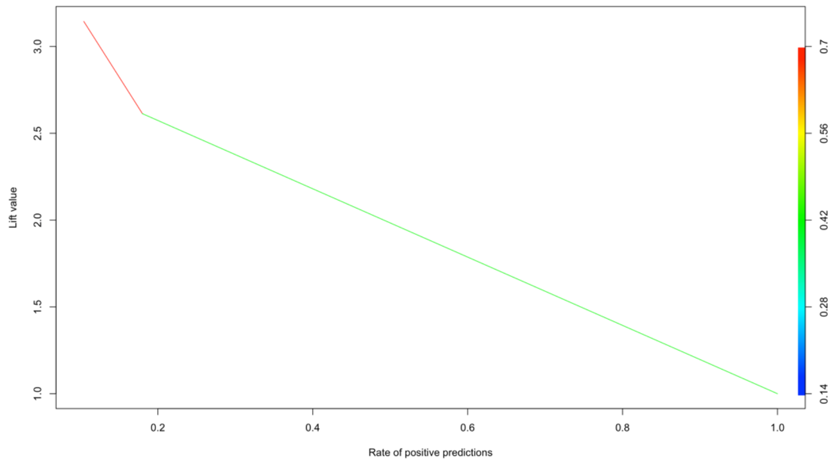
⊛Classification tree

```
tree(formula = Y_class ~ X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8 +
    X9 + X10 + X11 + X12 + X13 + X14 + X15 + X16 + X17 + X18 +
    X19 + X20 + X21 + X22 + X23, data = mydata, subset = tree.train)
Variables actually used in tree construction:
[1] "X6"  "X7"  "X20"
Number of terminal nodes:  4
Residual mean deviance:  0.8922 = 20070 / 22500
Misclassification error rate: 0.1801 = 4052 / 22500
```

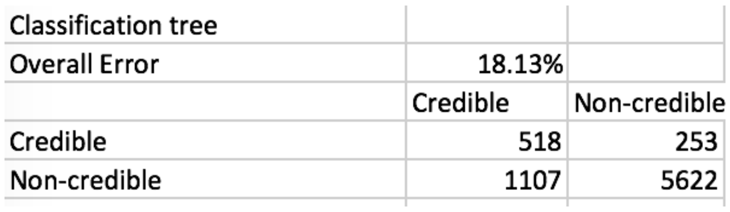
Plot ROC curve:



Plot Lift charts:



Confusion metrics:



Classification tree		
Overall Error	18.13%	
	Credible	Non-credible
Credible	518	253
Non-credible	1107	5622

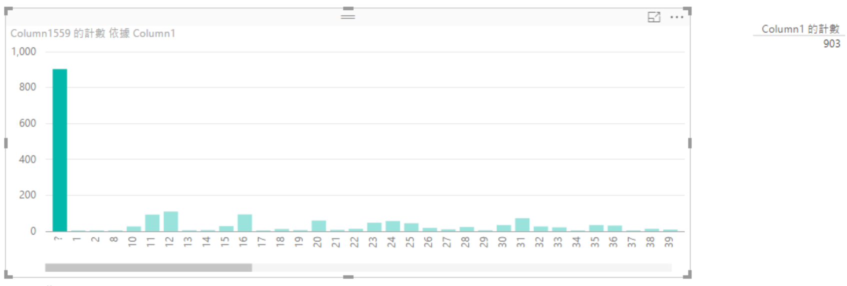
4. Discussion

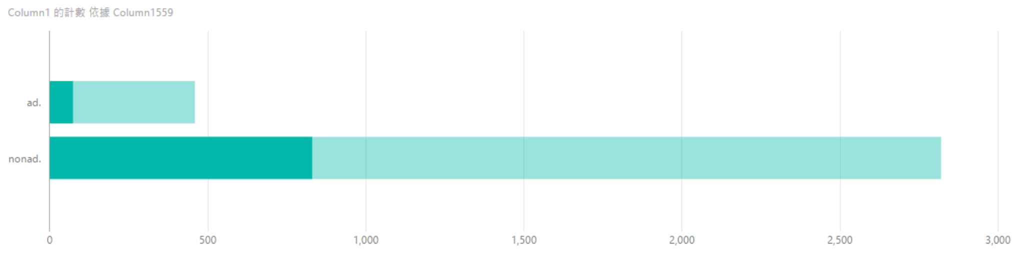
According to overall error rates as well as ROC curves and Lift charts of the three models, the classification tree model has the best performance, and the neural net model the worst.

Problem2:

1. Data Observations using Power BI

We use Power BI to get the observation of raw data, and the results as follows.
This dataset is used to predict whether a website is advertisement or not, and uses ad, or nonad, to represent the result. First, we utilize Power BI to make observation of dataset. The graph is shown below.





In this graph, there are 903 records have no height, weight and aratio represented by "?", we need to modify these values.

2. Data Clean and Pre-process

First, read the file, and change the first three column value to numbers, and change the last column which contains "ad.", "nonad." We change it to binary 0 and 1 to train model. (1 represents "ad." ; 0 represents "nonad.")

It generates some NA when we change the three columns into numbers. So we need to find a way to deal with them.

First way: we fill the missing data with the average of the all column in first three columns. Fill the 4th column with the mode of the column.

But it is not a very good way to fill the missing data. We came up another way: we can build a model based on the correct data, and then make prediction. Filling the missing data with the prediction.

In order to do this, we first delete the rows, which contains the missing data.

Then we observed the data, and we found that there are some column is all 0, which is useless in building models. So we find all these columns and delete them.

Then we use this data build model. Make prediction to predict the missing data in the first three columns.
Make prediction to predict the missing data in the first three columns.

```
library(caTools)

PreModelV1 = DeleteDataDeleteColumn[,-2:-3]
attach(PreModelV1)
lmV1 = lm(V1~.,PreModelV1)
PreForecaseV1 = AverageData[,-2:-3]
ForecastV1 = round(predict(lmV1,PreForecaseV1),2)
remove(PreModelV1,PreForecaseV1)

PreModelV2 = DeleteDataDeleteColumn[,-1]
PreModelV2 = PreModelV2[,-2]
attach(PreModelV2)
lmV2 = lm(V2~.,PreModelV2)
PreForecaseV2 = AverageData[,-1]
PreForecaseV2 = PreForecaseV2[,-2]
ForecastV2 = round(predict(lmV2,PreForecaseV2),2)
remove(PreModelV2,PreForecaseV2)

PreModelV3 = DeleteDataDeleteColumn[,-1:-2]
attach(PreModelV3)
lmV3 = lm(V3~.,PreModelV3)
PreForecaseV3 = AverageData[,-1:-2]
ForecastV3 = round(predict(lmV3,PreForecaseV3),4)
remove(PreModelV3,PreForecaseV3)
```

Fill the predicted data into the missing data in right place

```
Missing = read.csv("p2-data-MissingData.csv",sep = ",")
Missing1 = as.integer(na.exclude(Missing[,2]))
Missing2 = as.integer(na.exclude(Missing[,3]))
Missing3 = as.integer(na.exclude(Missing[,4]))
remove(Missing)

FillData = AverageData
for(i in Missing1){
  FillData[i,1] = ForecastV1[i]
}
for(i in Missing2){
  FillData[i,2] = ForecastV2[i]
}
for(i in Missing3){
  FillData[i,3] = ForecastV3[i]
}
```

Save it into Save it into p2-data-FillData.csv, for the following model training and evaluation.

3. Classification models

We split the data into training dataset for building the models and testing dataset for validating the models.

①Logistic regression

lg = glm(V1559~.,family=binomial(link='logit'),lgTraining)
summary(lg)

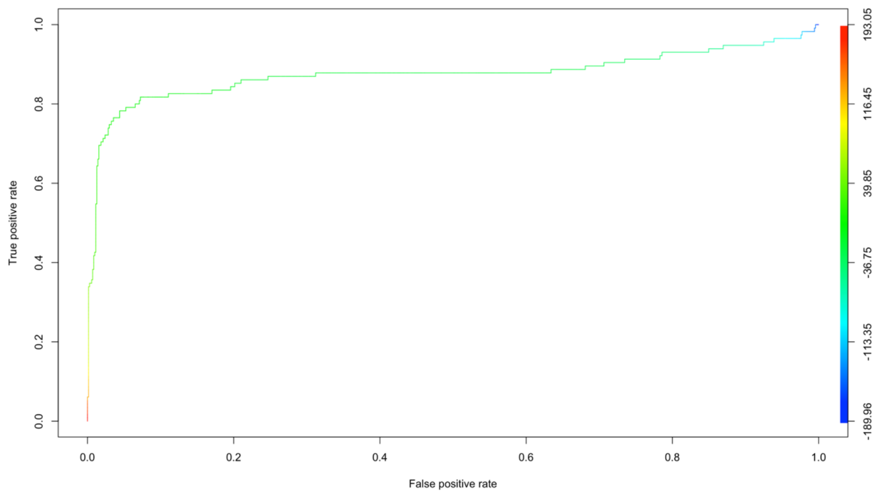
```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1591.746  on 1966  degrees of freedom
Residual deviance:   33.414  on 1298  degrees of freedom
AIC: 1371.4

Number of Fisher Scoring iterations: 24
```

After running the glm() function, we got a warning message: “glm.fit: fitted probabilities numerically 0 or 1 occurr.” The reason is perfect separation in logistic regression. A solution to this is to utilize a form of penalized regression.

ff = V1559~.
mat = model.matrix(ff,model.frame(ff,mydata))
lg2 = glmnet(model.matrix(ff,model.frame(ff,mydata)),V1559)
summary(lg2)

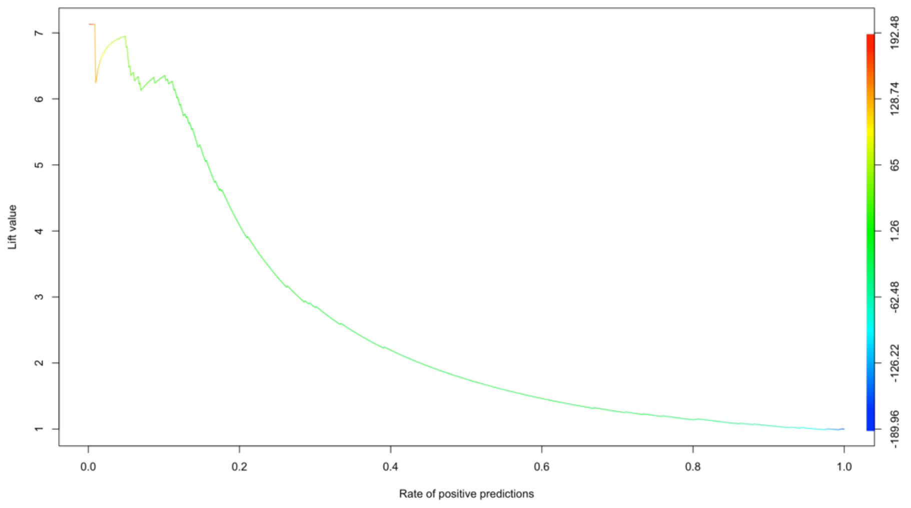
```
           Length Class     Mode
a0            100 -none-    numeric
beta       155900 dgCMatrix S4
df            100 -none-    numeric
dim             2 -none-    numeric
lambda        100 -none-    numeric
dev.ratio     100 -none-    numeric
nulldev         1 -none-    numeric
npasses         1 -none-    numeric
jerr            1 -none-    numeric
offset          1 -none-    logical
call            3 -none-    call
nobs            1 -none-    numeric
```

Plot ROC curve:



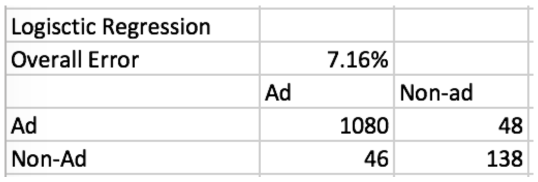
The diagonal line divides the ROC space. Points above the diagonal represent good classification results (better than random), points below the line poor results (worse than random). The bigger the area under the curve(AUC), the better the performance of a binary classifier system. Under the best circumstance, the AUC is approximate to 1. Thus, generally speaking, the curve indicates that this model is pretty.

Plot Lift charts:



Lift charts also measure the performance of the model against random guessing, indicative of the models ability to accurately predict within a training population. Higher lift value, or steeper curve, shows better performance.

Confusion Metrics:



Logisctic Regression		
Overall Error	7.16%	
	Ad	Non-ad
Ad	1080	48
Non-Ad	46	138

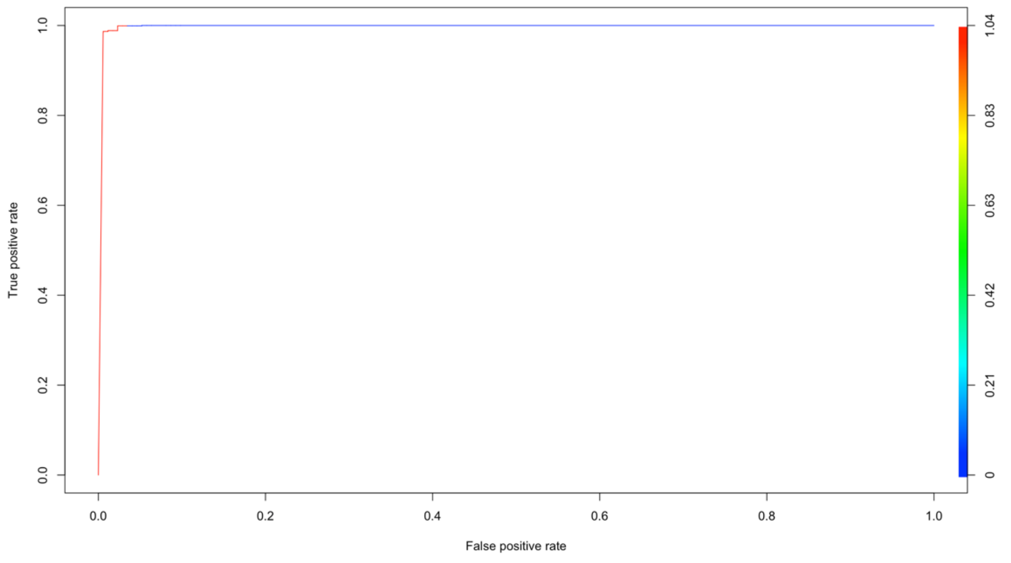
②Neural network

```
nn = nnet(trainY_class~.,train,size=1,MaxNWts=1800)
summary(nn)
```

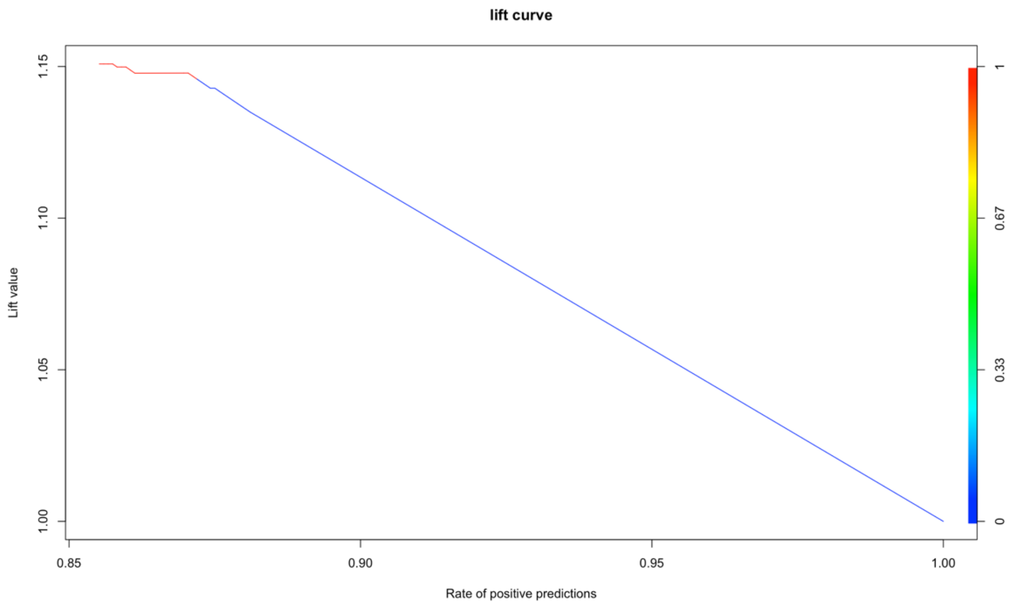
There are quite a lot of weights in the network due to too many variables.

```
a 1559-1-1 network with 1562 weights
options were - entropy fitting
    b->h1    i1->h1    i2->h1    i3->h1    i4->h1    i5->h1    i6->h1    i7->h1    i8->h1    i9->h1   i10->h1   i11->h1 
    -2.04      0.41      3.97     -1.68     -0.65      0.41      0.28     -0.38     -0.09      0.05     -0.18     -0.30 
  i12->h1   i13->h1   i14->h1   i15->h1   i16->h1   i17->h1   i18->h1   i19->h1   i20->h1   i21->h1   i22->h1   i23->h1 
     0.10     -0.24     -0.02     -0.01      0.42      0.48     -0.34      0.13     -0.55     -0.47     -0.22      0.08 
  i24->h1   i25->h1   i26->h1   i27->h1   i28->h1   i29->h1   i30->h1   i31->h1   i32->h1   i33->h1   i34->h1   i35->h1 
    -0.03      0.30     -0.36     -0.70      0.56     -0.54     -0.66      0.16      0.27      0.37      0.11     -0.49 
  i36->h1   i37->h1   i38->h1   i39->h1   i40->h1   i41->h1   i42->h1   i43->h1   i44->h1   i45->h1   i46->h1   i47->h1 
    -0.54     -0.69     -0.37      0.18      0.34      0.45     -0.66     -0.38      0.24      0.61     -0.08      0.70 
  i48->h1   i49->h1   i50->h1   i51->h1   i52->h1   i53->h1   i54->h1   i55->h1   i56->h1   i57->h1   i58->h1   i59->h1 
     0.00     -0.53     -0.44     -0.14      0.64      0.46     -0.34     -0.08      0.48      0.22      0.29      0.00 
  i60->h1   i61->h1   i62->h1   i63->h1   i64->h1   i65->h1   i66->h1   i67->h1   i68->h1   i69->h1   i70->h1   i71->h1 
    -0.43      0.51      0.09      0.30     -0.61     -0.55      0.55      0.68      0.37      0.25      0.03      0.55 
  i72->h1   i73->h1   i74->h1   i75->h1   i76->h1   i77->h1   i78->h1   i79->h1   i80->h1   i81->h1   i82->h1   i83->h1 
     0.54     -0.58      0.17     -0.54      0.24     -0.22      0.33      0.46     -0.27     -0.20     -0.23      0.49 
  i84->h1   i85->h1   i86->h1   i87->h1   i88->h1   i89->h1   i90->h1   i91->h1   i92->h1   i93->h1   i94->h1   i95->h1 
    -0.45      0.70      0.70      0.48     -0.29      0.00     -0.30     -0.21      0.54     -0.47      0.61      0.47 
  i96->h1   i97->h1   i98->h1   i99->h1  i100->h1  i101->h1  i102->h1  i103->h1  i104->h1  i105->h1  i106->h1  i107->h1 
     0.29     -0.25      0.66      0.18     -0.31      0.46      0.42      0.04     -0.05     -0.45      0.03     -0.68 
 i108->h1  i109->h1  i110->h1  i111->h1  i112->h1  i113->h1  i114->h1  i115->h1  i116->h1  i117->h1  i118->h1  i119->h1 
    -0.26      0.04      0.35      0.14      0.11     -0.35     -0.46      0.16     -0.01      0.09     -0.38      0.27 
 i120->h1  i121->h1  i122->h1  i123->h1  i124->h1  i125->h1  i126->h1  i127->h1  i128->h1  i129->h1  i130->h1  i131->h1 
    -0.10     -0.32     -0.27     -0.52     -0.16     -0.63     -0.13      0.21     -0.37     -0.62     -0.06     -0.18 
```

Plot ROC curve:



Plot Lift chart:



The above two curves both indicate a bad performance of the model.

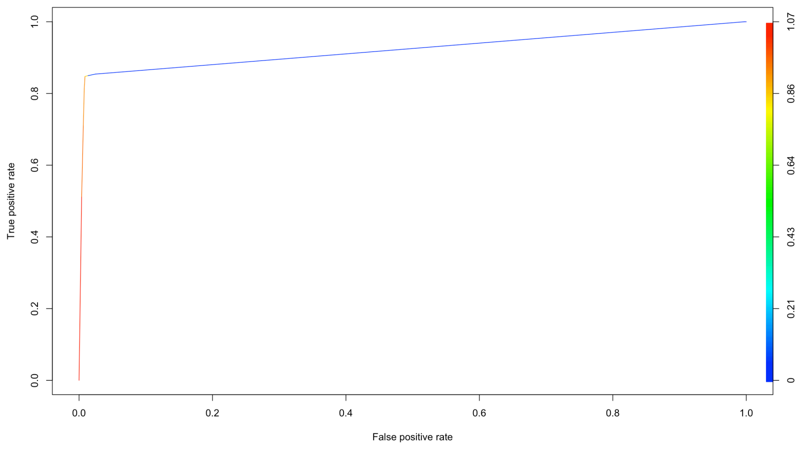
Confusion metrics:

Neural network		
Overall Error	0.46%	
	Ad	Non-Ad
Ad	168	1
Non-Ad	5	1138

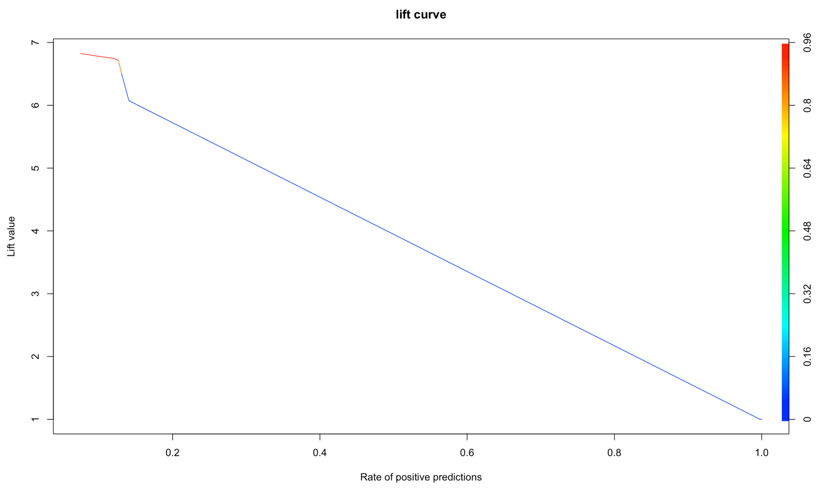
③Classification tree

```
Classification tree:
tree(formula = V1559_class ~ ., data = mydata, subset = tree.train)
Variables actually used in tree construction:
[1] "V1559"
Number of terminal nodes:  2
Residual mean deviance:  0 = 0 / 1998
Misclassification error rate: 0 = 0 / 2000
```

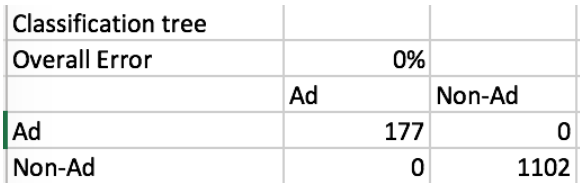
Plot ROC curve:



Plot Lift charts:



Confusion metrics:



Classification tree		
Overall Error	0%	
	Ad	Non-Ad
Ad	177	0
Non-Ad	0	1102

4. Discussion

According to overall error rates as well as ROC curves and Lift charts of the three models, the classification tree model has the best performance. The neural net model also has a pretty good performance.

Problem3:

1. Data Observations using Power BI

We use Power BI to get the observation of raw data, and the results as follows.
For this problem, we have 7 different datasets are used for train dataset. We need to use Power BI to check whether there is abnormal record in 7 datasets. We add a new index column in every dataset which can help build the observation graph.

2. Data Clean and Pre-process

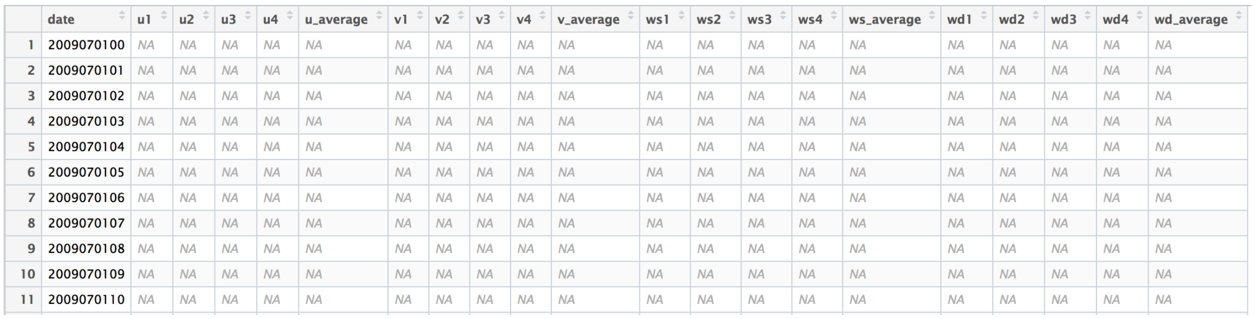
After reading the assignment requirement and observation the raw data, we are supposed to train model and do the evaluation from 2009070100 to 2012280911. So we first make a list, which contains hourly times from 2009070100 to 2012280911.

Then we notice that there are some data is missing in train.csv, we need to fine them. There are 7440 data missing in the train.csv. For example: from Jan/1/2011 1:00 to Jan/3/2011 0:00. We save it to “wf.MissingWP.csv” for reference

After analysis and observation the raw data, we notice that every 12 hours, there is 48 observations begin from that moment. Which means, most times, there are 4 observations, except the beginning and the last. We decided to use the average of these 4 observations to do the model training and evaluation.

We create a new data frame, which contains the following columns:

1:	“data” column, to show the hourly times;
2-5:	“u1”, “u2”, “u3”, “u4” record 4 observations for “u” in every hour;
6:	“u_average” to calculate the average of the u;
7-10:	“v1”, “v2”, “v3”, “v4” record 4 observations for “v” in every hour;
11:	“v_average” to calculate the average of the v
12-15:	“ws1”, “ws2”, “ws3”, “ws4” record observations for “ws” in every hour;
16:	“ws_average” to calculate the average of the ws
17-20:	“wd1”, “wd2”, “wd3”, “wd4” record observations for “wd” in every hour;
21:	“wd_average” to calculate the average of the wd



	date	u1	u2	u3	u4	u_average	v1	v2	v3	v4	v_average	ws1	ws2	ws3	ws4	ws_average	wd1	wd2	wd3	wd4	wd_average
1	2009070100	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	2009070101	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	2009070102	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	2009070103	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
5	2009070104	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
6	2009070105	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
7	2009070106	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
8	2009070107	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
9	2009070108	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
10	2009070109	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
11	2009070110	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

We take the data from raw data and locate them in the new data frame in the right place.

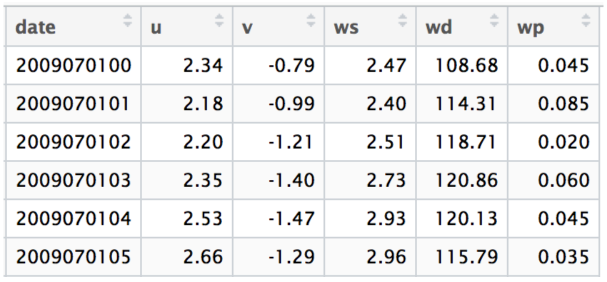
```
#u: 2:5 6
for(i in 1:(length(wf[,1])/48)){
  tempn = i%%4
  wf.new[ (1+(i-1)*12):(48+(i-1)*12), 2+tempn] = wf[(1+(i-1)*48):(48+(i-1)*48),3]
}

# v: 7-10 11
for(i in 1:(length(wf[,1])/48)){
  tempn = i%%4
  wf.new[ (1+(i-1)*12):(48+(i-1)*12), 7+tempn] = wf[(1+(i-1)*48):(48+(i-1)*48),4]
}

# ws: 12-15 16
for(i in 1:(length(wf[,1])/48)){
  tempn = i%%4
  wf.new[ (1+(i-1)*12):(48+(i-1)*12), 12+tempn] = wf[(1+(i-1)*48):(48+(i-1)*48),5]
}

# wd: 17-20 21
for(i in 1:(length(wf[,1])/48)){
  tempn = i%%4
  wf.new[ (1+(i-1)*12):(48+(i-1)*12), 17+tempn] = wf[(1+(i-1)*48):(48+(i-1)*48),6]
}
remove(tempn)
```

Then we calculate the average of the 4 observations in every hour. If there is only 1or 2 oe3 observations in that hour, we just user the average of the 1 or 2 or 3 observations.

Then we take the average data out and take the right column from train.csv, because there are some data missing in the train.csv, so we need to check if the date is matched.

```
#take the average column out
#build a new data frame to train model
u = wf.new[,6]
v = wf.new[,11]
ws = wf.new[,16]
wd = wf.new[,21]
wp = c(NA)
wf.final = cbind(date,u,v,ws,wd,wp)
remove(u,v,ws,wd,wp)

count = 1
for( i in 1: length(date) ) {
  if(count<=length(train[,1])){
    if( date[i] == train[count,1] ){
      wf.final[i,6] = train[count,WFnumber+1]
      count = count+1
    }
  }
}
remove(count)
```

After merge the average data from windforecast.csv and the data from train.csv, which will come to the right format to do the model training and evaluation.



date	u	v	ws	wd	wp
2009070100	2.34	-0.79	2.47	108.68	0.045
2009070101	2.18	-0.99	2.40	114.31	0.085
2009070102	2.20	-1.21	2.51	118.71	0.020
2009070103	2.35	-1.40	2.73	120.86	0.060
2009070104	2.53	-1.47	2.93	120.13	0.045
2009070105	2.66	-1.29	2.96	115.79	0.035

Then we separate the data into train and evaluation.

Train: 2009/7/1 to 2010/12/31

Evaluation: 2011/1/1 to 2012/6/28

There are some data is missing in the train, which is useless for the model training, so we will delete the missing data in the train. But we will keep missing ones in the evaluation.

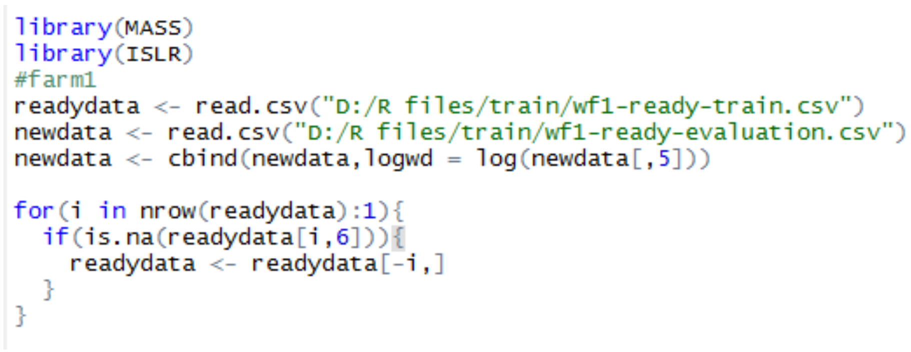
Save two files to “wf.train.csv” and “wf.evaluation.csv”.

We did all the process to all 1-7 windforecast files, and we will generate 7 wf.train.csv and 7 wf.evaluation.csv. They will be used to do the following model training and evaluation.

3. Build Model

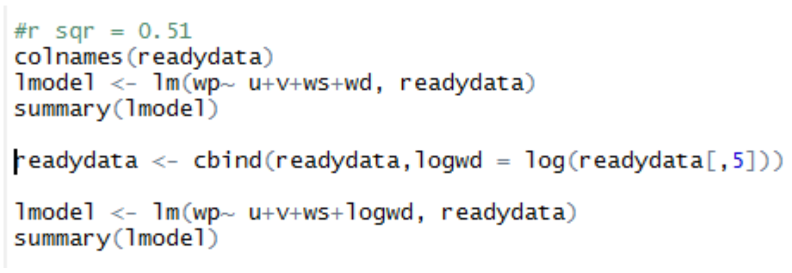
As the requirements of assignment, we need to build model by Regression, Neural Net, and Tree Regression. Different from Problem 1 & 2, we decide to use lm() method, known as Multiple Linear Regression, to get regression model. The reasons are that the predict value is not simple as 0 and 1, the only way to calculate is using MLR for regression.

To build model, we need to use two kind of data, training and evaluation, and get 7 pairs of these data for 7 farms. First, we use lm(), neuralnet() and tree() method to build model by using training data. To increase the accuracy of model, we use feature selection and feature transformation to modify original data. We notice that date column is irrelative with wind power, we delete this column by using for loop. And the values of “wd” column is pretty larger than other columns ’value, thus we use log() method to reduce them and add a new column logwd into every training data.

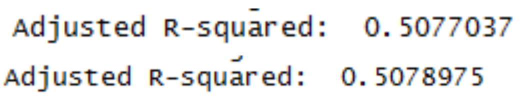


```r
library(MASS)
library(ISLR)
#farm1
readydata <- read.csv("D:/R files/train/wf1-ready-train.csv")
newdata <- read.csv("D:/R files/train/wf1-ready-evaluation.csv")
newdata <- cbind(newdata,logwd = log(newdata[,5]))

for(i in nrow(readydata):1){
  if(is.na(readydata[i,6])){
    readydata <- readydata[-i,]
  }
}
```

After feature transformation the R Square of linear model increase by 0.01 percent.



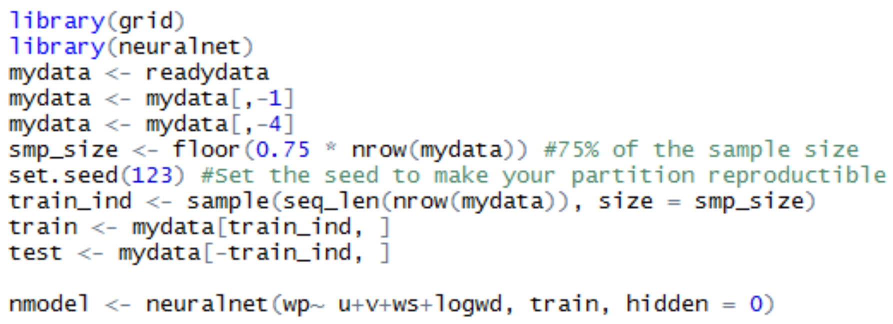
```r
#r sqr = 0.51
colnames(readydata)
lmodel <- lm(wp~ u+v+ws+wd, readydata)
summary(lmodel)

readydata <- cbind(readydata,logwd = log(readydata[,5]))

lmodel <- lm(wp~ u+v+ws+logwd, readydata)
summary(lmodel)
```



Before: `Adjusted R-squared:  0.5077037`

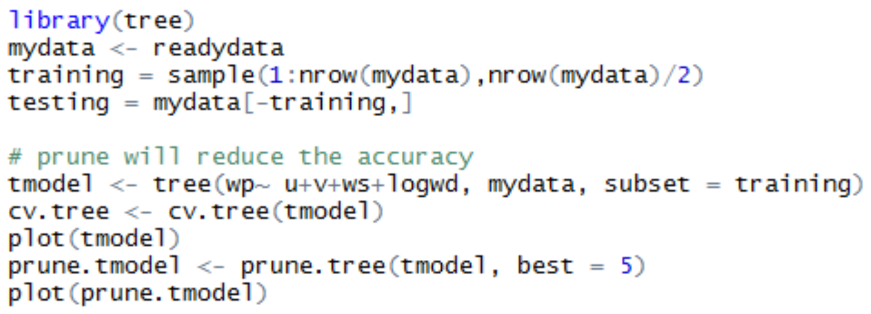
After: `Adjusted R-squared:  0.5078975`

We divide training data into two part, train and test to get performance matrixes



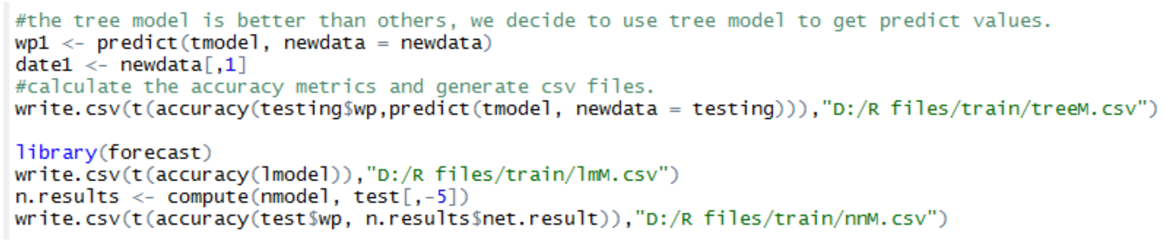
```r
library(grid)
library(neuralnet)
mydata <- readydata
mydata <- mydata[,-1]
mydata <- mydata[,-4]
smp_size <- floor(0.75 * nrow(mydata)) #75% of the sample size
set.seed(123) #Set the seed to make your partition reproductible
train_ind <- sample(seq_len(nrow(mydata)), size = smp_size)
train <- mydata[train_ind, ]
test <- mydata[-train_ind, ]

nmodel <- neuralnet(wp~ u+v+ws+logwd, train, hidden = 0)
```

We use tree method to build tree regression model and use pruning to get the best 5 points.



```r
library(tree)
mydata <- readydata
training = sample(1:nrow(mydata),nrow(mydata)/2)
testing = mydata[-training,]

# prune will reduce the accuracy
tmodel <- tree(wp~ u+v+ws+logwd, mydata, subset = training)
cv.tree <- cv.tree(tmodel)
plot(tmodel)
prune.tmodel <- prune.tree(tmodel, best = 5)
plot(prune.tmodel)
```

Using t() method and write.csv() method to generate performance matrixes.

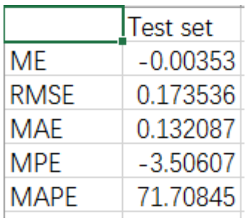


```r
#the tree model is better than others, we decide to use tree model to get predict values.
wp1 <- predict(tmodel, newdata = newdata)
date1 <- newdata[,1]
#calculate the accuracy metrics and generate csv files.
write.csv(t(accuracy(testing$wp,predict(tmodel, newdata = testing))),"D:/R files/train/treeM.csv")

library(forecast)
write.csv(t(accuracy(lmodel)),"D:/R files/train/lmM.csv")
n.results <- compute(nmodel, test[,-5])
write.csv(t(accuracy(test$wp, n.results$net.result)),"D:/R files/train/nnM.csv")
```

4. Performance Matrix Analysis

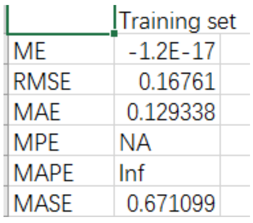
According to the performance matrixes, tree regression model's MAPE is highest, performs better than else, so we decide to use tree regression model to predict value.

Tree:



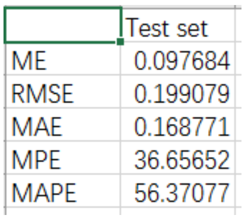
	Test set
ME	-0.00353
RMSE	0.173536
MAE	0.132087
MPE	-3.50607
MAPE	71.70845

MLM:



	Training set
ME	-1.2E-17
RMSE	0.16761
MAE	0.129338
MPE	NA
MAPE	Inf
MASE	0.671099

NN:



	Test set
ME	0.097684
RMSE	0.199079
MAE	0.168771
MPE	36.65652
MAPE	56.37077

Same operations to last 6 farms

```
#farm2
readydata <- read.csv("D:/R files/train/wf2-ready-train.csv")
newdata <- read.csv("D:/R files/train/wf2-ready-evaluation.csv")
newdata <- cbind(newdata,logwd = log(newdata[,5]))
date2 <- newdata[,1]
for(i in nrow(readydata):1){
  if(is.na(readydata[i,6])){
    readydata <- readydata[-i,]
  }
}

lmodel <- lm(wp~., data = readydata)
summary(lmodel)
#r sqr = 0.51

colnames(readydata)
lmodel <- lm(wp~ u+v+ws+wd, readydata)
#r sqr = 0.50

summary(lmodel)
readydata <- cbind(readydata,logwd = log(readydata[,5]))

lmodel <- lm(wp~ u+v+ws+logwd, readydata)
summary(lmodel)

library(grid)
library(neuralnet)
mydata <- readydata
mydata <- mydata[,-1]
mydata <- mydata[,-4]

nmodel <- neuralnet(wp~ u+v+ws+logwd, mydata, hidden = 0)

library(tree)
mydata <- readydata

# prune will reduce the accuracy
tmodel <- tree(wp~ u+v+ws+logwd, mydata)
cv.tree <- cv.tree(tmodel)
plot(tmodel)
prune.tmodel <- prune.tree(tmodel, best = 5)
plot(prune.tmodel)

#the tree model is better than others, we decide to use tree model to get predict values.
wp2 <- predict(tmodel, newdata = newdata)
```

Last, combine all the wind power result together and generate result.csv file

```
bind = c(NA)
date = date1
test2 = cbind(date1,bind)
id <- c(1:length(date))
result <- cbind(id, date, wp1,wp2,wp3,wp4,wp5,wp6,wp7)
write.csv(result, "D:/R files/train/result.csv", row.names = F)
```