Mini Project: Predictive Modeling

Model Report

GREAT LAKES INSTITUTE OF MANAGEMENT

Table of Contents

1 Project Objective and Scope

1.1 Objective

What determines the outcome of an election? Researchers, media, political parties and people at large have put forth several factors such as good governance, stable government, economic reforms, performance of the economy, anti-incumbency wave, corruption, misrule, regionalism, alliance partners, caste and religion to explain election outcomes at the national or regional level and also at the constituency level. On rare occasions, unexpected outcomes happen.

The Objective of the project is to study the data from the 2017 assembly elections from the various states and draw insights into the factors that played a significant role in determining outcome of the election.

In 2017, the assembly elections took place in following states of India:

- Utter Pradesh
- Uttarakhand
- Goa
- Manipur
- Punjab

Students are expected to analyze the 2017 Election data for any (or all) of the state and provide observations on factors affecting the election results, using various Data Mining and Prediction Techniques learnt so far.

Since it is expected to understand the existing data, validation of the Model on Holdout Sample is not expected.

1.2 Scope

This Model solution is prepared to analyze the Assembly Election Results of Utter Pradesh.

Similar approach may be applied on other States.

2 Project Approach

A typical Development Lifecycle can be adopted for this assignment, as follows:

1. Discovery
2. Data Preparation
3. Model Planning
4. Model Building
5. Communicating Results.

2.1 Discovery:

2.1.1 The Business Domain:

Being an Election domain, following factors are worth noticing for the five states in scope for 2017 Assembly Election:

Utter Pradesh:

Demonetization and the split in Samajwadi Party were two major issues in UP. Caste also was a major player as per as voting to the candidates was concerned, despite from the fact that the Supreme Court had clearly said that Caste, religion and Language cannot be cited during the elections.

The coalition of SP and Indian National Congress Party had added much needed spice in the Election Rallies during the Assembly Election.

Uttarakhand:

Uttarakhand saw a lot of political instability following a revolt against CM of the state Harish Rawat. This led to the imposition of President's Rule in the state, but the government was reinstated by the Supreme Court after two months. Will Rawat earn the sympathy vote or will issues such a corruption ruin his chances were the topics of debate during the Election campaign.

Goa:

It was Aam Aadmi Party’s first ever campaign in Goa. Political pundits were repeatedly warning not to ignore AAP. While this party was one of the factors, the bigger issue for the BJP was the RSS revolt. Former state head of the RSS, Subhash Velingkar had quit the organisation and announced an alliance with the Shiv Sena and the Maharashtrawadi Gomantak Party.

Manipur:

The biggest factor in Manipur was that of anti-incumbency. The Congress has ruled this state for 15 years and the BJP wanted to pull of an Assam in Manipur as well. While the BJP was training its guns in this state, pollsters point towards the decision by CM, Okram Ibobi Singh to carve out 8 new districts. This was likely to help the Congress retain Manipur. The other factor was Irom Sharmila. After breaking her fast that lasted 16 years, she decided to come on to the poll stage and challenge none other than the CM himself. She was new to politics and political experts voiced out that a win was unlikely for her. Her presence would have once again raised the issue regarding the Armed Forces Special Powers Act.

Punjab:

Drugs and anti-incumbency were the key issues during the Punjab elections. The sings of anti-incumbency was writ large during the 2014 general elections. The AAP had put up an impressive performance as a result of which they were enthused to contest the 2017 assembly polls. The AAP eventually became one of the major players and the Punjab polls witnessed a three corner contest with the Congress and BJP-Akali combine.

Drugs are a major issue in Punjab. The ruling Akali Dal has often been accused of doing very little to curb this problem. The Congress and the AAP had made it a poll issue and promise to wipe out the mafia. The other issue was of farmers' suicides. Punjab comes second after Maharashtra in the number of farmers' suicides. Several farmers have been pushed to take the extreme step. The allegations against the ruling dispensation was that the farmers were not provided enough aid.

It would be interesting to see if the data analysis relates to some of the issues highlighted above for the assembly elections 2017.

2.1.2 The Initial Hypothesis:

We would like to see if the Election Results were dependant on factors such as

- Anti-incumbency,
- Caste,
- Impact of a specific party,
- Criminal background of the candidates,
- Assets and Liabilities of the Candidates.

The study can be further extended to see an impact of Literacy, Crime Rate, etc.

2.1.3 The Potential Data Sources:

Students may source the needed data from sources such as:

- Election Commission of India: http://eci.nic.in/eci/eci.html
- My Neta Portal: http://myneta.info/
- Elections in India: http://www.elections.in/
- NITI Aayog: http://niti.gov.in/
- National Crime Records Bureau: http://ncrb.gov.in/
- Open Government Data Platform: https://data.gov.in/

2.2 Data Preparation:

We have sourced the data from following two portals:

1. Assembly Election Results for UP – 2017 and 2012 from Election Commission of India web site.
2. Candidates profile for UP – 2017 and 2012 from My Neta website.

The data shall be explored as part of Data Visualization and Descriptive Statistics. Necessary preparation shall be performed thereafter, such as outlier / missing values detection, treatment, merging the two data sets, etc.

Preliminary characteristics of the two data sets in scope are as follows:

2.2.1 Assembly Election Dataset

Sr. No.	Feature Name	Feature Description
1	Sr_No	Serial Number of the Candidate
2	ST_CODE	State Code, for Utter Pradesh, it is S24
3	ST_NAME	State Name.
4	MONTH	Month in which the Assembly Elections were conducted.
5	YEAR	Year of Assembly Election
6	DIST_NAME	District
7	AC_NO	Constituency Number
8	AC_NAME	Constituency Name
9	AC_TYPE	Constituency Type: e.g. General / SC / ST
10	Name	Name of the Candidate
11	CAND_SEX	Candidate Gender
12	CAND_CATEGORY	Candidate Category (General / SC / ST)
13	CAND_AGE	Candidate Age
14	PARTYABBRE	Party Abbreviation the Candidate belongs to.
15	TOTALVALIDVOTES POLLED	Total Valid Votes Polled to the Candidate
16	POSITION	Candidate Position after Election. (1: Winner, Other: Looser)

2.2.2 My Neta Dataset

Sr. No.	Feature Name	Feature Description
1	Sr_No	Serial Number of the Candidate
2	Name	Name of the Candidate
3	Constituency	Constituency Name
4	Party	Party of the Candidate
5	Criminal Case	Number of Criminal Cases against the Candidate
6	Education	Education Level of the Candidate.
7	Total Assets	Value of Total Assets the Candidate holding.
8	Liabilities	Value of Liabilities against the Candidate.

2.3 Model Planning:
In order to verify the initial hypothesis developed in section 2.1.2, this seems to be a typical Classification problem, wherein we would want to see the impact of various features over the Election Results. Hence, up front, CART and Random Forest seems to be a good choice.

We may revisit this section as we gain more understanding and clarity about the data.

2.4 Model Building:
After the data preparation, we shall proceed for Model Building.

2.5 Communicating Results:
The outcome of the Model shall interpreted and communicated in Business Language.

3 Data Preparation

3.1 Exploratory Data Analysis and Descriptive Statistics

3.2 Key observations:

Number of Rows and Columns:

- The number of Candidates in the Assembly Election 2017 are 7942 for all the five states.
- The number of columns (Features) in the dataset are 15.
- The MyNeta Website contains details of 4823 Candidates.

3.3 Data Cleansing / Pre Processing:

3.3.1 Candidate Data formatting

- Remove two extra rows from the Candidates Data File
- Rename following Features names which are ambiguous in nature.
  - Candidate Name
  - Criminal Case
  - Total Assets
- Remove Data Garbage from Candidate Data file such as:
  - '~.*', 'Rs', Comma, Blanks,
- Default Numeric Columns to zero, where data is not provided:
  - Total Assets: Set to zero where 'Nil' is captured.

3.3.2 Merge Election Data and MyNeta Data

- Merge the Data based on Candidate Name and Constituency.
- The Data Merging is done using 'Levinshtein Distance'[1] and Excel.
- Use the Master CSV File thereafter for further processing.

3.3.3 Pre-Processing on Master Data file for UP:

- Remove Records where Party abbreviation as 'NOTA'. (None of the Above).
- The Candidate Position is converted into 1 as Winning Candidate and 0 as Loosing Candidate.
- Default values for non-matching candidates between Assembly Election and My Neta Data:
  - No of Criminal Cases are set to zero.
  - Education to "Others".
  - Total Assets to zero.
  - Liabilities to zero.
  - Total Vote Poll to zero.
- Transform the Variables / Create New Features for the following:
  - Candidate Sex: 1 for Male, 2 For Female, and 3 for Other.
  - Candidate Financial Status is added as Dummy Variable as follows:

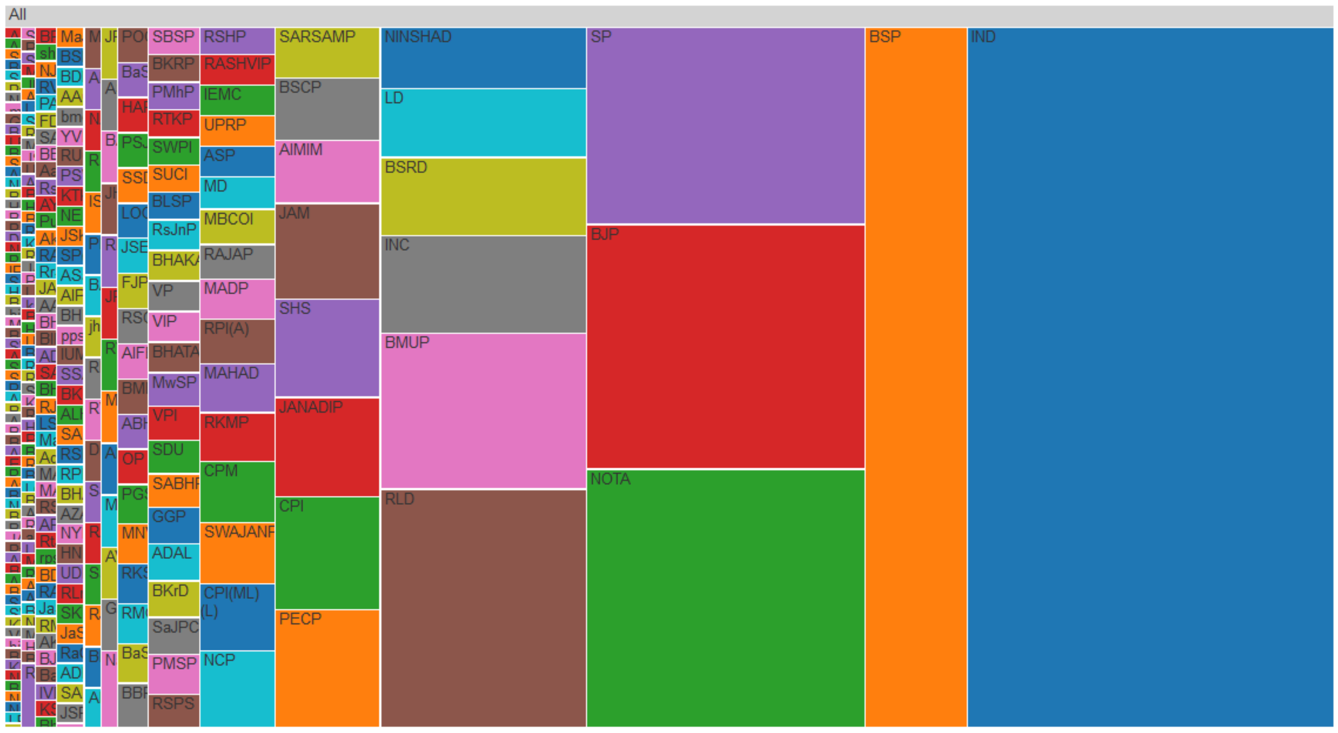
[1] Levenshtein Distance: See Appendix A.

- Total Assets < 1,00,000 🡐 Poor.
- Total Assets between 1,00,000 and 100,00,000 🡐 Lakhpati.
- Total Assets > 100,00,000 🡐 Crorepati.

  - Candidate Age Bracket is added as a Dummy Variable as follows:
    - Bracket 1: 25-35 Age
    - Bracket 2: 36-45 Age
    - Bracket 3: 46-55 Age
    - Bracket 4: 55+ Age
- Combine Parties who have not won a single seat into "Others".

4 Data Visualization

4.1 Party wise Candidates:



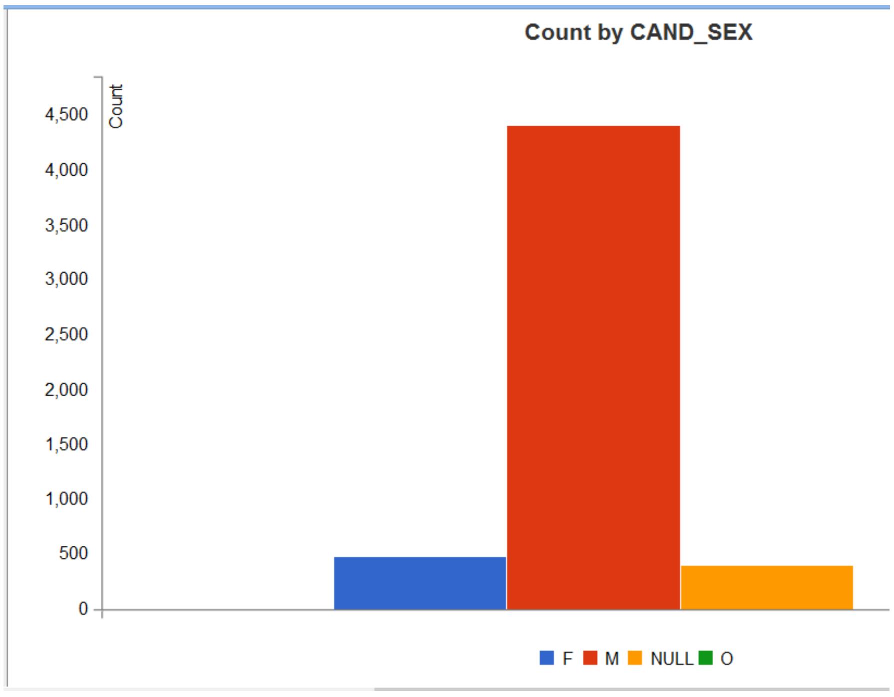
PARTYABBRE	Totals
IND	1,476
BSP	407
NOTA	407
BJP	388
SP	314
RLD	280
BMUP	183
INC	115
BSRD	92
LD	81
NINSHAD	72
PECP	70
CPI	68
JANADIP	60
SHS	59

Influential Political Parties:

Key Observations:

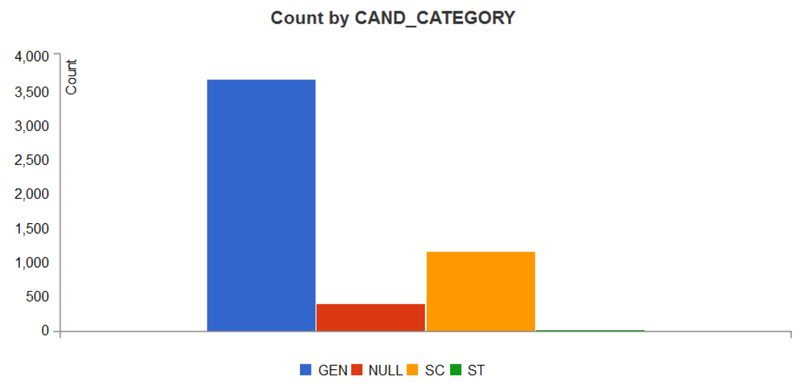
- Total 5307 Candidates contested in Utter Pradesh.
- As many as 1476 Independent candidates were testing their luck in the Assembly Election.
- BSP was contesting on maximum seats with 407 candidates, followed by BJP with 388 Candidates, SP with 388 and RLD with 280.
- INC, alliancing with SP was contesting on 115 Seats.

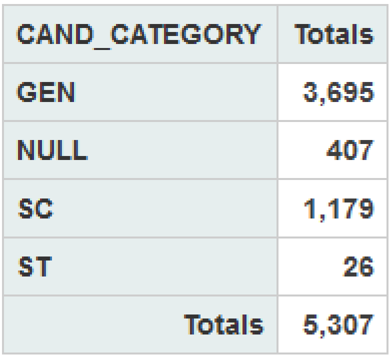
4.2 Gender wise candidate distribution:



CAND_SEX	Totals
F	484
M	4,415
NULL	407
O	1
Totals	5,307

4.3 Category wise candidate distribution:



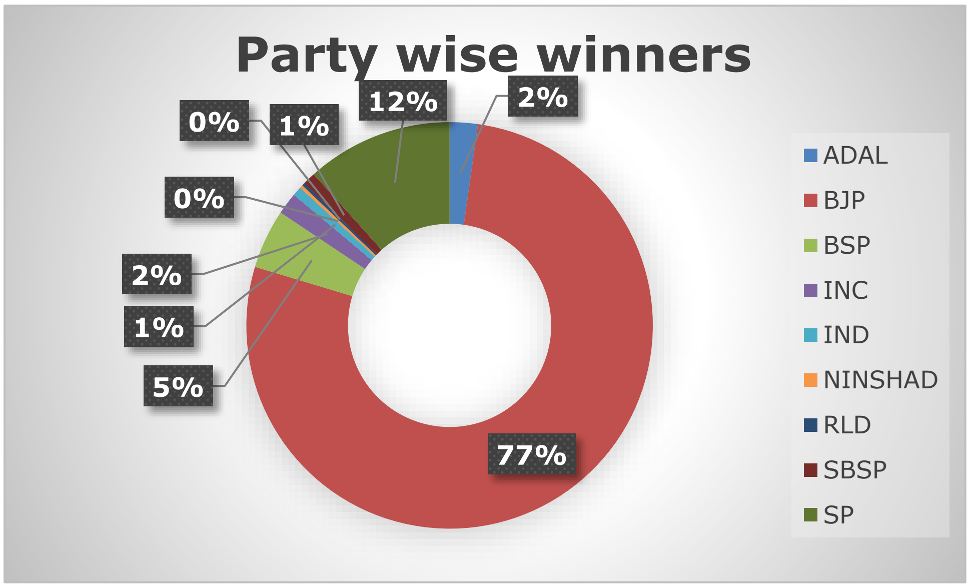


CAND_CATEGORY	Totals
GEN	3,695
NULL	407
SC	1,179
ST	26
Totals	5,307

Interpretation:

- Male candidates were more (4415) compared to Female candidates (484).
- Also General candidates were more (3695) compared to SC and ST, who were 1179 and 26 respectively.

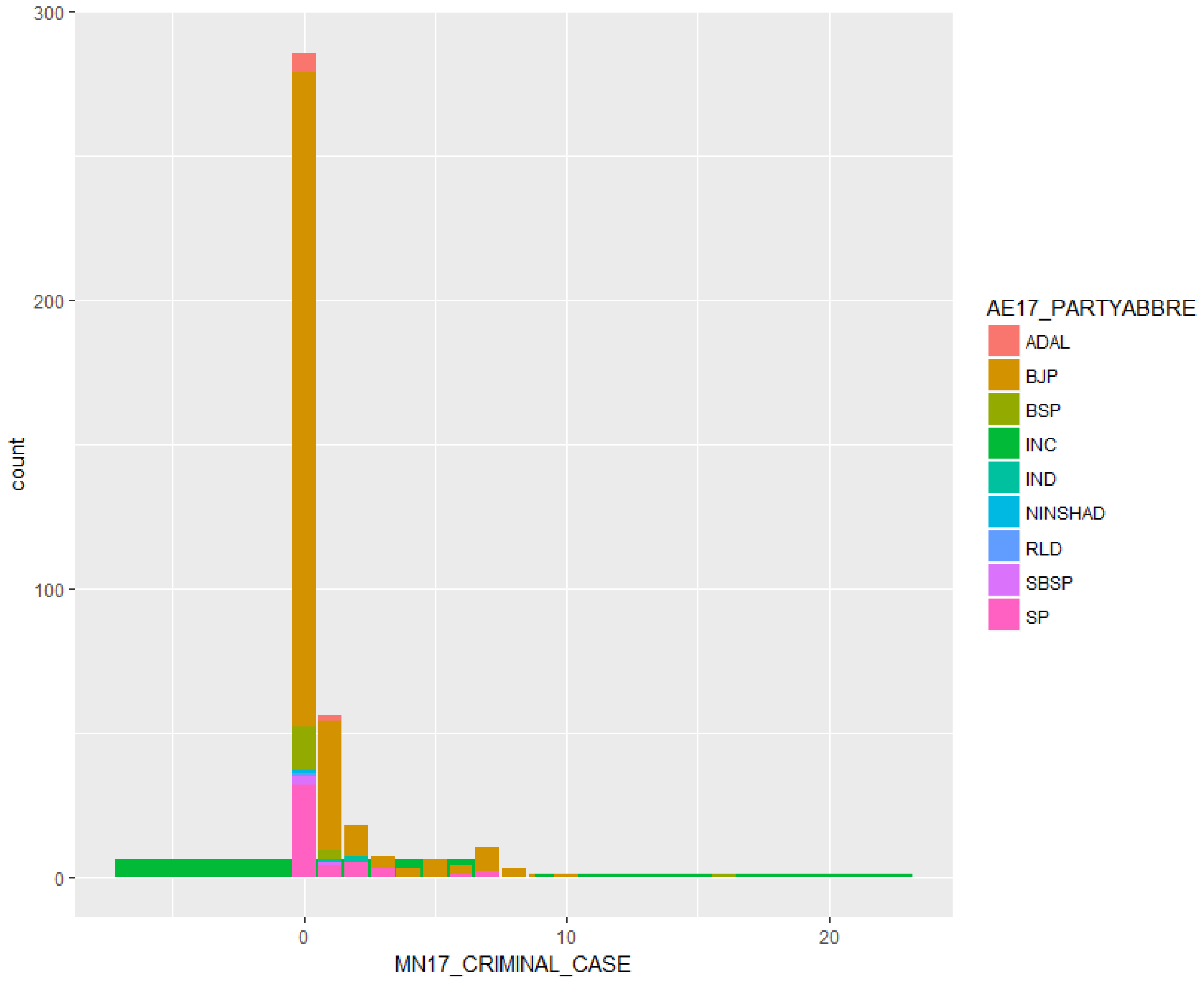
4.4 Party wise Distribution of Winning Candidates:



Interpretation:

- As per as winners are concerned, BJP was a clear leader, winning 77% seats followed by SP with 12% candidates.

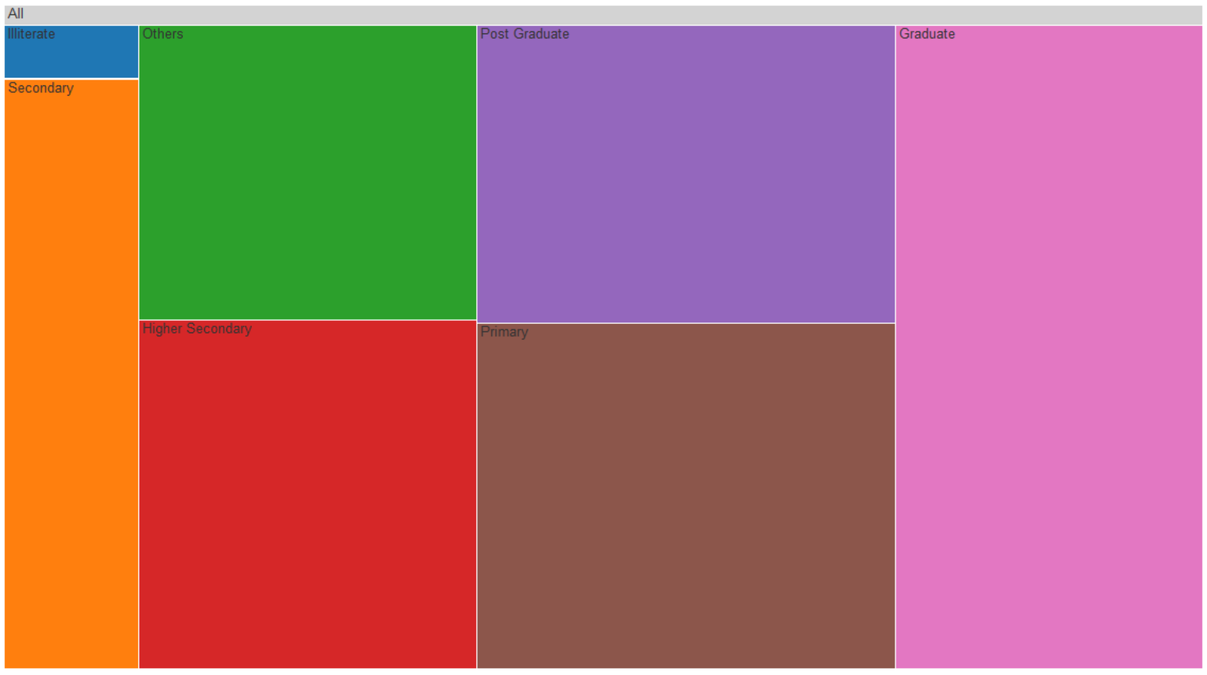
4.5 Party wise Criminal Cases against Candidates:



Interpretation:

- There were 242 Independent Candidates contesting elections who were having Criminal cases against those.
- 175 Candidates each from BJP and BSP also were having criminal cases, followed by 131 of SP Candidates.

4.6 Candidate distribution by Education:



MN17_EDUCATION_BKT	Totals
Graduate	1,256
Primary	920
Post Graduate	792
Higher Secondary	749
Others	633
Secondary	504
Illiterate	46
Totals	4,900

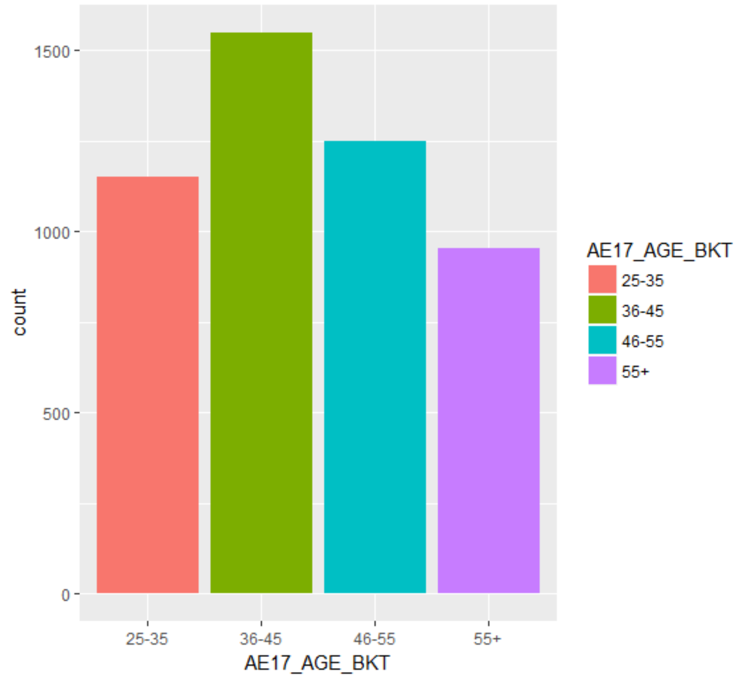
Key Observations:

Out of 4900 candidates, 2048 (1256+792) are Graduates and above. (Graduate, Post Graduate, Graduate Professional, Doctorate.

In terms of percentage, it comes to fairly 42%.

4.7 Candidate distribution by Age:

AE17_AGE_BKT	Totals
36-45	1,549
46-55	1,248
25-35	1,150
55+	953
Totals	4,900



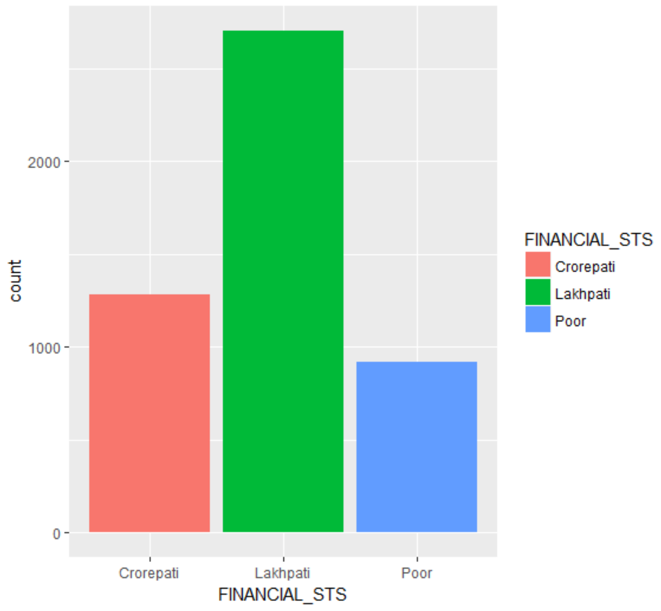
Key Observations:

Majority Candidates are falling in the Age Bracket of 36-55.

Young Candidates (25-35) forms total percentage of 23%. (1150 Candidates).

4.8 Candidate distribution by Financial Status:

FINANCIAL_STS	Totals
Lakhpati	2,702
Crorepati	1,281
Poor	917
Totals	4,900



Key Observations:

- 55% Candidates are Lakhpati. (2702) followed by
- 26% Crorepati Candidates. (1281)

5 Model Planning and Building

We are planning for three models as follows:

1. **Classification and Regression Tree:** A Prediction Model to predict the winning candidate, and in turn understand the features behind 2017 Election Result.
2. **Random Forest:** Another Prediction Model to predict the winning Candidates and Reason's behind 2017 Election Results. Both CART and RF shall be compared with each other.
3. **Linear Regression Model:** A Continuous Prediction Model to predict the Total Vote Share, and also consider Assembly Election 2012 Vote Shares for the candidates contesting elections in 2012 and 2017. This way, we also want to see if there is any incumbency (or Anti) Effect.

5.1 Classification and Regression Tree

The initial CART Model is built by setting up Control Parameters as follows:

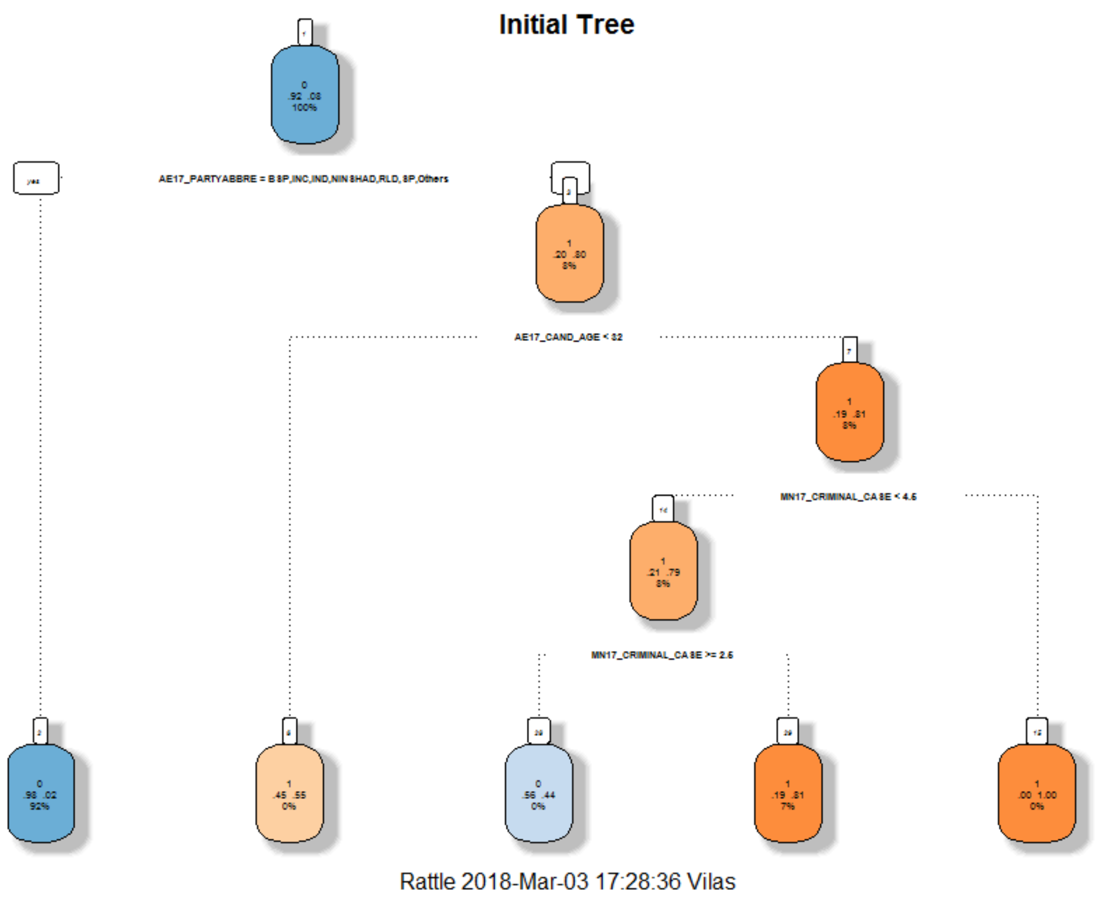
```
# Setting the control paramter inputs for rpart.
r.ctrl = rpart.control(minsplit=100,
                       minbucket = 10,
                       cp = 0, xval = 10)

# Build the Model.
m1.CART <- rpart(formula = AE17_POSITION ~ .,
                 data = Master_CT[,-c(1:10,15,21:28,32)], method = "class",
                   control = r.ctrl)

m1.CART

## n= 4900
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 4900 403 0 (0.91775510 0.08224490)
##    2) AE17_PARTYABBRE=BSP,INC,IND,NINSHAD,RLD,SP,Others 4493  78 0
(0.98263966 0.01736034) *
##    3) AE17_PARTYABBRE=ADAL,BJP,SBSP 407  82 1 (0.20147420 0.79852580)
##      6) AE17_CAND_AGE< 31.5 11   5 1 (0.45454545 0.54545455) *
##      7) AE17_CAND_AGE>=31.5 396  77 1 (0.19444444 0.80555556)
##       14) MN17_CRIMINAL_CASE< 4.5 374  77 1 (0.20588235 0.79411765)
##         28) MN17_CRIMINAL_CASE>=2.5 16   7 0 (0.56250000 0.43750000) *
##         29) MN17_CRIMINAL_CASE< 2.5 358  68 1 (0.18994413 0.81005587) *
##       15) MN17_CRIMINAL_CASE>=4.5 22   0 1 (0.00000000 1.00000000) *
```

The Classification Tree without Pruning:



Business Rules for determining the winning Candidate:

node number: 3 (One)
AE17_PARTYABBRE=ADAL,BJP,SBSP

node number: 7 (One)
AE17_PARTYABBRE=ADAL,BJP,SBSP
AE17_CAND_AGE>=31.5

node number: 2 (Zero)
AE17_PARTYABBRE=BSP,INC,IND,
NINSHAD,RLD,SP,Others

node number: 14 (One)
AE17_PARTYABBRE=ADAL,BJP,SBSP
AE17_CAND_AGE>=31.5
MN17_CRIMINAL_CASE< 4.5

node number: 6 (Zero)
AE17_PARTYABBRE=ADAL,BJP,SBSP
AE17_CAND_AGE< 31.5

node number: 28(One)
AE17_PARTYABBRE=ADAL,BJP,SBSP
AE17_CAND_AGE>=31.5
MN17_CRIMINAL_CASE< 4.5
MN17_CRIMINAL_CASE>=2.5

node number: 29(One)
AE17_PARTYABBRE=ADAL,BJP,SBSP
AE17_CAND_AGE>=31.5
MN17_CRIMINAL_CASE< 4.5
MN17_CRIMINAL_CASE< 2.5

node number: 15 (One)
AE17_PARTYABBRE=ADAL,BJP,SBSP
AE17_CAND_AGE>=31.5
MN17_CRIMINAL_CASE>=4.5

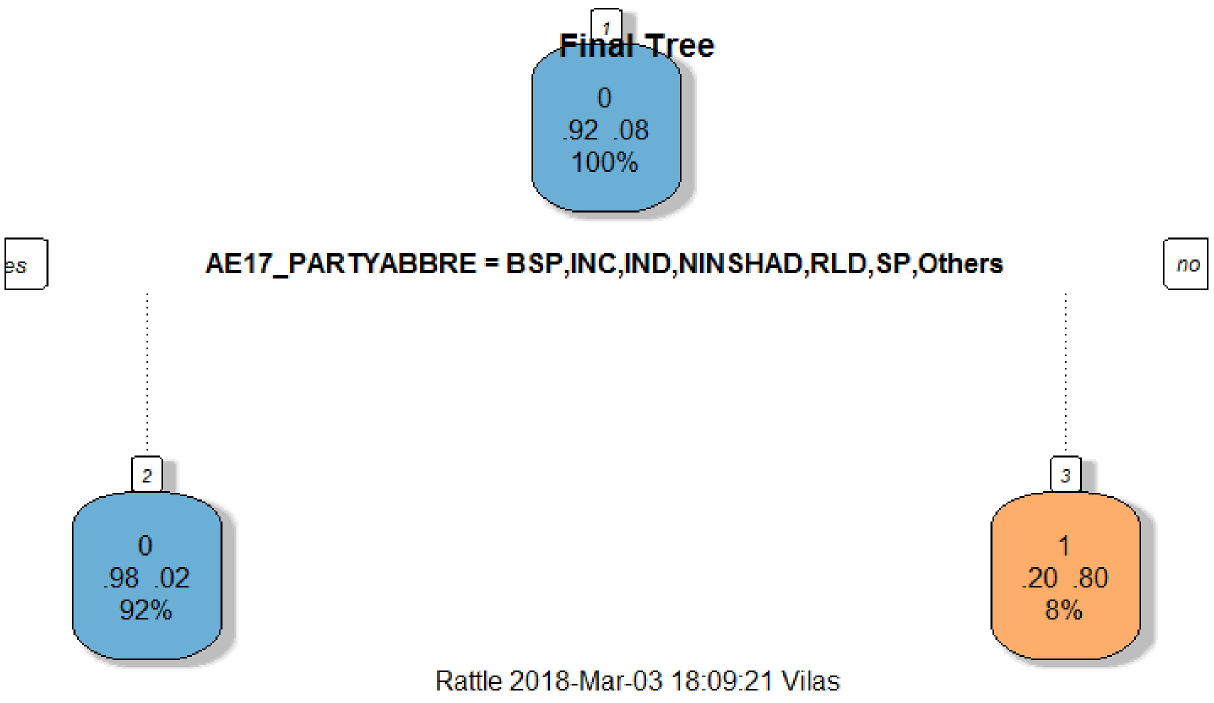
Pruning the Tree:

```
# Pruning the Tree
ptree<- prune(m1.CART, cp= 0.032 ,"CP")
printcp(ptree)

##
## Classification tree:
```

```
## rpart(formula = AE17_POSITION ~ ., data = Master_CT[, -c(1:10,
##     15, 21:28, 32)], method = "class", control = r.ctrl)
##
## Variables actually used in tree construction:
## [1] AE17_PARTYABBRE
##
## Root node error: 403/4900 = 0.082245
##
## n= 4900
##
##         CP nsplit rel error  xerror     xstd
## 1 0.60298      0   1.00000 1.00000 0.047721
## 2 0.03200      1   0.39702 0.39702 0.030871
```

Pruned Tree:


Final Tree
1
0
.92 .08
100%
yes
AE17_PARTYABBRE = BSP,INC,IND,NINSHAD,RLD,SP,Others
no
2
0
.98 .02
92%
3
1
.20 .80
8%
Rattle 2018-Mar-03 18:09:21 Vilas

Interpretation:

- The Pruned Tree is using only one Variable, Party Abbreviation.
- This may not be the best fit.
- The Unpruned Tree uses various features such as:
    - Party Name
    - Candidate Age
    - And Criminal Case

5.1.1 Performance Measures

The following model performance measures will be calculated on entire data set to gauge the goodness of the model:

- Rank Ordering
- KS
- Area Under Curve (AUC)
- Gini Coefficient
- Classification Error

5.1.1.1 Model Performance Measure - Rank Ordering

The rank order table is provided below:

deciles	Cnt	cnt_resp	cnt_ non_resp	rrate	cum_resp	cum_ non_resp	cum_ rel_resp	cum_rel_ non_resp	ks
10	4900	403	4497	8.22%	403	4497	100%	100%	0

Interpretation:

- The baseline Response Rate is 8.22%.
- The KS is above 0%, indicating it to be a **poor model**.

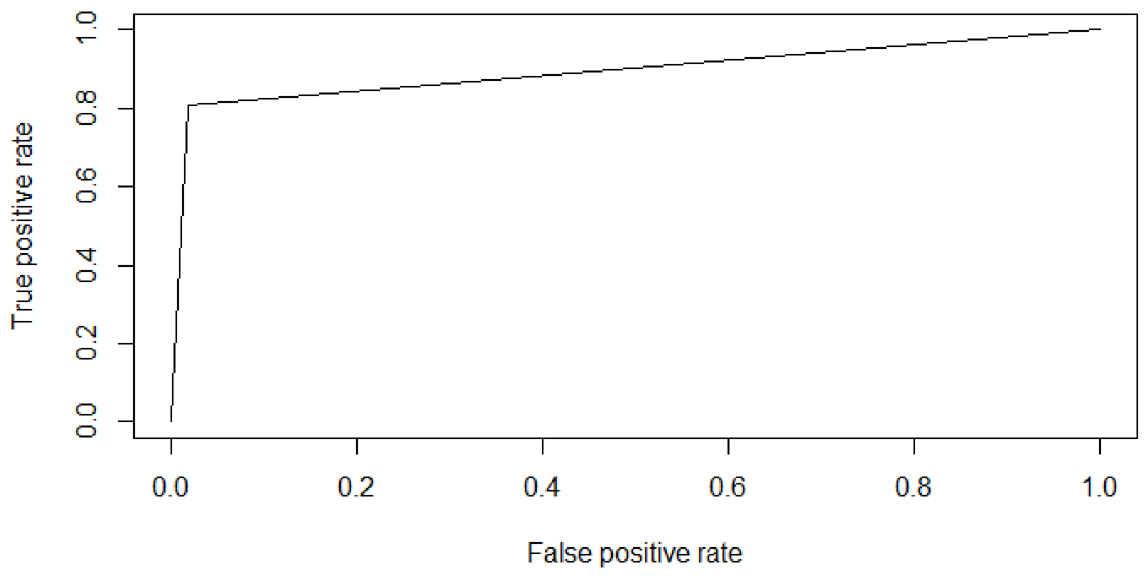
5.1.1.2 Model Performance Measure – KS , Area under Curve & Gini

The KS & AUC values are provided below:

Measure	Value
KS	78%
Area Under Curve	89%
Gini	80%

The AUC value around 89% indicates the good performance of the model.

Graphical representation of the Area Under Curve is as follows:



5.1.1.3 *Model Performance Measure – Confusion Matrix*

Confusion Matrix for our given Model is as follows:

```
##                predict.class
## AE17_POSITION    0    1
##               0 4415   82
##               1   78  325
```

```
##        predict.class
## TARGET     0     1
##       0 10257  2015
##       1  1568 10704
```

Accuracy = (4415+325)/(4415+325+82+78) = 96.73%

Classification Error Rate = 1- Accuracy = 3.26%.

The lower the classification error rate, higher the model accuracy, resulting in a better model.

5.1.1.4 *Model Performance Measure – Summary*

The summary of the various model performance measures on the training set are as follows:

Measure	Value
KS	78%
Area Under Curve	89%
Gini	80%

Accuracy	97%
Classification Error	3%

The model observed to perform as per expectations on majority of the model performance measures, indicating it to be a good model.

5.2 Random Forest

5.2.1 The initial build & Optimal No of Trees

The model is built with dependant variable as POSITION as follows:

```
names(Master_RF)

##  [1] "Sr_No"                "AE17_ST_CODE"         "AE17_ST_NAME"
##  [4] "AE17_MONTH"           "AE17_YEAR"            "AE17_DIST_NAME"
##  [7] "AE17_AC_NO"           "AE17_AC_NAME"         "AE17_AC_TYPE"
## [10] "AE17_CAND_NAME"       "AE17_CAND_SEX"        "AE17_CAND_CATEGORY"
## [13] "AE17_CAND_AGE"        "AE17_PARTYABBRE"      "AE17_TOTVOTPOLL"
## [16] "AE17_POSITION"        "MN17_CRIMINAL_CASE"   "MN17_EDUCATION"
## [19] "MN17_TOT_ASSETS"      "MN17_TOT_LIABILITIES" "MN17_SEQ_NO"
## [22] "AE12_SEQ_NO"          "AE12_CAND_NAME"       "AE12_CAND_SEX"
## [25] "AE12_CAND_CATEGORY"   "AE12_CAND_AGE"        "AE12_PARTYABBRE"
## [28] "AE12_TOTVOTPOLL"      "AE12_POSITION"        "FINANCIAL_STS"
## [31] "AE17_AGE_BKT"         "AE12_AGE_BKT"

# Random Forest

RF = randomForest( as.factor(AE17_POSITION) ~ .,
                   data = Master_RF[, -c(1:10,14,15,21:28,32)],
                   ntree = 500, mtry = 7, nodesize = 100,
                   importance = TRUE )
#

print(RF)

##
## Call:
##  randomForest(formula = as.factor(AE17_POSITION) ~ AE17_CAND_SEX + AE17_CAND_CATEGORY + AE17_CAND_AGE + MN17_CRIMINAL_CASE +      MN17_EDUCATION + MN17_TOT_ASSETS + MN17_TOT_LIABILITIES +      FINANCIAL_STS + AE17_AGE_BKT, data = Master_RF, ntree = 500,      mtry = 7, nodesize = 100, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 8.22%
## Confusion matrix:
##      0 1 class.error
## 0 4497 0           0
## 1  403 0           1
```

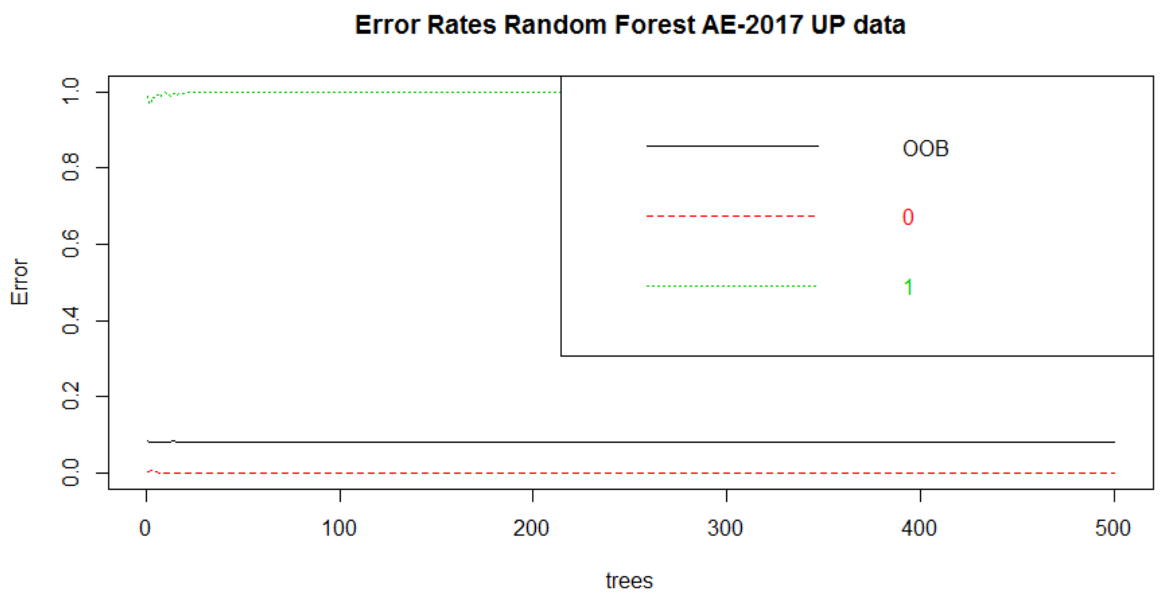
```
#
dev.off()

## null device
##           1

plot(RF, main="")
legend("topright", c("OOB", "0", "1"), text.col=1:6, lty=1:3, col=1:3)
title(main="Error Rates Random Forest AE-2017 UP data")
##
```

The Out-of-Bag Estimate of Error Rate for our given Random Forest in our case is 8.22%

The graphical output for the OOB estimate of error rate is provided below:



The minimum OOB estimate of error rate is provided below:

```
rf_err_rate$ID <- seq.int(nrow(rf_err_rate))

#Checking minimum error rate

rf_err_rate[which(rf_err_rate$OOB==min(rf_err_rate$OOB)),]

##          OOB        LOST       WON  ID
## 4 0.08074382 0.007144747 0.9837662  4

#Storing min try

min_tree<-
min(rf_err_rate[which(rf_err_rate$OOB==min(rf_err_rate$OOB)),]$ID)
```

It is observed that as the number of tress increases, the OOB error rate starts decreasing with OOB = 0.082 (the minimum value). After this, the OOB doesn't decrease further and remains largely steady. **Hence, the optimal number of trees would be 4.**

5.2.2 Variable Importance

To understand the important variables in Random Forest, the following measures are generally used:

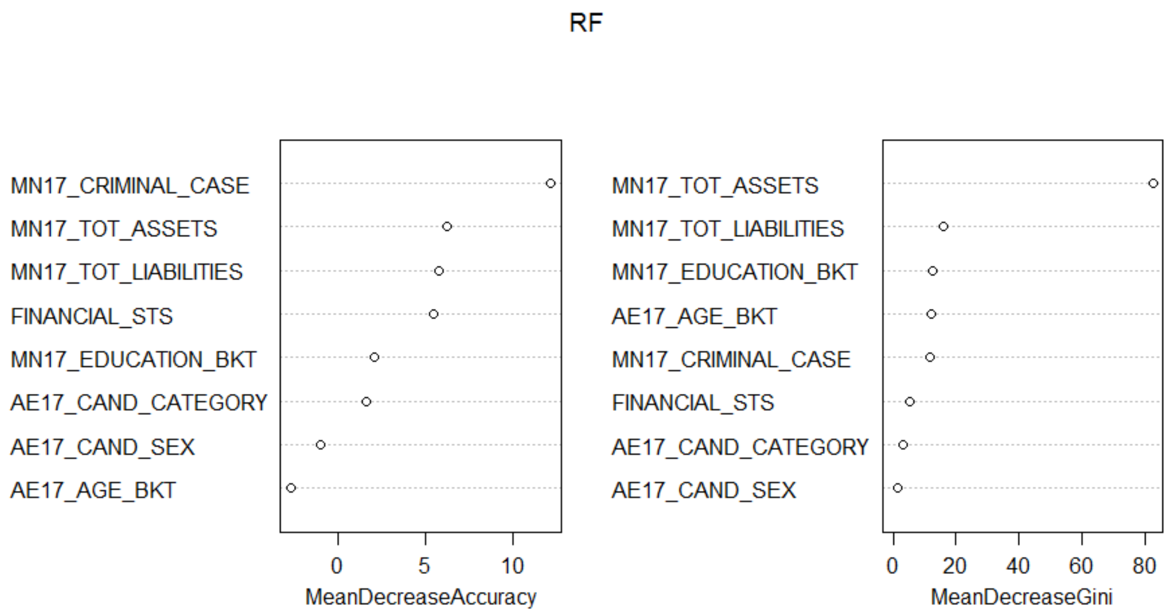
- Mean Decrease in Accuracy is based on permutation
  - Randomly permute values of a variable for which importance is to be computed in the OOB sample
  - Compute the Error Rate with permuted values
  - Compute decrease in OOB Error rate (Permuted - Not permuted)
  - Average the decrease over all the trees
- Mean Decrease in Gini is computed as "total decrease in node impurities from splitting on the variable, averaged over all trees"

The variables importance is computed as follows:

```
# List the importance of the variables.
impVar <- round(randomForest::importance(RF), 2)
# impVar[order(impVar[,3], decreasing=TRUE),]
# impVar[order(impVar[,4], decreasing=TRUE),]
impVar[order(impVar[,1], decreasing=TRUE),]

##                           0     1 MeanDecreaseAccuracy MeanDecreaseGini
## MN17_TOT_LIABILITIES 11.14 -2.92                11.83            15.94
## MN17_CRIMINAL_CASE    9.07  6.88                11.52            10.37
## MN17_EDUCATION        8.38  5.31                10.64            24.68
## MN17_TOT_ASSETS       7.91 -5.41                 7.80            66.54
## FINANCIAL_STS         6.37 -5.20                 6.57             7.44
## AE17_CAND_CATEGORY    2.41 -4.26                 1.33             2.91
## AE17_CAND_SEX         1.92 -0.84                 1.61             1.18
## AE17_AGE_BKT         -0.76  1.25                -0.47             2.49
## AE17_CAND_AGE        -4.17  6.02                -1.68            26.01
```

Variable Importance: Graphical representation



Interpretation:
As can be seen, using the Mean Decrease Gini Measure, following Variables are playing significant role in deciding the Winning Candidate:

1. Total Assets
2. Total Liabilities
3. Candidate Education
4. Candidate Age and
5. Criminal Case against the Candidate.

5.2.3 Optimal mtry value

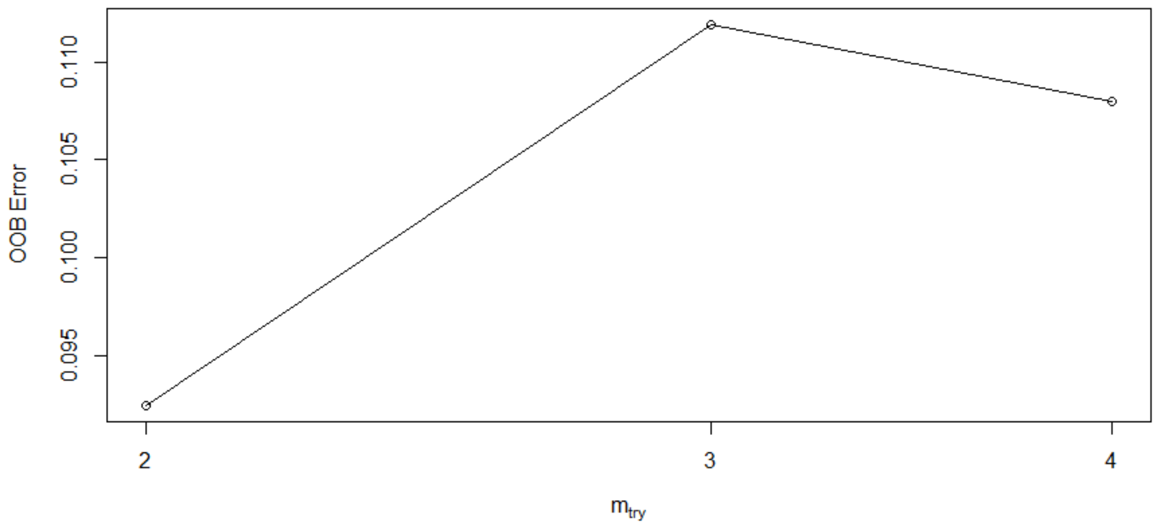
In the random forests literature, the number of variables available for splitting at each tree node is referred to as the **mtry parameter.**

The optimum number of variables is obtained using tuneRF function as follows:

```
tune_rf_model <- tuneRF(x = Master_RF[,-c(1:10,14:16,21:28,32)],
              y=as.factor(Master_RF$AE17_POSITION),
              mtryStart = 3, # Approx. Sqrt of Tot. No of Variables
              ntreeTry=min_tree,
              stepFactor = 1.5,
              improve = 0.0001,
              trace=TRUE,
              plot = TRUE,
              doBest = TRUE,
              nodesize = 10,
              importance=TRUE
)

## mtry = 3  OOB error = 9.12%
## Searching left ...
## mtry = 2     OOB error = 8.57%
## 0.06040268 1e-04
## Searching right ...
## mtry = 4      OOB error = 8.82%
## -0.02857143 1e-04
```

Output: OOB Error Vs Mtry:



As can be seen, the optimum number of Variables is 4 to get the optimal Out of Bag Error of 8.82%.

5.2.4 Performance Measures

The following model performance measures will be calculated on the training set to gauge the goodness of the model:

- Rank Ordering
- KS
- Area Under Curve (AUC)
- Gini Coefficient
- Classification Error

5.2.5 Model Performance Measure - Rank Ordering

Rank Ordering as model performance measures helps:

- Assess the ability of the model to relatively rank the customers
- How well the model separates the Responder Class from the Non-Responder
- Track the utility of the model on an ongoing basis

The rank order table is provided below:

Sr. No.	deciles	cnt	cnt_resp	cnt_ non_resp	rrate	cum_ resp	cum_ non_resp	cum_ rel_resp	cum_rel_ non_resp	ks
1	10	497	285	212	57%	285	212	70.70%	4.70%	66.01
2	9	497	48	449	10%	333	661	82.60%	14.70%	67.93
3	8	497	1	496	0.2%	334	1157	82.90%	25.70%	57.15
4	7	623	3	620	0.5%	337	1777	83.60%	39.50%	44.1
5	6	357	2	355	0.6%	339	2132	84.10%	47.40%	36.71
6	5	2429	64	2365	2.6%	403	4497	100.00%	100.00%	0

Interpretation:

- The baseline Response Rate is 8%, whereas the response rate in top two deciles is 57% and 10% respectively.
- With top 2 deciles, the KS is 68%, indicating it to be **a good model.**

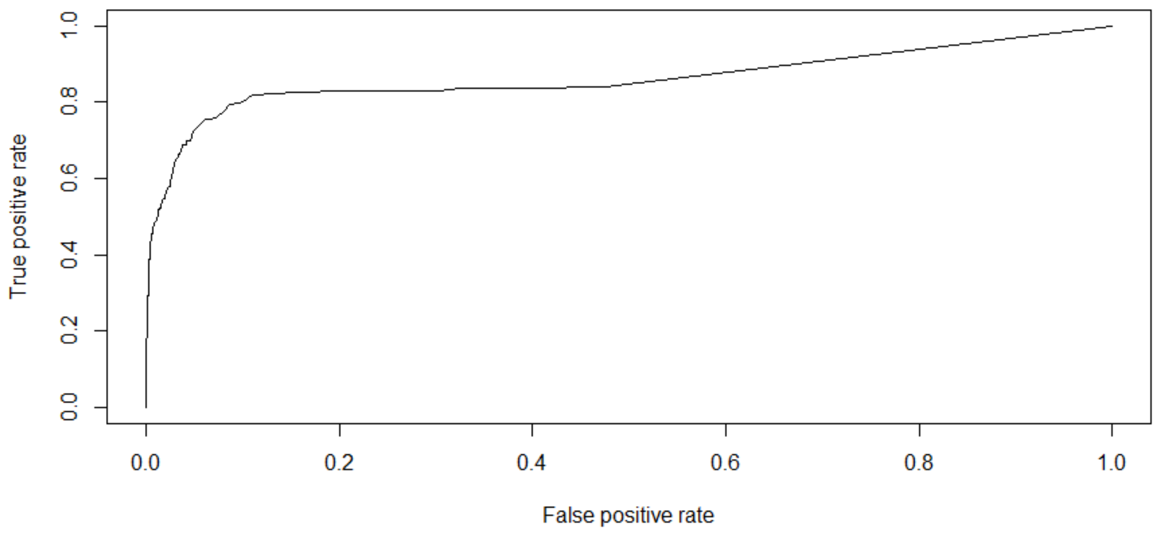
5.2.6 Model Performance Measure – KS and Area under Curve

The KS & AUC values are provided below:

Measure	Value
KS	70.78%
Area Under Curve	86.22%

The AUC value above 85% indicates fairly good performance of the model.

Graphical representation of the Area Under Curve is as follows:



5.2.7 Model Performance Measure – Gini Coefficient

The Gini Coefficient is the ratio of the area between the line of perfect equality and the observed Lorenz curve to the area between the line of perfect equality and the line of perfect inequality. The higher the coefficient, the more unequal the distribution is.

Gini coefficient can be straight away derived from the AUC ROC number.

Gini above 60% is a good model.

For our given dataset and the model we built, the Gini Coefficient is 82%, indicating a robust model.

5.2.8 Model Performance Measure – Confusion Matrix

A confusion matrix is an N X N matrix, where N is the number of classes being predicted. For the problem in hand, we have N=2, and hence we get a 2 X 2 matrix.

Few performance parameters we can obtain with the help of confusion matrix are as follows:

- **Accuracy:** the proportion of the total number of predictions that were correct.
- **Positive Predictive Value or Precision:** the proportion of positive cases that were correctly identified.
- **Negative Predictive Value:** the proportion of negative cases that were correctly identified.
- **Sensitivity or Recall:** the proportion of actual positive cases which are correctly identified.
- **Specificity:** the proportion of actual negative cases which are correctly identified.

Confusion Matrix		**Target**			
		Positive	Negative		
Model	Positive	a	b	*Positive Predictive Value*	a/(a+b)
	Negative	c	d	*Negative Predictive Value*	d/(c+d)
		Sensitivity	*Specificity*	**Accuracy** = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

Confusion Matrix for our given Model is as follows:

```
##          predict.class
## POSITION       0     1
##          0  4497     0
##          1   388    15
```

Accuracy = (4497+15)/(4497+15+0+388) = 92%

Classification Error Rate = 1- Accuracy = 8%.

The lower the classification error rate, higher the model accuracy, resulting in a better model. The classification error rate can be reduced if there were more independent variables were present for modeling.

5.2.9 Model Performance Measure – Summary

The summary of the various model performance measures on the training set are as follows:

Measure	Value
KS	70.78%
Area Under Curve	86.22%
Gini	82.68%
Accuracy	92.00%
Classification Error Rate	8%

The model observed to perform fairly decent on majority of the model performance measures, indicating it to be a good model.

5.3 Linear Regression Model

We shall build Linear Regression Prediction Model to predict the Vote Share.

Data from Assembly Election 2017 shall be merged with My Neta 2017, and with Assembly 2012 Data to see the effect of Incumbency or Anti Incumbency.

In order to see the impact of 2012 Election Results, we shall consider the candidates which are common in both 2012 and 2017.

We get total 802 such matching records, as follows:

```
Master_12_17 <- Master[which(Master$AE12_PARTYABBRE != "#N/A"),]
nrow(Master_12_17)

## [1] 802
```

5.3.1 Correlation between Variables

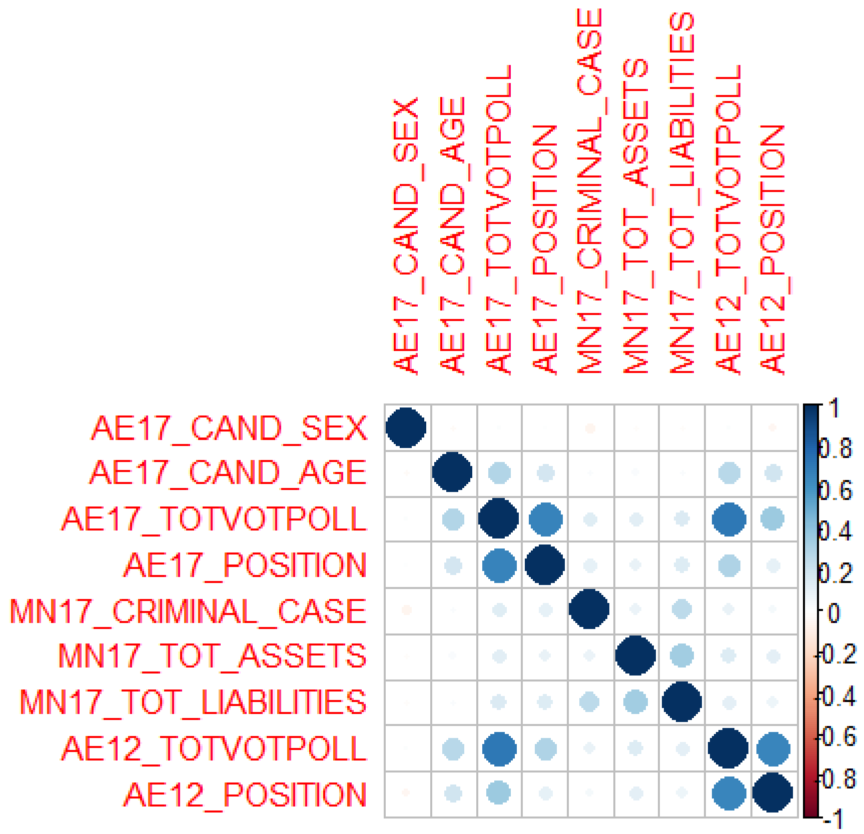
Let’s check correlation of various features with each other, especially with Total Vote Poll.

```
AE17_Vote_Poll_Rel <-subset(Master_12_17[,c(11,13,15:17,19:20,28,29)])

cor(AE17_Vote_Poll_Rel)

##                      AE17_CAND_SEX AE17_CAND_AGE AE17_TOTVOTPOLL
## AE17_CAND_SEX          1.000000000   -0.02145643      0.01451856
## AE17_CAND_AGE         -0.021456428    1.00000000      0.29208691
## AE17_TOTVOTPOLL        0.014518564    0.29208691      1.00000000
## AE17_POSITION          0.005905108    0.18395591      0.66632915
## MN17_CRIMINAL_CASE    -0.056781836    0.02186201      0.13333941
## MN17_TOT_ASSETS       -0.015498515    0.02955731      0.12883149
## MN17_TOT_LIABILITIES  -0.016835755    0.02406122      0.15864243
## AE12_TOTVOTPOLL        0.012279345    0.27780718      0.71857489
## AE12_POSITION         -0.041841681    0.19484776      0.36275876
##                      AE17_POSITION MN17_CRIMINAL_CASE MN17_TOT_ASSETS
## AE17_CAND_SEX          0.005905108        -0.05678184     -0.01549852
## AE17_CAND_AGE          0.183955913         0.02186201      0.02955731
## AE17_TOTVOTPOLL        0.666329151         0.13333941      0.12883149
## AE17_POSITION          1.000000000         0.10918734      0.09887042
## MN17_CRIMINAL_CASE     0.109187342         1.00000000      0.08418050
## MN17_TOT_ASSETS        0.098870422         0.08418050      1.00000000
## MN17_TOT_LIABILITIES   0.146863200         0.26236249      0.34147958
## AE12_TOTVOTPOLL        0.306871753         0.09530881      0.14479708
## AE12_POSITION          0.101609618         0.04738144      0.11839467
##                      MN17_TOT_LIABILITIES AE12_TOTVOTPOLL AE12_POSITION
## AE17_CAND_SEX                 -0.01683575      0.01227934   -0.04184168
## AE17_CAND_AGE                  0.02406122      0.27780718    0.19484776
## AE17_TOTVOTPOLL                0.15864243      0.71857489    0.36275876
## AE17_POSITION                  0.14686320      0.30687175    0.10160962
## MN17_CRIMINAL_CASE             0.26236249      0.09530881    0.04738144
## MN17_TOT_ASSETS                0.34147958      0.14479708    0.11839467
## MN17_TOT_LIABILITIES           1.00000000      0.11000423    0.07576649
## AE12_TOTVOTPOLL                0.11000423      1.00000000    0.65157286
## AE12_POSITION                  0.07576649      0.65157286    1.00000000

correlation <- cor(AE17_Vote_Poll_Rel)
corrplot(correlation, method="circle")
```



Interpretation:

- AE17 Total Vote Poll Vs AE17 Position has strong Correlation with correlation coefficient of +0.66
- AE17 Total Vote Poll Vs AE12 Total Vote Poll has strong Correlation with correlation coefficient of +0.72
- AE17 Total Vote Poll Vs AE12 Position has positive Correlation with correlation coefficient of +0.36

5.3.2 The initial Regression Model

```
names(AE17_Vote_Poll_Rel)

## [1] "AE17_CAND_SEX"        "AE17_CAND_AGE"        "AE17_TOTVOTPOLL"
## [4] "AE17_POSITION"        "MN17_CRIMINAL_CASE"   "MN17_TOT_ASSETS"
## [7] "MN17_TOT_LIABILITIES" "AE12_TOTVOTPOLL"      "AE12_POSITION"


OLS1 <- lm(AE17_TOTVOTPOLL~., data = AE17_Vote_Poll_Rel)
summary(OLS1)

##
## Call:
## lm(formula = AE17_TOTVOTPOLL ~ ., data = AE17_Vote_Poll_Rel)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -75624  -8379  -4113  10280 126074
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -2.614e+03  4.677e+03  -0.559  0.57642
## AE17_CAND_SEX           3.884e+02  2.984e+03   0.130  0.89648
## AE17_CAND_AGE           1.882e+02  6.854e+01   2.747  0.00616 **
## AE17_POSITION           4.554e+04  1.868e+03  24.376  < 2e-16 ***
## MN17_CRIMINAL_CASE      3.128e+02  2.844e+02   1.100  0.27163
## MN17_TOT_ASSETS        -2.766e-06  7.175e-06  -0.385  0.69997
## MN17_TOT_LIABILITIES    7.268e-05  5.835e-05   1.246  0.21326
## AE12_TOTVOTPOLL         8.248e-01  3.361e-02  24.541  < 2e-16 ***
## AE12_POSITION          -8.667e+03  2.012e+03  -4.307 1.86e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19760 on 793 degrees of freedom
## Multiple R-squared:  0.7451, Adjusted R-squared:  0.7425
## F-statistic: 289.7 on 8 and 793 DF,  p-value: < 2.2e-16
```

Interpretation:

- Following Features are having significant effect on Total Votes polled:
  - AE17 Candidate Age
  - AE17 Candidate Position
  - AE12 Total Votes Polled
  - AE12 Position 🡐 Negative Effect
- Adjusted R Square is 0.74, indicating a good model fit.

Let's remove the insignificant Variables and refine the initial built model.

5.3.3 The refined Regression Model

```
# Removing the Variables having least impact.
# Gender, Criminal Cases, Total Assets and Liabilities
#
OLS2 <- lm(AE17_TOTVOTPOLL~., data = AE17_Vote_Poll_Rel[,-c(1,5,6,7)])
summary(OLS2)

##
## Call:
## lm(formula = AE17_TOTVOTPOLL ~ ., data = AE17_Vote_Poll_Rel[,
##     -c(1, 5, 6, 7)])
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -75814  -8483  -4456  10344 125802
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.861e+03  3.370e+03  -0.552  0.58098
## AE17_CAND_AGE     1.856e+02  6.848e+01   2.710  0.00688 **
## AE17_POSITION    4.598e+04  1.850e+03  24.850  < 2e-16 ***
## AE12_TOTVOTPOLL  8.281e-01  3.342e-02  24.778  < 2e-16 ***
## AE12_POSITION   -8.673e+03  2.005e+03  -4.325 1.72e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19760 on 797 degrees of freedom
## Multiple R-squared:  0.7439, Adjusted R-squared:  0.7426
## F-statistic: 578.7 on 4 and 797 DF,  p-value: < 2.2e-16
```

Interpretation:

- Same results as seen in initial Model build, and confirms that the **AE12 Position has Negative Impact on Total Votes Polled in AE2017.**
- In other words, we can see anti-incumbency effect on Assembly Election 2017.

Check Multi-Collinearity Effect:

```
# Check Multi collinearity effect.
# VIF = 1      - Not Correlated
# VIF > 1 < 5 - Moderately Correlated
# VIF > 5      - Highly Correlated
#
library(car)

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

vif(OLS1)

##        AE17_CAND_SEX        AE17_CAND_AGE        AE17_POSITION
##             1.008840             1.099138             1.160738
##    MN17_CRIMINAL_CASE      MN17_TOT_ASSETS MN17_TOT_LIABILITIES
##             1.086607             1.149549             1.219868
##       AE12_TOTVOTPOLL        AE12_POSITION
##             2.006545             1.783739

# As can be seen, VIF is just slightly greater than 1, hence we can
# Conclude that our variables are moderately correlated.
```

6 Communicating Results – Conclusion

From the three models we built on Assembly Election 2017 for Utter Pradesh, we can conclude that the Major parameters impacting the Winners were:

1. Total Assets
2. Total Liabilities
3. Candidate Education
4. Candidate Age and
5. Criminal Case against the Candidate.

From the Linear Regression Model, we could also conclude that there was an Anti-Incumbency effect, which could be easily correlated with the Election Results wherein the SP Candidates had won in majority in 2012, whereas in 2017, BJP did a clean sweep.

7 Appendix A – Data Merge using Levinshtein Distance

This is a sample example of how Levenshtein Method can be used to compare distance between two strings.

Strings with perfect match would have 100% match, whereas with minor discrepancies in two strings, the percentage match may vary from 0.7 to 0.90.

One may also apply Soundex and phonetic algorithms along with Levenshtein to achieve more accuracy.

```
#  Get current working directory
# LOAD PACKAGE "RecordLinkage" FOR CALCULATING EDIT DISTANCE
library(RecordLinkage)

# SOME EXAMPLES OF DISTANCE CALCULATION
#levenshteinSim("apple", "apple")
#levenshteinSim("apple", "aaple")
#levenshteinSim("apple", "appled")
#levenshteinSim("appl", "apple")

levenshteinSim("anisharamukhulat", "nisharamukhulat")

## [1] 0.9375

levenshteinSim("ajayjashughodo","ajayjasugholi")

## [1] 0.7857143

levenshteinSim("chaitnabajranggorte","chetnabajranggorat")

## [1] 0.7894737

levenshteinSim("darshanajaniyadhoom","darshnajanyadhum")

## [1] 0.7894737

levenshteinSim("meenapraksahgorat","minaprakashgorat")

## [1] 0.7647059
```

8 Appendix B – Sample Source Code

```
#============================================================================
#
#   Mini Project 5: Predictive Modeling
#
#   1. Environment Setup and Data Import
#   2. Data Pre Processing and Cleansing
#   3. Data Visualization
#   4. Model Building - CART
#   5. Model Building - Random Forest
#   6. Model Building - Linear Regression
#
#==========================================================================
# 1. Environment Set up and Data Import
#==========================================================================
#Install Libraries and Packages
#===========================================================================#
install.packages("rpivotTable")
#install.packages("readxl")
#install.packages("caret")
#install.packages("rpart.plot")
#install.packages("randomForest")
#install.packages("RGtk2")

library(readxl) # For Reading Excel data format
library(XML)
library(plyr)
library(dplyr)

library(data.table)

library('ggplot2')
library('caret')

library(rpart)
library(rpart.plot)
library(randomForest)

library(rattle)

library(RColorBrewer)

#
# Setup Working Directory
#===========================================================================s
etwd("D:\\00 Great Lakes Engagement\\04 Week 5 Mini Project")
getwd()

#
#Reading Election Commission Data from Excel which is downloaded from election
commission site

LA_2017 <- read_excel("LA 2017.xlsx")
# View(LA_2017)
```

```
#Reading UP data from Mynetha website

MyNeta_UP<-
"http://www.myneta.info/uttarpradesh2017/index.php?action=summary&subAction=ca
ndidates_analyzed&sort=candidate#summary"

#Reading Candidate education,assets and criminal data from table 3

Candidate_Data<-readHTMLTable(MyNeta_UP, which=3)

# Verify Data Structure
str(LA_2017)

#Converting into data frame
LA_2017 <-data.frame(LA_2017)

#Checking dimentions of data
dim(LA_2017)

## [1] 7942   15

dim(Candidate_Data)

## [1] 4825    8

#Check Sample Data
head(LA_2017)

head(Candidate_Data)

##
Sno
## 1 Order by: Candidate |Constituency |Party|Criminal Cases|Education|Total
Assets|Liabilities
## 2
Sno
## 3
1
## 4
2
## 5
3
## 6
4
##              Candidateâˆ‡ Constituency               Party  Criminal Case
## 1                    <NA>          <NA>                <NA>           <NA>
## 2       Candidate<U+2207>  Constituency               Party  Criminal Case
## 3                  A Hasiv    ARYA NAGAR                 BSP              0
## 4                  A Wahid      GAINSARI                 IND              0
## 5 Aan Shikhar Shrivastava   GOSHAINGANJ Satya Shikhar Party              0
## 6        Aaptab Urf Aftab    MUBARAKPUR     Islam Party Hind              0
##   Education                Total Assets             Liabilities
## 1       <NA>                       <NA>                    <NA>
## 2  Education               Total Assets             Liabilities
## 3  12th Pass Rs 3,94,24,827 ~ 3 Crore+ Rs 58,46,335 ~ 58 Lacs+
## 4  10th Pass      Rs 75,106 ~ 75 Thou+                 Rs 0 ~
## 5   Graduate      Rs 41,000 ~ 41 Thou+                 Rs 0 ~
## 6 Illiterate      Rs 20,000 ~ 20 Thou+                 Rs 0 ~

# As we can see, Column names and data needs correction.
#Removing two extra unwanted rows from candidate dataset

Candidate_Data<-Candidate_Data[-c(1,2),]
# Recheck sample data
head(Candidate_Data)

##   Sno            Candidateâ^‡ Constituency               Party
## 3   1                 A Hasiv    ARYA NAGAR                 BSP
## 4   2                 A Wahid      GAINSARI                 IND
## 5   3 Aan Shikhar Shrivastava   GOSHAINGANJ Satya Shikhar Party
## 6   4         Aaptab Urf Aftab    MUBARAKPUR    Islam Party Hind
## 7   5              Aashi Gaur      KHATAULI             Lok Dal
## 8   6              Aashif Beg      BARKHERA                 IND
##   Criminal Case  Education               Total Assets
## 3              0  12th Pass Rs 3,94,24,827 ~ 3 Crore+
## 4              0  10th Pass      Rs 75,106 ~ 75 Thou+
## 5              0   Graduate      Rs 41,000 ~ 41 Thou+
## 6              0 Illiterate      Rs 20,000 ~ 20 Thou+
## 7              0   Literate   Rs 34,68,543 ~ 34 Lacs+
## 8              0  Not Given   Rs 13,30,000 ~ 13 Lacs+
##              Liabilities
## 3 Rs 58,46,335 ~ 58 Lacs+
## 4                 Rs 0 ~
## 5                 Rs 0 ~
## 6                 Rs 0 ~
## 7                 Rs 0 ~
## 8                 Rs 0 ~

#
#==========================================================================
# Data Pre Processing and cleansing
#==========================================================================
# Some Column names really needs cleaning
names(Candidate_Data)

## [1] "Sno"             "Candidateâ^‡"   "Constituency "  "Party "
## [5] "Criminal Case " "Education "     "Total Assets"   "Liabilities"

Candidate_Data$Candidate <- Candidate_Data$`Candidateâ^???`
Candidate_Data$Criminal_Case <- Candidate_Data$`Criminal Case `
Candidate_Data$Total_Assets <- Candidate_Data$`Total Assets`

names(Candidate_Data)

##  [1] "Sno"             "Candidateâ^‡"   "Constituency "  "Party "
##  [5] "Criminal Case " "Education "     "Total Assets"   "Liabilities"
##  [9] "Criminal_Case"  "Total_Assets"
```

```r
Candidate_Data$`Candidateâ^???` <- NULL
Candidate_Data$`Criminal Case ` <- NULL
Candidate_Data$`Total Assets` <- NULL

names(Candidate_Data)

## [1] "Sno"             "Candidateâ^‡"  "Constituency " "Party "
## [5] "Education "      "Liabilities"   "Criminal_Case" "Total_Assets"

Candidate_Data$Total_Assets<-gsub('~.*','',Candidate_Data$Total_Assets)
Candidate_Data$Liabilities<-gsub('~.*','',Candidate_Data$Liabilities)
Candidate_Data$Total_Assets<-gsub('Rs','',Candidate_Data$Total_Assets)
Candidate_Data$Liabilities<-gsub('Rs','',Candidate_Data$Liabilities)
Candidate_Data$Total_Assets<-gsub(',','',Candidate_Data$Total_Assets)
Candidate_Data$Liabilities<-gsub(',','',Candidate_Data$Liabilities)
Candidate_Data$Total_Assets<-
gsub('[[:blank:]]','',Candidate_Data$Total_Assets)
Candidate_Data$Liabilities<-gsub('[[:blank:]]','',Candidate_Data$Liabilities)
Candidate_Data$Total_Assets[Candidate_Data$Total_Assets=='Nil']<-0
Candidate_Data$Total_Assets<-as.numeric(Candidate_Data$Total_Assets)
Candidate_Data$Liabilities<-as.numeric(Candidate_Data$Liabilities)

# Recheck Sample Data after cleanup
head(Candidate_Data)

##   Sno            Candidateâ^‡ Constituency               Party  Education
## 3   1                 A Hasiv    ARYA NAGAR                 BSP  12th Pass
## 4   2                A Wahid       GAINSARI                 IND  10th Pass
## 5   3 Aan Shikhar Shrivastava   GOSHAINGANJ Satya Shikhar Party   Graduate
## 6   4         Aaptab Urf Aftab    MUBARAKPUR    Islam Party Hind Illiterate
## 7   5              Aashi Gaur     KHATAULI             Lok Dal   Literate
## 8   6              Aashif Beg     BARKHERA                 IND  Not Given
##   Liabilities Criminal_Case Total_Assets
## 3     5846335             0     39424827
## 4           0             0        75106
## 5           0             0        41000
## 6           0             0        20000
## 7           0             0      3468543
## 8           0             0      1330000

# Check the Dataset Structure
str(Candidate_Data)

#========================================================================
# Merging the Election Data with MyNeta based on two Keys:
# Candidate Name and Constituency.
# The Data Merging is done outside of this program with the help of
# R Programming using "Levenshtein Distance" and Excel.
# The Merged csv file for UP State is used thereafter.
#========================================================================
#
Master <-read.csv("Master_AE17_12_MyNeta17_12.csv", header = TRUE)
names(Master)
```

```
##  [1] "Sr_No"                "AE17_ST_CODE"         "AE17_ST_NAME"
##  [4] "AE17_MONTH"           "AE17_YEAR"            "AE17_DIST_NAME"
##  [7] "AE17_AC_NO"           "AE17_AC_NAME"         "AE17_AC_TYPE"
## [10] "AE17_CAND_NAME"       "AE17_CAND_SEX"        "AE17_CAND_CATEGORY"
## [13] "AE17_CAND_AGE"        "AE17_PARTYABBRE"      "AE17_TOTVOTPOLL"
## [16] "AE17_POSITION"        "MN17_CRIMINAL_CASE"   "MN17_EDUCATION"
## [19] "MN17_TOT_ASSETS"      "MN17_TOT_LIABILITIES" "MN17_SEQ_NO"
## [22] "AE12_SEQ_NO"          "AE12_CAND_NAME"       "AE12_CAND_SEX"
## [25] "AE12_CAND_CATEGORY"   "AE12_CAND_AGE"        "AE12_PARTYABBRE"
## [28] "AE12_TOTVOTPOLL"      "AE12_POSITION"

str(Master)

## 'data.frame':    5307 obs. of  29 variables:
View(Master)
#
#==========================================================================
# Data Pre Processing
#==========================================================================
# Remove records where PARTYABBRE = NOTA.
# Position: Keep 1 and all <> 1 as 0.
# Set Criminal Cases to 0 for #N/A.
# Set Education to "Others" for #N/A.
# Set Total Assets to 0 for #N/A. and "Nil Rs"
# Set Liabilities to 0 for #N/A.
#==========================================================================
# Remove records with NOTA
nrow(Master)

## [1] 5307

Master <- Master[!(Master$AE17_PARTYABBRE == "NOTA"),]
nrow(Master)

## [1] 4900

Master$AE17_POSITION <-ifelse(Master$AE17_POSITION ==1,1,0)
Master$AE12_POSITION <-ifelse(Master$AE12_POSITION ==1,1,0)
levels(Master$MN17_CRIMINAL_CASE)[1] <- "0"
levels(Master$MN17_EDUCATION)[1] <- "Others"
levels(Master$MN17_TOT_ASSETS)[1] <- "0"
levels(Master$MN17_TOT_LIABILITIES)[1] <- "0"
levels(Master$AE12_TOTVOTPOLL)[1] <- "0"
# levels(Master$MN17_CRIMINAL_CASE)  # To Check levels of a given feature.
# write.csv(Master, file = "MasterTemp.csv")
str(Master)

## 'data.frame':    4900 obs. of  29 variables:
##  $ Sr_No               : int  1 2 3 4 6 7 8 9 10 11 ...
##  $ AE17_ST_CODE        : Factor w/ 1 level "S24": 1 1 1 1 1 1 1 1 1 1 ...
##  $ AE17_ST_NAME        : Factor w/ 1 level "Uttar Pradesh": 1 1 1 1 1 1 1 1
1 1 ...
names(Master)
```

```
##  [1] "Sr_No"                "AE17_ST_CODE"         "AE17_ST_NAME"        
##  [4] "AE17_MONTH"           "AE17_YEAR"            "AE17_DIST_NAME"      
##  [7] "AE17_AC_NO"           "AE17_AC_NAME"         "AE17_AC_TYPE"        
## [10] "AE17_CAND_NAME"       "AE17_CAND_SEX"        "AE17_CAND_CATEGORY"  
## [13] "AE17_CAND_AGE"        "AE17_PARTYABBRE"      "AE17_TOTVOTPOLL"     
## [16] "AE17_POSITION"        "MN17_CRIMINAL_CASE"   "MN17_EDUCATION"      
## [19] "MN17_TOT_ASSETS"      "MN17_TOT_LIABILITIES" "MN17_SEQ_NO"         
## [22] "AE12_SEQ_NO"          "AE12_CAND_NAME"       "AE12_CAND_SEX"       
## [25] "AE12_CAND_CATEGORY"   "AE12_CAND_AGE"        "AE12_PARTYABBRE"     
## [28] "AE12_TOTVOTPOLL"      "AE12_POSITION"
```

```r
#
#=======================================================================
# Variable Transformation / Feature Creation
#=======================================================================
Master$AE17_CAND_SEX <- ifelse(Master$AE17_CAND_SEX == "M", 1,
                          ifelse(Master$AE17_CAND_SEX == "F", 2,3))

Master$AE12_CAND_SEX <- ifelse(Master$AE12_CAND_SEX == "M", 1,
                               ifelse(Master$AE12_CAND_SEX == "F", 2,3))
# Convert Candidate Age and Criminal Case as Numeric:
Master$AE17_CAND_AGE<-as.numeric(as.character(Master$AE17_CAND_AGE))
Master$AE12_CAND_AGE<-as.numeric(as.character(Master$AE12_CAND_AGE))

## Warning: NAs introduced by coercion

Master$MN17_CRIMINAL_CASE<-as.numeric(as.character(Master$MN17_CRIMINAL_CASE))
Master$AE12_TOTVOTPOLL<-as.numeric(as.character(Master$AE12_TOTVOTPOLL))

# Creating Dummy Variables for Total Assets

Master$MN17_TOT_ASSETS <- as.numeric(as.character(Master$MN17_TOT_ASSETS))
Master$MN17_TOT_LIABILITIES <-
as.numeric(as.character(Master$MN17_TOT_LIABILITIES))

Master$FINANCIAL_STS <- ifelse(Master$MN17_TOT_ASSETS < 100000, "Poor",
                          ifelse(Master$MN17_TOT_ASSETS <
10000000,"Lakhpati","Crorepati"))

Master$FINANCIAL_STS <- as.factor(Master$FINANCIAL_STS)
levels(Master$FINANCIAL_STS)

## [1] "Crorepati" "Lakhpati"  "Poor"

Master$AE17_AGE_BKT <- ifelse(Master$AE17_CAND_AGE < 36, "25-35",
                            ifelse(Master$AE17_CAND_AGE < 46,"36-45",
                                   ifelse(Master$AE17_CAND_AGE < 56,"46-
55","55+")))
Master$AE17_AGE_BKT <- as.factor(Master$AE17_AGE_BKT)
#
Master$AE12_AGE_BKT <- ifelse(Master$AE12_CAND_AGE < 36, "25-35",
                              ifelse(Master$AE12_CAND_AGE < 46,"36-45",
                                     ifelse(Master$AE12_CAND_AGE < 56,"46-
55","55+")))
Master$AE12_AGE_BKT <- as.factor(Master$AE12_AGE_BKT)
```

```
#levels(Master$AE17_AGE_BKT)

str(Master)

names(Master)

##  [1] "Sr_No"                "AE17_ST_CODE"         "AE17_ST_NAME"
##  [4] "AE17_MONTH"           "AE17_YEAR"            "AE17_DIST_NAME"
##  [7] "AE17_AC_NO"           "AE17_AC_NAME"         "AE17_AC_TYPE"
## [10] "AE17_CAND_NAME"       "AE17_CAND_SEX"        "AE17_CAND_CATEGORY"
## [13] "AE17_CAND_AGE"        "AE17_PARTYABBRE"      "AE17_TOTVOTPOLL"
## [16] "AE17_POSITION"        "MN17_CRIMINAL_CASE"   "MN17_EDUCATION"
## [19] "MN17_TOT_ASSETS"      "MN17_TOT_LIABILITIES" "MN17_SEQ_NO"
## [22] "AE12_SEQ_NO"          "AE12_CAND_NAME"       "AE12_CAND_SEX"
## [25] "AE12_CAND_CATEGORY"   "AE12_CAND_AGE"        "AE12_PARTYABBRE"
## [28] "AE12_TOTVOTPOLL"      "AE12_POSITION"        "FINANCIAL_STS"
## [31] "AE17_AGE_BKT"         "AE12_AGE_BKT"

#
# Combine Parties who have not won a single seat into Others.

Master_Others <- Master[which(Master$AE17_PARTYABBRE != "ADAL"&
                                Master$AE17_PARTYABBRE != "BJP"&
                                Master$AE17_PARTYABBRE != "BSP"&
                                Master$AE17_PARTYABBRE != "INC"&
                                Master$AE17_PARTYABBRE != "IND"&
                                Master$AE17_PARTYABBRE != "NINSHAD"&
                                Master$AE17_PARTYABBRE != "RLD"&
                                Master$AE17_PARTYABBRE != "SBSP"&
                                Master$AE17_PARTYABBRE != "SP"),]
#
Master_Winner <- Master[which(Master$AE17_PARTYABBRE == "ADAL"|
                                Master$AE17_PARTYABBRE == "BJP"|
                                Master$AE17_PARTYABBRE == "BSP"|
                                Master$AE17_PARTYABBRE == "INC"|
                                Master$AE17_PARTYABBRE == "IND"|
                                Master$AE17_PARTYABBRE == "NINSHAD"|
                                Master$AE17_PARTYABBRE == "RLD"|
                                Master$AE17_PARTYABBRE == "SBSP"|
                                Master$AE17_PARTYABBRE == "SP"),]
#
Master_Others$AE17_PARTYABBRE <- "Others"

Master <- rbind(Master_Winner,Master_Others)

nrow(Master_Others)

## [1] 1829

nrow(Master_Winner)

## [1] 3071

nrow(Master)
```

```
## [1] 4900

View(Master)
#
#===========================================================================
# Data Visualization
#===========================================================================
# Education wise Candidates
Master %>%
  ggplot(aes(x=MN17_EDUCATION)) +
  geom_bar(aes(fill=MN17_EDUCATION))

#
# Category wise Candidates
Master %>%
  ggplot(aes(x=AE17_CAND_CATEGORY)) +
  geom_bar(aes(fill=AE17_CAND_CATEGORY))

#
# Gender wise Candidates
Master %>%
  ggplot(aes(x=AE17_CAND_SEX)) +
  geom_bar(aes(fill=AE17_CAND_SEX))

#
# Age Bracket wise Candidates
Master %>%
  ggplot(aes(x=AE17_AGE_BKT)) +
  geom_bar(aes(fill=AE17_AGE_BKT))

#
# Financial Status wise Candidates
Master %>%
  ggplot(aes(x=FINANCIAL_STS)) +
  geom_bar(aes(fill=FINANCIAL_STS))

#
# Party wise winning Candidates
Master %>%
  filter(AE17_POSITION == 1) %>%
  ggplot(aes(x=AE17_PARTYABBRE)) +
  geom_bar(aes(fill=AE17_PARTYABBRE))

#
# Party wise winning Candidates with Criminal Cases
Master %>%
  filter(AE17_POSITION == 1) %>%
  ggplot(aes(x=MN17_CRIMINAL_CASE)) +
  geom_bar(aes(fill=AE17_PARTYABBRE))

## Warning: position_stack requires non-overlapping x intervals

#
#===========================================================================
# Model Building
#===========================================================================
# We need parameters explaining the Resons behind 2017 Election
# Hence we are going to build the Model on entire data set.
#=======================================================================
# Check No. of Winners Vs Loosers
table(Master$AE17_POSITION)

##
##    0    1
## 4497  403

prop.table(table(Master$AE17_POSITION))

##
##         0         1
## 0.9177551 0.0822449

#
#=======================================================================
# Model Building - CART: to Predict Position (1- Winner, 0 - Looser)
#=======================================================================
#
Master_CT <- Master
# Setting the control paramter inputs for rpart.
r.ctrl = rpart.control(minsplit=100,
                       minbucket = 10,
                       cp = 0, xval = 10)

# Build the Model.
names(Master_CT)

##  [1] "Sr_No"                "AE17_ST_CODE"         "AE17_ST_NAME"
##  [4] "AE17_MONTH"           "AE17_YEAR"            "AE17_DIST_NAME"
##  [7] "AE17_AC_NO"           "AE17_AC_NAME"         "AE17_AC_TYPE"
## [10] "AE17_CAND_NAME"       "AE17_CAND_SEX"        "AE17_CAND_CATEGORY"
## [13] "AE17_CAND_AGE"        "AE17_PARTYABBRE"      "AE17_TOTVOTPOLL"
## [16] "AE17_POSITION"        "MN17_CRIMINAL_CASE"   "MN17_EDUCATION"
## [19] "MN17_TOT_ASSETS"      "MN17_TOT_LIABILITIES" "MN17_SEQ_NO"
## [22] "AE12_SEQ_NO"          "AE12_CAND_NAME"       "AE12_CAND_SEX"
## [25] "AE12_CAND_CATEGORY"   "AE12_CAND_AGE"        "AE12_PARTYABBRE"
## [28] "AE12_TOTVOTPOLL"      "AE12_POSITION"        "FINANCIAL_STS"
## [31] "AE17_AGE_BKT"         "AE12_AGE_BKT"

m1.CART <- rpart(formula = AE17_POSITION ~ .,
                 data = Master_CT[,-c(1:10,15,21:28,32)], method = "class",
                   control = r.ctrl)

m1.CART

## n= 4900
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 4900 403 0 (0.91775510 0.08224490)
##    2) AE17_PARTYABBRE=BSP,INC,IND,NINSHAD,RLD,SP,Others 4493  78 0
```

```
(0.98263966 0.01736034) *
##    3) AE17_PARTYABBRE=ADAL,BJP,SBSP 407  82 1 (0.20147420 0.79852580)
##      6) AE17_CAND_AGE< 31.5 11   5 1 (0.45454545 0.54545455) *
##      7) AE17_CAND_AGE>=31.5 396  77 1 (0.19444444 0.80555556)
##       14) MN17_CRIMINAL_CASE< 4.5 374  77 1 (0.20588235 0.79411765)
##         28) MN17_CRIMINAL_CASE>=2.5 16   7 0 (0.56250000 0.43750000) *
##         29) MN17_CRIMINAL_CASE< 2.5 358  68 1 (0.18994413 0.81005587) *
##       15) MN17_CRIMINAL_CASE>=4.5 22   0 1 (0.00000000 1.00000000) *

#
printcp(m1.CART)

##
## Classification tree:
## rpart(formula = AE17_POSITION ~ ., data = Master_CT[, -c(1:10,
##     15, 21:28, 32)], method = "class", control = r.ctrl)
##
## Variables actually used in tree construction:
## [1] AE17_CAND_AGE      AE17_PARTYABBRE    MN17_CRIMINAL_CASE
##
## Root node error: 403/4900 = 0.082245
##
## n= 4900
##
##          CP nsplit rel error  xerror     xstd
## 1 0.6029777      0   1.00000 1.00000 0.047721
## 2 0.0016543      1   0.39702 0.39702 0.030871
## 3 0.0000000      4   0.39206 0.42928 0.032056

dev.off()

## null device
##           1

plotcp(m1.CART)
#
# Plotting the Tree
fancyRpartPlot(m1.CART,
               uniform = TRUE,
               main = "Initial Tree",
               palettes = c("Blues", "Oranges"))

# Pruning the Tree
ptree<- prune(m1.CART, cp= 0.032 ,"CP")
printcp(ptree)

##
## Classification tree:
## rpart(formula = AE17_POSITION ~ ., data = Master_CT[, -c(1:10,
##     15, 21:28, 32)], method = "class", control = r.ctrl)
##
## Variables actually used in tree construction:
## [1] AE17_PARTYABBRE
##
## Root node error: 403/4900 = 0.082245
```

```
##
## n= 4900
##
##         CP nsplit rel error  xerror     xstd
## 1 0.60298      0   1.00000 1.00000 0.047721
## 2 0.03200      1   0.39702 0.39702 0.030871

fancyRpartPlot(ptree,
               uniform = TRUE,
               main = "Final Tree",
               palettes = c("Blues", "Oranges"))


# Measure Model Performance: We don't have a Training / Testing Data set.

Master_CT$predict.class <- predict(ptree, Master_CT, type="class")
Master_CT$predict.score <- predict(ptree, Master_CT, type="prob")
#
## deciling code
decile <- function(x){
  deciles <- vector(length=10)
  for (i in seq(0.1,1,.1)){
    deciles[i*10] <- quantile(x, i, na.rm=T)
  }
  return (
    ifelse(x<deciles[1], 1,
           ifelse(x<deciles[2], 2,
                  ifelse(x<deciles[3], 3,
                         ifelse(x<deciles[4], 4,
                                ifelse(x<deciles[5], 5,
                                       ifelse(x<deciles[6], 6,
                                              ifelse(x<deciles[7], 7,
                                                     ifelse(x<deciles[8], 8,

ifelse(x<deciles[9], 9, 10
                                                            ))))))))))
}



## deciling
Master_CT$deciles <- decile(Master_CT$predict.score[,2])

## Ranking code
##install.packages("data.table")
##install.packages("scales")
library(data.table)
library(scales)
tmp_DT = data.table(Master_CT)
rank <- tmp_DT[, list(
  cnt = length(AE17_POSITION),
  cnt_resp = sum(AE17_POSITION),
  cnt_non_resp = sum(AE17_POSITION == 0)) ,
  by=deciles][order(-deciles)]
```

```
rank$rrate <- round(rank$cnt_resp / rank$cnt,4);
rank$cum_resp <- cumsum(rank$cnt_resp)
rank$cum_non_resp <- cumsum(rank$cnt_non_resp)
rank$cum_rel_resp <- round(rank$cum_resp / sum(rank$cnt_resp),4);
rank$cum_rel_non_resp <- round(rank$cum_non_resp / sum(rank$cnt_non_resp),4);
rank$ks <- abs(rank$cum_rel_resp - rank$cum_rel_non_resp) * 100;
rank$rrate <- percent(rank$rrate)
rank$cum_rel_resp <- percent(rank$cum_rel_resp)
rank$cum_rel_non_resp <- percent(rank$cum_rel_non_resp)

View(rank)

#install.packages("ROCR")
#install.packages("ineq")
library(ROCR)

library(ineq)
pred <- prediction(Master_CT$predict.score[,2], Master_CT$AE17_POSITION)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)

gini = ineq(Master_CT$predict.score[,2], type="Gini")

with(Master_CT, table(AE17_POSITION, predict.class))

##                predict.class
## AE17_POSITION    0    1
##               0 4415   82
##               1   78  325

auc

## [1] 0.8941086

KS

## [1] 0.7882172

Gini

## [1] 0.7233904

View(rank)

## Syntax to get the node path
tree.path <- path.rpart(ptree, node = c(2,3))

##
##  node number: 2
##    root
##    AE17_PARTYABBRE=BSP,INC,IND,NINSHAD,RLD,SP,Others
##
##  node number: 3
##    root
##    AE17_PARTYABBRE=ADAL,BJP,SBSP

#
#===========================================================================
# Model Building - RANDOM FOREST: to Predict Position
#===========================================================================
#install.packages("randomForest")
library(randomForest)
names(Master)

##  [1] "Sr_No"                 "AE17_ST_CODE"          "AE17_ST_NAME"
##  [4] "AE17_MONTH"            "AE17_YEAR"             "AE17_DIST_NAME"
##  [7] "AE17_AC_NO"            "AE17_AC_NAME"          "AE17_AC_TYPE"
## [10] "AE17_CAND_NAME"        "AE17_CAND_SEX"         "AE17_CAND_CATEGORY"
## [13] "AE17_CAND_AGE"         "AE17_PARTYABBRE"       "AE17_TOTVOTPOLL"
## [16] "AE17_POSITION"         "MN17_CRIMINAL_CASE"    "MN17_EDUCATION"
## [19] "MN17_TOT_ASSETS"       "MN17_TOT_LIABILITIES" "MN17_SEQ_NO"
## [22] "AE12_SEQ_NO"           "AE12_CAND_NAME"        "AE12_CAND_SEX"
## [25] "AE12_CAND_CATEGORY"    "AE12_CAND_AGE"         "AE12_PARTYABBRE"
## [28] "AE12_TOTVOTPOLL"       "AE12_POSITION"         "FINANCIAL_STS"
## [31] "AE17_AGE_BKT"          "AE12_AGE_BKT"

str(Master)

Master_RF <- Master

head(Master_RF)

# Random Forest

RF = randomForest( as.factor(AE17_POSITION) ~ .,
                   data = Master_RF[, -c(1:10,14,15,21:28,32)],
                   ntree = 500, mtry = 7, nodesize = 100,
                   importance = TRUE )
#
names(Master_RF)

##  [1] "Sr_No"                 "AE17_ST_CODE"          "AE17_ST_NAME"
##  [4] "AE17_MONTH"            "AE17_YEAR"             "AE17_DIST_NAME"
##  [7] "AE17_AC_NO"            "AE17_AC_NAME"          "AE17_AC_TYPE"
## [10] "AE17_CAND_NAME"        "AE17_CAND_SEX"         "AE17_CAND_CATEGORY"
## [13] "AE17_CAND_AGE"         "AE17_PARTYABBRE"       "AE17_TOTVOTPOLL"
## [16] "AE17_POSITION"         "MN17_CRIMINAL_CASE"    "MN17_EDUCATION"
## [19] "MN17_TOT_ASSETS"       "MN17_TOT_LIABILITIES" "MN17_SEQ_NO"
## [22] "AE12_SEQ_NO"           "AE12_CAND_NAME"        "AE12_CAND_SEX"
## [25] "AE12_CAND_CATEGORY"    "AE12_CAND_AGE"         "AE12_PARTYABBRE"
## [28] "AE12_TOTVOTPOLL"       "AE12_POSITION"         "FINANCIAL_STS"
## [31] "AE17_AGE_BKT"          "AE12_AGE_BKT"


print(RF)

##
## Call:
```

```
##  randomForest(formula = as.factor(AE17_POSITION) ~ AE17_CAND_SEX +
AE17_CAND_CATEGORY + AE17_CAND_AGE + MN17_CRIMINAL_CASE +      MN17_EDUCATION
+ MN17_TOT_ASSETS + MN17_TOT_LIABILITIES +      FINANCIAL_STS + AE17_AGE_BKT,
data = Master_RF, ntree = 500,      mtry = 7, nodesize = 100, importance =
TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 8.22%
## Confusion matrix:
##      0 1 class.error
## 0 4497 0           0
## 1  403 0           1

#
dev.off()

plot(RF, main="")
legend("topright", c("OOB", "0", "1"), text.col=1:6, lty=1:3, col=1:3)
title(main="Error Rates Random Forest AE-2017 UP data")

rf_err_rate <- as.data.frame(RF$err.rate)

rf_err_rate$ID <- seq.int(nrow(rf_err_rate))

#Checking minimum error rate


#Storing min try

min_tree<-min(rf_err_rate[which(rf_err_rate$OOB==min(rf_err_rate$OOB)),]$ID)


# List the importance of the variables.
impVar <- round(randomForest::importance(RF), 2)
# impVar[order(impVar[,3], decreasing=TRUE),]
# impVar[order(impVar[,4], decreasing=TRUE),]
impVar[order(impVar[,1], decreasing=TRUE),]

##                          0     1 MeanDecreaseAccuracy MeanDecreaseGini
## MN17_TOT_LIABILITIES 11.14 -2.92                11.83            15.94
## MN17_CRIMINAL_CASE    9.07  6.88                11.52            10.37
## MN17_EDUCATION        8.38  5.31                10.64            24.68
## MN17_TOT_ASSETS       7.91 -5.41                 7.80            66.54
## FINANCIAL_STS         6.37 -5.20                 6.57             7.44
## AE17_CAND_CATEGORY    2.41 -4.26                 1.33             2.91
## AE17_CAND_SEX         1.92 -0.84                 1.61             1.18
## AE17_AGE_BKT         -0.76  1.25                -0.47             2.49
## AE17_CAND_AGE        -4.17  6.02                -1.68            26.01

varImpPlot(RF)

# Tuning Random Forest
names(Master_RF)
```

```
tune_rf_model <- tuneRF(x = Master_RF[,-c(1:10,14:16,21:28,32)],
              y=as.factor(Master_RF$AE17_POSITION),
              mtryStart = 3, # Approx. Sqrt of Tot. No of Variables
              ntreeTry=min_tree,
              stepFactor = 1.5,
              improve = 0.0001,
              trace=TRUE,
              plot = TRUE,
              doBest = TRUE,
              nodesize = 10,
              importance=TRUE
)

## mtry = 3  OOB error = 9.12%
## Searching left ...
## mtry = 2     OOB error = 8.57%
## 0.06040268 1e-04
## Searching right ...
## mtry = 4     OOB error = 8.82%
## -0.02857143 1e-04

#
# Scoring syntax
Master_RF$predict.class <- predict(tune_rf_model, Master_RF[,-
c(1:10,14:16,21:28,32)], type="class")
Master_RF$predict.score <- predict(tune_rf_model, Master_RF[,-
c(1:10,14:16,21:28,32)], type="prob")

#
#Checking Variable Importance
varImpPlot(tune_rf_model)

#
## deciling code
decile <- function(x){
  deciles <- vector(length=10)
  for (i in seq(0.1,1,.1)){
    deciles[i*10] <- quantile(x, i, na.rm=T)
  }
  return (
    ifelse(x<deciles[1], 1,
           ifelse(x<deciles[2], 2,
                  ifelse(x<deciles[3], 3,
                         ifelse(x<deciles[4], 4,
                                ifelse(x<deciles[5], 5,
                                       ifelse(x<deciles[6], 6,
                                              ifelse(x<deciles[7], 7,
                                                     ifelse(x<deciles[8], 8,

ifelse(x<deciles[9], 9, 10
                                                                  ))))))))))
}

class(Master_RF$predict.score)
```

```
## [1] "matrix" "votes"

## deciling
Master_RF$deciles <- decile(Master_RF$predict.score[,2])
# View(Master_RF)

## Ranking code
##install.packages("data.table")
##install.packages("scales")
library(data.table)
library(scales)
tmp_DT = data.table(Master_RF)
rank <- tmp_DT[, list(
  cnt = length(AE17_POSITION),
  cnt_resp = sum(AE17_POSITION),
  cnt_non_resp = sum(AE17_POSITION == 0)) ,
  by=deciles][order(-deciles)]
rank$rrate <- round(rank$cnt_resp / rank$cnt,4);
rank$cum_resp <- cumsum(rank$cnt_resp)
rank$cum_non_resp <- cumsum(rank$cnt_non_resp)
rank$cum_rel_resp <- round(rank$cum_resp / sum(rank$cnt_resp),4);
rank$cum_rel_non_resp <- round(rank$cum_non_resp / sum(rank$cnt_non_resp),4);
rank$ks <- abs(rank$cum_rel_resp - rank$cum_rel_non_resp) * 100;
rank$rrate <- percent(rank$rrate)
rank$cum_rel_resp <- percent(rank$cum_rel_resp)
rank$cum_rel_non_resp <- percent(rank$cum_rel_non_resp)

View(rank)

#install.packages("ROCR")
#install.packages("ineq")
library(ROCR)
library(ineq)
pred <- prediction(Master_RF$predict.score[,2], Master_RF$AE17_POSITION)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)

gini = ineq(Master_RF$predict.score[,2], type="Gini")

with(Master_RF, table(AE17_POSITION, predict.class))

##                predict.class
## AE17_POSITION    0    1
##              0 4497    0
##              1  379   24

auc

## [1] 0.9077218

KS
```

```
## [1] 0.7190352

gini

## [1] 0.8049511

# View(rank)
#========================================================================
# Model Building - Linear Regression: to Predict Votes Share
#========================================================================
# Votes Share Prediction Model.
# We shall use the same Master file, which has been merged with MyNeta and
# AE-2012 Results for Utter Pradesh.
# We found total 802 Candidates, which will be the input records.
# While building the Vote Share Prediction Model, we shall see the impact of
# Incumbency (Anti Incumbency on Vote Share)
#============================================================================

Master_12_17 <- Master[which(Master$AE12_PARTYABBRE != "#N/A"),]
nrow(Master_12_17)

## [1] 802

names(Master_12_17)

##  [1] "Sr_No"                "AE17_ST_CODE"         "AE17_ST_NAME"
##  [4] "AE17_MONTH"           "AE17_YEAR"            "AE17_DIST_NAME"
##  [7] "AE17_AC_NO"           "AE17_AC_NAME"         "AE17_AC_TYPE"
## [10] "AE17_CAND_NAME"       "AE17_CAND_SEX"        "AE17_CAND_CATEGORY"
## [13] "AE17_CAND_AGE"        "AE17_PARTYABBRE"      "AE17_TOTVOTPOLL"
## [16] "AE17_POSITION"        "MN17_CRIMINAL_CASE"   "MN17_EDUCATION"
## [19] "MN17_TOT_ASSETS"      "MN17_TOT_LIABILITIES" "MN17_SEQ_NO"
## [22] "AE12_SEQ_NO"          "AE12_CAND_NAME"       "AE12_CAND_SEX"
## [25] "AE12_CAND_CATEGORY"   "AE12_CAND_AGE"        "AE12_PARTYABBRE"
## [28] "AE12_TOTVOTPOLL"      "AE12_POSITION"        "FINANCIAL_STS"
## [31] "AE17_AGE_BKT"         "AE12_AGE_BKT"

View(Master_12_17)

# Visualization

AE17_Vote_Poll_Rel <-subset(Master_12_17[,c(11,13,15:17,19:20,28,29)])
str(AE17_Vote_Poll_Rel)

cor(AE17_Vote_Poll_Rel)

##                      AE17_CAND_SEX AE17_CAND_AGE AE17_TOTVOTPOLL
## AE17_CAND_SEX          1.000000000   -0.02145643      0.01451856
## AE17_CAND_AGE         -0.021456428    1.00000000      0.29208691
## AE17_TOTVOTPOLL        0.014518564    0.29208691      1.00000000
## AE17_POSITION          0.005905108    0.18395591      0.66632915
## MN17_CRIMINAL_CASE    -0.056781836    0.02186201      0.13333941
## MN17_TOT_ASSETS       -0.015498515    0.02955731      0.12883149
## MN17_TOT_LIABILITIES  -0.016835755    0.02406122      0.15864243
## AE12_TOTVOTPOLL        0.012279345    0.27780718      0.71857489
## AE12_POSITION         -0.041841681    0.19484776      0.36275876
##                        AE17_POSITION MN17_CRIMINAL_CASE MN17_TOT_ASSETS
## AE17_CAND_SEX            0.005905108        -0.05678184     -0.01549852
## AE17_CAND_AGE            0.183955913         0.02186201      0.02955731
## AE17_TOTVOTPOLL          0.666329151         0.13333941      0.12883149
## AE17_POSITION            1.000000000         0.10918734      0.09887042
## MN17_CRIMINAL_CASE       0.109187342         1.00000000      0.08418050
## MN17_TOT_ASSETS          0.098870422         0.08418050      1.00000000
## MN17_TOT_LIABILITIES     0.146863200         0.26236249      0.34147958
## AE12_TOTVOTPOLL          0.306871753         0.09530881      0.14479708
## AE12_POSITION            0.101609618         0.04738144      0.11839467
##                        MN17_TOT_LIABILITIES AE12_TOTVOTPOLL AE12_POSITION
## AE17_CAND_SEX                   -0.01683575      0.01227934   -0.04184168
## AE17_CAND_AGE                    0.02406122      0.27780718    0.19484776
## AE17_TOTVOTPOLL                  0.15864243      0.71857489    0.36275876
## AE17_POSITION                    0.14686320      0.30687175    0.10160962
## MN17_CRIMINAL_CASE               0.26236249      0.09530881    0.04738144
## MN17_TOT_ASSETS                  0.34147958      0.14479708    0.11839467
## MN17_TOT_LIABILITIES             1.00000000      0.11000423    0.07576649
## AE12_TOTVOTPOLL                  0.11000423      1.00000000    0.65157286
## AE12_POSITION                    0.07576649      0.65157286    1.00000000

correlation <- cor(AE17_Vote_Poll_Rel)
#install.packages("corrplot")
library(corrplot)

## corrplot 0.84 loaded

dev.off()

## null device
##           1

corrplot(correlation, method="circle")
#

attach(AE17_Vote_Poll_Rel)

OLS1 <- lm(AE17_TOTVOTPOLL~., data = AE17_Vote_Poll_Rel)
summary(OLS1)

# Removing the Variables having least impact.
# Gender, Criminal Cases, Total Assets and Liabilities
#
OLS2 <- lm(AE17_TOTVOTPOLL~., data = AE17_Vote_Poll_Rel[,-c(1,5,6,7)])
summary(OLS2)

#
# AE12_POSITION is negatively impacting the Total Votes Polled in 2017
# Indicating Anti-Incumbency Effect.

#
# Check Multi collinearity effect.
# VIF = 1      - Not Correlated
# VIF > 1 < 5 - Moderately Correlated
# VIF > 5      - Highly Correlated
```

```
#
library(car)

vif(OLS1)

##        AE17_CAND_SEX        AE17_CAND_AGE        AE17_POSITION
##             1.008840             1.099138             1.160738
##    MN17_CRIMINAL_CASE      MN17_TOT_ASSETS MN17_TOT_LIABILITIES
##             1.086607             1.149549             1.219868
##      AE12_TOTVOTPOLL        AE12_POSITION
##             2.006545             1.783739

# As can be seen, VIF is just slightly greater than 1, hence we can
# Conclude that our variables are moderately correlated.

#=============================================================================
#
#                              T H E - E N D
#
#=============================================================================
```