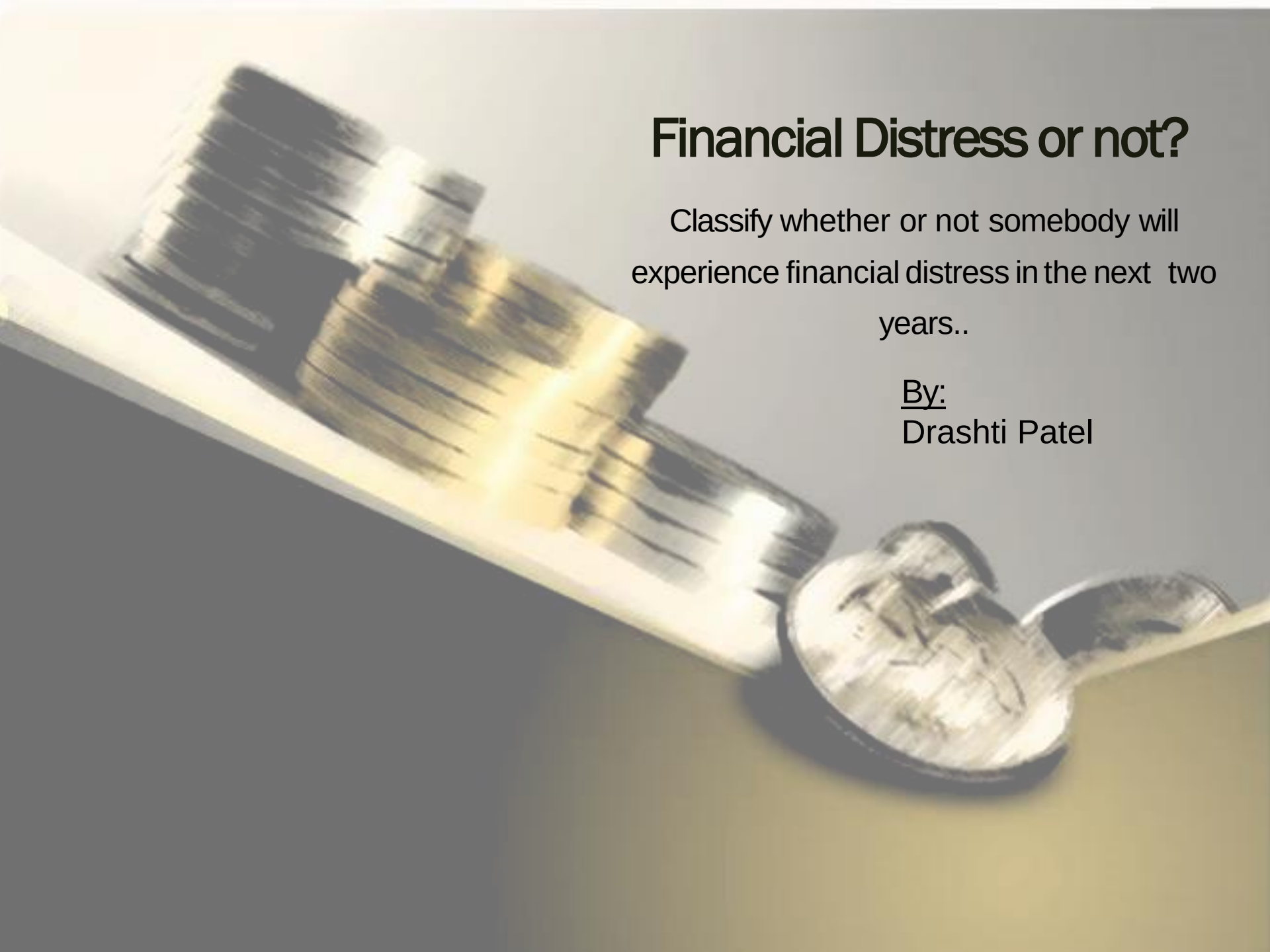


Financial Distress or not?

Classify whether or not somebody will experience financial distress in the next two years..

By:
Drashti Patel

Project Overview

- Based on an individual's credit data, classify whether or not somebody will experience financial distress in the next two years…
- Credit scoring algorithms make a guess at the chances of default
- The purpose of building our model is to make it accessible to the borrowers
- Banks play a crucial role in market economies
- Data Source: https://www.kaggle.com/c/GiveMeSomeCredit/data

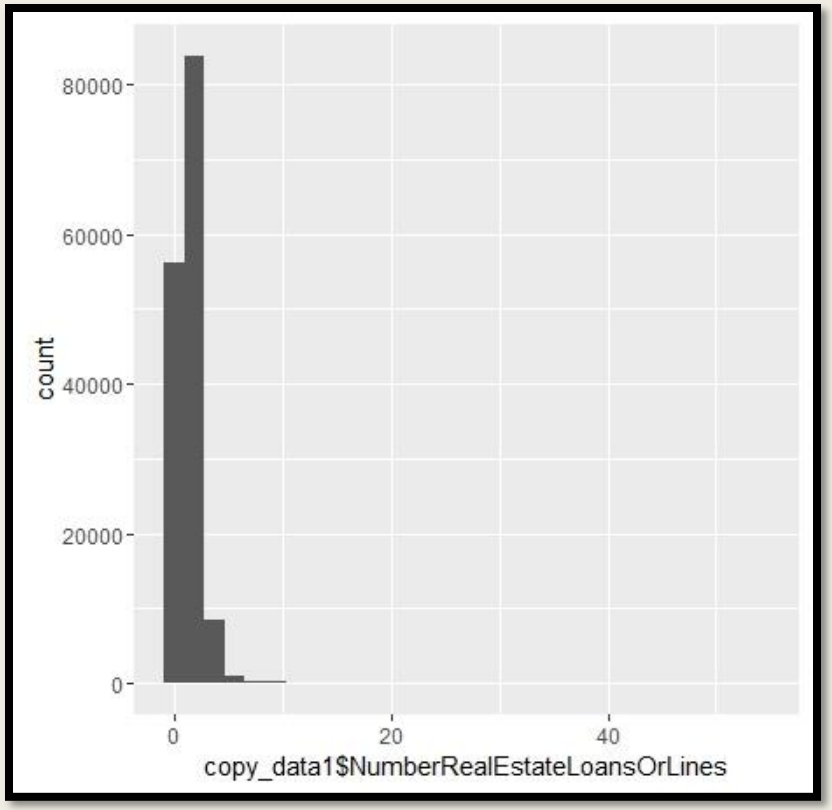
Data Dictionary

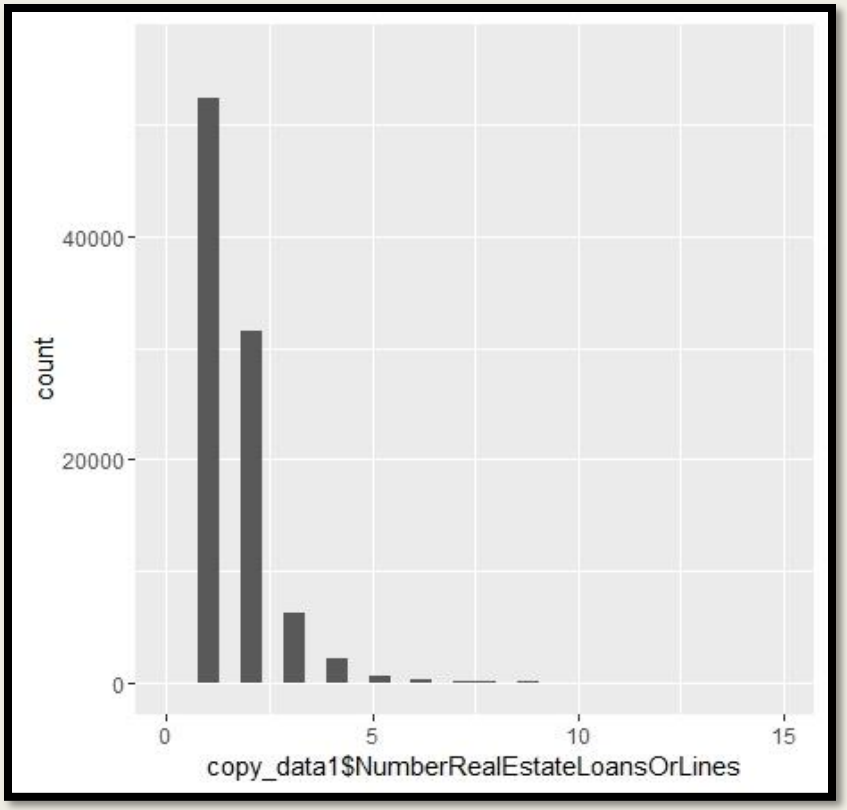
Variable Name	Description	Type
SeriousDlqin2yrs	**Person experienced 90 days past due delinquency or**	**Y/N**
RevolvingUtilizationOfUnsecuredLines	Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits	percentage
age	Age of borrower in years	integer
NumberOfTime30-59DaysPastDueNotWorse	Number of times borrower has been 30-59 days past due b	integer
DebtRatio	Monthly debt payments, alimony,living costs divided by m	percentage
MonthlyIncome	Monthly income	real
NumberOfOpenCreditLinesAndLoans	Number of Open loans (installment like car loan or mortga	integer
NumberOfTimes90DaysLate	Number of times borrower has been 90 days or more past	integer
NumberRealEstateLoansOrLines	Number of mortgage and real estate loans including home	integer
NumberOfTime60-89DaysPastDueNotWorse	Number of times borrower has been 60-89 days past due b	integer
NumberOfDependents	Number of dependents in family excluding themselves (sp	integer

150,000 records in dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	SeriousDl	Revolving	age	NumberO	DebtRatio	MonthlyIr	NumberO	NumberO	NumberR	NumberO	NumberOfDependents		
2	1	0.766127	45	2	0.802982	9120	13	0	6	0	2		
3	0	0.957151	40	0	0.121876	2600	4	0	0	0	1		
4	0	0.65818	38	1	0.085113	3042	2	1	0	0	0		
5	0	0.23381	30	0	0.03605	3300	5	0	0	0	0		
6	0	0.907239	49	1	0.024926	63588	7	0	1	0	0		
7	0	0.213179	74	0	0.375607	3500	3	0	1	0	1		
8	0	0.305682	57	0	5710	NA	8	0	3	0	0		
9	0	0.754464	39	0	0.20994	3500	8	0	0	0	0		
10	0	0.116951	27	0	46	NA	2	0	0	0	NA		
11	0	0.189169	57	0	0.606291	23684	9	0	4	0	2		
12	0	0.644226	30	0	0.309476	2500	5	0	0	0	0		
13	0	0.018798	51	0	0.531529	6501	7	0	2	0	2		
14	0	0.010352	46	0	0.298354	12454	13	0	2	0	2		
15	1	0.964673	40	3	0.382965	13700	9	3	1	1	2		
16	0	0.019657	76	0	477	0	6	0	1	0	0		

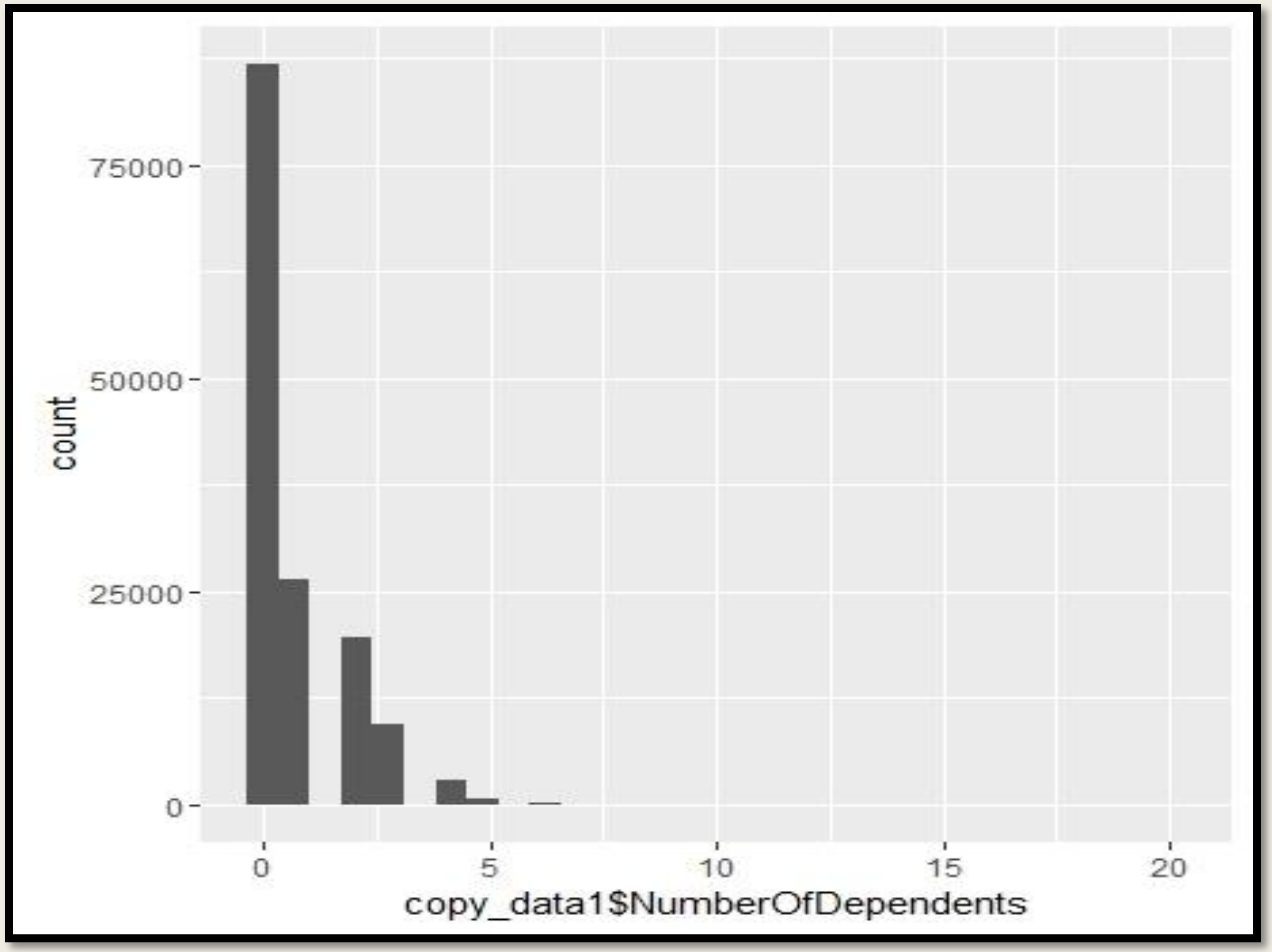
Summary and Qplot of Data Columns






```
> summary(copy_data1$NumberRealEstateLoansOrLines)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   1.000   1.018   2.000  54.000
```

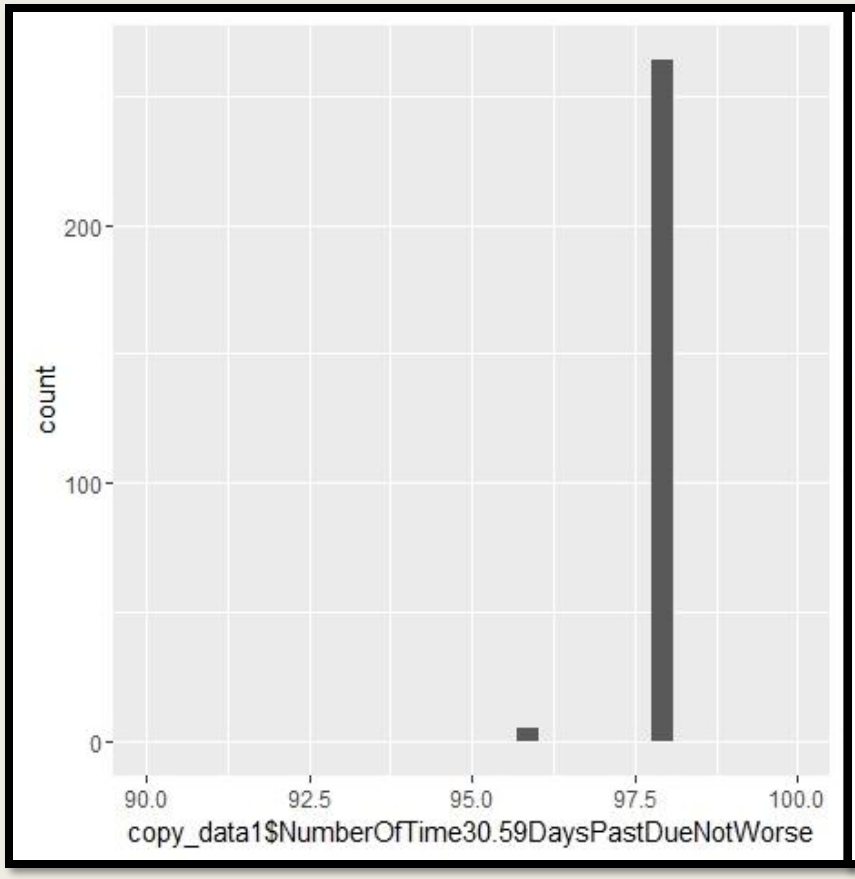
Summary and Qplot of Data Columns

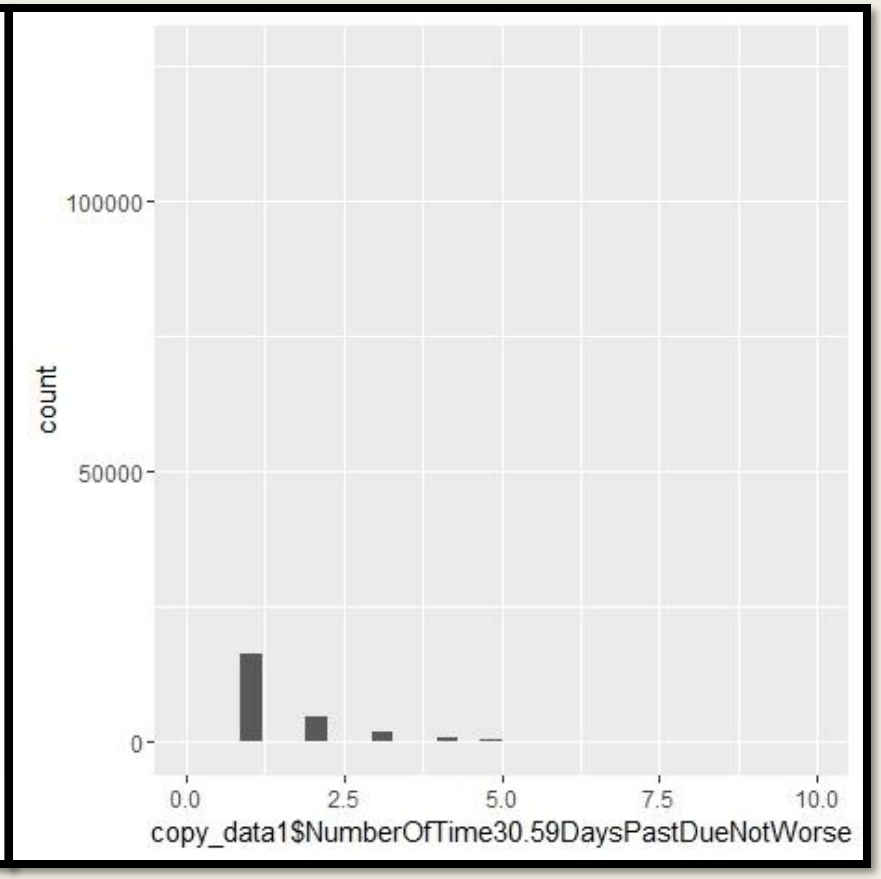



```
> summary(copy_data1$NumberOfDependents)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  0.000   0.000   0.000   0.757   1.000  20.000    3924
```

Summary and Qplot of Data Columns

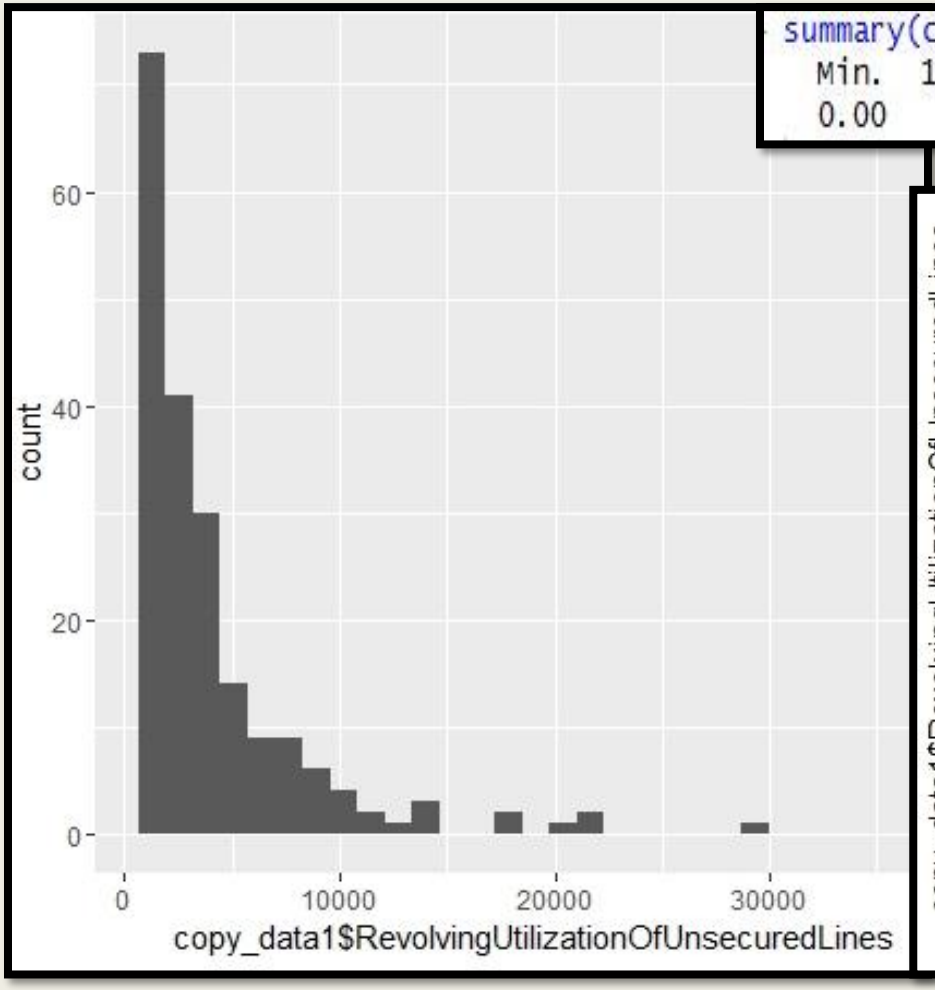
```
> summary(copy_data1$NumberOfTime30.59DaysPastDueNotWorse)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   0.000   0.421   0.000  98.000
```

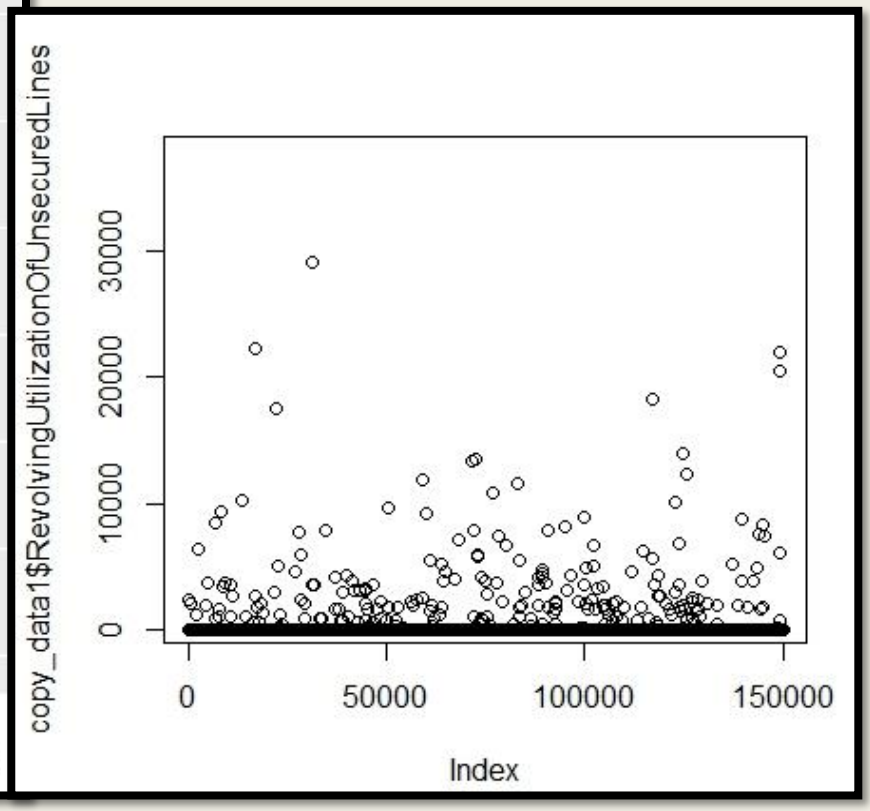




Summary and Boxplot of Data Columns

```
summary(copy_data1$RevolvingUtilizationOfUnsecuredLines)
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
   0.00     0.03     0.15     6.05     0.56 50710.00
```

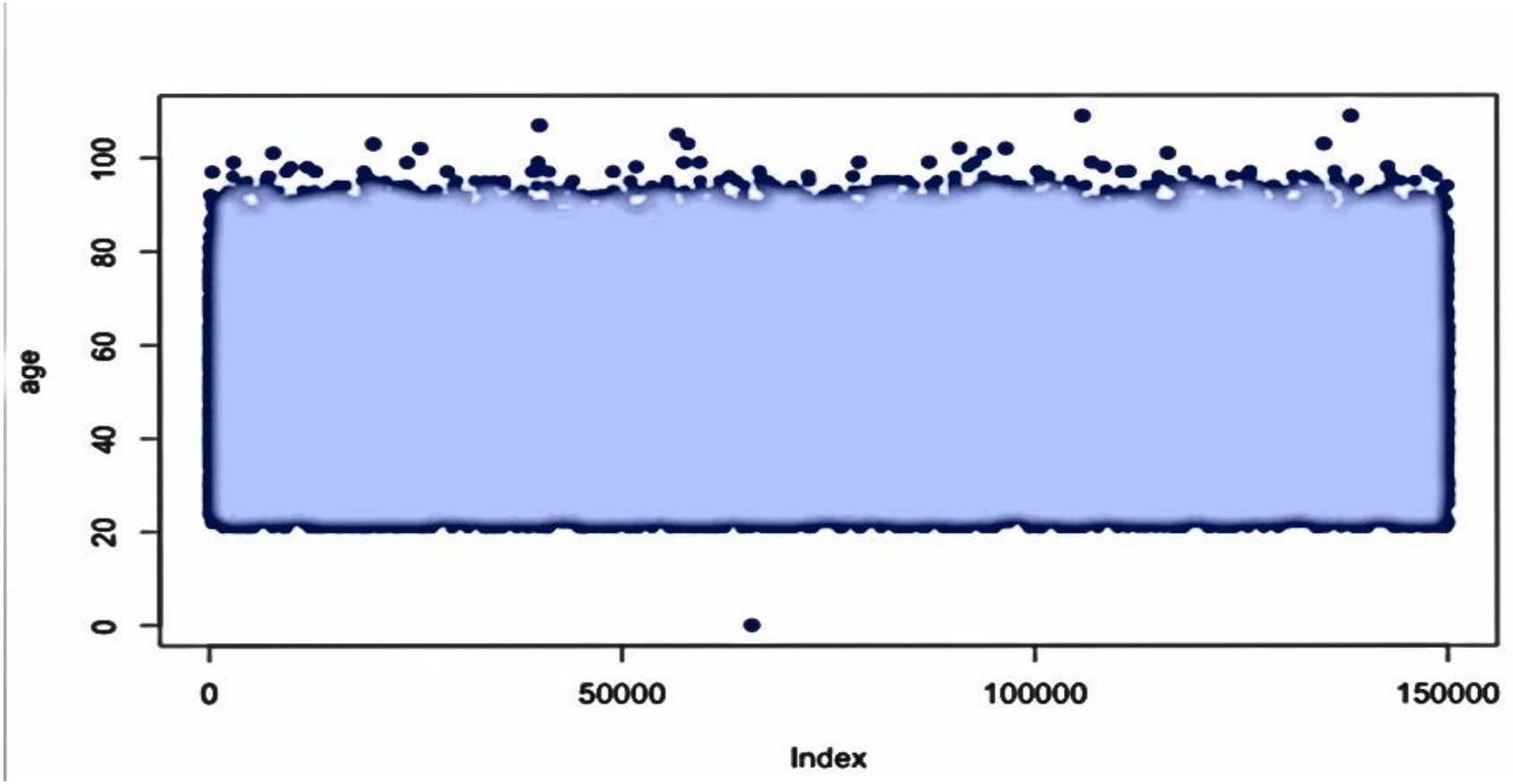




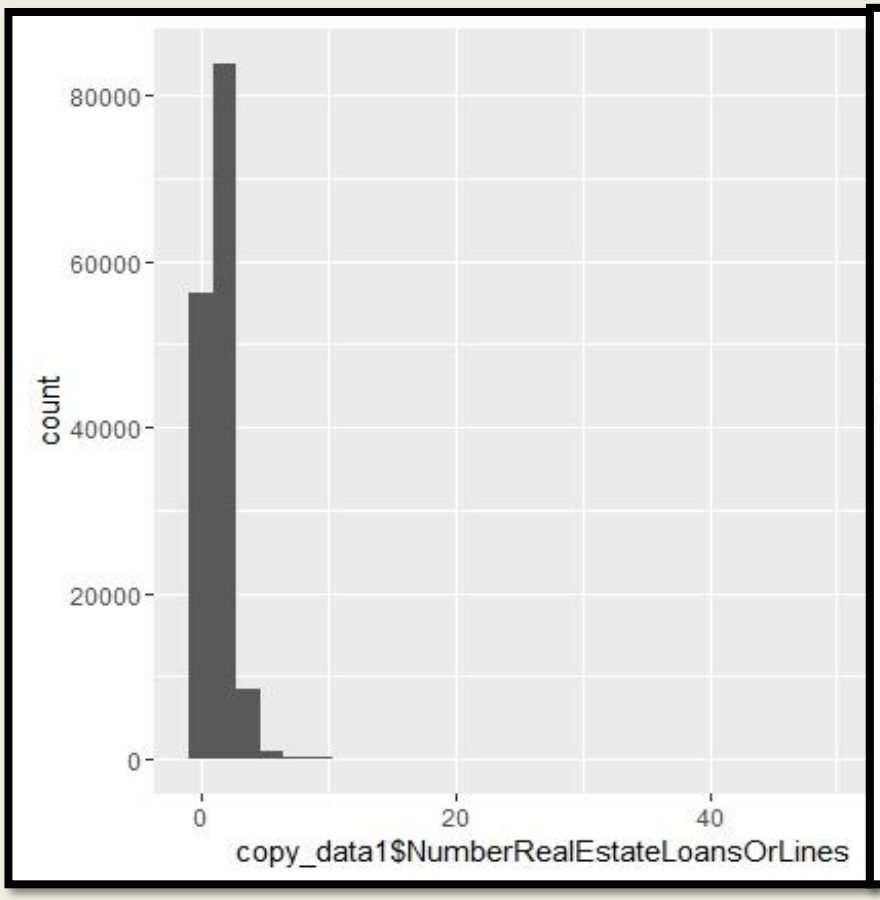
Data Exploration

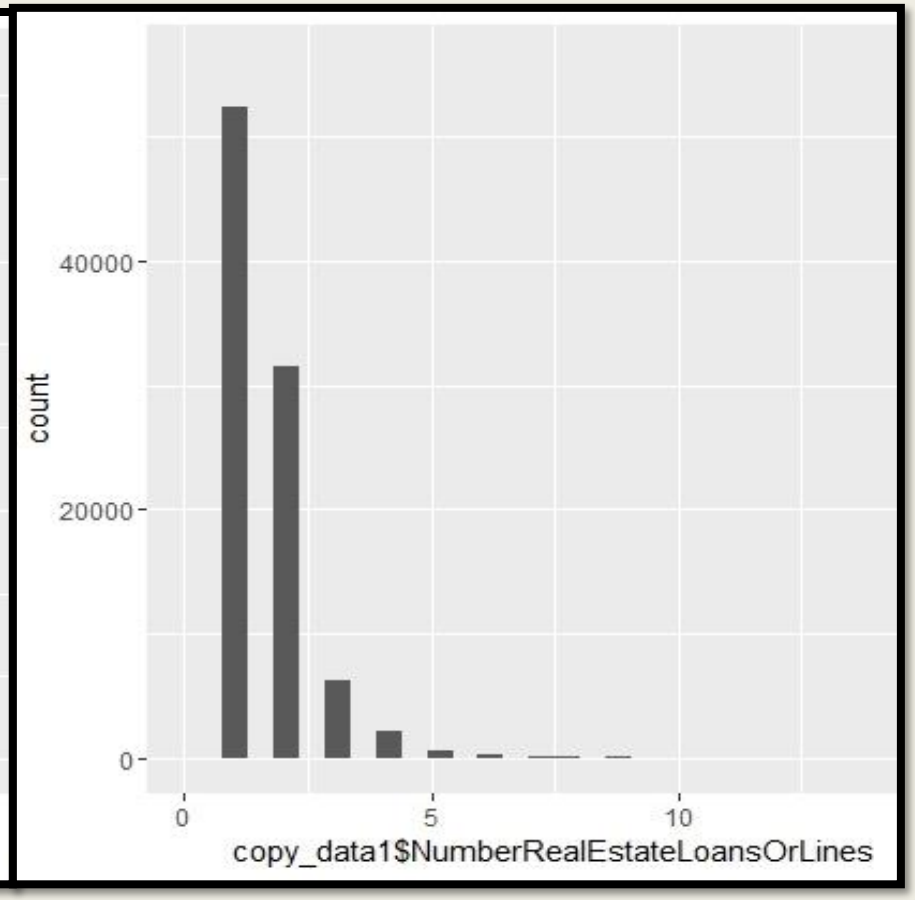
Scatter Plot of Age:

```
> summary(copy_data1$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    41.0    52.0    52.3    63.0   109.0
```



```
> summary(copy_data1$NumberRealEstateLoansOrLines)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   1.000   1.018   2.000  54.000
```



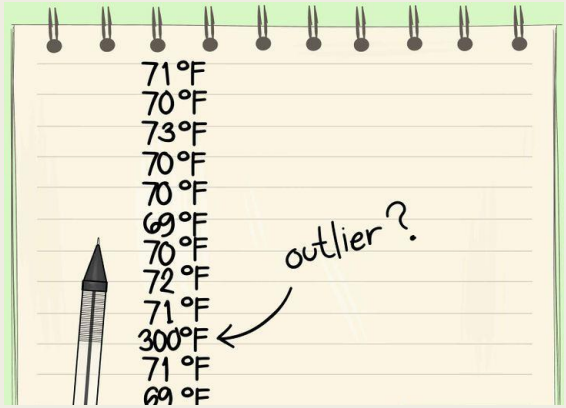


Outliers:

Finding Outliers Numerically

- The Interquartile range (IQR) is a measure of variability that represents the spread of the middle 50% of the data

  IQR= Q3 - Q1
- A data value is an outlier if: it is located 1.5 (IQR) or more below Q1, or it is located 1.5 (IQR) or more above Q3.



- Age =0 (1 record) → Replaced with Median
- Revolving Utilization >3 (292 records) → Deleted
- Monthly Income=0 → Replaced with 1
- Debt Ratio = 0 (4016 records) → Deleted
- Number of Real Estate Loans =54 (1 record) → Replaced with Median
- NumberOfTime60.89DaysPastDue, NumberOfTimes90DaysLate, NumberOfTime30.59DaysPastDue = 98 (269 rows) → No Action

Missing Data

- 29,731 Records with Monthly Income missing
- 3,924 Records with Number of Dependents missing

➢ Handling Missing Data

- Delete rows with missing values
- Replace missing values by 1 or Median
- Predict missing values using KNN's Accuracy
- For Number of Dependents, we replaced the Missing values with 0.

Prediction of Missing Values

- Implemented KNN to predict Monthly Income
- Firstly replacing with 1,
  - ➢ Using K=3 Error Rate= 8.11%
  - ➢ Using K=10, Error Rate= 6.81%
  - ➢ Using K=50, Error Rate= 6.76%
- Replacing with Median,
  - ➢ Using K=3 Error Rate= 8.81%
  - ➢ Using K=10, Error Rate= 6.49%
  - ➢ Using K=50, Error Rate= 6.46%
- Replacing Missing values with **Median** resulted in 6.49% error rate
- Replacing Missing values with **1** resulted in 6.81% error rate

Addition of a new field

DebtRatio	MonthlyIncome
0.802982129	9120
0.121876201	2600
0.085113375	3042
0.036049682	3300
0.024925695	63588
0.375606969	3500
5710	NA
0.209940017	3500
46	NA
0.606290901	23684
0.30947621	2500
0.53152876	6501
0.298354075	12454
0.382964747	13700
477	0
0.209891754	11362
2058	NA
0.18827406	8800
0.527887839	3280
0.065868263	333
0.430046338	12300

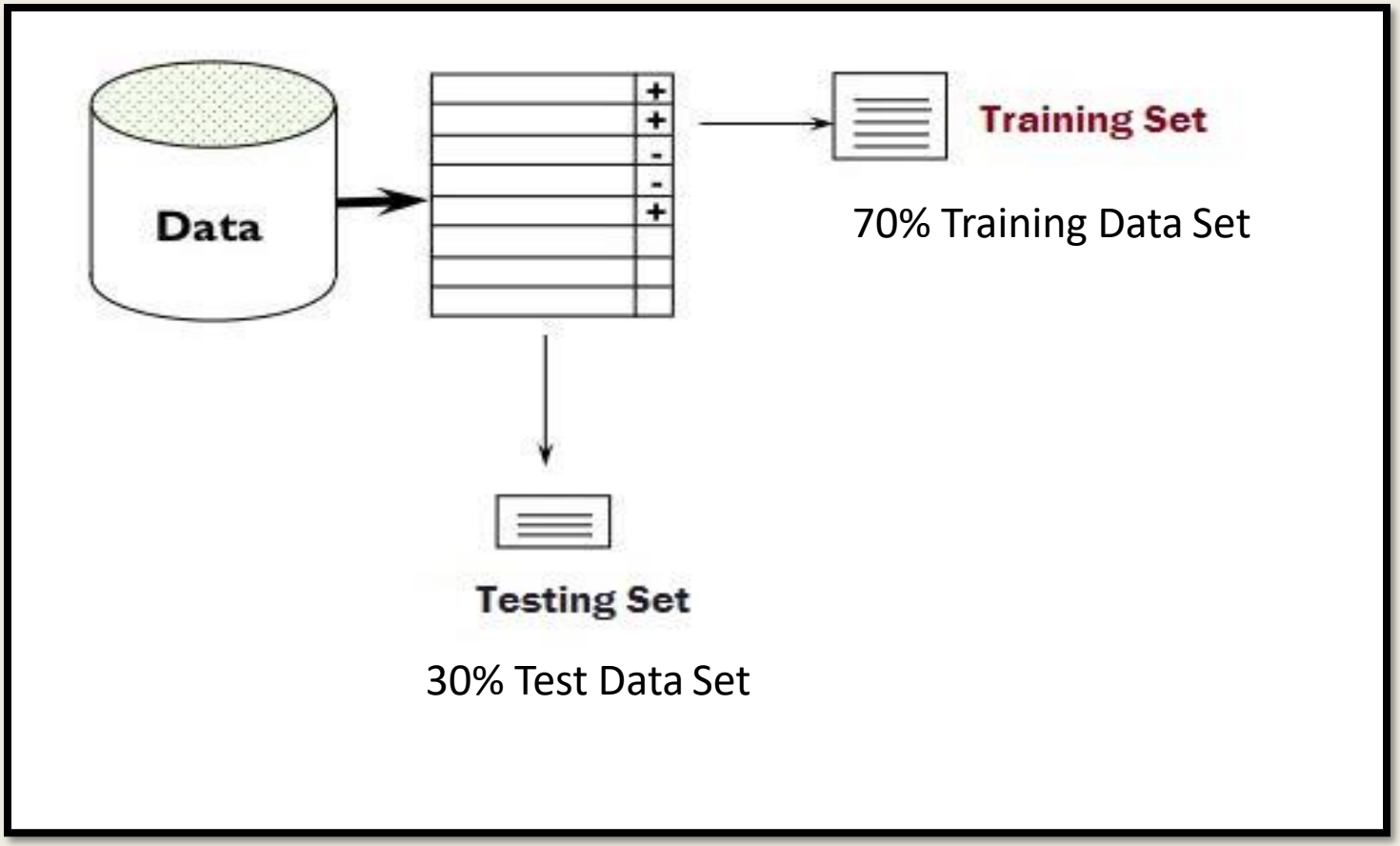
Expenditure = Debt Ratio * Monthly Income

- Not a good idea to predict or even replace monthly income with "Mean"
- Large values of Debt Ratio where Monthly Income is not available
- To implement the above formula correctly, replace 'NA' and '0' values for Monthly Income with '1' & 'Median'.
- Afterwards we removed Debt Ratio and Monthly Income columns from our dataset
- Added Expenditure column to our dataset

expenditure
7323.197016
316.8781226
258.9148868
118.9639506
1584.975094
1314.624392
30834000
734.7900595
248400
14359.3937
773.690525
3455.468469
3715.70165
5246.617034
477
2384.790109
11113200
1656.811728
1731.472112
21.93413158
5289.569957

Split Data into Train and Test sets

- Uniform Division of Data

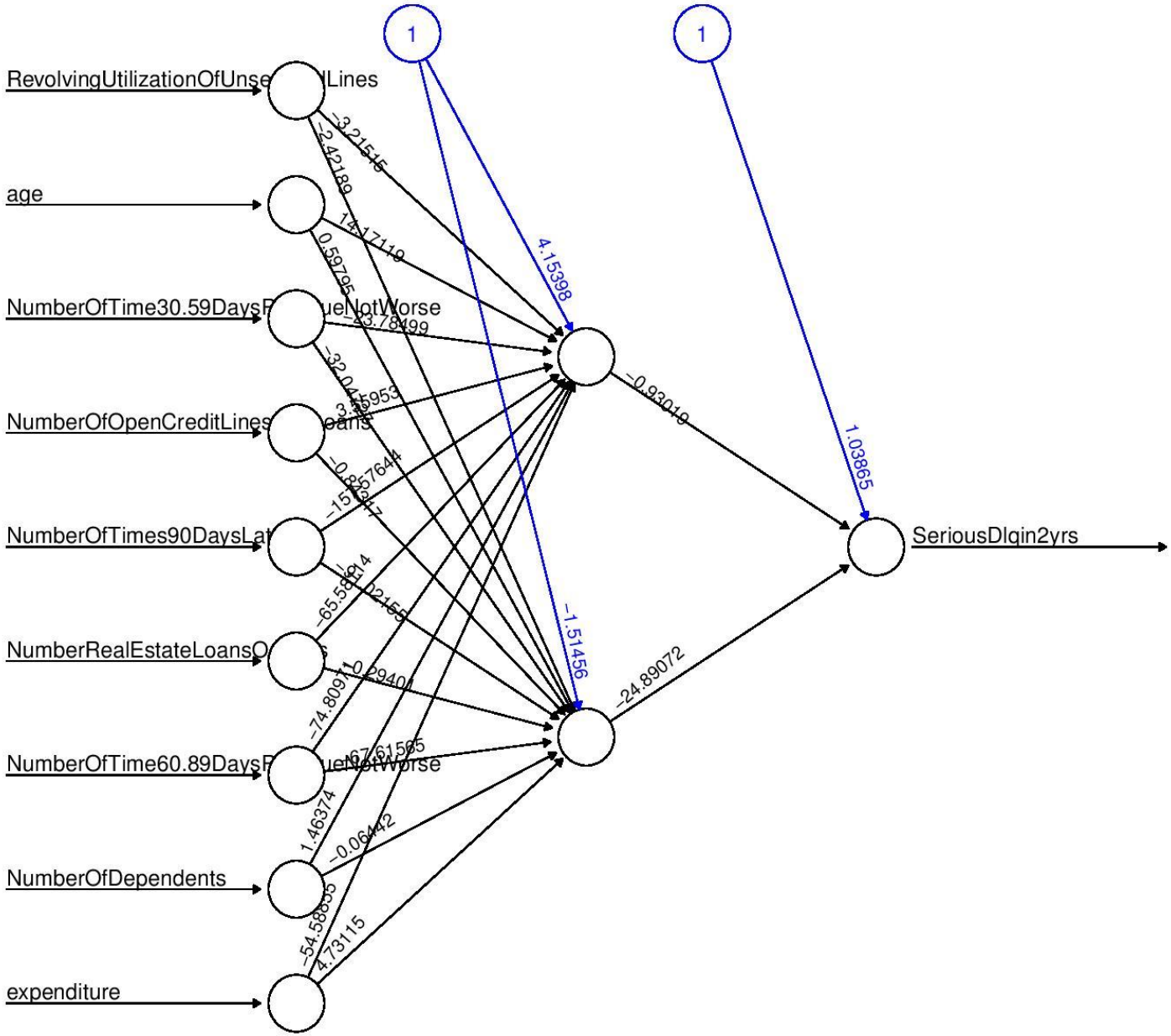


Classification using ANN

- We tried to implement the Artificial Neural Network.
- Runtime: 3 hours for only 10 seeds

Output:

```
> Accuracy <- 1- Error_Rate
> Accuracy
[1] 0.8213057
```



Error: 26017.340394 Steps: 75691

Classification using KNN

➢ K=03

Accuracy:	Error rate:
92.73%	For 1:88.19%

Actual

predict_knn_k03	0	1
0	40255	2495
1	595	334

Predicted

➢ K=10

Accuracy:	Error rate:
93.53%	For 1: 97.91%

Actual

predict_knn_k10	0	1
0	40663	2650
1	187	179

Predicted

➢ K=50

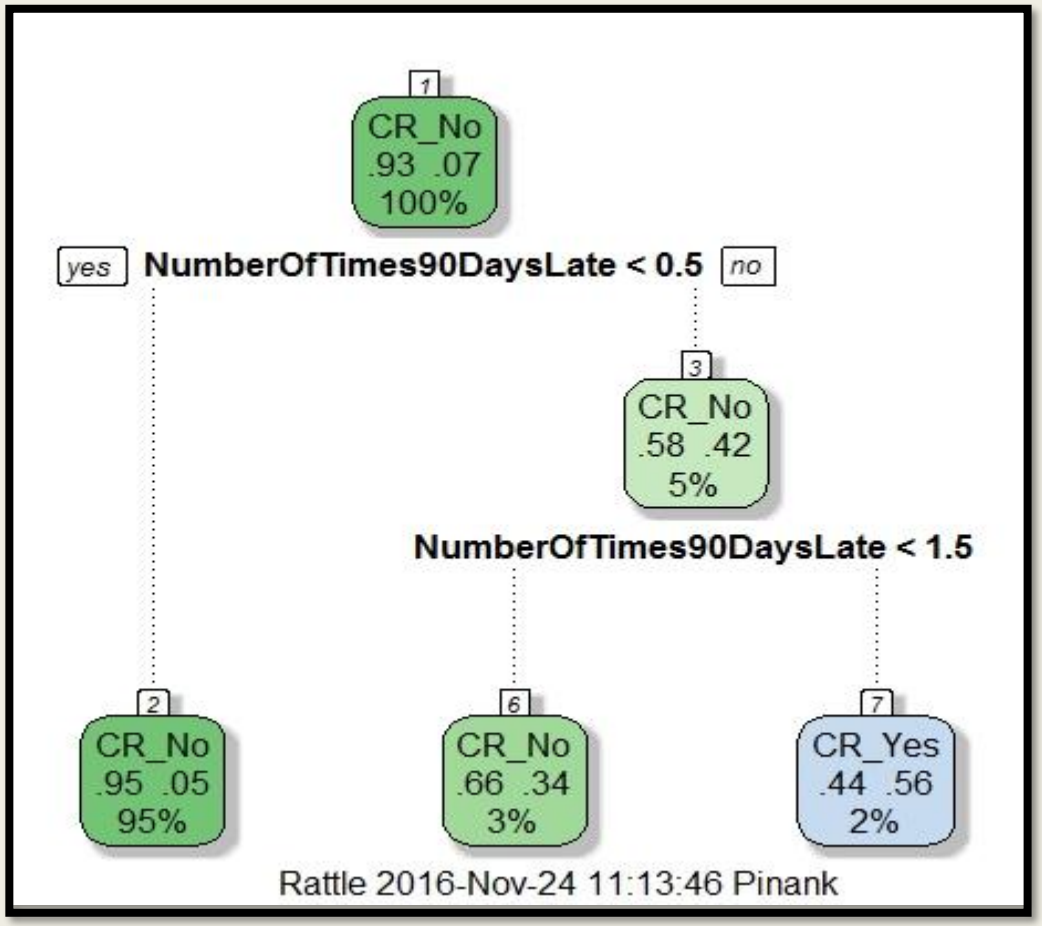
Accuracy:	Error rate:
93.50%	For 1: 93.67%

Actual

predict_knn_k50	0	1
0	40794	2770
1	56	59

Predicted

Classification using CART



Actual

	CR_No	CR_Yes
CR_No	40411	2454
CR_Yes	344	470

Accuracy:	Error rate:
93.59%	For 1: 83.92%

Classification using C5.0

Predicted \ Actual	y = 0	y = 1
x = 0	40416	2427
x = 1	339	497

Accuracy:	Error rate:
93.67%	For 1: 83.00%

- Even after cleaning the data the error rate is very high for 1's. So, it is definitely the imbalanced data with minority class of 1's.
- Illusion of high accurate model

How to deal with "Imbalanced" data??

Data Balancing using Sampling:

- The False Positive ratio is unacceptable and the accuracy is just an illusion with the model.
- We have tried to balance the data using Sampling procedure by applying **Under-Sampling & Over-Sampling** using SMOTE.
- We have under-sampled 60% of 0's related data and over-sampled 40% of 1's data.

Overall Data and probability of the table before balancing it.

```
table(trainSplit$target)
#0     1
#67947  4851
prop.table(table(trainSplit$target))
#0          1
#0.93336355 0.06663645
```

Overall Data and probability of the table After balancing it.

```
> table(trainSplit$target)

    0     1
28320 14160
> prop.table(table(trainSplit$target)) #overall Probability of the Table

        0         1
0.6666667 0.3333333
```

- Sampling with 60 - 40 ratio is the best suitable for our dataset.

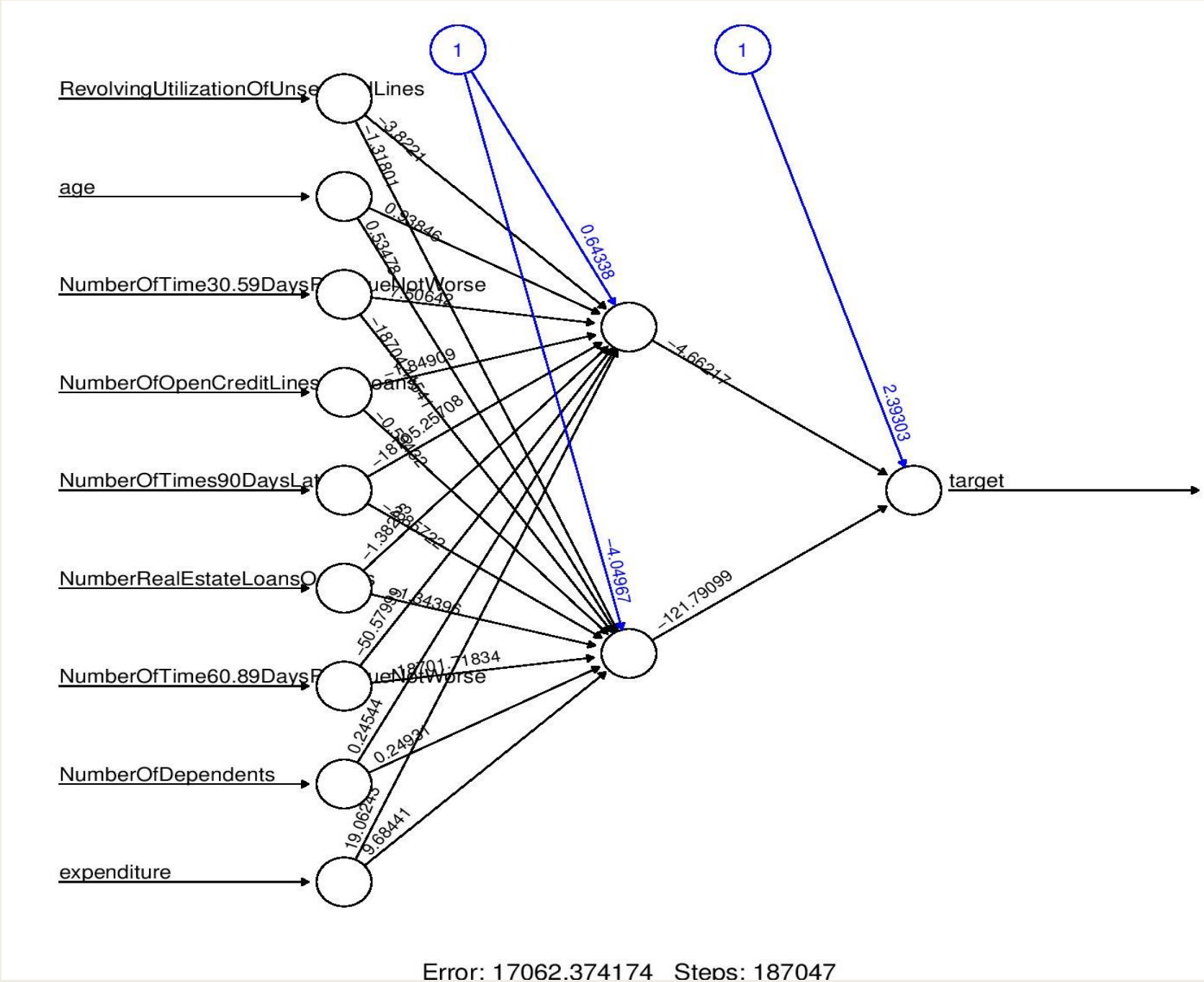
```
#Call: for 200 & 300
# roc.default(response = testSplit$target, predictor = pred)

#Data: pred in 67991 controls (testSplit$target 0) < 4806 cases (testSplit$target 1).
#Area under the curve: 0.8315
```

Classification using ANN (Balanced Data)

- We tried to implement the Artificial Neural Network.
- Runtime: again 3.5 hours for only 10 seeds

Output:

```
> Accuracy <- 1- Error_rate
> Accuracy
[1] 0.6091986
```



Error: 17062.374174 Steps: 187047

Classification using KNN (Balanced Data)

➢ K=03

Actual

```
predict_knn_k03    0    1
Predicted       0 7515  914
                1 1249 3420
> accuracy <- (7515+3420)/(7515+914+1249+3420)
> Accuracy
[1] 0.8700565
```

- Error rate:
- For 1: 21.1%
- Accuracy: **87.00%**

➢ K=10

Actual

```
predict_knn_k10    0    1
Predicted       0 7560 1417
                1 1204 2917
> Accuracy<- (7560+2917)/(7560+2917+1204+1217)
> Accuracy
[1] 0.8122965
```

- Error rate:
- For 1: 32.69%
- Accuracy: 81.23%

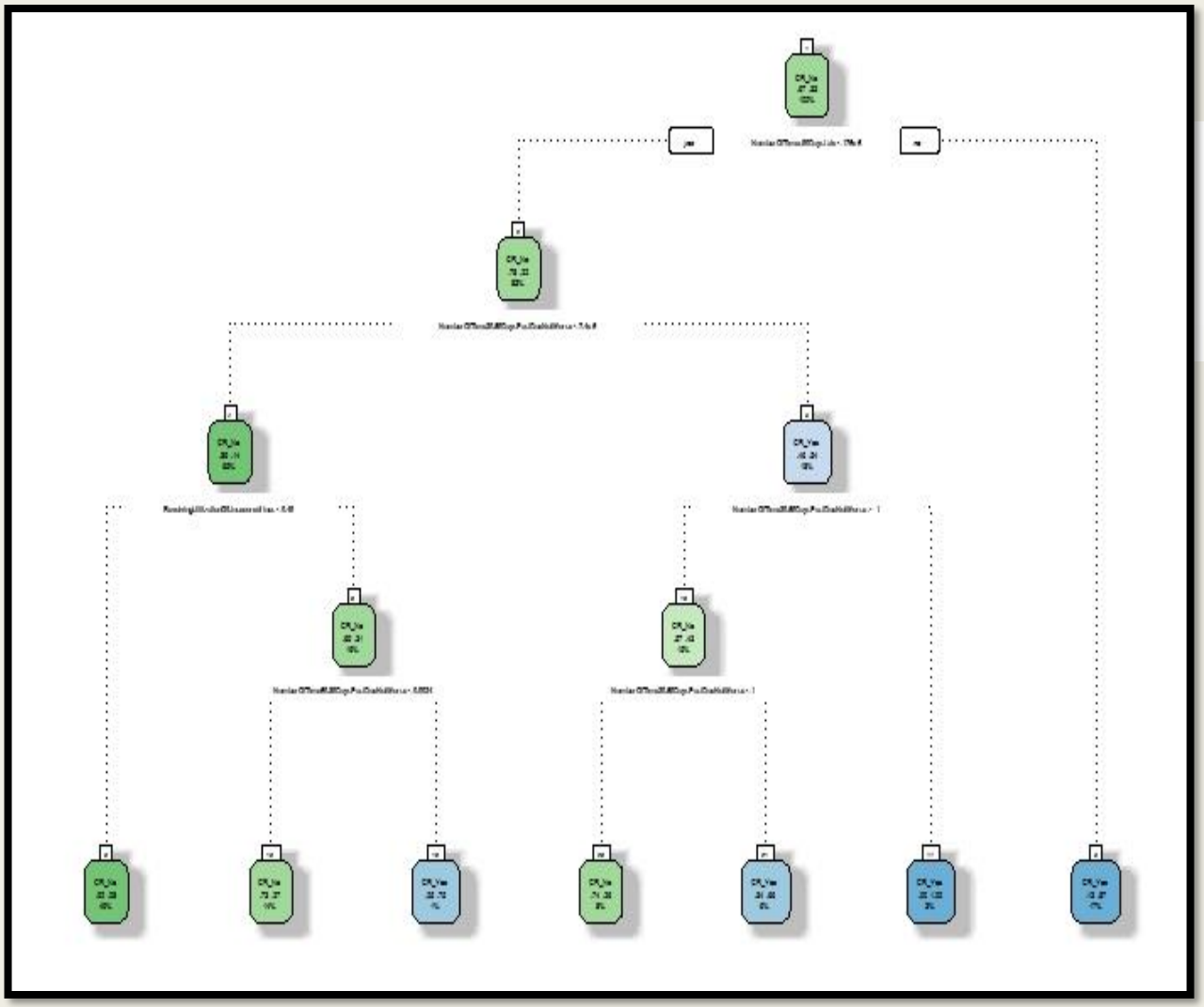
➢ K=50

Actual

```
predict_knn_k50    0    1
Predicted       0 7612 1845
                1 1152 2489
> Accuracy<- (7612+2489)/(7612+1845+1152+2489)
> Accuracy
[1] 0.7711864
```

- Error Rate:
- For 1: 42.57%
- Accuracy: 77.119%

Classification using CART (Balanced Data)



Actual

	CR_No	CR_Yes
CR_No	8072	1320
CR_Yes	591	3115

- Error Rate:
- For 1: 29.76%
- **Accuracy: 85.41%**

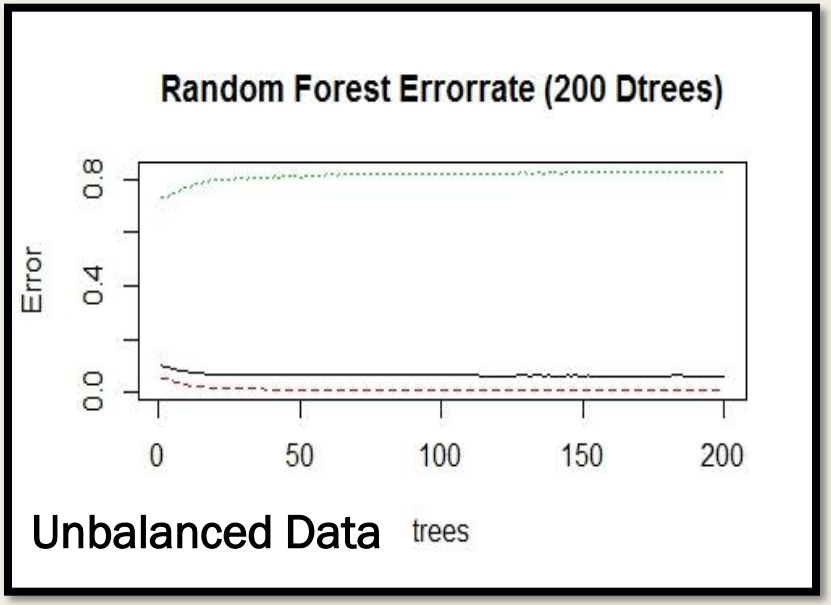
Classification using C5.0

	y	Actual
x	1	2
1	8309	1371
2	354	3064

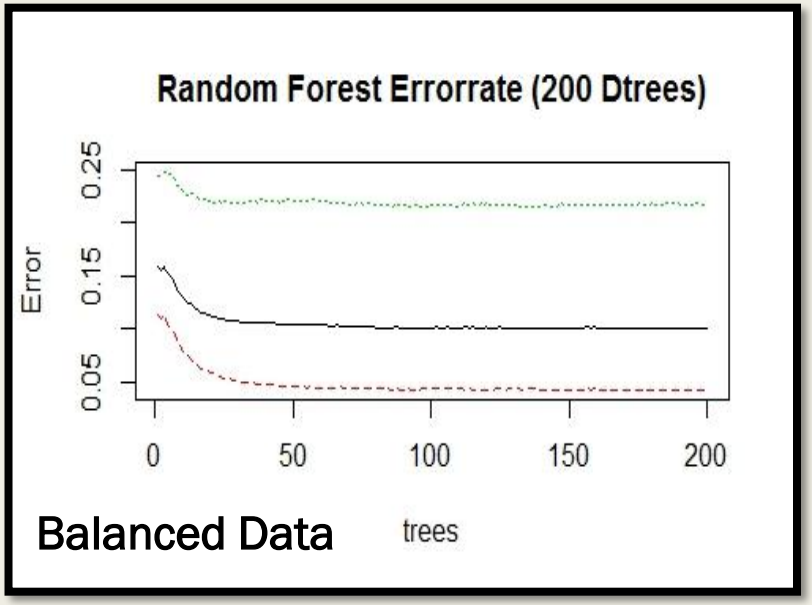
- Error Rate:
- For 1: 30.91%
- **Accuracy: 86.83%**

- After Balancing the data, the best method is **KNN with 87% accuracy** and **21% error-rate** which is 80% less error-rate compare to unbalanced data with 96-97% error rate.

Classification using Random Forest



Unbalanced Data



Balanced Data

Prediction	Reference 0	1
0	95046	179
1	0	6584

Accuracy:	Error rate:
98.2%	2.64%

Prediction	Reference 1	2
1	20372	33
2	0	10176

Accuracy:	Error rate:
98.73%	0.32%

Overall Comparison:

Unbalanced Data

Algorithm	Accuracy	False Positive
KNN = 3	92.73%	88.19%
KNN = 10	93.53%	97.91%
KNN = 50	93.50%	93.67%
CART	93.59%	83.92%
C5.0	93.67%	83.00%
Random Forest	**98.20%**	**02.64%**

Balanced Data

Algorithm	Accuracy	False Positive
KNN = 3	87.00%	21.10%
KNN = 10	81.23%	32.69%
KNN = 50	77.12%	42.57%
CART	85.41%	29.76%
C5.0	86.83%	30.91%
Random Forest	**98.73%**	**00.32%**

Conclusions:

- Best Method For our Problem is Random Forest.
  1. *Least Percent Error*
  2. *Easy to Implement*
  3. *Best Fit for Classification Problems*
- Cleaning the data takes usually the most time (at least 70% of the time)
- Our team spent at least 80% of the time cleaning the data, which takes more effort than implementing the algorithm
- Outlier must be analyzed (not just ignored or removed) because they may be relevant to the dataset sometimes.

Thank you!