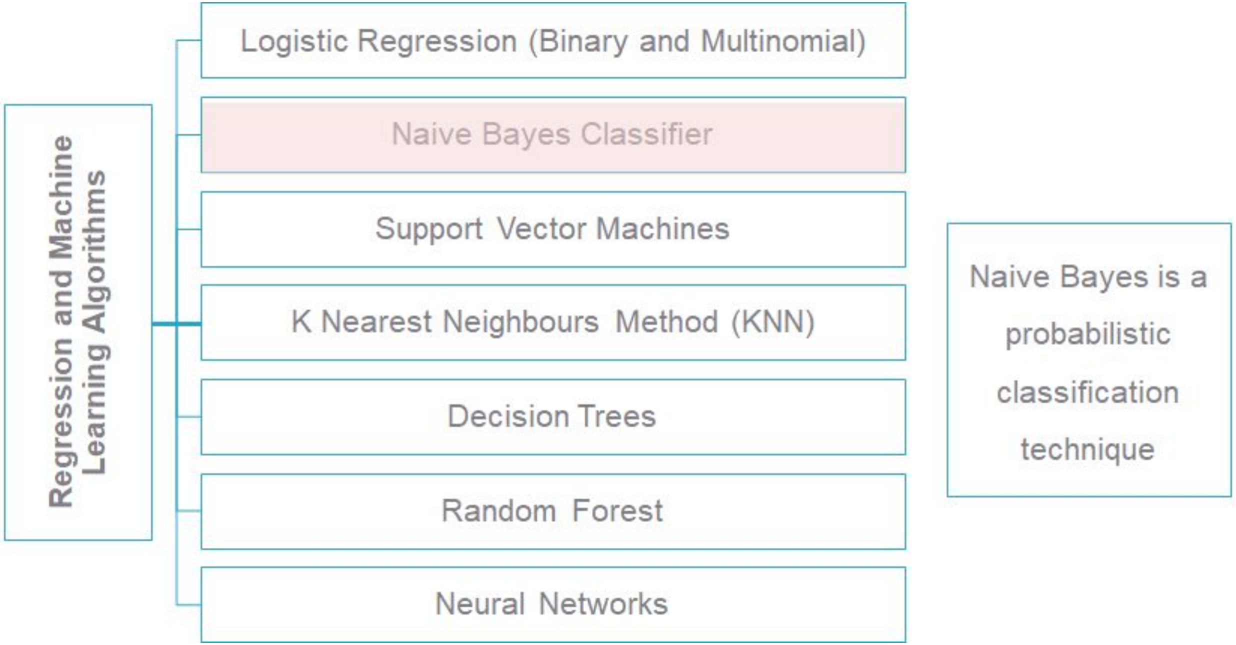
Naive Bayes Classifier I

Classifier Based on Bayes' Theorem

Contents

Classification Methods

Apart from logistic regression, several types of machine learning algorithms are effective in classification and prediction.



About Naive Bayes Classifier

- Simple probabilistic classifier based on Bayes Theorem.
- It can be used as an alternative method to logistic regression (Binary or Multinomial).
- It assumes conditional independence among the predictors.
- It is particularly suited when the dimensionality of the inputs is high.

Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

Conditional Probability

The conditional probability of an event B is the probability that event B will occur given the knowledge that an event A has already occurred. This probability is written as $P(B|A)$.

- If A and B are independent events then

$$P(B|A) = P(B)$$

- An unbiased die, with numbers 1-6 is tossed

A: Getting a number greater than 1

B: Getting an even number

$$P(A) = 5/6$$

$$P(B) = 3/6$$

$$P(B|A) = 3/5$$

Here the sample space has 5 points given A has occurred.

Bayes Theorem

Conditional Probability

Posterior Probability

$$P(B|A) = \frac{P(A|B)\,P(B)}{P(A)}$$

Prior Probability

Prior Probability

where

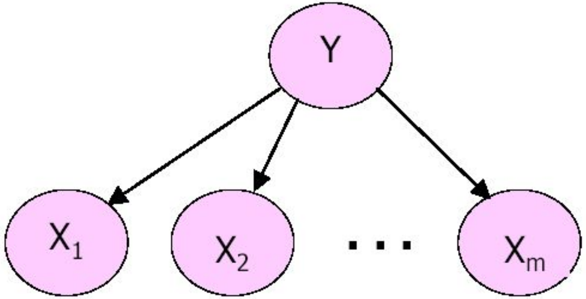
P(A) : Prior probability or marginal probability of A

P(A | B) : Conditional probability of A given B

P(B | A) : Conditional probability of B given A

P(B) : Prior or marginal probability of B

Naive Bayes Framework



Y : Categorical Dependent Variable

X_i : Categorical/Continuous Independent Variable

Objective: To estimate Y given the values of X_i's or

To estimate $P(Y|X_1, X_2, \ldots, X_m)$ using the Naïve Bayes Classifier

Assumption: All X_i's are conditionally independent of each other

Naive Bayes Framework - Example

Consider a simple example where Y is binary (response to a certain question) with 2 independent categorical variables X_1 and X_2

We classify	**Y = 1 "Buyer"** **Y = 0 "Non-Buyer"**
Let $\mathbf{X_1}$ denote **age of the individual**	$X_1 = 0$ for age group 25-30 years $X_1 = 1$ for age group 31-40 years
Let $\mathbf{X_2}$ denote **gender**	$X_2 = 0$ if Gender=female $X_2 = 1$ if Gender=male

Classification Rule

For the given values of X_1 and X_2 we want to know if the individual will be a potential buyer or not. Using Naive Bayes classifier we estimate:

$$P(Y = 0|X_1 = a_1, X_2 = a_2)$$

&

$$P(Y = 1|X_1 = a_1, X_2 = a_2)$$

where a_1 and a_2 are values of X_1 and X_2 for a particular respondent

We classify $Y = 0$ if $P(Y = 0|X_1 = a_1, X_2 = a_2) > 0.5$ OR

$Y = 1$ if $P(Y = 1|X_1 = a_1, X_2 = a_2) > 0.5$

In the general case i.e. when Y has more than 2 categories we compare $P(Y = y_k \mid X)$ for all values of y_k and classify $Y = y_k$ for which $P(Y = y_k \mid X)$ is the maximum

Expected Output

Once the classification rule is applied the output can be shown as follows:

Case#	X1	X2	$P(Y=1/X_1,X_2)$	$P(Y=0/X_1,X_2)$	Y classified as
1	1	0	0.44	0.56	0
2	1	1	0.7	0.3	1
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
240	0	0	0.2	0.8	0

Advantages of Naive Bayes Method

- Classification rule is simple to understand.
- The method requires a small amount of training data to estimate the parameters necessary for classification.
- The evaluation of the classifier is quick and easy.
- The method can be a good alternative to logistic regression.

Limitations of Naive Bayes Method

- Assumption of conditional independence of the independent variables is highly impractical.
- In case of continuous independent variables the density function must be known or assumed to be normal.
- In case of categorical independent variables the probabilities cannot be calculated if the count in any conditional category is zero. For instance: If there are no respondents in the age group 25-30 yrs. then $P(X_1=0 \mid Y=1) = 0$

?

How to deal with such cases?

If a category has zero entries we replace 0 by 0.5/n (n = sample size) so that the probability expression does not reduce to zero.

Case Study – Modeling Loan Defaults

Background

- A bank possesses demographic and transactional data of its loan customers. If the bank has a model to predict defaulters it can help in loan disbursal decision making.

Objective

- To predict whether the customer applying for the loan will be a defaulter or not.

Available Information

- Sample size is 700
- Independent Variables: Age group, Years at current address, Years at current employer, Debt to Income Ratio, Credit Card Debts, Other Debts. The information on predictors was collected at the time of loan application process.
- Dependent Variable: Defaulter (=1 if defaulter ,0 otherwise). The status is observed after loan is disbursed.

Bank Loan Data

Independent Variables

Dependent Variable

SN	AGE	EMPLOY	ADDRESS	DEBTINC	CREDDEBT	OTHDEBT	DEFAULTE
1	3	17	12	9.3	11.36	5.01	1
2	1	10	6	17.3	1.36	4	0

Column	Description	Type	Measurement	Possible Values
SN	Serial Number	numeric	-	-
AGE	Age Groups	Categorical	1(<28 years),2(28-40 years),3(>40 years)	3
EMPLOY	Number of years customer working at current employer	Continuous	-	Positive value
ADDRESS	Number of years customer staying at current address	Continuous	-	Positive value
DEBTINC	Debt to Income Ratio	Continuous	-	Positive value
CREDDEBT	Credit Card Debt	Continuous	-	Positive value
OTHDEBT	Other Debt	Continuous	-	Positive value
DEFAULTER	Whether customer defaulted on loan	Binary	1(Defaulter), 0(Non-Defaulter)	2

Logistic Regression in R

```
# Importing data and checking data structure
```

```
bankloan<-read.csv("BANK LOAN.csv",header=T)

str(bankloan)
```

```
# Output
```

```
> str(bankloan)
'data.frame':   700 obs. of  8 variables:
 $ SN       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ AGE      : int  3 1 2 3 1 3 2 3 1 2 ...
 $ EMPLOY   : int  17 10 15 15 2 5 20 12 3 0 ...
 $ ADDRESS  : int  12 6 14 14 0 5 9 11 4 13 ...
 $ DEBTINC  : num  9.3 17.3 5.5 2.9 17.3 10.2 30.6 3.6 24.4 19.7 ...
 $ CREDDEBT : num  11.36 1.36 0.86 2.66 1.79 ...
 $ OTHDEBT  : num  5.01 4 2.17 0.82 3.06 ...
 $ DEFAULTER: int  1 0 0 0 1 0 0 0 1 0 ...
```

```
bankloan$AGE<-factor(bankloan$AGE)

riskmodel<-glm(DEFAULTER~AGE+EMPLOY+ADDRESS+DEBTINC+CREDDEBT+OTHDEBT,
                family=binomial,data=bankloan)
```

glm() fits a generalised linear model. **family=binomial** ensures that a binary regression is used.

Model Summary

```
summary(riskmodel)
```

```
# Output
```

summary() generates model summary.

```
Call:
glm(formula = DEFAULTER ~ AGE + EMPLOY + ADDRESS + DEBTINC +
    CREDDEBT + OTHDEBT, family = binomial, data = bankloan)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3495  -0.6601  -0.2974   0.2509   2.8583

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.78821    0.26407  -2.985  0.00284 **
AGE2         0.25202    0.26651   0.946  0.34433
AGE3         0.62707    0.36056   1.739  0.08201 .
EMPLOY      -0.26172    0.03188  -8.211  < 2e-16 ***
ADDRESS     -0.09964    0.02234  -4.459 8.22e-06 ***
DEBTINC      0.08506    0.02212   3.845  0.00012 ***
CREDDEBT     0.56336    0.08877   6.347 2.20e-10 ***
OTHDEBT      0.02315    0.05709   0.405  0.68517
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 804.36  on 699  degrees of freedom
Residual deviance: 553.41  on 692  degrees of freedom
AIC: 569.41

Number of Fisher Scoring iterations: 6
```

Interpretation: EMPLOY, ADDRESS, DEBTINC and CREDDEBT are statistically significant.

Excluding Insignificant Variables

```
riskmodel<-glm(DEFAULTER~EMPLOY+ADDRESS+DEBTINC+CREDDEBT,
               family=binomial,data=bankloan)

summary(riskmodel)
```

```
# Output
```

```
Call:
glm(formula = DEFAULTER ~ EMPLOY + ADDRESS + DEBTINC + CREDDEBT,
    family = binomial, data = bankloan)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4483  -0.6396  -0.3108   0.2583   2.8496

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.79107    0.25154  -3.145  0.00166 **
EMPLOY      -0.24258    0.02806  -8.646  < 2e-16 ***
ADDRESS     -0.08122    0.01960  -4.144 3.41e-05 ***
DEBTINC      0.08827    0.01854   4.760 1.93e-06 ***
CREDDEBT     0.57290    0.08725   6.566 5.17e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 804.36  on 699  degrees of freedom
Residual deviance: 556.74  on 695  degrees of freedom
AIC: 566.74

Number of Fisher Scoring iterations: 6
```

Interpretation :
All four variables remain significant.

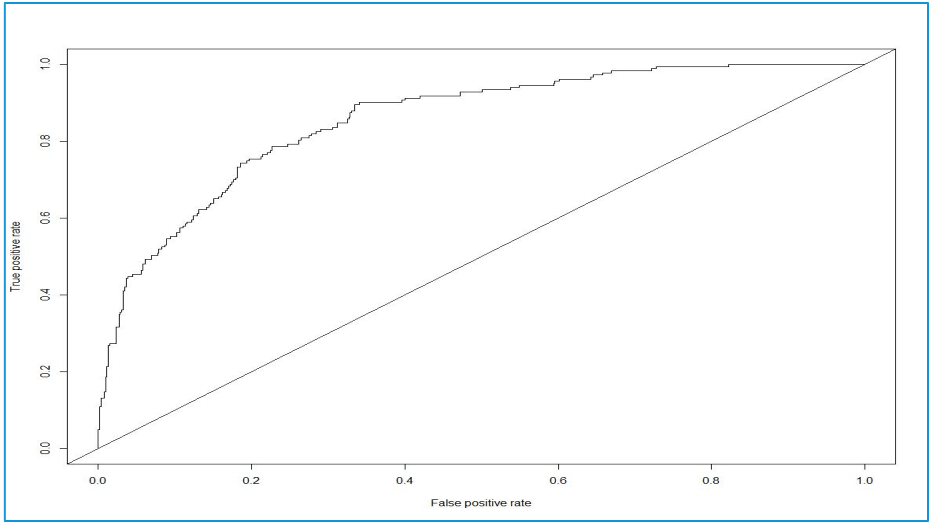
ROC Curve and Area Under ROC Curve

```
# ROC Curve
install.packages("ROCR")
library(ROCR)

bankloan$predprob<-fitted(riskmodel)

pred<-prediction(bankloan$predprob,bankloan$DEFAULTER)

perf<-performance(pred,"tpr","fpr")

plot(perf)

abline(0,1)
```

- **prediction()** function prepares data required for ROC curve.
- **performance()** function creates performance objects, "tpr“ (True positive rate), "fpr” (False positive rate).
- **plot()** function plots the objects created using performance
- **abline()** adds a straight line to the plot.

ROC Curve and Area Under ROC Curve

```
# Output
```




```
auc<-performance(pred,"auc")

auc@y.values
[[1]]
[1] 0.8556193
```

Estimates area under the ROC curve. Here it is 0.8556

Naive Bayes Method in R

```
# Install and load package “e1071”.
# Model Fitting
```

```
install.packages("e1071")
library(e1071)

riskmodel2<-naiveBayes(DEFAULTER~AGE+EMPLOY+ADDRESS+DEBTINC+CREDDEBT+OTHDEBT,
                data=bankloan)

riskmodel2
```

- **naiveBayes()** fits a Naive Bayes algorithm.
- It computes the conditional posterior probabilities of customer being defaulter/Non defaulter given values of independent variables using the Bayes rule.

Naive Bayes Model Output

```
# Output
```

```
> riskmodel2

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1 
0.7385714 0.2614286 

Conditional probabilities:
   AGE
Y           1         2         3
  0 0.3017408 0.4313346 0.2669246
  1 0.4699454 0.3333333 0.1967213

   EMPLOY
Y       [,1]     [,2]
  0 9.508704 6.663741
  1 5.224044 5.542946

   ADDRESS
Y       [,1]     [,2]
  0 8.945841 7.000621
  1 6.393443 5.925208

   DEBTINC
Y        [,1]     [,2]
  0  8.679304 5.615197
  1 14.727869 7.902798

   CREDDEBT
Y       [,1]     [,2]
  0 1.245397 1.422238
  1 2.423770 3.232645

   OTHDEBT
Y       [,1]     [,2]
  0 2.773230 2.813970
  1 3.863388 4.263394
```

Interpretation :

- Output shows a list of tables, one for each predictor variable. If the variable is categorical it shows the conditional probabilities for each class. For a numeric variable, for each target class, mean and standard deviation are shown.
- Eg. For EMPLOY, mean for “Defaulter” status = 0 is 9.51 and sd is 6.66.

Predicted Probabilities

```
# Predicted Probabilities
prednb<-predict(riskmodel2,bankloan,type='raw')


head(prednb)
```

- **predict()** returns predicted probabilities based on the model results and historical data.
- **type="raw"** returns raw probabilities. If not specified, predicted class is returned for each case

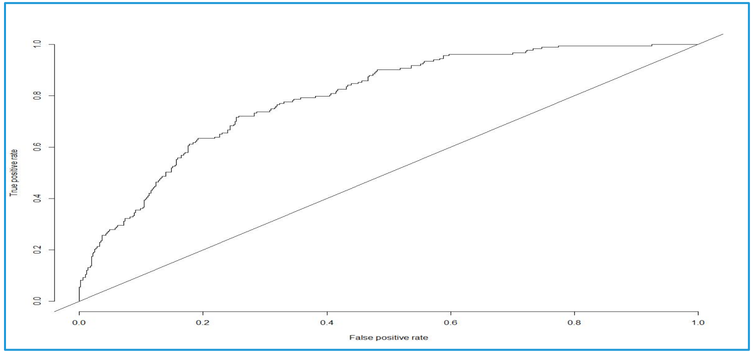
```
# Output
```

```
> head(prednb)
                0           1
[1,] 4.269360e-08 0.999999957
[2,] 7.182632e-01 0.281736806
[3,] 9.930388e-01 0.006961177
[4,] 9.885979e-01 0.011402100
[5,] 4.889496e-01 0.511050425
[6,] 9.177660e-01 0.082234006
```

Interpretation :
Column 2 gives probability of default (=1)

ROC Curve and Area Under ROC Curve

```
# ROC Curve and Area Under ROC Curve
pred<-prediction(prednb[,2],bankloan$DEFAULTER)
perf<-performance(pred,"tpr","fpr")
plot(perf)
abline(0,1)
```




```
# Area Under ROC Curve
auc<-performance(pred,"auc")
auc@y.values
[[1]]
[1] 0.794971
```

* The column having probability of the event under study must be selected while creating the prediction object. In this case, we are predicting the likelihood of default and default is represented by 1, hence column index [,2] is taken.

Quick Recap

Conditional Probability and Bayes' Theorem

- The conditional probability of an event B is the probability that event B will occur given the knowledge that an event A has already occurred.
- $P(B|A) = P(A|B)\,P(B) / P(A)$

Naive Bayes Classifier

- To estimate Y given the values of X_i's or $P(Y|X_1, X_2, \ldots, X_m)$ using the Naïve Bayes Classifier.
- **Assumption:** All X_i's are conditionally independent of each other.
-

Naive Bayes in R

- `naiveBayes()` in package **e1071**

Naive Bayes Classifier - II

Contents

Laplace Smoothing

- If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero.
- This is problematic because it will wipe out all information in the other probabilities when they are multiplied.
- Therefore, it is often desirable to incorporate a small-sample correction, called pseudo-count, in all probability estimates such that no probability is ever set to be exactly zero.
- This way of regularising naive Bayes is called Laplace Smoothing when the pseudo count is one, and Lidstone Smoothing in the general case.

Laplace Smoothing

$$P(x = x_i \mid y = y_j) = f_i / N_j$$

This prob will be 0 if numerator count (f_i) is 0

Laplace smoothing will replace this probability with a value obtained by the formula:

$$\boxed{\hat{\theta}_i = \frac{F_i + \alpha}{Nj + \alpha d}}$$

where

α : Smoothing Parameter

N_j : Number of observations for $Y = y_j$

d_i : Number of classes of x_i

Laplace Smoothing in R

```
# Importing Data
data1<-read.csv("Data for Laplace Smoothing.csv",header=T)

data1$X1<-as.factor(data1$X1)

data1
```

Y	X1	X2	X3
0	1	M	A
0	2	M	A
0	2	M	A
0	1	M	A
0	2	F	A
1	2	F	A
1	2	M	B
1	2	M	B
1	2	M	B
1	2	M	B
1	2	F	B
1	2	F	B
1	2	M	B
1	2	M	A
0	2	M	A
0	2	F	A
0	1	F	A
0	1	F	B
0	1	F	B
1	2	M	B
1	2	M	A
1	2	M	A
1	2	M	A
1	2	F	B
1	2	F	B
1	2	F	B
1	2	M	B
1	2	M	B

Variable X1 is a factor with two levels, 1 & 2. There is no observation in the data with X1 =1 when the dependent variable Y =1. Hence, P(X1 | Y=1) =0. We thus introduce smoothing, to avoid loss of information.

Laplace Smoothing in R

```
# Naive Bayes Model with Laplace Smoothing

model<-naiveBayes(Y~X1+X2+X3,data=data1)

laplacemodel<-naiveBayes(Y~X1+X2+X3,data=data1,laplace=2)
```

We first run the default Naive Bayes model.

laplace= tells R the value of pseudo-count to be used to smoothen the model.

$$\hat{\theta}_i = \frac{F_i + \alpha}{Nj + \alpha d} = \frac{2}{18 + 2 \times 2} = 0.09090$$

Since there are no observations for XI=I and Y=I,
$f_i=0$; $\alpha=2$
Total no. of Y=I is I8
XI is a factor with two classes, hence d=2

* There is no rule for choosing the appropriate pseudo-count. The number should be low, but not so low that the resulting probability is almost 0.

Laplace Smoothing in R

```
model
```

```
# Output
```

```
> model

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1 
0.3571429 0.6428571 

Conditional probabilities:
   X1
Y     1   2
  0 0.5 0.5
  1 0.0 1.0

   X2
Y           F         M
  0 0.5000000 0.5000000
  1 0.3333333 0.6666667

   X3
Y           A         B
  0 0.8000000 0.2000000
  1 0.2777778 0.7222222
```

Interpretation : Conditional probability of X1=1|Y=1 is 0.

Laplace Smoothing in R

```
laplacemodel
```

```
# Output
```

```
> laplacemodel

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1 
0.3571429 0.6428571 

Conditional probabilities:
   X1
Y            1          2
  0 0.50000000 0.50000000
  1 0.09090909 0.90909091

   X2
Y           F         M
  0 0.5000000 0.5000000
  1 0.3636364 0.6363636

   X3
Y           A         B
  0 0.7142857 0.2857143
  1 0.3181818 0.6818182
```

Interpretation :
R has now replaced 0 with 0.0909 (Calculated using the Laplace smoothing formula).

Predictions After Smoothing

```
# Importing and Readying New Data
newdata1<-read.csv("New Data for Laplace Predictions.csv",header=T)

newdata1
# Output
 Y X1 X2 X3
1 1  1  M  A
2 0  2  M  A
3 0  2  M  A
4 1  1  M  A

newdata1$X1<-as.factor(newdata1$X1)
```

- New data to be used for predictions is saved as an object named **newdata1**.
- New data contains observations which were absent in training data, i.e. conditional probability in training data was zero.
- **as.factor()** converts X1 to factor variable.

```
# Predictions
prednew<-predict(laplacemodel,newdata1,type="raw")
prednew1<-predict(laplacemodel,newdata1,type="raw",
                  threshold=0.1,eps=0.1)
```

- **threshold=** and **eps=** are added to ensure predicted probabilities are not too low. Threshold is the value that replaces values within the eps range.
- Here, probabilities<=0.1 are replaced by 0.1. Defaults are **threshold=**0.001 and **eps=**0.

Predictions After Smoothing

```
# Predictions
prednew
```

```
# Output
```

```
> prednew
             0         1
[1,] 0.8434941 0.1565059
[2,] 0.3502079 0.6497921
[3,] 0.3502079 0.6497921
[4,] 0.8434941 0.1565059
```

```
prednew1
```

```
# Output
```

```
> prednew1
             0         1
[1,] 0.8304964 0.1695036
[2,] 0.3502079 0.6497921
[3,] 0.3502079 0.6497921
[4,] 0.8304964 0.1695036
```

Interpretation :
Predicted probabilities using just the smoothened model and by using additional constraints of epsilon and threshold are different.

Quick Recap

Laplace Smoothing

- If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero.
- A pseudo-count is incorporated, in all probability estimates such that no probability is ever set to be exactly zero.
- This way of regularizing naive Bayes is called Laplace Smoothing
- `naiveBayes(`Y~X_i`,data=,laplace=)`

K Nearest Neighbours Classifier

Learn how a Simple Lazy Learning Algorithm Works

Contents

Introduction to KNN

Machine Learning Algorithms

Eager Learners

Learn a model that maps relationship between the predictors and response variable and then give a decision.

Lazy Learners

Base their decisions simply on the patterns found in training data. Generalisation beyond training data is delayed until a query is made.

- K Nearest Neighbours is one of the simplest lazy learner algorithms.
- The algorithm can be used for both classification and regression problems.
- Conceptually simple yet capable of solving complex problems.

KNN stores all available cases and classifies (or gives expected value of) new cases based on a similarity measure

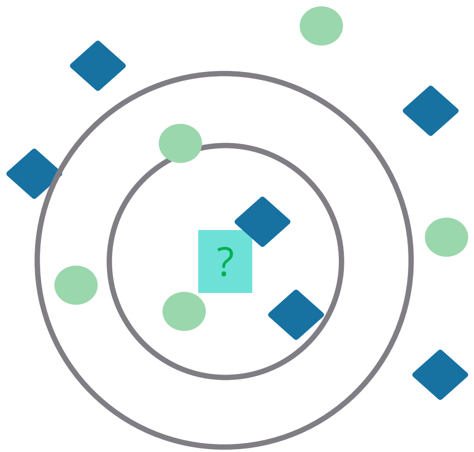
KNN for Classification

- Training dataset has 11 observations belonging to two categories.
- 12th observation is introduced, class of which is not known.
- **Nearest neighbour algorithm classifies new observation to the class of the training observation closest to it.**

When K=1, nearest one case is considered

As we go on increasing K, classification may vary

K	Classification
1	Blue
3	Blue
5	Orange



Three most important components of this method are **Distance** between cases, **Value of K** and **Voting** criteria.

Measuring Distance

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$D_1 = \sqrt{(25 - 48)^2 + (40000 - 142000)^2}$$

$$D_1 = 102000$$

Age	Current Debt	Default	Distance
25	40,000	N	102000
35	60,000	N	82000
45	80,000	N	62000
20	20,000	N	122000
35	120,000	N	22000
52	18,000	N	124000
23	95,000	Y	47000
40	62,000	Y	80000
60	100,000	Y	42000
48	220,000	Y	78000
33	150,000	Y	8000
48	**142,000**	**?**	

Here k=1, so the least distance from New observation is 8000, so it will be classified as “Y”.

Measuring Distance

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Euclidean Distance

Age	Current Debt	Default
25	40,000	N
35	60,000	N
45	80,000	N
20	20,000	N
35	120,000	N
52	18,000	N
23	95,000	Y
40	62,000	Y
60	100,000	Y
48	220,000	Y
33	150,000	Y
48	**142,000**	**Y**

Distance
102000
82000
62000
122000
22000
124000
47000
80000
42000
78000
8000

However, we can see that both these **attributes** are of different scales.

So rescaling of variables before calculating distances is the preferred approach.

Distance Based on Standardised Variables

$$X_s = \frac{X - Min}{Max - Min}$$

Alternatively, $\frac{(X-Mean)}{SD}$ can also be used

Age	Current Debt	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	0.61	?	

New observation will be classified as "N"

Selection of K

The second component of KNN model is selecting the appropriate value for K

- If K = 1, the case is classified using the nearest neighbour
- However, K is usually greater than 1. Consider the following when choosing K :
  - Mostly odd numbered K is preferred to avoid tie.
  - For a very large K the classifier may result in misclassification, as group of nearest neighbours may include data points which are actually located far away from it.

Thumb Rule :

K = sqrt (n)

n is the number of observations in training data

Voting Criteria

Most common criteria for classification decision is **Majority Voting.**

Frequency of each class in K instances is measured. Class having the highest frequency is attributed to the new case.

Eg. Suppose for K = 7, 4 cases belong to class A and 3 to class B. New case is given class A

Drawback:

Classification is inappropriate when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number.

?

Is there a way to correct this?
One option to remove this drawback is to create training data with equal class frequency. However, this is possible only if data is very large.

Voting Criteria

Another approach is **Inverse Distance Weighted Voting**

This approach assigns higher weights to closer neighbours. Votes are then summed and class with the highest votes is assigned to the new case.

Eg. Suppose for K = 3, 2 cases belong to class A and 1 to class B. with distances 0.4, 0.5 and 0.2 respectively. Sum of inverse distances are

Class A (1/0.4)+(1/0.5) = 4.5 and Class B (1/0.2)=5

This rule will allot class A to the new observation

Some studies also recommend inverse of squared distances or Kernel functions

Handling Ties

KNN may result in **'Ties' – nearest neighbours may have equal class frequencies or equal inverse distance sums**

Such ties are solved by either of the following ways:

- In case of binary class variables, avoid using even numbered Ks
- Increasing or decreasing K until ties are broken

Case Study – Predicting Loan Defaulters

Background

- The bank possesses demographic and transactional data of its loan customers. If the bank has a robust model to predict defaulters it can undertake better resource allocation.

Objective

- To predict whether the customer applying for the loan will be a defaulter

Available Information

- Sample size is 389
- Age group, Years at current address, Years at current employer, Debt to Income Ratio, Credit Card Debts, Other Debts are the independent variables
- Defaulter (=1 if defaulter, 0 otherwise) is the dependent variable

Data Snapshot

BANK LOAN KNN

Variables

SN	AGE	EMPLOY	ADDRESS	DEBTINC	CREDDEBT	OTHDEBT	DEFAULTER

Column	Description	Type	Measurement	Possible Values
SN	Serial Number		-	-
AGE	Age Groups	Categorical	1(<28 years),2(28-40 years),3(>40 years)	3
EMPLOY	Number of years customer working at current employer	Continuous	-	Positive value
ADDRESS	Number of years customer staying at current address	Continuous	-	Positive value
DEBTINC	Debt to Income Ratio	Continuous	-	Positive value
CREDDEBT	Credit Card Debt	Continuous	-	Positive value
OTHDEBT	Other Debt	Continuous	-	Positive value
DEFAULTER	Whether customer defaulted on loan	Binary	1(Defaulter), 0(Non-Defaulter)	2

KNN Classification in R

```
# Importing the Data
bankloan<-read.csv("BANK LOAN KNN.csv",header=T)

# Preparing data by removing unwanted variables
bankloan2<-subset(bankloan,select=c(-AGE,-SN,-DEFAULTER))
```

subset() is used to remove unwanted variables. AGE is removed because it is a categorical variable.

```
head(bankloan2)
```

```
# Output
```

```
> head(bankloan2)
  EMPLOY ADDRESS DEBTINC CREDDEBT OTHDEBT
1     17      12     9.3    11.36    5.01
2      2       0    17.3     1.79    3.06
3     12      11     3.6     0.13    1.24
4      3       4    24.4     1.36    3.28
5     24      14    10.0     3.93    2.47
6      6       9    16.3     1.72    3.01
```

KNN Classification in R

```
# Preparing Variables
bankloan3<-scale(bankloan2)
```

scale() in base R is a generic function used for centering or scaling columns of a numeric matrix. The default method for scaling is (X-Mean)/SD.

```
head(bankloan3)
```

```
# Output
```

```
> head(bankloan3)
      EMPLOY    ADDRESS    DEBTINC     CREDDEBT     OTHDEBT
1  1.5656796  0.6216799 -0.2881684 3.8774339687  0.51519694
2 -0.8239988 -1.1852951  0.7889154 0.0289356115 -0.02571385
3  0.7691201  0.4710987 -1.0555906 -0.6386200074 -0.53056393
4 -0.6646869 -0.5829701  1.7448273 -0.1439854223  0.03531198
5  2.6808628  0.9228424 -0.1939235 0.8895193612 -0.18937404
6 -0.1867512  0.1699362  0.6542799 0.0007856758 -0.03958336
```

- All the continuous predictors are now scaled

KNN Classification in R

```
# Training and Testing Data Sets
```

```r
install.packages("caret")
library(caret)

index<-createDataPartition(bankloan$SN,p=0.7,list=FALSE)
head(index)
```

```
# Output
```

```
> head(index)
     Resample1
[1,]         1
[2,]         3
[3,]         4
[4,]         5
[5,]         7
[6,]         8
```

```r
traindata<-bankloan3[index,]
testdata<-bankloan3[-index,]

dim(traindata)
[1] 273 5
dim(testdata)
[1] 116 5
```

KNN Classification in R

```
# Creating Class Vectors
Ytrain<-bankloan$DEFAULTER[index]

Ytest<-bankloan$DEFAULTER[-index]
```

Interpretation :

- Training and testing datasets are created with a 70:30 division.
- Class variable is also split by the same proportion.

KNN Classification Using Package "class"

```
# KNN Using Package "class"
install.packages("class")
library(class)
# KNN Classification (Continuous Predictors)
model<-knn(traindata,testdata,k=20,cl=Ytrain)
```

- **knn()** in package **“class”** performs k-nearest neighbour classification of test data using train data. Distance is calculated by Euclidean measure, and the classification is decided by majority vote, with ties broken at random.
- **k=** specifies the value of k.
- **cl=** is the vector of observed Y values

KNN Classification Using Package "class"

```
table(Ytest,model)
     model
Ytest  0  1
    0 45 20
    1 13 38
```

table() gives the cross tabulation of actual and predicted classes for test data

```
# Confusion Matrix
```

```
class(model)
[1] "factor"

class(Ytest)
[1] "integer"

Ytest <- as.factor(Ytest)

library(caret)
confusionMatrix(Ytest,model)
```

* Note : Output will be slightly different as observations are randomly assigned to train-test data.

KNN Classification in R

```
# Output
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 45 20
         1 13 38

               Accuracy : 0.7155
                 95% CI : (0.6243, 0.7954)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 1.933e-06

                  Kappa : 0.431

 Mcnemar's Test P-Value : 0.2963

            Sensitivity : 0.7759
            Specificity : 0.6552
         Pos Pred Value : 0.6923
         Neg Pred Value : 0.7451
             Prevalence : 0.5000
         Detection Rate : 0.3879
   Detection Prevalence : 0.5603
      Balanced Accuracy : 0.7155

       'Positive' Class : 0
```

Interpretation :

- From the confusion matrix we can see that sensitivity is 77.6% & specificity 65.5%.

KNN for Regression

KNN algorithm can also be extended to regression problems, i.e. **when the dependent variable is continuous**

Process flow for classification and regression is the same, except for the last step

Standardise Variables → Calculate Distances → Choose Appropriate k → **Average the Response Variable for k Neighbours**

Average value of the response variable for k neighbours is calculated and assigned to the new case.

> **`knn.reg()`** from package **`"FNN"`** can be used to run k-nearest neighbour regression in R, using the syntax : **`knn.reg`**`(train, test, y, k)`
>
> **y** is the response for each observation in training set

Get an Edge!

- KNN can be used for categorical variables as well.
- Before executing knn on train-test data, categorical variables have to be converted to continuous variables by creating dummy variables.

Quick Recap

KNN for Classification	• Three most important components of this method are **Distance** between cases, **Value of K** and **Voting** criteria.
KNN for Classification in R	• **knn()** in package **class.**
KNN for Regression	• KNN algorithm can also be extended to regression problems **when the dependent variable is continuous.**
KNN for Regression in R	• **`knn.reg()`** from package **`FNN`**

Support Vector Machines in R

Contents

Introduction to Support Vector Machines

- Support Vector Machines (SVM's) are a relatively new learning method generally used for classification problem.
- Although the first paper dates way back to early 1960's it is only in 1992-1995 that this powerful method was universally adopted as a mainstream machine learning paradigm

The basic idea is to find a hyper plane which separates the d-dimensional data perfectly into its classes. However, since training data is often not linearly separable, SVM's introduce the notion of a "Kernel-induced Feature Space" which casts the data into a higher dimensional space where the data is separable.

What is a Hyper Plane

In two dimensions, a hyper plane is defined by the equation:

$$\boxed{W_1X_1 + W_2X_2 + b = 0}$$

This is nothing but **equation of line.**

The above equation can be easily extended to the p-dimensional setting:

$$\boxed{W_1X_1 + W_2X_2 + \cdots + W_pX_p + b = 0}$$

In short,

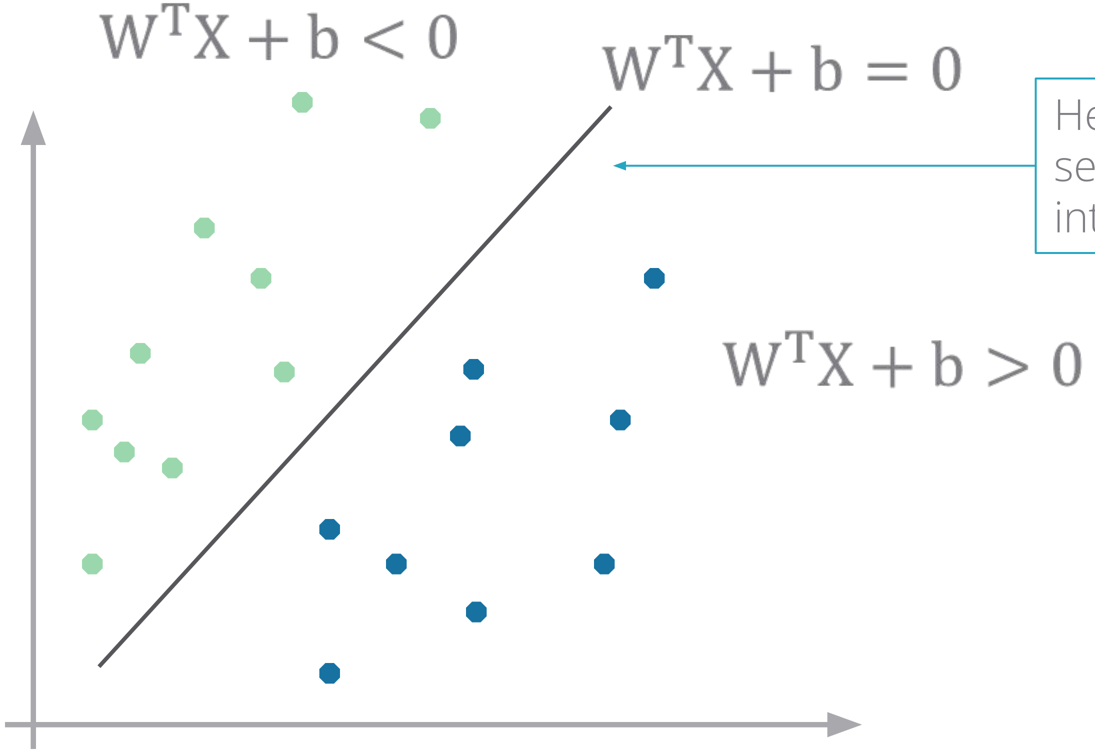
$$\boxed{\mathbf{W^TX + b = 0}}$$

In p > 3 dimensions, it can be hard to visualize a hyper planes.

Separating a Hyper Plane

- Binary classification can be viewed as the task of separating classes in feature space:

Fig. 01: Binary Classification

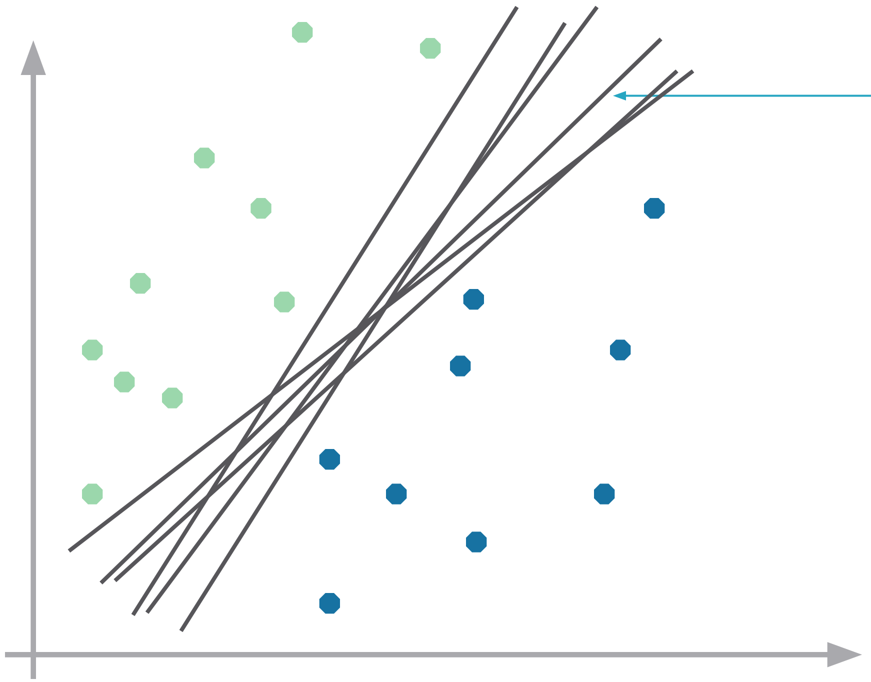


Here line separates data into two parts

Linear Separators

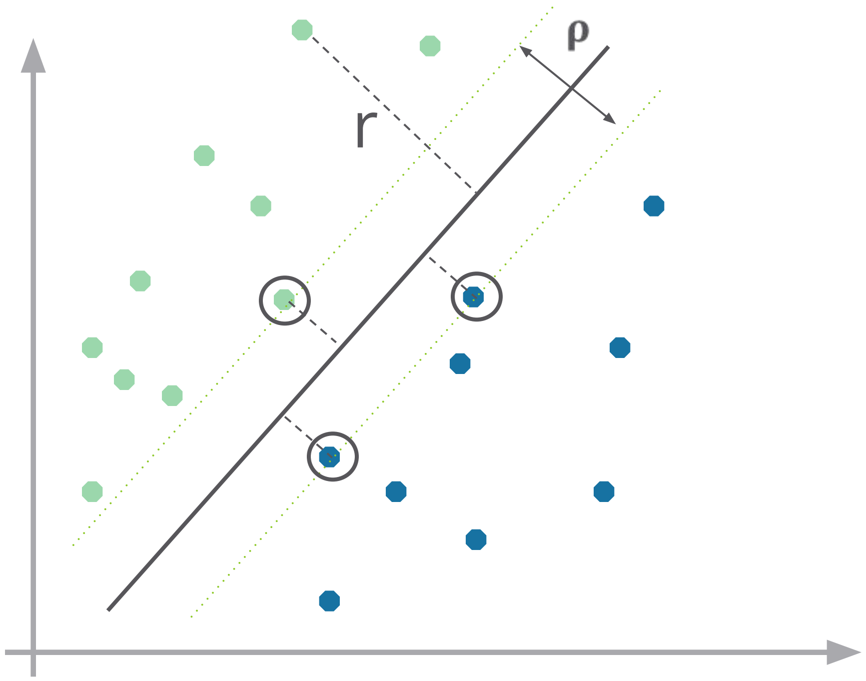
The objective in SVM is to find optimum separator

Fig. 02: Linear Separators



Which of the linear separators is optimal?

Classification Margin



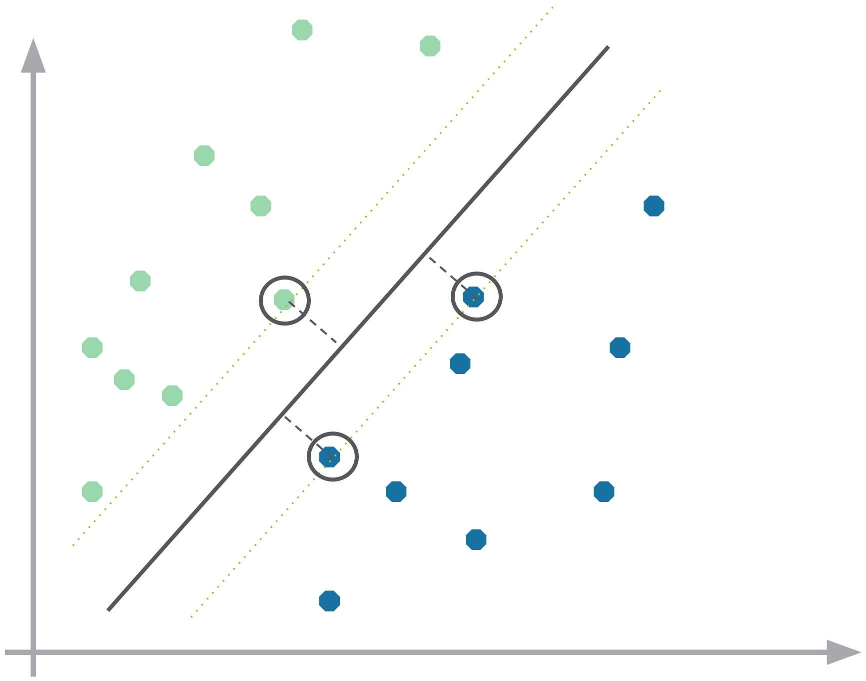
- Distance from case $\mathbf{x}_i$ to the separator is

$$r = \frac{w^T x_i + b}{\| w \|}$$

Here $\| w \|$ is length of a vector given by sqrt(sum(W^2))

- **Cases closest to the hyper plane are Support Vectors**
- **Margin ρ of the separator is the distance between support vectors**

Maximum Margin Classification



- The objective is now to maximize the margin ρ of the separator
- The focus is on 'Support Vectors'
- Other cases are not considered in the algorithm

Mathematical Approach to Linear SVM

Let training set be separated by a hyper plane with margin ρ. Then for each training observation

$$\begin{aligned} w^T x_i + b &\leq -\rho/2 \quad \text{if } y_i = -1 \\ w^T x_i + b &\geq \rho/2 \quad \text{if } y_i = 1 \end{aligned} \leftrightarrow \boxed{y_i(w^T x_i + b) \geq \rho/2}$$

For every support vector x_s the above inequality is an equality

After rescaling w and b by $\rho/2$ in the equality, we obtain that distance between each x_s and the hyper plane is

$$r = \frac{y_i(w^T x_s + b)}{\| w \|} = \frac{1}{\| w \|}$$

Margin can be expressed through (

$$\boxed{\rho = 2r = \frac{2}{\| w \|}}$$

Mathematical Approach to Linear SVM

Quadratic Optimisation problem is:

Find w and b such that

$\rho = \frac{2}{\|\mathrm{w}\|}$ is maximised

and

$$\mathrm{y_i(w^T x_i + b) \geq 1}$$

which can be reformulated as:

Find w and b such that

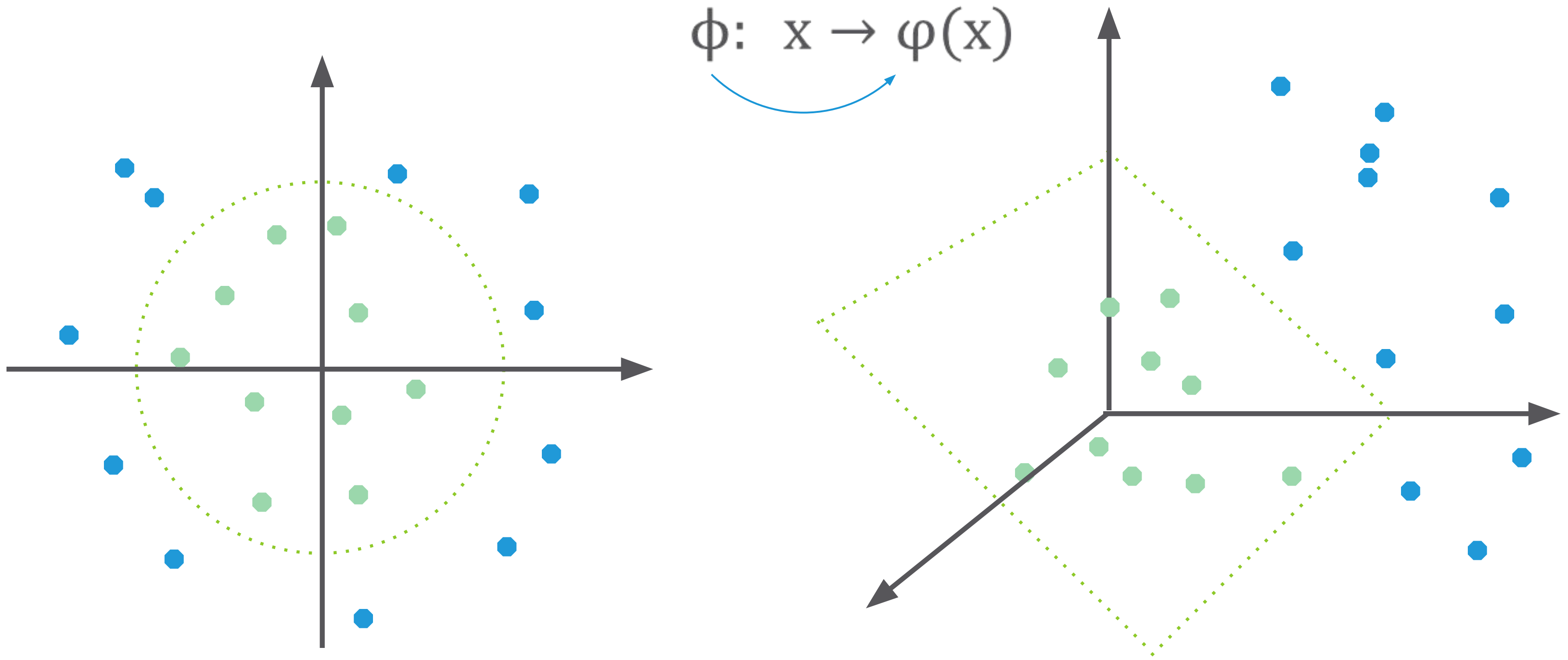
$\phi(w) = w^T w$ is minimised

and

$$\mathrm{y_i(w^T x_i + b) \geq 1}$$

Non-Linear SVMs – Feature Spaces

General idea: The original feature space can always be mapped to some higher-dimensional feature space where the training set is separable



The "Kernel Trick"

The linear classifier relies on inner product between vectors

$$K(x_i, x_j) = x_i^T x_j$$

If every data point is mapped into high-dimensional space via some transformation $\phi: \; x \to \varphi(x)$

then the inner product becomes

$$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$$

A kernel function is a function that is equivalent to an inner product in some feature space

The "Kernel Trick"

Example:

2-dimensional vector $x = [\,x_1 \quad x_2\,]$;

Let $K(x_i, x_j) = (1 + x_i^T x_j)^2$

Need to show that $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$:

$K(x_i, x_j) = (1 + x_i^T x_j)^2$

$= 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$

$= [1 \quad x_{i1}^2 \; \sqrt{2}x_{i1}x_{i2} \quad x_{i2}^2 \; \sqrt{2}x_{i1} \quad \sqrt{2}x_{i2}]\ T\ [1$

$x_{j1}^2 \; \sqrt{2}x_{j1}x_{j2} \quad x_{j2}^2 \; \sqrt{2}x_{j1} \quad \sqrt{2}x_{j2}]$

$= \varphi(x_i)^T \varphi(x_j)$ where $\varphi(x) = [1 \; x_1^2 \sqrt{2}x_1x_2 \; x_2^2 \sqrt{2}x_1 \sqrt{2}x_2]$

Thus, a kernel function implicitly maps data to a high-dimensional space (Without the need to compute each $\varphi(x)$ explicitly)

Examples of Kernel Functions

Linear

$$K(x_i, x_j) = x_i{}^T x_j$$

Mapping ϕ

$x \rightarrow \varphi(x)$ where $\varphi(x)$ is x itself

Polynomial of power ρ

$$K(x_i, x_j) = (1 + x_i{}^T x_j)^\rho$$

Gaussian (Radial basis function)

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$

Case Study – Predicting Loan Defaulters

Background

- The bank possesses demographic and transactional data of its loan customers. If the bank has a robust model to predict defaulters it can undertake better resource allocation.

Objective

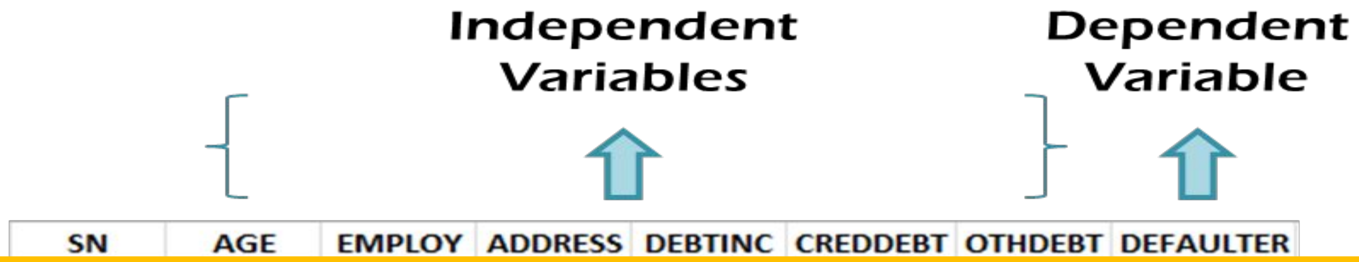
- To predict whether the customer applying for the loan will be a defaulter

Available Information

- Sample size is 700
- Age group, Years at current address, Years at current employer, Debt to Income Ratio, Credit Card Debts, Other Debts are the independent variables
- Defaulter (=1 if defaulter, 0 otherwise) is the dependent variable

Data Snapshot

BANK LOAN



Column	Description	Type	Measurement	Possible Values
SN	Serial Number	-	-	-
AGE	Age Groups	Integer	1(<28 years),2(28-40 years),3(>40 years)	3
EMPLOY	Number of years customer working at current employer	Integer	-	Positive value
ADDRESS	Number of years customer staying at current address	Integer	-	Positive value
DEBTINC	Debt to Income Ratio	Continuous	-	Positive value
CREDDEBT	Credit to Debit Ratio	Continuous	-	Positive value
OTHDEBT	Other Debt	Continuous	-	Positive value
DEFAULTER	Whether customer defaulted on loan	Integer	1(Defaulter), 0(Non-Defaulter)	2

SVM in R

```
# Importing and Readying the Data
```

```
bankloan<-read.csv("BANK LOAN.csv",header=T)

bankloan$AGE<-as.factor(bankloan$AGE)

str(bankloan)
```

as.factor() changes age from an integer to a factor variable.

str() is used to check if the conversion to factor has taken place and if all other variable formats are appropriate, before moving to SVM modeling.

```
# Output
```

```
'data.frame':   700 obs. of  8 variables:
 $ SN       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ AGE      : Factor w/ 3 levels "1","2","3": 3 1 2 3 1 3 2 3 1 2 ...
 $ EMPLOY   : int  17 10 15 15 2 5 20 12 3 0 ...
 $ ADDRESS  : int  12 6 14 14 0 5 9 11 4 13 ...
 $ DEBTINC  : num  9.3 17.3 5.5 2.9 17.3 10.2 30.6 3.6 24.4 19.7 ...
 $ CREDDEBT : num  11.36 1.36 0.86 2.66 1.79 ...
 $ OTHDEBT  : num  5.01 4 2.17 0.82 3.06 ...
 $ DEFAULTER: int  1 0 0 0 1 0 0 0 1 0 ...
```

SVM in R

```
# SVM Using Package "e1071"
```

```
install.packages("e1071")
library(e1071)

model<-svm(formula=DEFAULTER~AGE+EMPLOY+ADDRESS+
           DEBTINC+CREDDEBT+OTHDEBT,data=bankloan,
           type="C",probability=TRUE,kernel="linear")

model
```

- ❑ **svm()** trains a support vector machine.
- ❑ **formula=** gives the model to be fit.
- ❑ **data=** specifies the data object.
- ❑ **type=** specifies whether SVM is used for classification or regression or novelty detection. Default for **type=** is "C“.
- ❑ **probability=** logical for indicating whether model should allow for probability predictions.
- ❑ **kernel=** specifies the kernel used in training and predicting. Here, we have kept kernel as linear.

SVM in R

```
# Output
```

```
> model

Call:
svm(formula = DEFAULTER ~ AGE + EMPLOY + ADDRESS + DEBTINC + CREDDEBT + OTHDEBT,
 data = bankloan, type = "C", probability = TRUE,
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  312
```

Predictions Based on SVM

```
# Predictions
```

```
pred1<-predict(model,bankloan,probability=TRUE)
```

- **predict()** returns predicted probabilities based on the model results and historical data.
- First argument is the **svm()** model object while the second argument is original dataset.
- **probability=TRUE** returns raw probabilities. This argument is valid only when **type="probability“** is specified in **svm()**.

```
pred2<-attr(pred1,"probabilities")[,1]
```

- **attr()**, from base R, is used get or set specific attributes of an object. Here, we want to get the predicted probabilities obtained by the **svm()** model.
- First argument is the name of the object whose attributes we want to extract.
- Second argument is the character string specifying which attribute is to be accessed. Check **pred1** to know the exact name, which is **"probabilities"**.

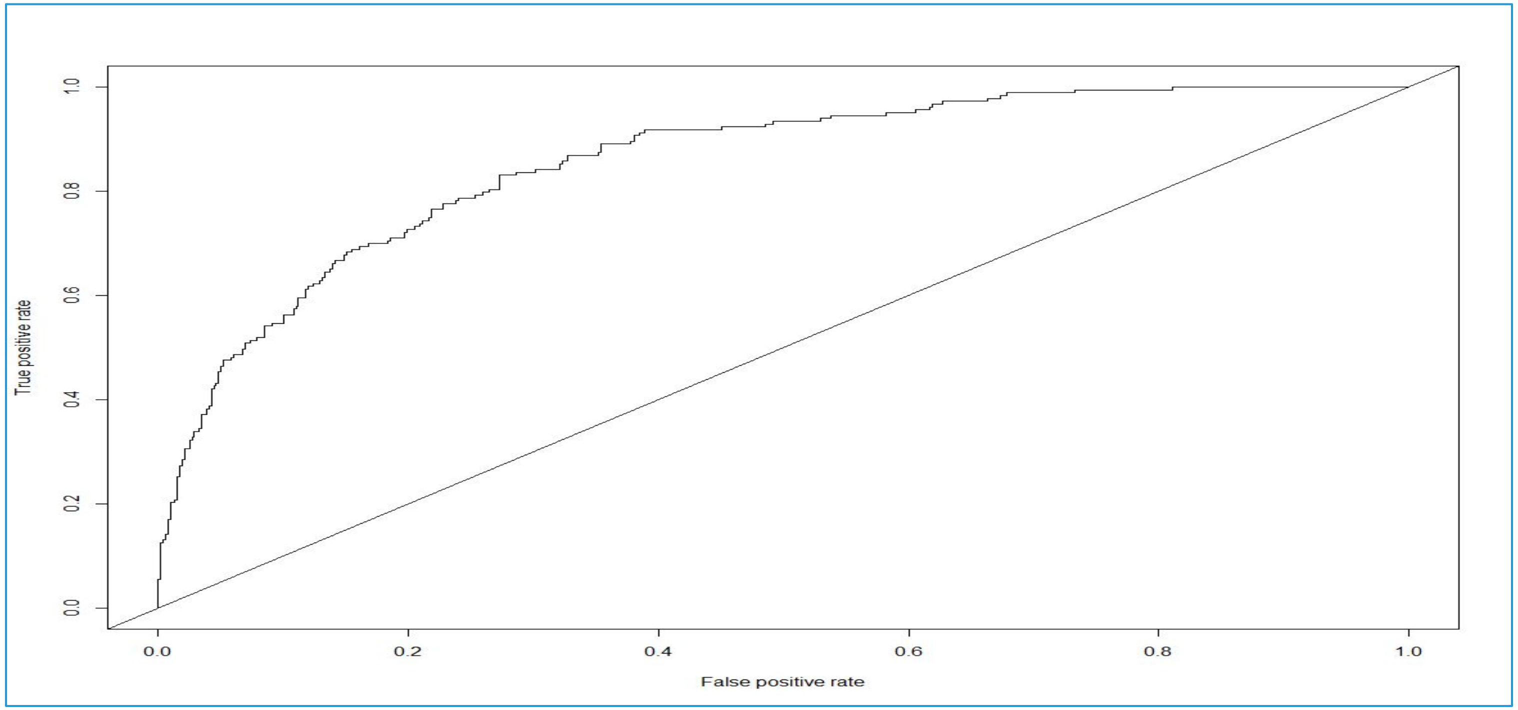
ROC Curve and Area Under ROC Curve

```
#ROC Curve

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

pred<-prediction(pred2,bankloan$DEFAULTER)

perf<-performance(pred,"tpr","fpr")

plot(perf)

abline(0,1)
```

prediction() creates object of class prediction, required for ROC curve. **performance()** calculates predictor evaluations. Using **measure="tpr", measure="fpr"** we can plot an ROC Curve.
abline()
adds a straight line to the plot.

ROC Curve and Area Under ROC Curve

```
# Output
```




```
# Area Under ROC Curve
```

```
auc<-performance(pred,"auc")
auc@y.values
[[1]]
[1] 0.855577
```

"auc" in performance() calculates Area Under ROC Curve.

Quick Recap

Support Vector Machines

- SVMs find a hyper plane which separates the d-dimensional data perfectly into its classes
- Since training data is often not linearly separable, SVM's introduce the notion of a “Kernel-induced Feature Space" which casts the data into a higher dimensional space where the data is separable

SVM in R

- Package **`"e1071"`** has **`svm()`** that trains a support vector machine
- The function takes arguments to specify whether **`svm()`** is to be used for classification or regression; if probabilities are to be returned and which kernel to use for training and predicting

Weight of Evidence (WoE)
and
Information Value (IV)

Contents

Handling Categorical Variables In Statistical Models

- In classification or regression problems independent variables can either be continuous or categorical. Categorical variable can have limited number of categories or many categories as shown in the following table.

Type	Example
Variable with limited categories	Age Groups: Below 15 = 1, 15-25 = 2, Above 25 = 3
Variables with large number of categories	City or Country will have many levels

- Generally, when independent variables in statistical models are categorical, they are replaced and represented by Dummy Variables.
- If there are k categorical variables, then they are represented by k-1 Dummy Variables.

Weight of Evidence (WoE)

- Weight of Evidence (WoE) estimates the **predictive power of an independent variable in relation to the dependent variable.**
- WoE is originally used in credit risk analytics, as the method of separation of "good" and "bad" customers (Non-defaulters: Y=0 and Defaulters: Y=1)
- WoE is defined as

$$\boxed{\ln\left(\frac{\text{Distribution of Good}_i}{\text{Distribution of Bad}_i}\right)}$$

- Here, Distribution of Good is the proportion of good customers in a category to total good customers. Similarly, Distribution of Bad is the proportion of bad customers in a category to total bad customers.
- Using WoE ,we can assign continuous value for each category. For instance, if there are 50 cities then there will be 50 WoE values.

Get an Edge!

Some thumb rules related to Weight of Evidence

- Each category (bin) should have at least 5% of the observations.
- Each category (bin) should be non-zero for both non-events and events.
- The WoE should be distinct for each category. Similar groups should be aggregated.
- The WoE should be monotonic, i.e. either growing or decreasing with the groupings (Not applicable when groups are for character strings).
- Missing values are binned separately.

Information Value (IV)

- Information Value (IV) is a highly useful tool for variable selection.
- The concept has its roots in entropy in information theory.
- IV of an independent variable **expresses the amount of diagnostic information of that variable for separating the Goods from the Bads.**
- IV is calculated as

$$\sum (\text{Distribution of Good}_i - \text{Distribution of Bad}_i) \times \underbrace{\ln\left(\frac{\text{Distribution of Good}_i}{\text{Distribution of Bad}_i}\right)}_{\text{Weight of Evidence}}$$

- IV helps in ranking variables based on their importance.

Information Value (IV)

By convention, information values can be interpreted as follows:

Value	Predictive
< 0.02	Not useful for prediction
0.02 to 0.1	Weak predictor
0.1 to 0.3	Medium predictor
0.3 to 0.5	Strong predictor
> 0.5	Suspicious predictive power

Case Study – Predicting Loan Defaulters

Background

- The bank possesses demographic and transactional data of its loan customers. If the bank has a robust model to predict defaulters it can undertake better resource allocation.

Objective

- To predict whether the customer applying for the loan will be a defaulter.

Available Information

- Sample size is 700
- Independent Variables: Age group, Town, Years at current address, Years at current employer, Debt to Income Ratio, Credit to Debit ratio, Other Debts
- Dependent Variables: Defaulter (=1 if defaulter, 0 otherwise)

Data Snapshot

BANK LOAN WOE-IV

Independent Variables

Dependent Variable

SN	AGE	TOWN	EMPLOY	ADDRESS	DEBTINC	CREDDEBT	OTHDEBT	DEFAULTER
1	3	Mumbai	17	12	9.3	11.36	5.01	1

Observations

Column	Description	Type	Measurement	Possible Values
SN	Serial Number	Numeric	-	-
AGE	Age Groups	Categorical	1(<28 years),2(28-40 years),3(>40 years)	3
TOWN	Customer Belonging to Which Town	Categorical	Mumbai, Delhi,etc..	15
EMPLOY	Number of years customer working at current employer	Continuous	-	Positive value
ADDRESS	Number of years customer staying at current address	Continuous	-	Positive value
DEBTINC	Debt to Income Ratio	Continuous	-	Positive value
CREDDEBT	Credit to Debit Ratio	Continuous	-	Positive value
OTHDEBT	Other Debt	Continuous	-	Positive value
DEFAULTER	Whether customer defaulted on loan	Binary	1(Defaulter), 0(Non-Defaulter)	2

WoE and IV in R

```
# Import the data
```

```r
data<-read.csv("BANK LOAN WOE-IV.csv",header=T)
head(data)
str(data)
```

```
# Convert AGE to Factor
```

```r
data$AGE<-as.factor(data$AGE)
```

- ❑ **read.csv**() is used to import csv file.
- ❑ **str**() shows class and levels of variables in the data.
- ❑ AGE is actually a categorical variable but represented numerically. We will convert it to factor using **as.factor**() and then calculate WoE and IV for AGE.

WoE and IV in R

```
# Output:
```

```
> data<-read.csv(file.choose())
> str(data)
'data.frame':   700 obs. of  9 variables:
 $ SN       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ AGE      : int  3 1 2 3 1 3 2 3 1 2 ...
 $ TOWN     : Factor w/ 15 levels "Ahmedabad","Bengaluru",..: 12 4 2 5 1 3 10 15 14 7 ...
 $ EMPLOY   : int  17 10 15 15 2 5 20 12 3 0 ...
 $ ADDRESS  : int  12 6 14 14 0 5 9 11 4 13 ...
 $ DEBTINC  : num  9.3 17.3 5.5 2.9 17.3 10.2 30.6 3.6 24.4 19.7 ...
 $ CREDDEBT : num  11.36 1.36 0.86 2.66 1.79 ...
 $ OTHDEBT  : num  5.01 4 2.17 0.82 3.06 ...
 $ DEFAULTER: int  1 0 0 0 1 0 0 0 1 0 ...
> data$AGE<-as.factor(data$AGE)
> str(data)
'data.frame':   700 obs. of  9 variables:
 $ SN       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ AGE      : Factor w/ 3 levels "1","2","3": 3 1 2 3 1 3 2 3 1 2 ...
 $ TOWN     : Factor w/ 15 levels "Ahmedabad","Bengaluru",..: 12 4 2 5 1 3 10 15 14 7 ...
 $ EMPLOY   : int  17 10 15 15 2 5 20 12 3 0 ...
 $ ADDRESS  : int  12 6 14 14 0 5 9 11 4 13 ...
 $ DEBTINC  : num  9.3 17.3 5.5 2.9 17.3 10.2 30.6 3.6 24.4 19.7 ...
 $ CREDDEBT : num  11.36 1.36 0.86 2.66 1.79 ...
 $ OTHDEBT  : num  5.01 4 2.17 0.82 3.06 ...
 $ DEFAULTER: int  1 0 0 0 1 0 0 0 1 0 ...
```

Interpretation:

- Age initially as integer is converted to factor with 3 levels 1,2,3.

WoE and IV in R

```
# Install and load package "Information"
```

```
install.packages("Information")
library(Information)
```

- The **Information** package is designed to perform exploratory data analysis and variable screening for binary classification models using WOE and IV. The package is specifically designed to perform data exploration by producing easy-to-read tables and graphs.

```
# Changing Binary Values for Defaulter
```

```
data$DEFAULTERNEW = 1-data$DEFAULTER
```

- Packages for computing WoE and IV consider binary value 0 to be 'bad' and 1 to be 'good'. However, in our data, (and as a general practice) 1 represents occurrence of an event and 0 otherwise. It is imperative to remember this before calculating WoE and IV tables.

WoE and IV in R

```
# Calculate WoE and IV
```

```
IV <- create_infotables(data=data, y="DEFAULTERNEW")
```

- create_infotable generates WoE and IV for all variables in the data except dependent variable which is specified as **"y="**.

```
# Get WoE and IV values for 'AGE' variable
```

```
woe_age<-as.data.frame(IV$Tables$AGE)
woe_age
```

- **create_infotables()** returns WOE tables as data.frames, and a data.frame with IV values for all predictive variables.
- IV$Tables$'predictor variable name' is created to store WoE and IV values in a dataframe which are used for further analysis.

WoE and IV in R

```
# Output :
```

```
> woe_age
  AGE   N   Percent        WOE         IV
1   1 242 0.3457143 -0.4430480 0.07452269
2   2 284 0.4057143  0.2577412 0.09978166
3   3 174 0.2485714  0.3051780 0.12120615
```

Interpretation:

- Output table contains categories of the variable, count and percent of observations for each category, WoE and IV values.

Appending WoE Values to Original Data

```
# Check the type of key variable before merging
```

```r
str(woe_age)
woe_age$AGE<-as.factor(woe_age$AGE)
str(woe_age)
```

- **‘Age’** is the common variable in the original data and WoE data.
- Type of ‘Age’ in both data should be common for merging datasets.]

```
# Output :
```

```
> str(woe_age)
'data.frame':	3 obs. of  5 variables:
 $ AGE    : chr  "1" "2" "3"
 $ N      : num  242 284 174
 $ Percent: num  0.346 0.406 0.249
 $ WOE    : num  -0.443 0.258 0.305
 $ IV     : num  0.0745 0.0998 0.1212
> woe_age$AGE<-as.factor(woe_age$AGE)
> str(woe_age)
'data.frame':	3 obs. of  5 variables:
 $ AGE    : Factor w/ 3 levels "1","2","3": 1 2 3
 $ N      : num  242 284 174
 $ Percent: num  0.346 0.406 0.249
 $ WOE    : num  -0.443 0.258 0.305
 $ IV     : num  0.0745 0.0998 0.1212
```

Appending WoE Values to Original Data

```
# Merging the datasets
```

```
leftjoin<-merge(data,woe_age,by="AGE", all.x = TRUE)
head(leftjoin)
```

- **merge()** with **all.x=TRUE** returns data with all rows from left table (here, data) and any rows with matching keys from the right table (here, woe_age).

```
# Output :
```

```
> head(leftjoin)
  AGE  SN      TOWN EMPLOY ADDRESS DEBTINC CREDDEBT OTHDEBT DEFAULTER DEFAULTERNEW   N   Percent       WOE         IV
1   1 523     Kochi      4       7     4.1     0.29    0.49         0            1 242 0.3457143 -0.443048 0.07452269
2   1 376     Kochi      1       4     2.5     0.13    0.29         0            1 242 0.3457143 -0.443048 0.07452269
3   1  39    Jaipur      1       8    17.1     1.34    2.77         1            0 242 0.3457143 -0.443048 0.07452269
4   1 201 Ahmedabad      3       7     4.1     0.26    0.52         0            1 242 0.3457143 -0.443048 0.07452269
5   1 245   Kolkata      3       4    13.3     1.60    3.05         0            1 242 0.3457143 -0.443048 0.07452269
6   1  46    Kanpur      0       1     6.8     0.15    0.94         0            1 242 0.3457143 -0.443048 0.07452269
```

Binary Logistic Model

```
# Binary Logistic Model with AGE as FACTOR
```

```
riskmodel1<-glm(DEFAULTER~AGE+EMPLOY+ADDRESS+DEBTINC+
                 CREDDEBT+OTHDEBT,
               family=binomial,data=data)

summary(riskmodel1)
```

```
# Binary Logistic Model using 'WOE' as a predictor instead of 'AGE'
```

```
riskmodel2<-glm(DEFAULTER~WOE+EMPLOY+ADDRESS+DEBTINC+
                 CREDDEBT+OTHDEBT,
               family=binomial,data=leftjoin)

summary(riskmodel2)
```

Binary Logistic Model

Output :

```
> summary(riskmodel1)

Call:
glm(formula = DEFAULTER ~ AGE + EMPLOY + ADDRESS + DEBTINC +
    CREDDEBT + OTHDEBT, family = binomial, data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3495  -0.6601  -0.2974   0.2509   2.8583

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.78821    0.26407  -2.985  0.00284 **
AGE2         0.25202    0.26651   0.946  0.34433
AGE3         0.62707    0.36056   1.739  0.08201 .
EMPLOY      -0.26172    0.03188  -8.211  < 2e-16 ***
ADDRESS     -0.09964    0.02234  -4.459 8.22e-06 ***
DEBTINC      0.08506    0.02212   3.845  0.00012 ***
CREDDEBT     0.56336    0.08877   6.347 2.20e-10 ***
OTHDEBT      0.02315    0.05709   0.405  0.68517
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 804.36  on 699  degrees of freedom
Residual deviance: 553.41  on 692  degrees of freedom
AIC: 569.41

Number of Fisher Scoring iterations: 6
```

Interpretation:

- Model with Age as factor creates 2 dummy variables.
- P values for both dummy variables are greater than 0.05. Therefore, the impact of AGE is statistically insignificant.

Binary Logistic Model

Output :

```
> summary(riskmodel2)

Call:
glm(formula = DEFAULTER ~ WOE + EMPLOY + ADDRESS + DEBTINC +
    CREDDEBT + OTHDEBT, family = binomial, data = leftjoin)

Deviance Residuals:
    Min       1Q    Median       3Q      Max
-2.3634  -0.6484  -0.3069   0.2472   2.9116

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.58887    0.28711  -2.051 0.040263 *
WOE          0.48221    0.36301   1.328 0.184048
EMPLOY      -0.26104    0.03187  -8.190 2.62e-16 ***
ADDRESS     -0.09535    0.02205  -4.325 1.53e-05 ***
DEBTINC      0.08242    0.02197   3.752 0.000176 ***
CREDDEBT     0.57151    0.08857   6.452 1.10e-10 ***
OTHDEBT      0.02922    0.05665   0.516 0.606014
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 804.36  on 699  degrees of freedom
Residual deviance: 554.64  on 693  degrees of freedom
AIC: 568.64

Number of Fisher Scoring iterations: 6
```

Interpretation:

- Model with WOE as a predictor gives one P value
- The P value for WOE is greater than 0.05. Therefore, the impact of AGE is statistically insignificant.

WoE of Character Variable

```
# WoE and IV for variable 'TOWN'
```

```
IV$Tables$TOWN
```

```
# Output :
```

```
> IV$Tables$TOWN
        TOWN  N    Percent         WOE         IV
1  Ahmedabad 34 0.04857143 -0.80216794 0.03627141
2  Bengaluru 35 0.05000000  0.17783860 0.03778456
3    Chennai 33 0.04714286 -0.20564760 0.03987340
4      Delhi 36 0.05142857  0.78599257 0.06552734
5  Hyderabad 40 0.05714286  0.34773764 0.07184911
6     Indore 39 0.05571429  0.16541608 0.07331250
7     Jaipur 40 0.05714286 -0.19125886 0.07549575
8     Kanpur 52 0.07428571  0.16541608 0.07744694
9      Kochi 66 0.09428571 -0.13284810 0.07916282
10   Kolkata 63 0.09000000 -0.27308888 0.08629521
11   Lucknow 51 0.07285714 -0.06669614 0.08662442
12    Mumbai 58 0.08285714 -0.24004903 0.09166334
13    Nagpur 59 0.08428571  0.12904844 0.09302324
14      Pune 50 0.07142857  0.22710965 0.09650390
15     Surat 44 0.06285714  0.46552068 0.10856864
```

WoE of Numeric Variable

```
# WoE and IV for variable 'EMPLOY'
IV$Tables$EMPLOY
```

- By default, **create_infotables()** categorizes numeric variable into 10 bins. You can change the number of bins by specifying bins='n' in the function.

```
# Output :
```

```
> IV$Tables$EMPLOY
    EMPLOY  N    Percent        WOE        IV
1    [0,0] 62 0.08857143 -1.1030952 0.1288816
2    [1,1] 49 0.07000000 -0.5817983 0.1555268
3    [2,3] 86 0.12285714 -0.9920367 0.2987787
4    [4,4] 47 0.06714286 -0.1811065 0.3010739
5    [5,6] 82 0.11714286  0.0277947 0.3011638
6    [7,8] 69 0.09857143  0.6239910 0.3336590
7   [9,10] 75 0.10714286  0.2663920 0.3407685
8  [11,13] 83 0.11857143  0.6449892 0.3822789
9  [14,17] 70 0.10000000  0.8750926 0.4424923
10 [18,31] 77 0.11000000  1.4323637 0.5922371
```

WoE of Numeric Variable

```
# WoE and IV for variable 'EMPLOY' with 3 bins
```

```
IV <- create_infotables(data=data, y="DEFAULTERNEW", bins = 3)
IV$Tables$EMPLOY
```

```
# Output :
```

```
> IV$Tables$EMPLOY
   EMPLOY   N   Percent        WOE        IV
1   [0,3] 197 0.2814286 -0.9267653 0.2845505
2   [4,9] 243 0.3471429  0.1441387 0.2915113
3 [10,31] 260 0.3714286  0.8898857 0.5217636
```

Information Value (IV) Interpretation

```
# Extracting IV for all predictor variables
```

```
IV <- create_infotables(data=data, y="DEFAULTERNEW")

IV_Value = data.frame(IV$Summary)
IV_Value
```

- **create_infotable** generates Tables and Summary objects.
- **Tables** object used earlier to extract WoE and IV for individual variables.
- **Summary** object contains IV for all predictor variables.

Information Value Interpretation

Output :

```
> IV_Value
   Variable        IV
6   DEBTINC 0.7871927
4    EMPLOY 0.5922371
5   ADDRESS 0.3359295
7  CREDDEBT 0.2835522
8   OTHDEBT 0.1453887
2       AGE 0.1212061
3      TOWN 0.1085686
1        SN 0.0424855
9 DEFAULTER 0.0000000
```

Interpretation:

- We will not consider IV for SN and DEFAULTER as they are not the predictor variables.
- With the help of table '**IV values and its Predictive Power**' on slide number 8 we can say that,
  - Town and Age are weak predictor.
  - Othdebt, Creddebt, Address have medium predictive power.
  - Employ and Debtinc are strong predictor.

Quick Recap

In this session, we learnt how to **compute and use Weight of Evidence and Information Value:**

Weight of Evidence	• Tells the predictive power of an independent variable in relation to the dependent variable • $\ln((\text{Distribution of Good}_i)/(\text{Distribution of Bad}_i))$
Information Value	• Expresses the amount of diagnostic information of that variable for separating the Goods from the Bads • $\sum(\text{Distribution of Good}_i - \text{Distribution of Bad}_i) \times \text{WoE}$
Weight of Evidence and Information Value in R	• Package "**Information**" in R contains functions for calculating weights of evidence, information value. • Function `create_infotables()` generates Tables and Summary objects. • Tables object used to extract WoE and IV of individual variable . • Summary object gives list of variables and its corresponding IV.