# Bayesian Networks Assignment

Benjamin Garcia

2025-04-14

# Data Prep

```
#remove.packages('bnlearn')
#install.packages('bnlearn')
library(readr)
loan_data <- read_csv("loan_data.csv")
```

```
## Rows: 24000 Columns: 7
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): Text, Employment_Status, Approval
## dbl (4): Income, Credit_Score, Loan_Amount, DTI_Ratio
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
summary(loan_data)
```

```
##      Text               Income        Credit_Score    Loan_Amount    
##  Length:24000       Min.   : 20001   Min.   :300.0   Min.   :  1005  
##  Class :character   1st Qu.: 65636   1st Qu.:437.0   1st Qu.: 16212  
##  Mode  :character   Median :110464   Median :575.0   Median : 35207  
##                     Mean   :110378   Mean   :575.7   Mean   : 44356  
##                     3rd Qu.:155187   3rd Qu.:715.0   3rd Qu.: 65623  
##                     Max.   :200000   Max.   :850.0   Max.   :158834  
##    DTI_Ratio      Employment_Status    Approval        
##  Min.   :  2.53   Length:24000       Length:24000      
##  1st Qu.: 14.51   Class :character   Class :character  
##  Median : 24.86   Mode  :character   Mode  :character  
##  Mean   : 34.72                                        
##  3rd Qu.: 41.84                                        
##  Max.   :246.33                                        
```

```
#install.packages('installr')
#installr::updateR()
#install.packages('bnlearn')
#install.packages('rlang', dependencies = T)
```

```
#loan_data$Employment_Status <- as.factor(ifelse(loan_data$Employment_Status == 'employed', 1,
0))
#loan_data$Approval <- as.factor(ifelse(loan_data$Approval == 'Approved' , 1, 0))
```

For employment status, 1 if employed, 0 if not employed. For approval, 1 if approved, 0 if not. Ended up not using these columns because aracne does not run with categorical/factor data.

```
#BiocManager::install('Rgraphviz')
#install.packages('Rgraphviz')
library(Rgraphviz)
```

```
## Loading required package: graph
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: generics
```

```
##
## Attaching package: 'generics'
```

```
## The following objects are masked from 'package:base':
##
##     as.difftime, as.factor, as.ordered, intersect, is.element, setdiff,
##     setequal, union
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, saveRDS, table, tapply, unique,
##     unsplit, which.max, which.min
```

```
## Loading required package: grid
```

# Build Models

Score Based Algorithm - Hill-climbing

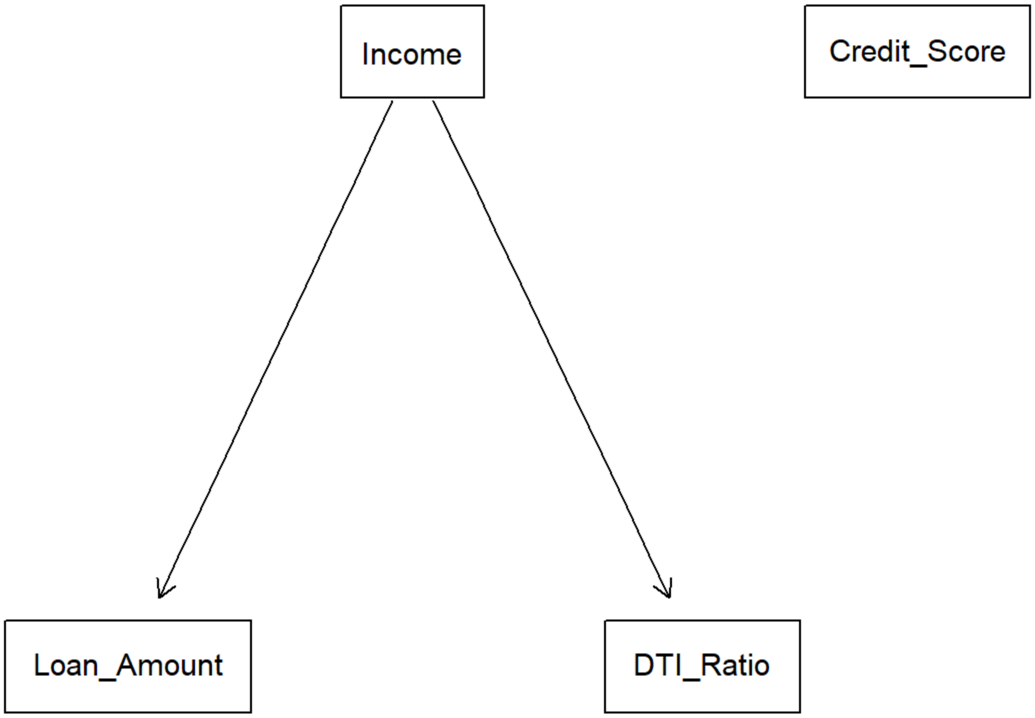
Constraint Based Algorithm - Incremental Association with FDR

Hybrid Algorithm - Hybrid HPC

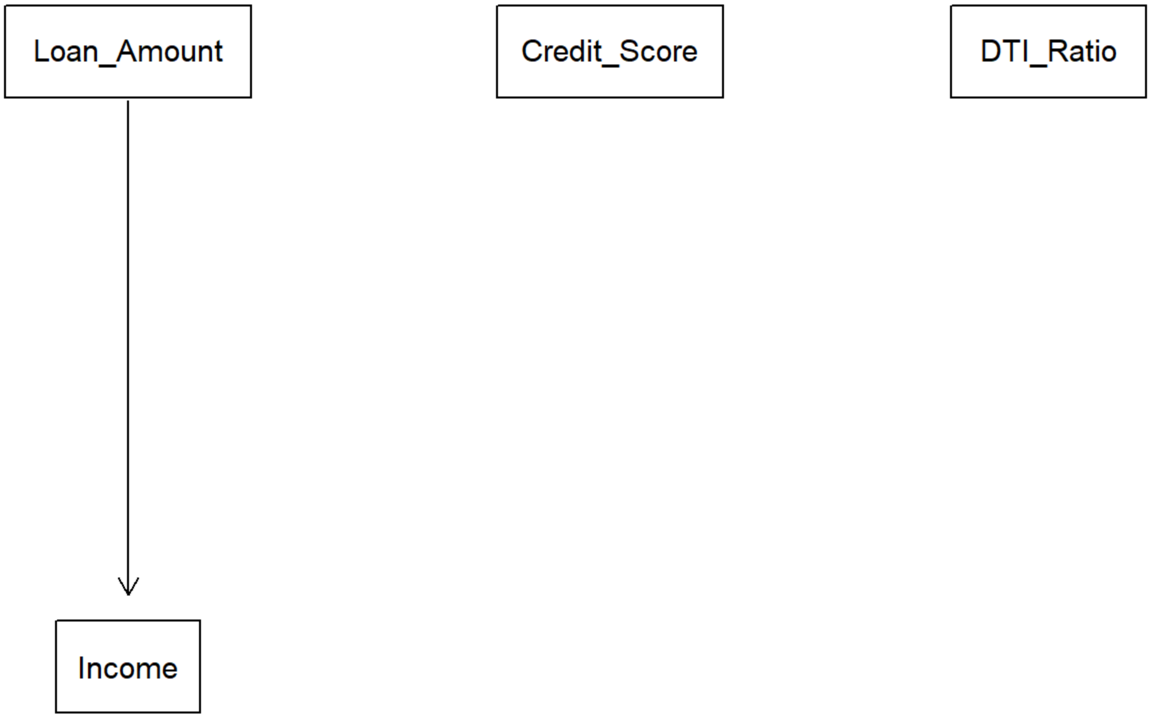
Local Discovery Algorithm - ARACNE

```
library(bnlearn)
model_hc <- bnlearn::hc(loan_data[,2:5])
model_iamb_fdr <- bnlearn::iamb.fdr(loan_data[,2:5])
model_h2pc <- bnlearn:::h2pc(loan_data[,2:5])
model_aracne <- bnlearn::aracne(loan_data[,2:5])
#bnlearn::graphviz.plot(model_hc)
```

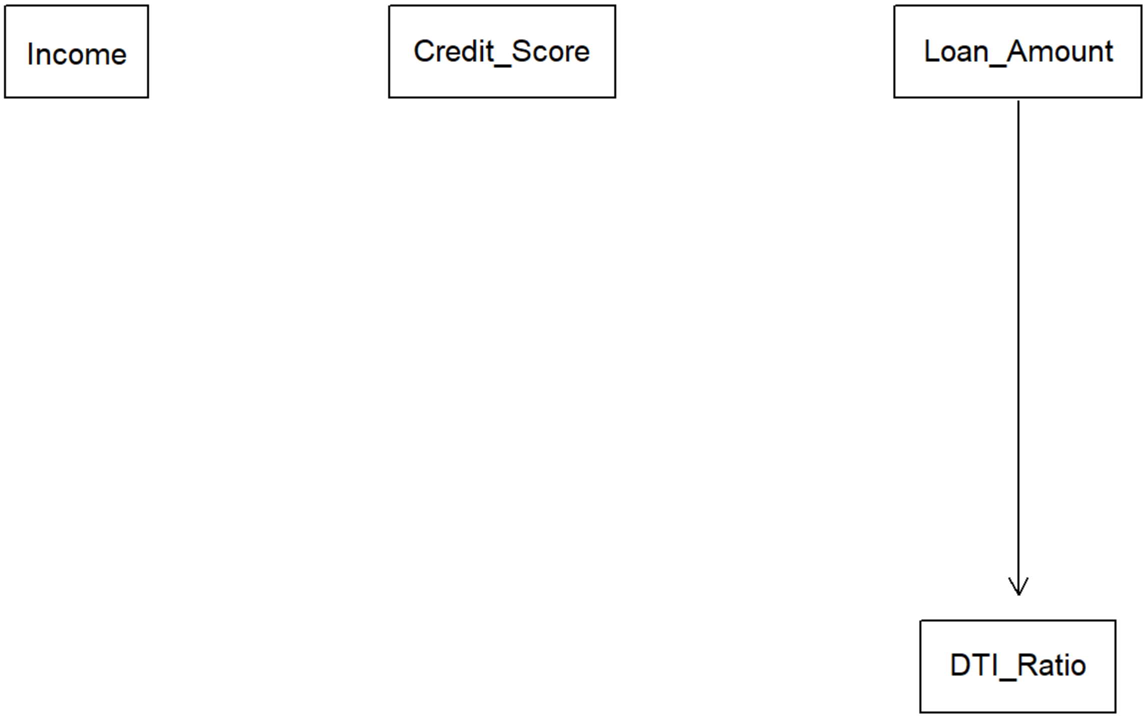
```
graphviz.plot(model_hc)
```



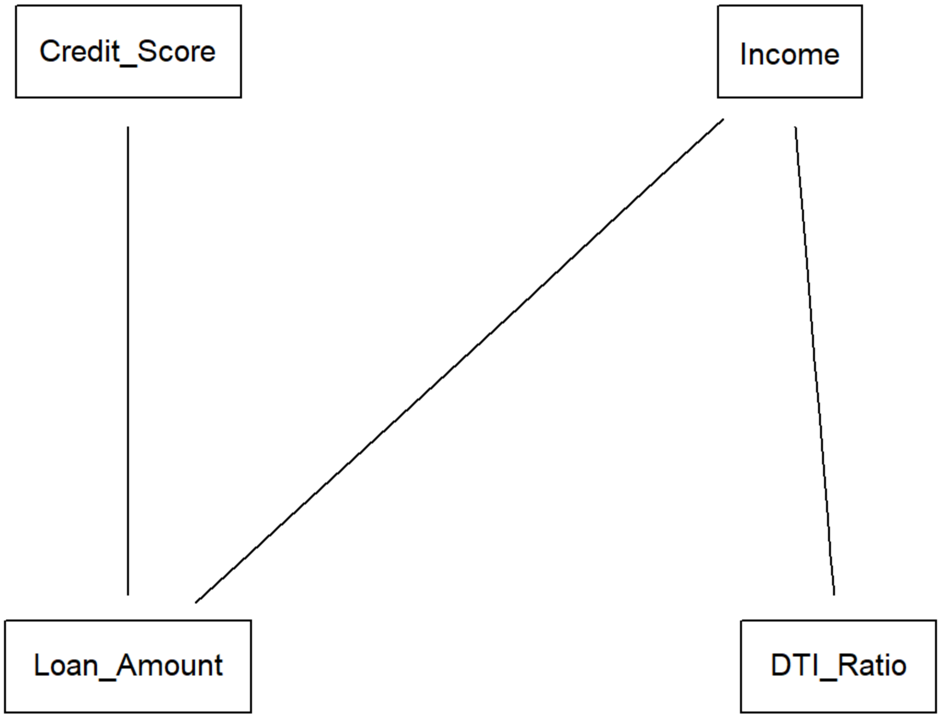
```
graphviz.plot(set.arc(model_iamb_fdr, from = 'Loan_Amount', to = 'Income'))
```



```
graphviz.plot(model_h2pc)
```



```
graphviz.plot(model_aracne)
```



```
arcs(model_aracne)
```

```
##      from          to
## [1,] "Income"       "Loan_Amount"
## [2,] "Loan_Amount"  "Income"
## [3,] "Income"       "DTI_Ratio"
## [4,] "DTI_Ratio"    "Income"
## [5,] "Credit_Score" "Loan_Amount"
## [6,] "Loan_Amount"  "Credit_Score"
```

```
M_arcs <- arcs(model_hc)
M_arcs2 <- arcs(model_iamb_fdr)
M_arcs3 <- arcs(model_h2pc)
M_arcs4 <- arcs(model_aracne)
```

```
#M_arcs5 <- M_arcs4
```

```
model_iamb_fdr <- set.arc(
x = model_iamb_fdr,
from = M_arcs2[1,1],
to = M_arcs2[1,2],
check.cycles = FALSE,
check.illegal = FALSE
)


model_aracne <- set.arc(
x = model_aracne,
from = M_arcs4[1,1],
to = M_arcs4[1,2],
)

model_aracne <- set.arc(
x = model_aracne,
from = M_arcs4[3,1],
to = M_arcs4[3,2],
)

model_aracne <- set.arc(
x = model_aracne,
from = M_arcs4[5,1],
to = M_arcs4[5,2],
)
```

```
arcs(model_aracne)
```

```
##      from           to
## [1,] "Income"       "Loan_Amount"
## [2,] "Income"       "DTI_Ratio"
## [3,] "Credit_Score" "Loan_Amount"
```

# Score Models

```
M_Score <- data.frame(Method = c('hc', 'iamb.fdr', 'h2pc', 'aracne'), Score = c(NA, NA, NA, NA))
```

```
M_Score[1,2] <- score(x = model_hc, data = loan_data[,2:5], type = 'bic-g')
M_Score[2,2] <- score(x = model_iamb_fdr, data = loan_data[,2:5], type = 'bic-g')
M_Score[3,2] <- score(x = model_h2pc, data = loan_data[,2:5], type = 'bic-g')
M_Score[4,2] <- score(x = model_aracne, data = loan_data[,2:5], type = 'bic-g')
```

The network-score I selected was ‘bic-g’. Since I had a gaussian bayesian network with just continuous variables (income, Credit_Score, Loan_Amount, and DTI_Ratio) I chose one of the gaussian scoring methods. I went with bic over aic, loglik and other scoring methods because I have relatively few variables, a somewhat large amount of observations (24,000), and am looking for the simplest model. BIC gives me the best chance to select the best simplest model given my data.

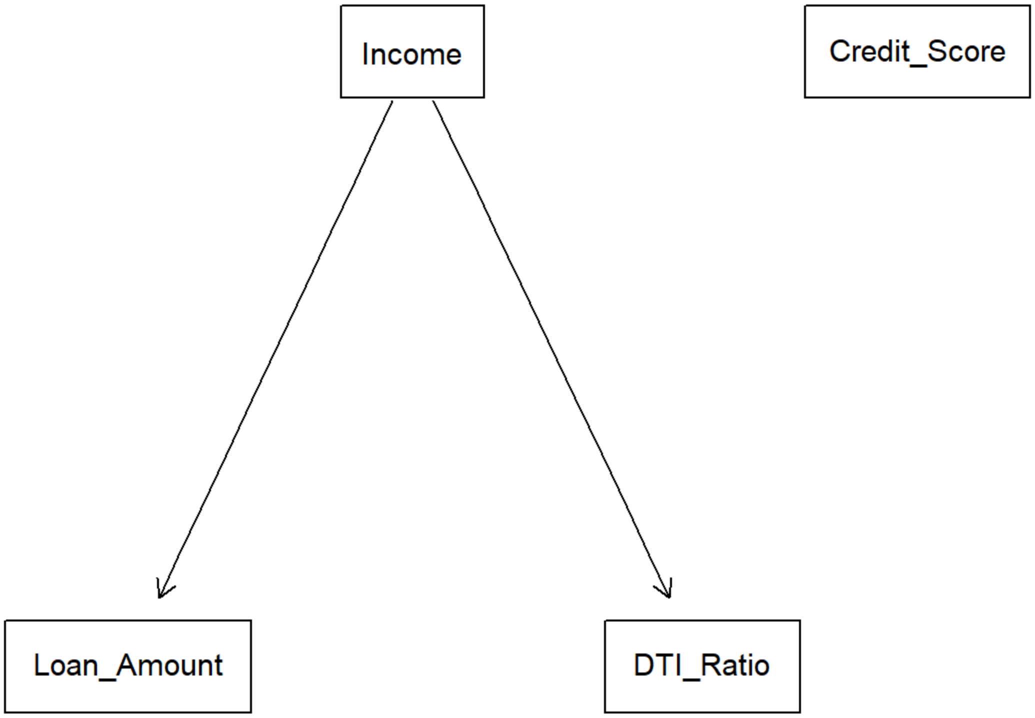
```
M_Score <- M_Score[order(M_Score$Score, decreasing = T),]
M_Score
```

```
##     Method     Score
## 1       hc -840872.6
## 4   aracne -840876.7
## 2 iamb.fdr -847604.6
## 3     h2pc -850798.7
```

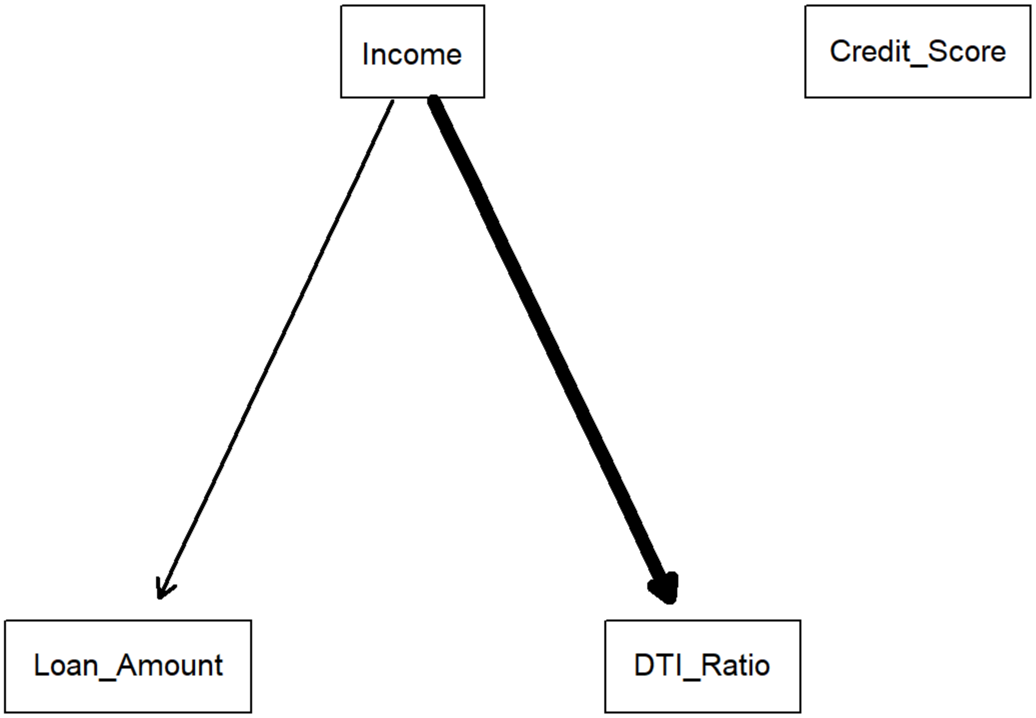
According to the table hc or hill-climbing algorithm which is a score-based algorithm did the best. Hc is the method with the greatest score value or value that is closest to zero since they are all negative.

# Visualize final model

```
graphviz.plot(model_hc)
```



```
strength_loan <- arc.strength(
x = model_hc,
data = loan_data[,2:5])
strength.plot(x = model_hc, strength = strength_loan)
```



# Predict and Evaluate Fit

```
bn_loan <- bn.fit(
x = model_hc,
data = loan_data[,2:5]
)
```

```
bn_loan_pred <- predict(
object = bn_loan,
data = loan_data[,2:5],
node = colnames(loan_data[,4]))
```

Mean Squared-Error (MSE)

```
mean((bn_loan_pred - loan_data$Loan_Amount )^2)
```

```
## [1] 783107169
```

The high MSE value indicates that the bayesian network model may not be best for this data. We must be aware of the extremely low score values for the different bayesian network methods. These values may indicate that the variables do not have causal or correlated relationship. While there may be some kind of relationship, it is hard for the bayesian methods to use the limited data to discover it. I would not recommend a bayesian network over a tradition linear regression or other machine learning methods for this data. Despite, the large amount of observations, the limiting factor may be the use of 4 continuous variables since the other 2 binary variables could not be used with the aracne method.