Causal Inference in Fraud Detection

1. Data Preprocessing

The dataset I will be working with has 1 million observations and 32 variables, both numeric and categorical. Out of them, 'fraud_bool' is the response variable holding binary values.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(fastDummies)
```

```
## Warning: package 'fastDummies' was built under R version 4.4.3
```

```
data = read.csv('Nigerian_Fraud_Dataset_Base.csv')
str(data)
```

```
## 'data.frame':    1000000 obs. of  32 variables:
##  $ fraud_bool                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ income                          : num  0.3 0.8 0.8 0.6 0.9 0.6 0.2 0.8 0.3 0.8 ...
##  $ name_email_similarity           : num  0.987 0.617 0.997 0.475 0.842 ...
##  $ prev_address_months_count       : int  -1 -1 9 11 -1 -1 22 -1 21 -1 ...
##  $ current_address_months_count    : int  25 89 14 14 29 369 4 103 2 134 ...
##  $ customer_age                    : int  40 20 40 30 40 30 40 40 30 20 ...
##  $ days_since_request              : num  0.00674 0.0101 0.01232 0.00699 5.74263 ...
##  $ intended_balcon_amount          : num  102.45 -0.85 -1.49 -1.86 47.15 ...
##  $ payment_type                    : chr  "AA" "AD" "AB" "AB" ...
##  $ zip_count_4w                    : int  1059 1658 1095 3483 2339 1204 1998 1548 1781 3113 ...
##  $ velocity_6h                     : num  13096 9223 4471 14432 7602 ...
##  $ velocity_24h                    : num  7851 5745 5472 6755 5124 ...
##  $ velocity_4w                     : num  6742 5942 5993 5970 5941 ...
##  $ bank_branch_count_8w            : int  5 3 15 11 1 705 28 6 2 14 ...
##  $ date_of_birth_distinct_emails_4w: int  5 18 11 13 6 5 8 7 10 20 ...
##  $ employment_status               : chr  "CB" "CA" "CA" "CA" ...
##  $ credit_risk_score               : int  163 154 89 90 91 134 72 163 35 201 ...
```

```
##  $ email_is_free                    : int  1 1 1 1 0 1 1 0 0 1 ...
##  $ housing_status                   : chr  "BC" "BC" "BC" "BC" ...
##  $ phone_home_valid                 : int  0 1 0 0 1 1 1 1 1 1 ...
##  $ phone_mobile_valid               : int  1 1 1 1 1 1 1 1 0 1 ...
##  $ bank_months_count                : int  9 2 30 1 26 30 1 25 2 15 ...
##  $ has_other_cards                  : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ proposed_credit_limit            : num  1500 1500 200 200 200 200 200 200 200 1500 ...
##  $ foreign_request                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ source                           : chr  "INTERNET" "INTERNET" "INTERNET" "INTERNET" ...
##  $ session_length_in_minutes        : num  16.22 3.36 22.73 15.22 3.74 ...
##  $ device_os                        : chr  "linux" "other" "windows" "linux" ...
##  $ keep_alive_session               : int  1 1 0 1 0 1 1 1 1 1 ...
##  $ device_distinct_emails_8w        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ device_fraud_count               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ month                            : int  0 0 0 0 0 0 0 0 0 0 ...
```

This dataset is highly imbalanced with nearly 99% data as genuine transactions.

```r
data$fraud_bool %>% as.factor() %>% summary
```

```
##      0      1
## 988971  11029
```

Because variable 'device_fraud_count' has only '0' value, this variable will be omitted.

```r
data <- data %>% select(-device_fraud_count)
```

Since Bayesian Network learning requires numeric variables, 5 categorical features of the dataset will be converted using one hot coding.

This process results in 26 new dummies, increasing the feature count to 52 (excluding 5 originally categorical features).

```r
cat_vars_ls <- c("payment_type", "employment_status", "housing_status",
                 "source", "device_os")
data <- dummy_cols(data, select_columns = cat_vars_ls,
                   remove_selected_columns = TRUE)
```

Since numeric variables of the dataset have different data ranges, I will apply Min-Max standardisation to scale them within range [0:1].

To make it simple, all 52 variables will be standardised since they all hold numeric values. The binary features still remain intact.

```r
df_num <- data %>%
  mutate(across(everything(), ~ (.-min(.)) / (max(.) - min(.))))
str(df_num)
```

```
## 'data.frame':    1000000 obs. of  52 variables:
##  $ fraud_bool                       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ income                           : num  0.25 0.875 0.875 0.625 1 0.625 0.125 0.875 0.25 0.875 ...
##  $ name_email_similarity            : num  0.987 0.617 0.997 0.475 0.842 ...
```

```
## $ prev_address_months_count      : num 0 0 0.026 0.0312 0 ...
## $ current_address_months_count   : num 0.0606 0.2098 0.035 0.035 0.0699 ...
## $ customer_age                   : num 0.375 0.125 0.375 0.25 0.375 0.25 0.375 0.375 0.25 0.125 .
## $ days_since_request             : num 8.58e-05 1.29e-04 1.57e-04 8.91e-05 7.32e-02 ...
## $ intended_balcon_amount         : num 0.918 0.114 0.109 0.106 0.488 ...
## $ zip_count_4w                   : num 0.158 0.247 0.163 0.52 0.349 ...
## $ velocity_6h                    : num 0.786 0.556 0.275 0.865 0.46 ...
## $ velocity_24h                   : num 0.798 0.542 0.508 0.665 0.466 ...
## $ velocity_4w                    : num 0.939 0.747 0.76 0.754 0.747 ...
## $ bank_branch_count_8w           : num 0.002096 0.001258 0.006289 0.004612 0.000419 ...
## $ date_of_birth_distinct_emails_4w: num 0.128 0.462 0.282 0.333 0.154 ...
## $ credit_risk_score              : num 0.596 0.58 0.463 0.465 0.467 ...
## $ email_is_free                  : num 1 1 1 1 0 1 1 0 0 1 ...
## $ phone_home_valid               : num 0 1 0 0 1 1 1 1 1 1 ...
## $ phone_mobile_valid             : num 1 1 1 1 1 1 1 1 0 1 ...
## $ bank_months_count              : num 0.303 0.0909 0.9394 0.0606 0.8182 ...
## $ has_other_cards                : num 0 0 0 0 0 0 0 1 0 0 ...
## $ proposed_credit_limit          : num 0.68586 0.68586 0.00524 0.00524 0.00524 ...
## $ foreign_request                : num 0 0 0 0 0 0 0 0 0 0 ...
## $ session_length_in_minutes      : num 0.1982 0.0502 0.2731 0.1866 0.0546 ...
## $ keep_alive_session             : num 1 1 0 1 0 1 1 1 1 1 ...
## $ device_distinct_emails_8w      : num 0.667 0.667 0.667 0.667 0.667 ...
## $ month                          : num 0 0 0 0 0 0 0 0 0 0 ...
## $ payment_type_AA                : num 1 0 0 0 1 0 0 0 0 0 ...
## $ payment_type_AB                : num 0 0 1 1 0 0 1 1 1 0 ...
## $ payment_type_AC                : num 0 0 0 0 0 0 0 0 0 0 ...
## $ payment_type_AD                : num 0 1 0 0 0 1 0 0 0 1 ...
## $ payment_type_AE                : num 0 0 0 0 0 0 0 0 0 0 ...
## $ employment_status_CA           : num 0 1 1 1 1 0 1 1 1 1 ...
## $ employment_status_CB           : num 1 0 0 0 0 1 0 0 0 0 ...
## $ employment_status_CC           : num 0 0 0 0 0 0 0 0 0 0 ...
## $ employment_status_CD           : num 0 0 0 0 0 0 0 0 0 0 ...
## $ employment_status_CE           : num 0 0 0 0 0 0 0 0 0 0 ...
## $ employment_status_CF           : num 0 0 0 0 0 0 0 0 0 0 ...
## $ employment_status_CG           : num 0 0 0 0 0 0 0 0 0 0 ...
## $ housing_status_BA              : num 0 0 0 0 0 0 0 0 0 0 ...
## $ housing_status_BB              : num 0 0 0 0 0 0 0 0 0 0 ...
## $ housing_status_BC              : num 1 1 1 1 1 0 1 0 1 0 ...
## $ housing_status_BD              : num 0 0 0 0 0 0 0 0 0 1 ...
## $ housing_status_BE              : num 0 0 0 0 0 1 0 1 0 0 ...
## $ housing_status_BF              : num 0 0 0 0 0 0 0 0 0 0 ...
## $ housing_status_BG              : num 0 0 0 0 0 0 0 0 0 0 ...
## $ source_INTERNET                : num 1 1 1 1 1 1 1 1 1 1 ...
## $ source_TELEAPP                 : num 0 0 0 0 0 0 0 0 0 0 ...
## $ device_os_linux                : num 1 0 0 1 0 1 0 0 0 0 ...
## $ device_os_macintosh            : num 0 0 0 0 0 0 0 0 0 0 ...
## $ device_os_other                : num 0 1 0 0 1 0 0 1 1 1 ...
## $ device_os_windows              : num 0 0 1 0 0 0 0 0 0 0 ...
## $ device_os_x11                  : num 0 0 0 0 0 0 1 0 0 0 ...
```

2. Modelling

a. Learning a Bayesian Network structure

I will use the PC-algorithm to generate a completed partially DAG with all 51 predictors.

The learning has taken for a while to produce a DAG of 276 edges within which 15 edges were left undirected.

```
library(bnlearn)
library(pcalg)
```

```
##
## Attaching package: 'pcalg'

## The following objects are masked from 'package:bnlearn':
##
##     dsep, pdag2dag, shd, skeleton
```

```
set.seed(789)
cpdag <- pc(suffStat = list(C = cor(df_num[ ,-1]), n = nrow(df_num)),
            indepTest = gaussCItest, labels = colnames(df_num[ ,-1]),
            alpha= .01, u2pd = "rand")
cpdag
```

```
## Object of class 'pcAlgo', from Call:
## pc(suffStat = list(C = cor(df_num[, -1]), n = nrow(df_num)),
##     indepTest = gaussCItest, alpha = 0.01, labels = colnames(df_num[,
##         -1]), u2pd = "rand")
## Number of undirected edges:  15
## Number of directed edges:    257
## Total number of edges:       272
```

```
# Check if cpdag1 is a valid CPDAG
cpdag_amat <- as(cpdag, "amat")
isValidGraph(cpdag_amat, type = "cpdag", verbose = TRUE)
```

```
## [1] TRUE
```

The code below is for visualisation's convenience.The above DAG is then exported to a pdf file.

```
# Export cpdag to a bn object for visualisation
cpdag_bn <- as.bn(cpdag, check.cycles = TRUE)

# Function to split variable names into 2 lines
split_var_names <- function(names) {
  sapply(names, function(name) {
    n <- nchar(name)
    if (n > 1) {
      mid <- ceiling(n / 2)
      paste(substr(name, 1, mid), substr(name, mid + 1, n), sep = "\n")
    } else {
```

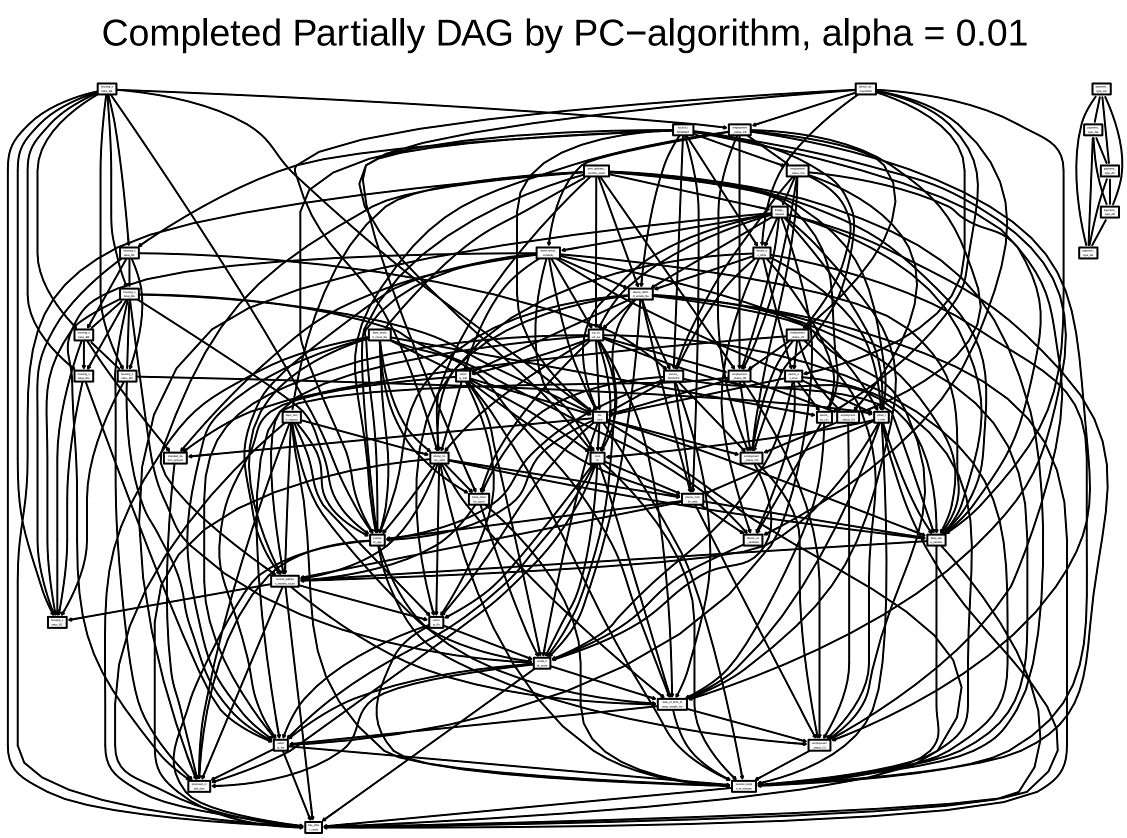
```
      name
    }
  })
}

# Apply the splitting function to the node names
original_names <- nodes(cpdag_bn)
new_names <- split_var_names(original_names)
nodes(cpdag_bn) <- new_names

# Plot the CPDAG with adjusted variable names
pdf("cpdag_plot.pdf", width = 11, height = 8.5)

graphviz.plot(cpdag_bn, layout = "dot", fontsize = 15, main = 'Completed Partially DAG by PC-algorithm,
```

```
## Loading required namespace: Rgraphviz
```

```
dev.off()
```

```
## pdf
##   2
```

```
graphviz.plot(cpdag_bn, layout = "dot", fontsize = 15, main = 'Completed Partially DAG by PC-algorithm,
```



b. Fraud Detection Model The PC-simple algorithm is used to find the immediate nodes surrounding

the class variable 'fraud_bool'. This method is widely accepted as a feature reduction tool for classification models.

```
# # Find the parent and children set of class variable
feature_select <- pcSelect(df_num[,1], df_num[,-1], alpha = 0.01)

df_feature_select <- data.frame(
  Variable = names(feature_select$G),  # Extract variable names
  Selected = feature_select$G,         # Extract TRUE/FALSE values
  zMin = feature_select$zMin,           # Extract zMin values
  row.names = NULL
)
head(df_feature_select)
```

```
##                        Variable Selected       zMin
## 1                        income     TRUE 17.4430261
## 2         name_email_similarity     TRUE 27.8653245
## 3     prev_address_months_count     TRUE 13.7694487
## 4 current_address_months_count    FALSE  1.8771027
## 5                  customer_age     TRUE 15.4186049
## 6            days_since_request    FALSE  0.5672787
```

From the PC-simple, 17 variables are identified as the most important predictors to incorporate into the classifier.

```
selected_vars_ls <- df_feature_select$Variable[df_feature_select$Selected]
selected_vars_ls
```

```
##  [1] "income"                           "name_email_similarity"
##  [3] "prev_address_months_count"        "customer_age"
##  [5] "date_of_birth_distinct_emails_4w" "credit_risk_score"
##  [7] "email_is_free"                    "phone_home_valid"
##  [9] "has_other_cards"                  "proposed_credit_limit"
## [11] "foreign_request"                  "keep_alive_session"
## [13] "device_distinct_emails_8w"        "payment_type_AC"
## [15] "employment_status_CB"             "housing_status_BA"
## [17] "device_os_windows"
```

A Naive Bayes model is trained using the above 17 input variables, optimised via 10-fold cross validation and SMOTE sampling to address imbalanced class.

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(naivebayes)
```

```
## naivebayes 1.0.0 loaded
```

```
## For more information please visit:

## https://majkamichal.github.io/naivebayes/
```

```r
df_num$fraud_bool <- as.factor(df_num$fraud_bool)

train_ctr <- trainControl(method = "cv", number = 10, sampling = "smote")
nb <- train(fraud_bool ~ ., data = df_num[, c(selected_vars_ls, "fraud_bool")],
                   method = "naive_bayes", trControl = train_ctr)
```

```
## Warning: package 'themis' was built under R version 4.4.3

## Loading required package: recipes

##
## Attaching package: 'recipes'

## The following object is masked from 'package:bnlearn':
##
##     discretize

## The following object is masked from 'package:stats':
##
##     step
```

```r
print(nb)
```

```
## Naive Bayes
##
## 1000000 samples
##      17 predictor
##       2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 900000, 900000, 900000, 900001, 900000, 900000, ...
## Addtional sampling using SMOTE
##
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy  Kappa
##   FALSE      0.776999  0.04978839
##    TRUE      0.949820  0.14814922
##
## Tuning parameter 'laplace' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = TRUE
##  and adjust = 1.
```

The model produces 2 results corresponding to whether to use of kernel density estimation (KDE). The outperforming accuracy in the case of KDE suggests that the input variables do not abide by normal distribution. In addition, the improved Kappa shows that the KDE model is better to deal with this imbalanced dataset.