library(tidyverse)  
library(caret)  
library(lubridate)

# Load data

df0=read\_csv("C:\\Users\\Arnold\\OneDrive\\R\_Python\_working\_directory\\Loan\_payments\_data.csv")

# Look at the structure

str(df0)

## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 500 obs. of 11 variables:  
## $ Loan\_ID : chr "xqd20166231" "xqd20168902" "xqd20160003" "xqd20160004" ...  
## $ loan\_status : chr "PAIDOFF" "PAIDOFF" "PAIDOFF" "PAIDOFF" ...  
## $ Principal : num 1000 1000 1000 1000 1000 300 1000 1000 1000 800 ...  
## $ terms : num 30 30 30 15 30 7 30 30 30 15 ...  
## $ effective\_date: chr "9/8/2016" "9/8/2016" "9/8/2016" "9/8/2016" ...  
## $ due\_date : chr "10/7/2016" "10/7/2016" "10/7/2016" "9/22/2016" ...  
## $ paid\_off\_time : chr "9/14/2016 19:31" "10/7/2016 9:00" "9/25/2016 16:58" "9/22/2016 20:00" ...  
## $ past\_due\_days : num NA NA NA NA NA NA NA NA NA NA ...  
## $ age : num 45 50 33 27 28 35 29 36 28 26 ...  
## $ education : chr "High School or Below" "Bechalor" "Bechalor" "college" ...  
## $ Gender : chr "male" "female" "female" "male" ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. Loan\_ID = col\_character(),  
## .. loan\_status = col\_character(),  
## .. Principal = col\_double(),  
## .. terms = col\_double(),  
## .. effective\_date = col\_character(),  
## .. due\_date = col\_character(),  
## .. paid\_off\_time = col\_character(),  
## .. past\_due\_days = col\_double(),  
## .. age = col\_double(),  
## .. education = col\_character(),  
## .. Gender = col\_character()  
## .. )

# Drop columns I won’t use

df=df0 %>% select(-Loan\_ID,-effective\_date,-loan\_status,-past\_due\_days)  
str(df)

## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 500 obs. of 7 variables:  
## $ Principal : num 1000 1000 1000 1000 1000 300 1000 1000 1000 800 ...  
## $ terms : num 30 30 30 15 30 7 30 30 30 15 ...  
## $ due\_date : chr "10/7/2016" "10/7/2016" "10/7/2016" "9/22/2016" ...  
## $ paid\_off\_time: chr "9/14/2016 19:31" "10/7/2016 9:00" "9/25/2016 16:58" "9/22/2016 20:00" ...  
## $ age : num 45 50 33 27 28 35 29 36 28 26 ...  
## $ education : chr "High School or Below" "Bechalor" "Bechalor" "college" ...  
## $ Gender : chr "male" "female" "female" "male" ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. Loan\_ID = col\_character(),  
## .. loan\_status = col\_character(),  
## .. Principal = col\_double(),  
## .. terms = col\_double(),  
## .. effective\_date = col\_character(),  
## .. due\_date = col\_character(),  
## .. paid\_off\_time = col\_character(),  
## .. past\_due\_days = col\_double(),  
## .. age = col\_double(),  
## .. education = col\_character(),  
## .. Gender = col\_character()  
## .. )

# Get data in shape

df = df %>% mutate(Principal=ordered(Principal),due\_date=mdy(due\_date),terms=ordered(  
terms),paid\_date=date(mdy\_hm(paid\_off\_time)),education=factor(education,ordered =  
T,levels = c("High School or Below","college","Bechalor","Master or Above")),  
Gender=factor(Gender)) %>% mutate(paid\_on\_time=factor(ifelse(paid\_date>due\_date | is.na(  
paid\_date),'N',"Y"))) %>% select(-due\_date,-paid\_date,-paid\_off\_time)

# Look at summary

summary(df)

## Principal terms age education   
## 300 : 6 7 : 21 Min. :18.00 High School or Below:209   
## 500 : 3 15:207 1st Qu.:27.00 college :220   
## 700 : 1 30:272 Median :30.00 Bechalor : 67   
## 800 :111 Mean :31.12 Master or Above : 4   
## 900 : 2 3rd Qu.:35.00   
## 1000:377 Max. :51.00   
## Gender paid\_on\_time  
## female: 77 N:201   
## male :423 Y:299   
##   
##   
##   
##

# It makes more sense to cut age into 10 years intervals

df=df %>% mutate(age=cut(age,breaks = c(1:4,5.2)\*10))

# Get row numbers for training data

trnidx=createDataPartition(df$paid\_on\_time,p = .7,list = F)  
trndf=df[trnidx,]  
ttdf=df[-trnidx,]

# Classification model to predict whether a person will pay on time or not.

## Naive Bayes model

f=formula(paid\_on\_time~.)  
tg=expand.grid(laplace=0:3,usekernel=c(T,F),adjust=1:3)  
ctr=trainControl(method = 'cv',number = 10)  
(nb=train(form=f,data=trndf,tuneGrid=tg,method='naive\_bayes'))

## Naive Bayes   
##   
## 351 samples  
## 5 predictor  
## 2 classes: 'N', 'Y'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 351, 351, 351, 351, 351, 351, ...   
## Resampling results across tuning parameters:  
##   
## laplace usekernel adjust Accuracy Kappa   
## 0 FALSE 1 0.5349528 0.08807129  
## 0 FALSE 2 0.5349528 0.08807129  
## 0 FALSE 3 0.5349528 0.08807129  
## 0 TRUE 1 0.5398972 0.04173839  
## 0 TRUE 2 0.5398972 0.04173839  
## 0 TRUE 3 0.5398972 0.04173839  
## 1 FALSE 1 0.5349528 0.08807129  
## 1 FALSE 2 0.5349528 0.08807129  
## 1 FALSE 3 0.5349528 0.08807129  
## 1 TRUE 1 0.5398972 0.04173839  
## 1 TRUE 2 0.5398972 0.04173839  
## 1 TRUE 3 0.5398972 0.04173839  
## 2 FALSE 1 0.5349528 0.08807129  
## 2 FALSE 2 0.5349528 0.08807129  
## 2 FALSE 3 0.5349528 0.08807129  
## 2 TRUE 1 0.5398972 0.04173839  
## 2 TRUE 2 0.5398972 0.04173839  
## 2 TRUE 3 0.5398972 0.04173839  
## 3 FALSE 1 0.5349528 0.08807129  
## 3 FALSE 2 0.5349528 0.08807129  
## 3 FALSE 3 0.5349528 0.08807129  
## 3 TRUE 1 0.5398972 0.04173839  
## 3 TRUE 2 0.5398972 0.04173839  
## 3 TRUE 3 0.5398972 0.04173839  
##   
## 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.

report=data.frame(Model='Naive Bayes','Test Accuracy'=postResample(pred = predict(  
nb,ttdf),obs = ttdf$paid\_on\_time)[[1]])

## Random Forest model

(rf=train(form=f,data=trndf,tuneLength=5,method='rf'))

## Random Forest   
##   
## 351 samples  
## 5 predictor  
## 2 classes: 'N', 'Y'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 351, 351, 351, 351, 351, 351, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.5715284 0.03129396  
## 5 0.5347346 0.02167812  
## 8 0.5362160 0.03670507  
## 11 0.5344969 0.03210383  
## 14 0.5362947 0.03864316  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

report=rbind(report,data.frame(Model='Random Forest','Test Accuracy'=postResample(pred =  
predict(rf,ttdf),obs = ttdf$paid\_on\_time)[[1]]))

## Kernal SVM model

(svm=train(form=f,data=trndf,tuneGrid=expand.grid(C=seq(0.001,2.5,length.out = 5),  
sigma=seq(0,1,length.out = 5)),method='svmRadial'))

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 351 samples  
## 5 predictor  
## 2 classes: 'N', 'Y'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 351, 351, 351, 351, 351, 351, ...   
## Resampling results across tuning parameters:  
##   
## C sigma Accuracy Kappa   
## 0.00100 0.00 0.6023388 0.00000000  
## 0.00100 0.25 0.6023388 0.00000000  
## 0.00100 0.50 0.6023388 0.00000000  
## 0.00100 0.75 0.6023388 0.00000000  
## 0.00100 1.00 0.6023388 0.00000000  
## 0.62575 0.00 0.6023388 0.00000000  
## 0.62575 0.25 0.5565033 0.02703921  
## 0.62575 0.50 0.5546694 0.02803827  
## 0.62575 0.75 0.5541802 0.02869712  
## 0.62575 1.00 0.5532779 0.02752703  
## 1.25050 0.00 0.6023388 0.00000000  
## 1.25050 0.25 0.5494909 0.02247418  
## 1.25050 0.50 0.5504347 0.02409250  
## 1.25050 0.75 0.5504347 0.02409250  
## 1.25050 1.00 0.5504347 0.02409250  
## 1.87525 0.00 0.6023388 0.00000000  
## 1.87525 0.25 0.5482107 0.02079658  
## 1.87525 0.50 0.5504347 0.02409250  
## 1.87525 0.75 0.5504347 0.02409250  
## 1.87525 1.00 0.5504347 0.02409250  
## 2.50000 0.00 0.6023388 0.00000000  
## 2.50000 0.25 0.5482107 0.02079658  
## 2.50000 0.50 0.5504347 0.02409250  
## 2.50000 0.75 0.5504347 0.02409250  
## 2.50000 1.00 0.5504347 0.02409250  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were sigma = 1 and C = 0.001.

report=rbind(report,data.frame(Model='Kernal SVM','Test Accuracy'=postResample(pred =  
predict(svm,ttdf),obs = ttdf$paid\_on\_time)[[1]]))

# Look at the proportion of people pay on time and not on time

ttdf %>% group\_by(paid\_on\_time) %>% tally() %>% mutate(p=n/sum(n))

## # A tibble: 2 x 3  
## paid\_on\_time n p  
## <fct> <int> <dbl>  
## 1 N 60 0.403  
## 2 Y 89 0.597

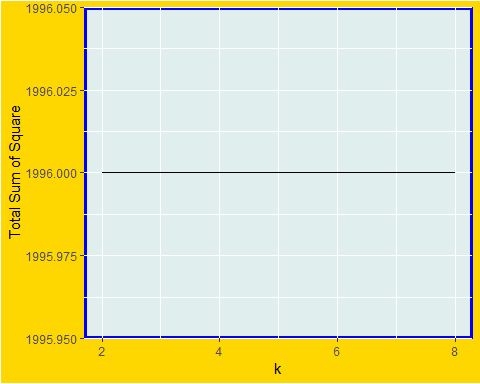
# Model comparison

report

## Model Test.Accuracy  
## 1 Naive Bayes 0.5033557  
## 2 Random Forest 0.5704698  
## 3 Kernal SVM 0.5973154

# K means clustering

kmdf=df0 %>% mutate(education=ordered(education,levels=c("High School or Below",  
"college","Bechalor","Master or Above")) %>% as.numeric()) %>% select(  
Principal,terms,age,education)  
pp = preProcess(kmdf, method = c("center", "scale"))  
kmdf=predict(pp,kmdf)  
tss=c()  
km=c()  
for (i in 2:8){  
 mod=kmeans(kmdf,i)  
 km = c(km,mod)  
 tss= c(tss,mod$totss)  
}  
data.frame(k=2:8,tss) %>% ggplot(aes(k,tss)) + geom\_line() + ylab('Total Sum of Square')



# Association rules minning

library(arules)  
rules = apriori(df, parameter = list(supp = 0.01, conf = 0.6),appearance = list(  
 rhs="paid\_on\_time=N")) %>% sort(by="confidence", decreasing=TRUE)

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.6 0.1 1 none FALSE TRUE 5 0.01 1  
## maxlen target ext  
## 10 rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 5   
##   
## set item appearances ...[1 item(s)] done [0.00s].  
## set transactions ...[21 item(s), 500 transaction(s)] done [0.00s].  
## sorting and recoding items ... [17 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 5 6 done [0.00s].  
## writing ... [6 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

inspect(rules)

## lhs rhs support confidence lift count  
## [1] {terms=30,   
## age=(20,30],   
## education=Bechalor} => {paid\_on\_time=N} 0.014 0.6363636 1.582994 7  
## [2] {Principal=1000,   
## terms=30,   
## age=(20,30],   
## education=Bechalor} => {paid\_on\_time=N} 0.014 0.6363636 1.582994 7  
## [3] {terms=30,   
## age=(20,30],   
## education=Bechalor,   
## Gender=male} => {paid\_on\_time=N} 0.014 0.6363636 1.582994 7  
## [4] {Principal=1000,   
## terms=30,   
## age=(20,30],   
## education=Bechalor,   
## Gender=male} => {paid\_on\_time=N} 0.014 0.6363636 1.582994 7  
## [5] {terms=15,   
## age=(40,52],   
## education=college} => {paid\_on\_time=N} 0.010 0.6250000 1.554726 5  
## [6] {age=(20,30],   
## education=Bechalor,   
## Gender=male} => {paid\_on\_time=N} 0.028 0.6086957 1.514168 14

# 