People's Interactive Classification Assignment

Saurav

April 30, 2017

Problem Statement: To predict the probability that a person X will buy a product Y, given a set of demographic related features available about person X and features available around X's past activities. (NOTE: All the independent variables ahs been masked)

1. Loading the dataset:

```
train<-read.table("ClassificationProblem1.txt",sep = "\t",header = T)

#Converting Outcome to catogories
train$C<-as.factor(train$C)

test<-read.table("Classification1Test.txt",sep = "\t",header = T)
```

2. Structure of datasets:

```
str(train)
```

```
## 'data.frame':    101180 obs. of  24 variables:
##  $ Index: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ F1   : num  0.225 0.321 0.893 0.321 0.476 ...
##  $ F2   : num  0.5 0.281 0.622 0.957 0.623 ...
##  $ F3   : num  0.49 0.907 0.999 0.346 0.545 ...
##  $ F4   : num  0.9024 0.7722 0.0984 0.6465 0.1597 ...
##  $ F5   : int  7934 -8238 8540 -7772 1571 -6554 -9455 7089 554 8952 ...
##  $ F6   : int  -6970 1219 5266 -383 -8039 8770 -9937 2404 -3388 6923 ...
##  $ F7   : int  -5714 1663 -9377 9681 -7961 1065 4079 3157 1279 3112 ...
##  $ F8   : int  9982 1287 -3504 -8661 -2385 -9720 8178 5484 4381 -7115 ...
##  $ F9   : int  -5697 -3658 -4511 3474 4407 5801 -663 -2829 -8957 -1413 ...
##  $ F10  : num  4.23e+09 -1.15e+09 5.95e+09 -5.72e+09 -3.10e+09 ...
##  $ F11  : num  -3.92e+09 -6.84e+09 6.88e+09 -6.01e+09 -9.76e+09 ...
##  $ F12  : num  3.16e+08 1.38e+09 -9.92e+09 6.55e+09 7.59e+08 ...
##  $ F13  : num  6.18e+09 -9.03e+09 -5.61e+09 -4.70e+09 9.98e+09 ...
##  $ F14  : num  -3.43e+09 6.09e+08 -8.98e+09 4.87e+09 9.76e+09 ...
##  $ F15  : Factor w/ 8031 levels "1/1/1978","1/1/1979",..: 1254 3070 77
2919 4062 5811 1009 2508 812 295 ...
##  $ F16  : Factor w/ 6300 levels "1/1/1981","1/1/1984",..: 6239 445 1323
4227 712 272 905 3163 3258 4074 ...
##  $ F17  : int  2 1 2 1 1 4 1 1 2 1 ...
##  $ F18  : int  1 1 1 1 1 1 2 1 1 2 ...
##  $ F19  : int  706 423 703 122 486 806 448 187 701 502 ...
##  $ F20  : int  305 206 315 304 240 157 702 123 34 706 ...
##  $ F21  : int  1 18 1 15 1 6 5 3 5 1 ...
```

```
##  $ F22  : int  2 7 4 1 1 5 1 1 1 1 ...
##  $ C    : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1 ...
```

3. Exploratory data analysis:

Let's look at the distribution of binary outcome:

```
library('ggplot2')

qplot(C,
      main = "Distribution of Binary outcome classes",
      xlab = "Buy or Not",
      data=train)
```

Class Imbalance, as expected with more proportion of :

```
base::table(train$C)/length(train$C)
```

```
##
##         0         1
## 0.7546254 0.2453746
```

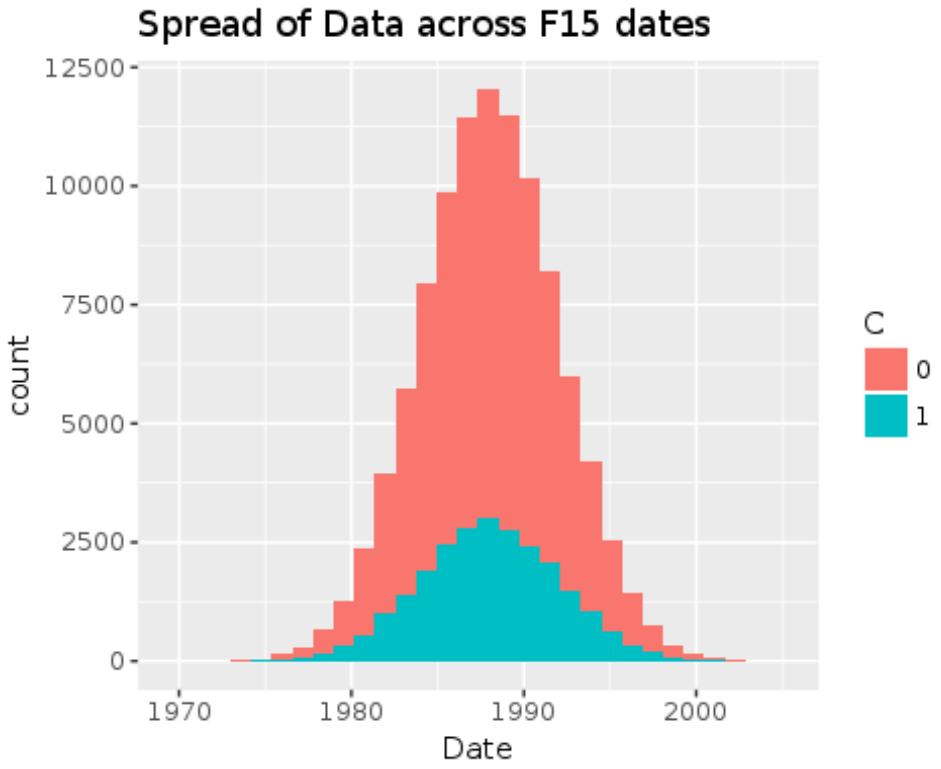
Its hard to comment for any feature with concrete coments but on first sight, F15 and F16 clearly looks to be date feilds. So let's convert them to date time feilds:

```
train$F15<-as.Date(train$F15,"%m/%d/%Y")
train$F16<-as.Date(train$F16,"%m/%d/%Y")
```

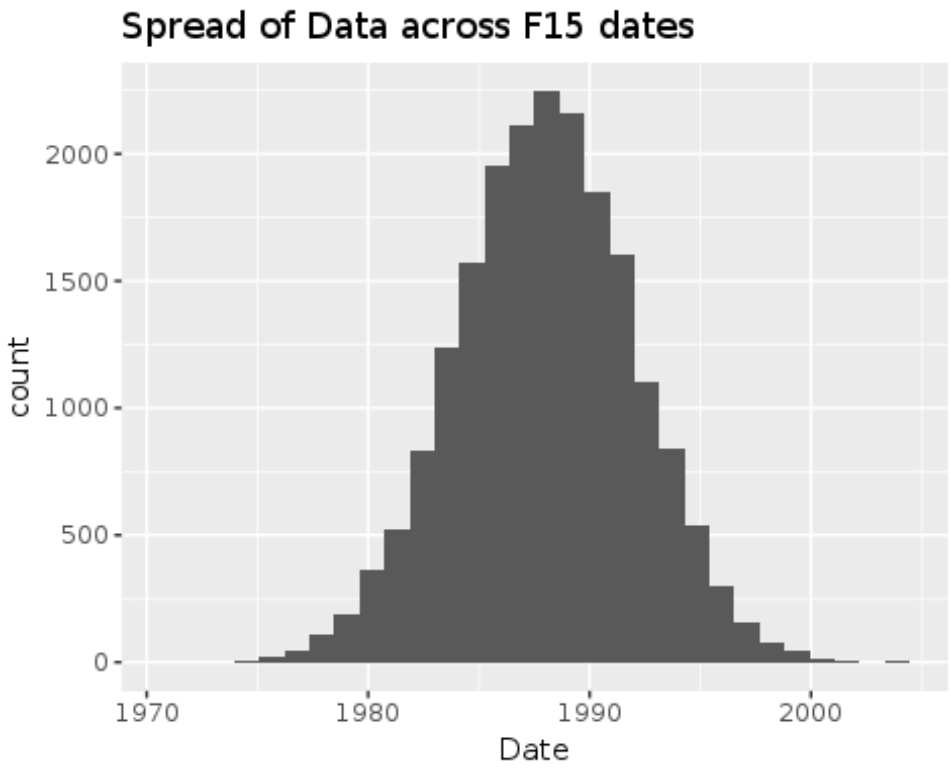
```
test$F15<-as.Date(test$F15,"%m/%d/%Y")
test$F16<-as.Date(test$F16,"%m/%d/%Y")
```

Let's look at the distribution of these data variables for train and test. Let's start with F15.

```
qplot(F15,
      main = "Spread of Data across F15 dates",
      xlab = "Date",
      fill=C,
      data=train)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



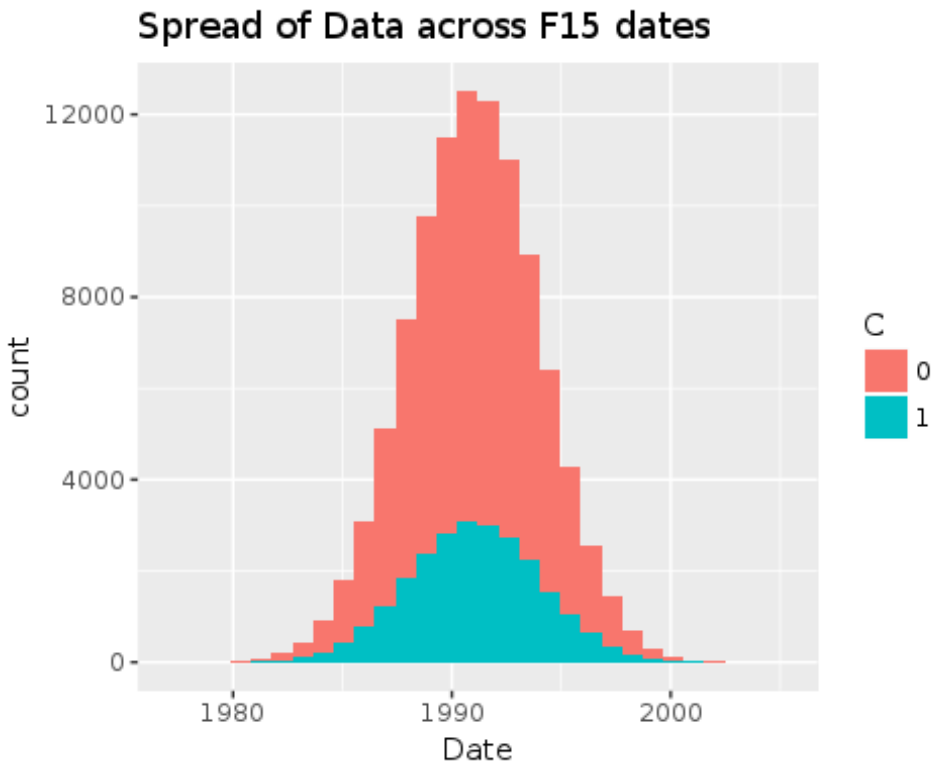

```
qplot(F15,
      main = "Spread of Data across F15 dates",
      xlab = "Date",
      data=test)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



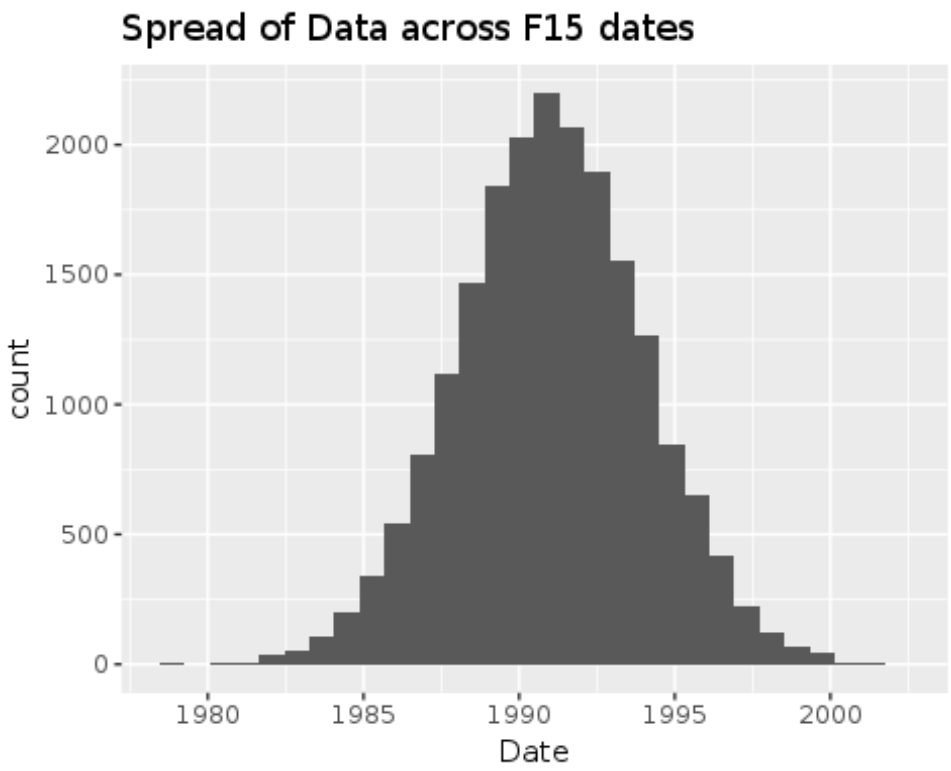
The train and test dates for F15 are in good sync. Lets see F16.

```
qplot(F16,
      main = "Spread of Data across F15 dates",
      xlab = "Date",
      fill=C,
      data=train)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
qplot(F16,
      main = "Spread of Data across F15 dates",
      xlab = "Date",
      data=test)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

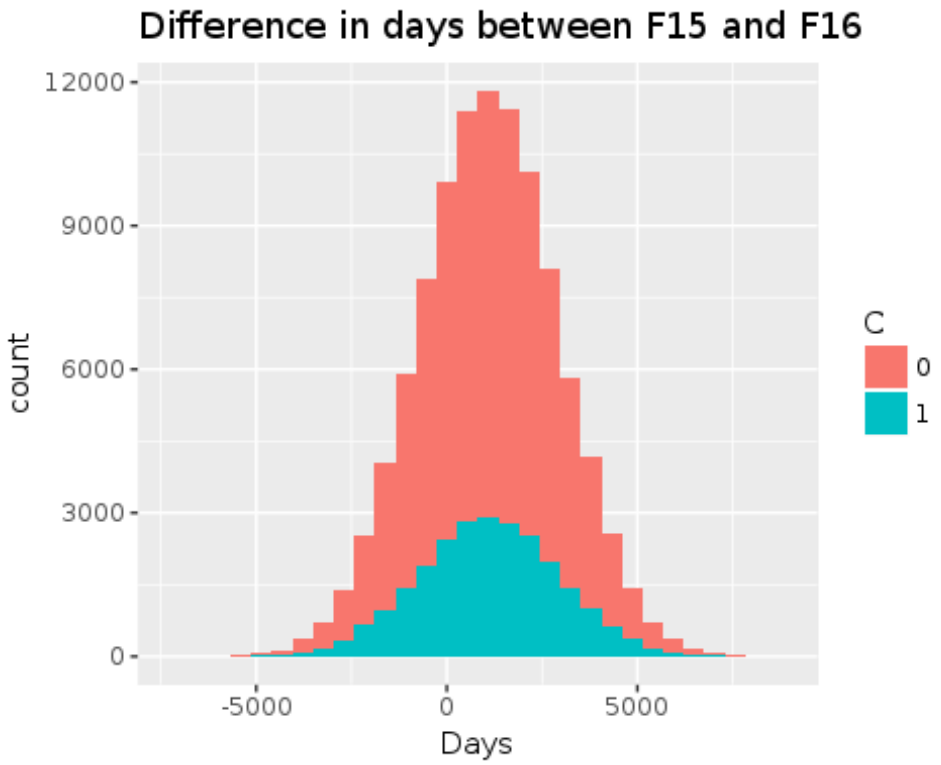


Again simmilar distribution of date across train and test. Therefore we can conclude that the train and test distribution is not time based, instead a random sampling probably. Therefore, its a fair expectation that train and test will have simmilar outcome distributions.

Let's find try to make some sense of of F15 and F16 by looking at their differences in days for train and test datasets.

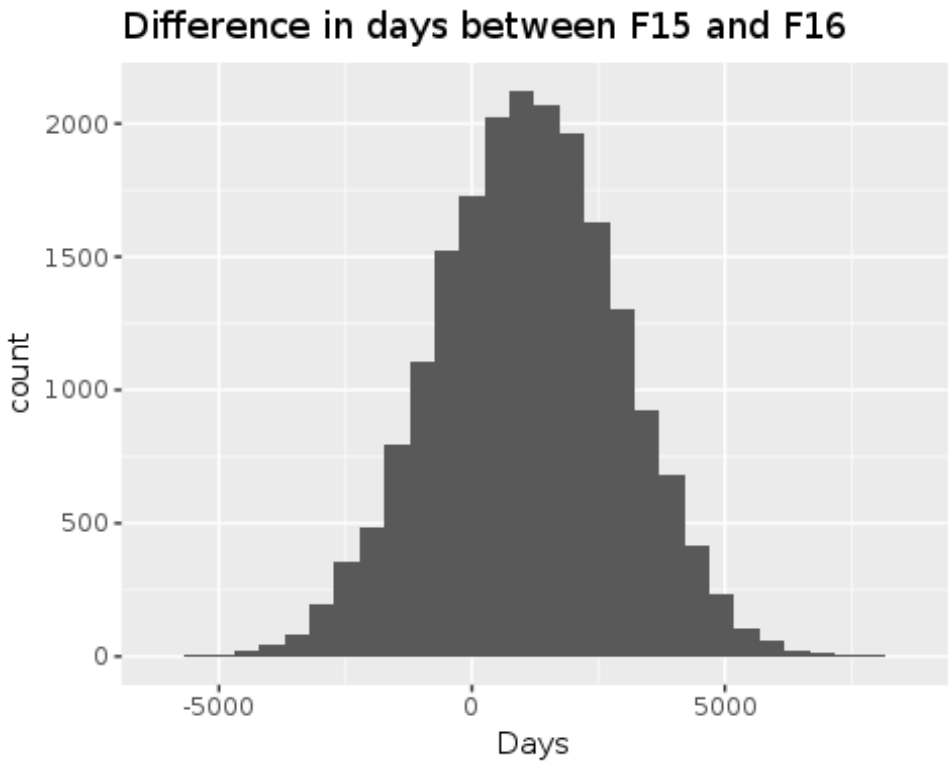
```
qplot(as.numeric(difftime(train$F16,train$F15,units = "days")),
      main = "Difference in days between F15 and F16",
      xlab = "Days",
      fill=C,
      data=train)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Let's look for the same in test data.

```
qplot(as.numeric(difftime(test$F16,test$F15,units = "days")),
      main = "Difference in days between F15 and F16",
      xlab = "Days",
      data=test)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

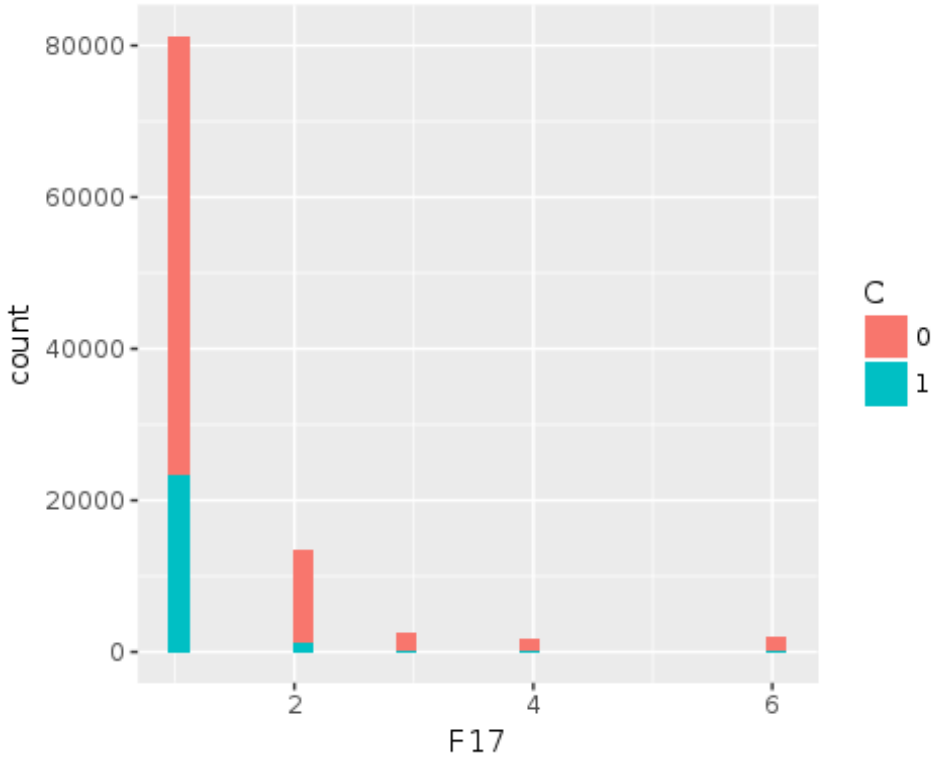


Again, it confirms that the test is randomly sampled.

Conclustion1: Using a K fold CV/ Stratified validation set will be more appropriate compared to using a time based validation set.

While exploring found the feature F17, F18, F21 and F22 to be catogorical.

```
qplot(F17,
      fill=C,
      data=train)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The porportions of outcome variable for each of the levels is different, let's look at the split of each catogory:

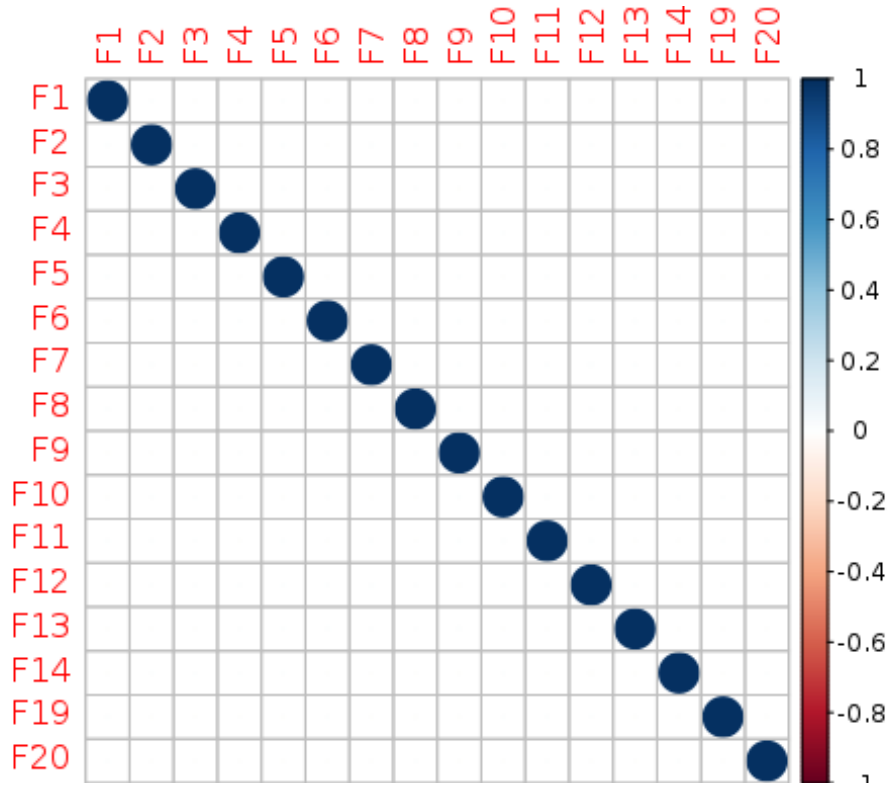
```
summary(as.factor(train$F17))/length(train$F17)

##          1          2          3          4          6
## 0.80371615 0.13378138 0.02397707 0.01738486 0.02114054
```

Let's convert F17, F18, F21 and F22 to catogorical and we'll use One hot encoding as without context, we're unsure whether these are ordinal or not.

```
train$F17<-as.factor(train$F17)
train$F18<-as.factor(train$F18)
train$F21<-as.factor(train$F21)
train$F22<-as.factor(train$F22)

test$F17<-as.factor(test$F17)
test$F18<-as.factor(test$F18)
test$F21<-as.factor(test$F21)
test$F22<-as.factor(test$F22)
```

All the other features are numerical. Let's check out the corelation between varaibles. This will help us identify multicolinearity. Multicolinearity is not that big of a problem for tree based algorithms, but if present, will be really really problametic for linear models.

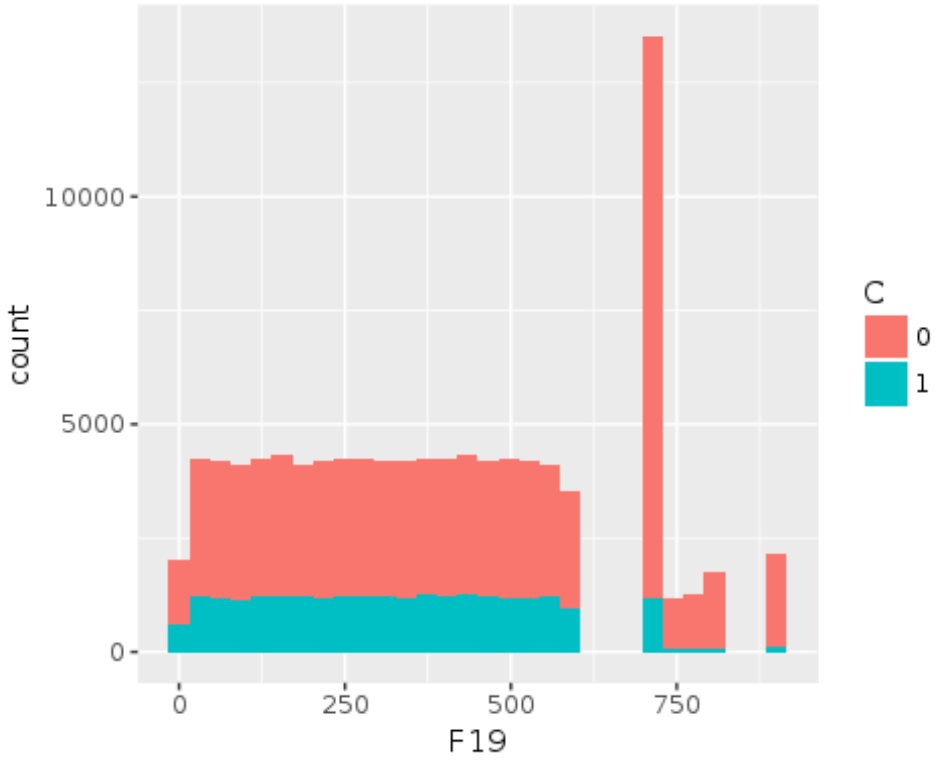
```
library(corrplot)

corrplot(cor(train[,c(-1,-16,-17,-18,-19,-22,-23,-24)]))
```



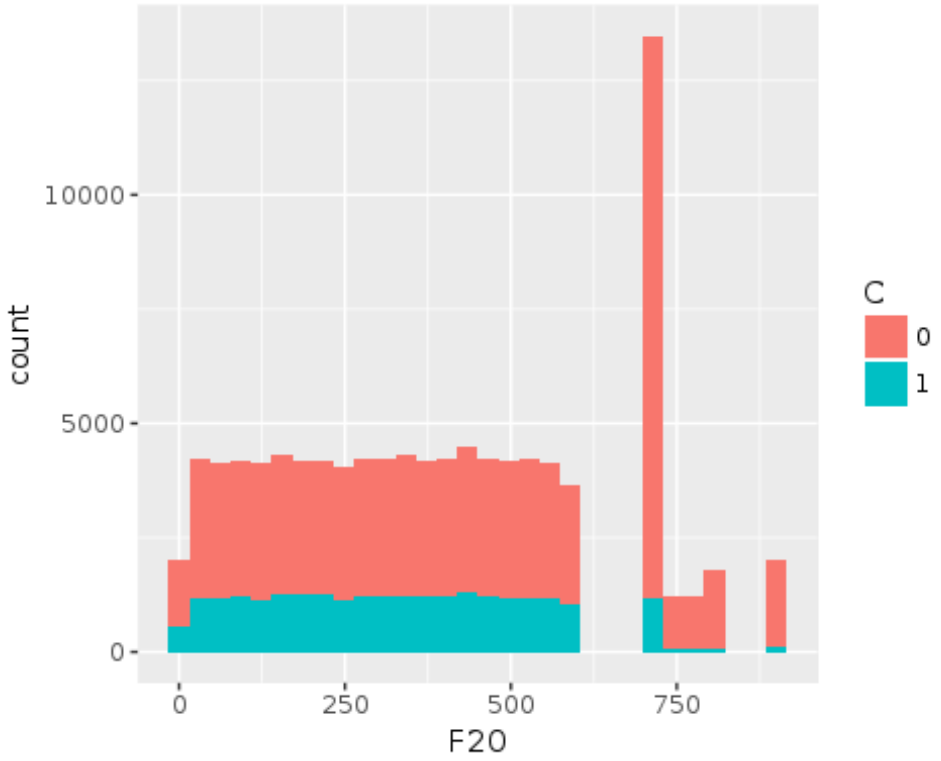
Fortunately, there's no multicolinearty.

Also, there is something really strange with the distribution of F19 and F20.

```
qplot(F19,
      fill=C,
      data=train)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
qplot(F20,
      fill=C,
      data=train)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The distribution is really ackward and will be raelly interesting to know what is causing it given the unmasked dataset.

Conclusion2: Although, one really crucial thing I found by this EDA is that most features especially F19 and F20 and does not exhibit a linear relationship with the outcome variable. Therefore, we'll use tree based models.

4. Data Cleaning

Let's are there any missing values?

```
library('VIM')
```

```
## Loading required package: colorspace
```

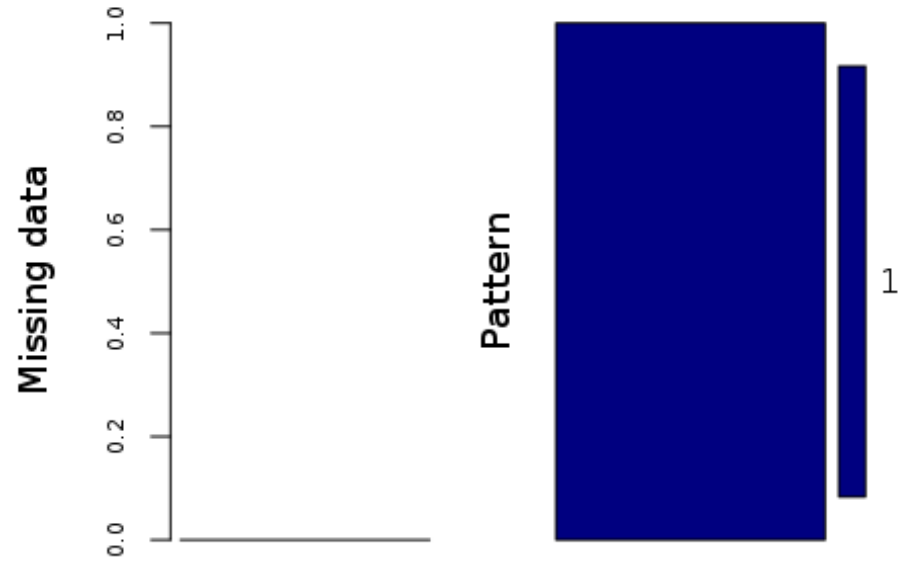
```
## Loading required package: grid
```

```
## Loading required package: data.table
```

```
## VIM is ready to use.
##  Since version 4.0.0 the GUI is in its own package VIMGUI.
##
##           Please use the package to use the new (and old) GUI.
```

```
## Suggestions and bug-reports can be submitted at:
https://github.com/alexkowa/VIM/issues
```

```
##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep

mice_plot <- aggr(all, col=c('navyblue','yellow'),
                  numbers=TRUE, sortVars=FALSE,
                  labels=names(all), cex.axis=.7,
                  gap=3, ylab=c("Missing data","Pattern"))

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
## 'closure'

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
## 'closure'
```



No missing values as it seems. Let's als see if test has missing values?

```
sum(is.na(test))

## [1] 0
```

Also, features aren't skewed in any direction, therefore no need preprocess them.

5. Feature Engineering

It's not easy to create features logically with the masked feature names which does not let anything out. Let's first start with creating features from individual date feilds, F15 and F16

```
train$day1<-as.numeric(strftime(train$F15,"%d"))
train$month1<-as.numeric(strftime(train$F15,"%m"))
train$year1<-as.numeric(strftime(train$F15,"%Y"))
train$weekday1<-as.factor(weekdays(train$F15))

test$day1<-as.numeric(strftime(test$F15,"%d"))
test$month1<-as.numeric(strftime(test$F15,"%m"))
test$year1<-as.numeric(strftime(test$F15,"%Y"))
test$weekday1<-as.factor(weekdays(test$F15))

train$day2<-as.numeric(strftime(train$F16,"%d"))
train$month2<-as.numeric(strftime(train$F16,"%m"))
train$year2<-as.numeric(strftime(train$F16,"%Y"))
train$weekday2<-as.factor(weekdays(train$F16))

test$day2<-as.numeric(strftime(test$F16,"%d"))
test$month2<-as.numeric(strftime(test$F16,"%m"))
test$year2<-as.numeric(strftime(test$F16,"%Y"))
test$weekday2<-as.factor(weekdays(test$F16))
```

Let's create another feature for the difference in dates of F15 and F16.

```
train$diff_days<-as.numeric(difftime(train$F16,train$F15,units = "days"))

test$diff_days<-as.numeric(difftime(test$F16,test$F15,units = "days"))
```

Also, there's no feature with high cardinality, so no need for count and posterier probability features.

6. Model Selection

As we concluded in conclusion 1, we'll choose a tree based mode. Typically, our choices are:

1. Decesion Tree Classifier.
2. Extra Tree Classifier.
3. Random Forest Clasiifier.
4. Gradient Boosting Machines.

The following algorithms are with increasing complexity from top to bottom and also increasing accuracy from top to bottom as each successive algorithm is built over the previous one alonng with additional concepts to reduce variance. Therefore this creates a trade-off.

Also, taking into consideration the risk to overfit the training data is high in GBM, I'll be using Random Forest.

Also, since Random Forest's R implementation can deal with catogorical features of upto 52 levels, we also don't need to perform One hor encoding for the catogorical variables.

7. Feature Selection

Let's use Recursive Feature elimination for feature selection.

```
library('caret')

## Loading required package: lattice

control <- rfeControl(functions = rfFuncs,
                      method = "cv",
                      number = 3,
                      allowParallel = T,
                      returnResamp = T,
                      verbose = FALSE)
outcomeName<-'C'

predictors<-c("F1", "F2", "F3", "F4", "F5", "F6", "F7", "F8", "F9", "F10",
"F11", "F12", "F13", "F14", "F17", "F18",
"F19", "F20", "F21", "F22", "day1", "month1", "year1", "weekday1", "day2",
"month2", "year2", "weekday2", "diff_days")

#Rather using all the features due to time and slow processor constraint

#Profile <- rfe(train[,predictors], train[,outcomeName], metric = "Kappa",
#rfeControl = control)


#plot(Profile, type = c("g", "o"))
```

8. Mertic Selection.

The majority class of "not buying" or 0 is about 75% while "buying" or 1 is just 25%. Therefore, instead of using accuracy, we'll be using Kappa as our evaluation metric.

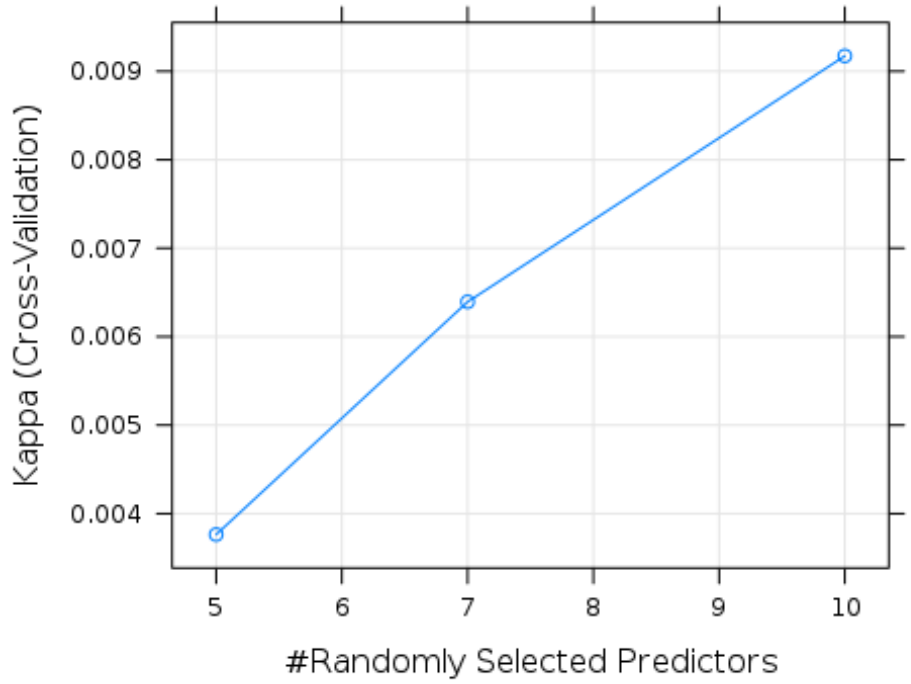
9. Training the model.

```
grid <- expand.grid(mtry=c(5,7,10))

fitControl <- trainControl(
  method = "cv",
  number = 3)


# training the model

model_rf<-train(train[,predictors],train[,outcomeName],method='rf',metric =
'Kappa',trControl=fitControl,tuneGrid=grid)
```

```
## Loading required package: randomForest

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

Let's plot the variable importance:

```
plot(model_rf)
```



10. Predicting and saving for submission.

```
predictions <- predict(model_rf, test[,predictors], type='prob')

sub<-test
sub$Class<-predictions$`1`
sub<-sub[,c('Index','Class')]


write.table(sub, file = "submission_Saurav.data.txt",
            sep = "\t", row.names=FALSE, quote=FALSE)
```