Code

Crowdfunding Platfrom - Modelling User Contribution

Allwyn Joseph - MLDM

3/17/2018

Please click here to head to an interactive html page of the report (http://rpubs.com/ajoseph/371399)

1 Problem Understanding

As of today, recommendation systems stand as one of the biggest driving forces of the e-commerce industry. Its success began provoking these systems to be put in place within social media, online entertainment and a myriad other industries. One such space where recommendaion systems are beginning to burgeon are within crowdfunding platfromes.

This report will focus on building a content based, robust and simple recommendation engine based on data sourced from a crowdfunding platfrom. The aim of the engine would be to try and set up a personalized recommendations engine for the users by means of trying and predicting the genré's of projects the users would be most inclined to contributing towards.

2 Data Understanding

This section would try and attain a holistic view of the dataset in three steps, Reading, Tweaking, and Exploring.

2.1 Reading and Tweaking

Loading the data set:

Hide

```
library(data.table)
library(dplyr)
library(ggplot2)
library(corrplot)
library(DT)
library(GGally)
library(tidyr)
library(e1071)
library(knitr)
library(caret)
library(randomForest)
library(unbalanced)

crowd_fund <- read.csv("recos_training.csv",
                    sep = ",",
                    header = TRUE,
                    na.strings = c("NA","#DIV/0!",""))
crowd_fund_test <- read.csv("recos_test.csv",
                    sep = ",",
                    header = TRUE,
                    na.strings = c("NA","#DIV/0!",""))
```

There are a total of 24 features per user. You can scroll the x-axis to see all of them. (The column discriptions for the dataset can be found in the Github repository.)

	id	owner_friend	dist	nb_copledgers	amount_copledgers	desc_score_mean	amou
1	0	0	0.079014	0	0	0.893994991935	
2	1	0	0.9456767	0.0049	339	0.861315066015	
3	2	0	0	0	0	0.887325569938	
4	3	0	0	0.2469	36	0.886948774142	
5	4	0	0	0.07	1875	0	
6	5	0	0	0.0558	13748	0.895523956954	
7	6	0	1	0.0271	10	0.869664232586	
8	7	0	1	0.0296	190	0.850021367941	
9	8	0	1	0.1741	935	0	
10	9	0	1	0.0835	515	0.917154205586	

Previous 1 2 3 4 5 … 10 Next

One can observe from the table above that the dataset being dealt with is numeric in nature. But, there are features within the dataset that are booleans. In the following step along with rectifying this. The 'id' column will be deleted as well since it serves no particular purpose.

Hide

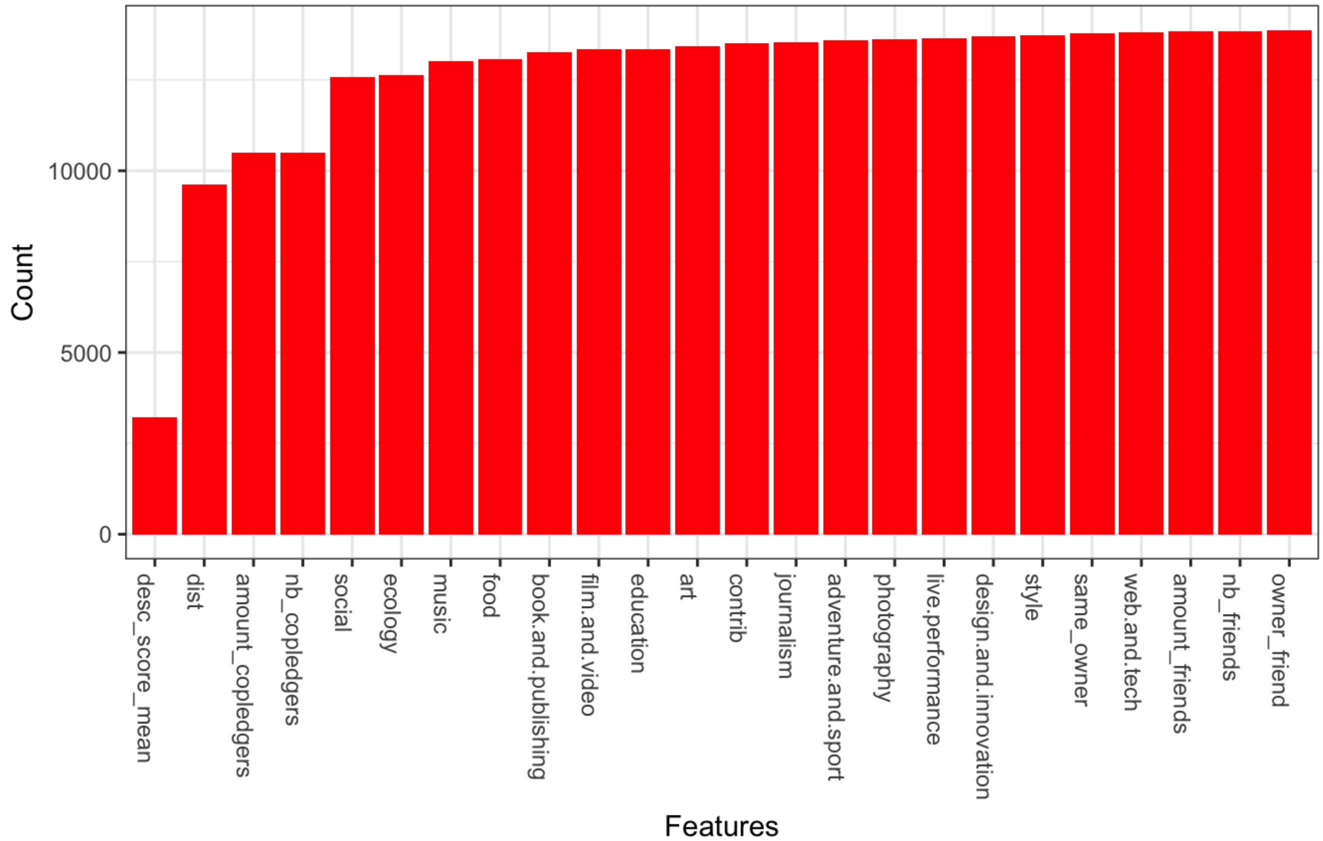
```
# nullifying 'id' column
crowd_fund$id <- NULL
crowd_fund_test$id <- NULL
train <- crowd_fund

# checking for NA values
any(is.na(train))

## converting 'numeric' boolean columns to factors

# which columns are boolean
which(apply(train,2,function(x) { all(x %in% 0:1) }))

# converting these to factors
train$owner_friend <- as.factor(train$owner_friend)
train$same_owner <- as.factor(train$same_owner)
train$contrib <- as.factor((train$contrib))

crowd_fund_test$owner_friend <- as.factor(crowd_fund_test$owner_friend )
crowd_fund_test$same_owner <- as.factor(crowd_fund_test$same_owner)
crowd_fund_test$contrib <- as.factor(crowd_fund_test$contrib)
```

2.2 Exploring

With prior knowledge that we are dealing with a sparce dataset, let's try and visualise how much of our data is actually zeros.

Hide

```
zero_values <- as.data.frame(colSums(train == 0))
ggplot(zero_values,aes(x=reorder(rownames(zero_values),zero_values[,]),y=zero_valu
  es[,])) +
  geom_bar(stat="identity",fill="red")+theme_bw() + ylab("Count") + xlab("Feature
  s") +
  theme(axis.text.x=element_text(angle = -90, hjust = 0))
```



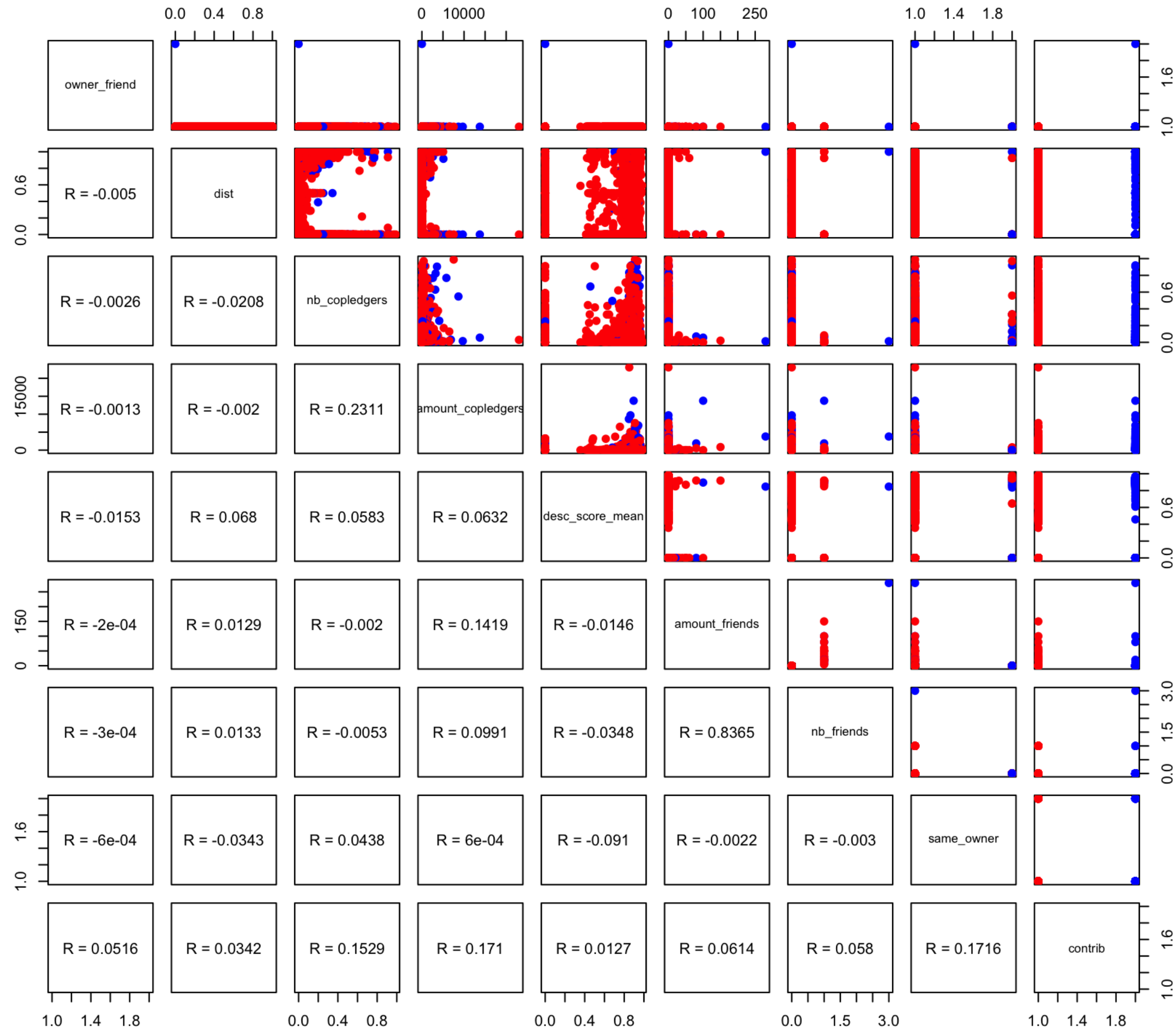
More than 90% of our data are zeros. Extrating meaningful information form such datasets do prove to be quite the task. So let's start talking to the data.

2.2.1 The Primary Attributes

These attributes consists of all the features excluding the different project genres offered on the crowdfunding platfrom. To understand the interation between these features let's display a combination of scatter and correlation plot.

Hide

```
my_cols <- c("red","blue")
panel.cor <- function(x, y){
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- round(cor(x, y), digits=4)
  txt <- paste0("R = ", r)
  text(0.5, 0.5, txt)
}
# Customize upper panel
upper.panel<-function(x, y){
  points(x,y, pch = 19, col = my_cols[train$contrib])
}
# Create the plots
pairs(train[,c(1:8,24)],
      lower.panel = panel.cor,
      upper.panel = upper.panel)
```



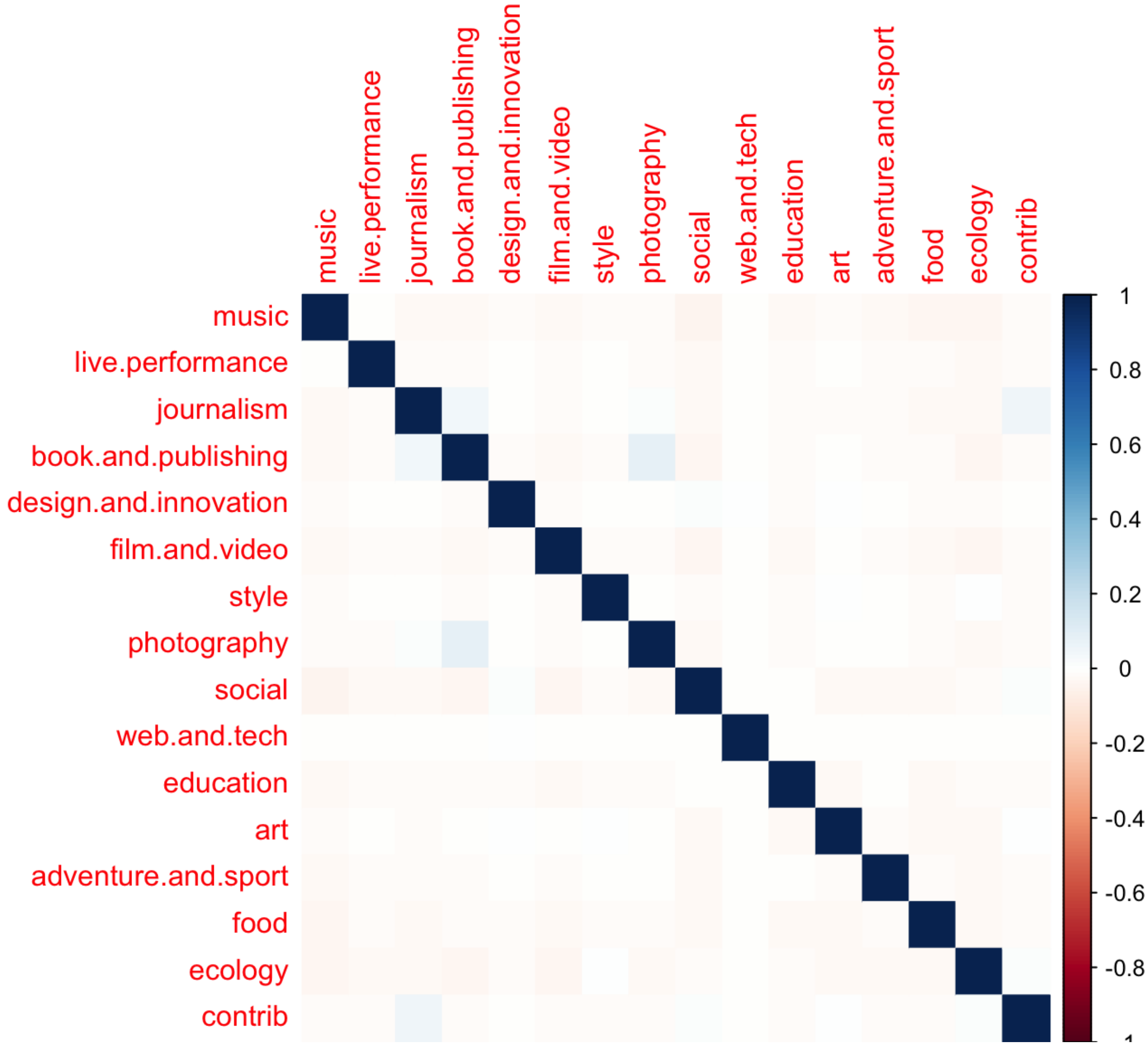
The plot shows very little correlation between the primary feature themselves and with the target variable 'contrib'. Also, from the scatter plot it's very difficult to see any real separation between the points plotted. When dealing with such parse datasets it sometimes might help to perform a PCA tranform to try and project the data differently. We'll come back to this during the data preparation section.

2.2.2 The Secondary Attributes

The secondary attributes mainly have to do with the project genres that are hosted on the crowdfunding platfrom. From the zeros plot in the above section it was evident that the 'genre' features had a majority of its rows populated with zeros. So let's see what its correlation plot has to say to us.

Hide

```
corrplot(cor(crowd_fund[,9:24]), method = "color")
```



Like with the primary features, the secondary ones too have very little correlation amongst themselves and the feature 'contrib'. Again, a PCA transform might help better with data extraction.

3 Data Preparation, Modelling and Evaluation

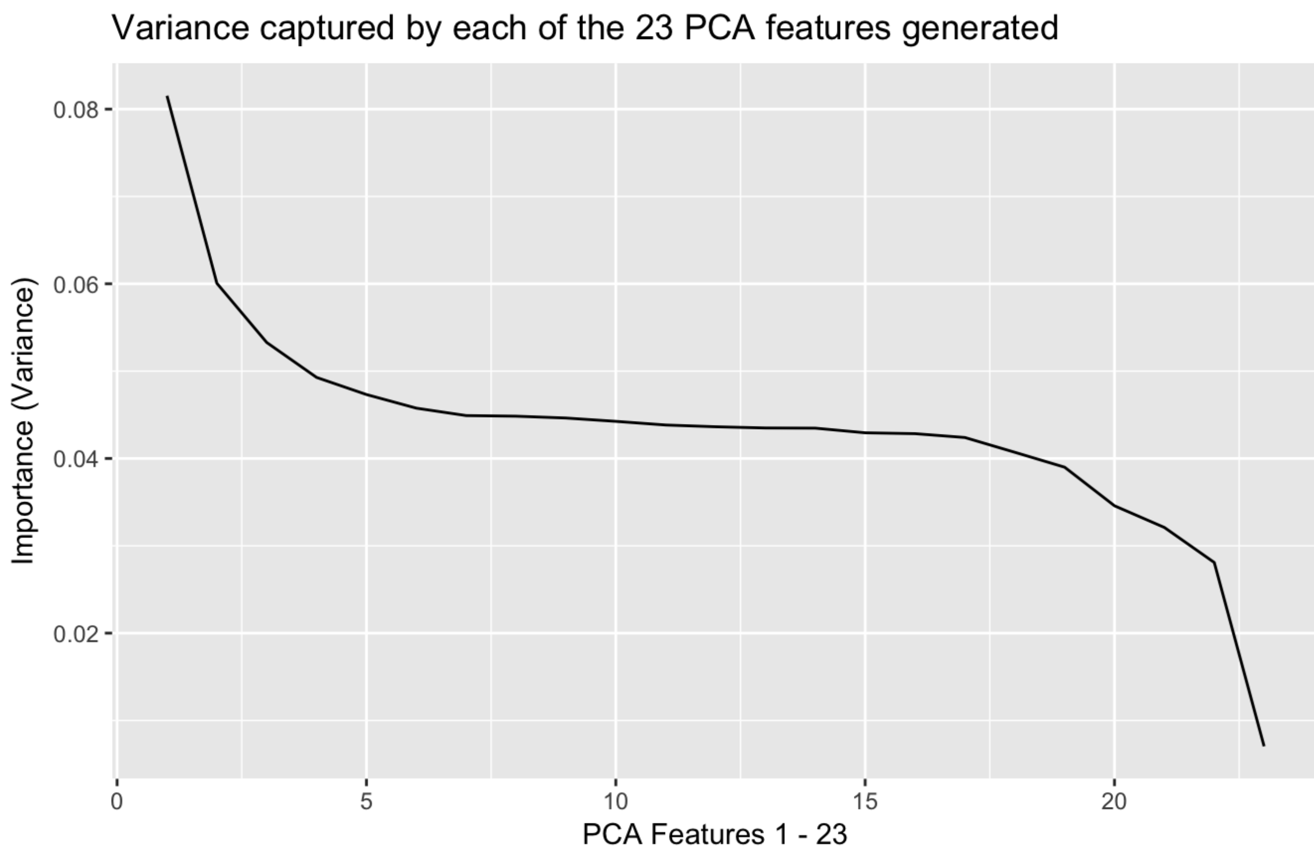
The dataset we are dealing with can be considered unbalanced, i.e when the class of interest (minority class) is much rarer than normal behaviour (majority class). In such datasets cost of missing a minority class is typically much higher that missing a majority class. Most learning systems are not prepared to cope with unbalanced data. So in this section we explore some of those techniques in combination with logistic regression, SVM and Random Forest classifiers and see what works best.

3.1 Data Preparation : Does PCA work ?

Since we couldn't really find correlations or patterns of any kind it would make sense to preprocess the data before fitting models onto it. Moreover, the dataset has a few attributes with suffixes '_amount', whose values vary over larger intervals as compared to other features. This would mean scaling and centering the dataset would work in favour of obtaining better fits for the classifier.

Hide

```
# Performing the PCA transform along with scaling and centering

crowd_fund_pca <- prcomp(crowd_fund[-24],center = TRUE,scale = TRUE)
summary <- summary(crowd_fund_pca)
qplot(1:23,summary$importance[2,], geom  = "line", xlab = "PCA Features 1 - 23",
      ylab = "Importance (Variance)",main = "Variance captured by each of the 23 P
  CA features generated")
```



From the above graph it becomes evident that no one PCA feature captures a majority of the information from the dataset. This could only mean two things: - PCA features generated are not going to be of significant help as compared to the actual features before preprocessing. - PCA features generated may allow better model fit despite lack of variance capture due to the fact that they are a linear combination of actual features.

To find out which of the aforementioned points are accurate we'll need to fit datasets on both preprocessed and original datasets. To help with model evaluation a train validation split would help too.

Hide

```
# Train set with PCA preprocessing
train_p<- predict(crowd_fund_pca) # suffix p stands for preprocess
train_p<- as.data.frame(train_p)
train_p <- cbind(train_p,train$contrib)
colnames(train_p)[24]<- "contrib"

# Train validation split
inTrain <- createDataPartition(crowd_fund$contrib, p=0.7, list=FALSE)
my_train_p<- train_p[inTrain, ]
my_test_p<- train_p[-inTrain, ]
my_train<- train[inTrain, ]
my_test<- train[-inTrain, ]
```

3.2 Simple and Weighted Logistic regression

In this section I try and validate the necessity of a PCA transform and model logistic regression to fit well the imbalanced dataset we are dealing with. This section entails training a normal and weighted logistic regression on the original and preprocessed dataset to try and observe its effect on modelling.

$$weight = \frac{number\ of\ instances}{(number\ of\ classes * number\ of\ majoirty/minoirty\ class\ instances)}$$

Another thing to keep in mind is that, in the following sections we are going to be mainly focusing on improving the specificity output of the models learnt, as we need to be able to predict with better accuracy users who will contribute towards a recommedation, rather than who won't. Of course, while we try and bring up specificity values, heed must be given so as to ensure sensitivity values aren't affected all too much. Hence we'll also be analysing the accuracy and sensitivity of our models too.

Hide

```
#-------------------------------------------------------------------normal_logisti
  c
### Normal logistic regression (logreg_1)

## Modeling

# With pca and other preprocess
logreg_1_p<- glm(contrib~., data = my_train_p, family  = "binomial")
pred_logr_1_p<- predict(logreg_1_p, my_test_p,type = "response")
#table(ActualValue = my_test_p$contrib, PredictedValue = pred_logr_1_p > 0.5)
#summary(logreg_1)
PredictedValue =  ifelse(pred_logr_1_p > 0.5,1,0)
cm_n_p<- confusionMatrix(PredictedValue, my_test_p$contrib)

# Without pca and other preprocess
logreg_1<- glm(contrib~., data = my_train, family  = "binomial")
pred_logr_1<- predict(logreg_1, my_test,type = "response")
#table(ActualValue = my_test$contrib, PredictedValue = pred_logr_1 > 0.5)
#summary(logreg_1)
PredictedValue =  ifelse(pred_logr_1 > 0.5,1,0)
cm_n <- confusionMatrix(PredictedValue, my_test$contrib)

#-------------------------------------------------------------------weighted_logis
  tic
### Weighted logistic regression (logreg_2)

## Modeling
wt<- ifelse(my_train$contrib == 0 ,0.5, 19)

# With pca and other preprocess
logreg_2_p<- glm(contrib~., data = my_train_p, family  = "binomial" ,weights = wt)
pred_logr_2_p<- predict(logreg_2_p, my_test_p,type = "response")
#table(ActualValue = my_test_p$contrib, PredictedValue = pred_logr_2_p > 0.5)
#summary(logreg_1)
PredictedValue =  ifelse(pred_logr_2_p > 0.5,1,0)
cm_w_p<- confusionMatrix(PredictedValue, my_test_p$contrib)

# Without pca and other preprocess
logreg_2<- glm(contrib~., data = my_train, family  = "binomial",weights = wt)
pred_logr_2<- predict(logreg_2, my_test,type = "response")
#table(ActualValue = my_test$contrib, PredictedValue = pred_logr_2 > 0.5)
#summary(logreg_1)
PredictedValue =  ifelse(pred_logr_2 > 0.5,1,0)
cm_w<- confusionMatrix(PredictedValue, my_test$contrib)


### printing Accuracy, Sensitivity and Specificity resutls after above modelling
norm_log <-  c(cm_n$overall['Accuracy'], cm_n$byClass['Sensitivity'],
               cm_n$byClass['Specificity'])
```

```
norm_log_p <- c(cm_n_p$overall['Accuracy'], cm_n_p$byClass['Sensitivity'],
                cm_n_p$byClass['Specificity'])
weight_log <- c(cm_w$overall['Accuracy'], cm_w$byClass['Sensitivity'],
                cm_w$byClass['Specificity'])
weight_log_p <- c(cm_w_p$overall['Accuracy'], cm_w_p$byClass['Sensitivity'],
                  cm_w_p$byClass['Specificity'])

df_resutls <- data.frame(norm_log, norm_log_p, weight_log, weight_log_p)
colnames(df_resutls)<- c("Logistic Reg", "Logistic Reg with preprocess",
                         "Weighted Logistic Reg", "Weighted Logistic Reg with prep
  rocess")

kable(df_resutls, caption = "Accuracy, Sensitivity and Specificity for the Weighte
  d and normal Logistic Regression models realised")
```

Accuracy, Sensitivity and Specificity for the Weighted and normal Logistic Regression models realised

	Logistic Reg	Logistic Reg with preprocess	Weighted Logistic Reg	Weighted Logistic Reg with preprocess
Accuracy	0.9740322	0.9740322	0.7860063	0.7860063
Sensitivity	0.9987670	0.9987670	0.7908755	0.7908755
Specificity	0.0096154	0.0096154	0.5961538	0.5961538

From the table above table it becomes evident of the negligible role PCA plays with this dataset towards ameliorating the models/classifer. Resutls show that with and without preprocessing the models' accuray, sensitivity and specificity remains the same. So in the following sections I will avoid the preprocessed data and instead only model on the original data.

While logistic model in itself has a poor specificity score, the weighted version of the same performs quite well with a score nearing 0.6. While attaining a score of 0.6 the weighted logistic regresion doen't compromise too much on the accuracy and sensitivity scores. Next, I put to test a few other sampling techniques to even the weights out between the majorit and minority classes.

3.3 Logistic regression, SVM and Randomforest with Under and Oversampling

In this section Logistic, SVM and Randomforest classifiers will be modeled on under and over sampled datasets and their performaces will be recored and tablulated.

3.3.1 Undersampling

I began with undersampling the original data, this is primarily done by ignoring some points from the majority class while keeping intact the points from the minority class. Once the undersampled dataset is created Logistic, SVM and Randomforest classifiers are modeled upon it.

Hide

```
### Undersampling with logistic regression, SVMs and Randomforest

#------------------------------------------------------------------undersampled_l
  ogistic_svm_rf

## Data preprocessing - Undersampling
temp <- data<-ubUnder(X=my_train[,-24], Y= my_train$contrib, perc = 50, method =
  "percPos")
data_under<-cbind(temp$X, temp$Y)
colnames(data_under)[24]<- "contrib"

## Modeling

#Logistic Regression
logreg_3<- glm(contrib~., data = data_under, family  = "binomial")
pred_logr_3<- predict(logreg_3, my_test,type = "response")
PredictedValue =  ifelse(pred_logr_3 > 0.5,1,0)
cm_u_log<- confusionMatrix(PredictedValue, my_test$contrib)

#SVM
svm_1 <- svm(contrib ~ ., data = data_under,kernel = "radial")
svm_pred_1  <- predict(svm_1, my_test, type = "class")
cm_u_svm<- confusionMatrix(svm_pred_1, my_test$contrib)

# Random Forest
rf_1<- randomForest(contrib ~ ., data = data_under, importance = TRUE, ntree = 100
  0)
rf_pred_1  <- predict(rf_1, my_test, type = "class")
cm_u_rf<- confusionMatrix(rf_pred_1, my_test$contrib)

#-----------------------------------------------------------dataframe with undersam
  pled results

## Storing Accuracy, Sensitivity and Specificity resutls after above modelling

Logistic_Undersampled <-  c(cm_u_log$overall['Accuracy'], cm_u_log$byClass['Sensit
  ivity'],
                            cm_u_log$byClass['Specificity'])
SVM_Undersampled <- c(cm_u_svm$overall['Accuracy'], cm_u_svm$byClass['Sensitivity'
  ],
                      cm_u_svm$byClass['Specificity'])
RandomForest_Undersampled <- c(cm_u_rf$overall['Accuracy'], cm_u_rf$byClass['Sensi
  tivity'],
                               cm_u_rf$byClass['Specificity'])

df_under <- data.frame(Logistic_Undersampled, SVM_Undersampled, RandomForest_Under
  sampled)
kable(df_under, caption = "Accuracy, Sensitivity and Specificity for
      the Logistic Regression, SVM and Random Forest models realised on Undersampl
  ed data ")
```

Accuracy, Sensitivity and Specificity for the Logistic Regression, SVM and Random Forest models realised on Undersampled data

	Logistic_Undersampled	SVM_Undersampled	RandomForest_Undersampled
Accuracy	0.8259197	0.8369800	0.7136331
Sensitivity	0.8327990	0.8441430	0.7139334
Specificity	0.5576923	0.5576923	0.7019231

From the table above we see similar performace with regard to specificity for the logistic and SVM classifiers without having compromised too much the accuracy and sensitivity of the data. Off the three, the random forest classifier achieved the best specificity score, but this came at the price of lower accuracy and sensitivity score.

3.3.2 Oversampling

I then moved onto oversampling the original data, this is primarily done by oversampling instances from the minority class while keeping intact the points from the majority class. Once the oversampled dataset is created Logistic, SVM and Randomforest classifiers are modeled upon it.

Hide

```
### Oversampling with logistic regression, SVMs and Randomforest
#------------------------------------------------------------------oversampled_lo
  gistic_svm_rf

## Data preprocessing - Oversampling
temp<- ubOver(X=my_train[,-24],  Y= my_train$contrib, k = 0, verbose=FALSE)
data_over<-cbind(temp$X, temp$Y)
colnames(data_over)[24]<- "contrib"

## Modeling

#Logistic Regression
logreg_4<- glm(contrib~., data = data_over, family  = "binomial")
pred_logr_4<- predict(logreg_4, my_test,type = "response")
PredictedValue =  ifelse(pred_logr_4 > 0.5,1,0)
cm_o_log<- confusionMatrix(PredictedValue, my_test$contrib)

# SVM
svm_2 <- svm(contrib ~ ., data = data_over,kernel = "radial")
svm_pred_2  <- predict(svm_2, my_test, type = "class")
cm_o_svm<- confusionMatrix(svm_pred_2, my_test$contrib)

# Random Forest
rf_2<- randomForest(contrib ~ ., data = data_over, importance = TRUE, ntree = 1000
  )
rf_pred_2  <- predict(rf_2, my_test, type = "class")
cm_o_rf<- confusionMatrix(rf_pred_2, my_test$contrib)

#-----------------------------------------------------------dataframe with oversamp
  led results

## Storing Accuracy, Sensitivity and Specificity resutls after above modelling

Logistic_Oversampled <-  c(cm_o_log$overall['Accuracy'], cm_o_log$byClass['Sensiti
  vity'],
                           cm_o_log$byClass['Specificity'])
SVM_Oversampled <- c(cm_o_svm$overall['Accuracy'], cm_o_svm$byClass['Sensitivity'
  ],
                     cm_o_svm$byClass['Specificity'])
RandomForest_Oversampled <- c(cm_o_rf$overall['Accuracy'], cm_o_rf$byClass['Sensit
  ivity'],
                              cm_o_rf$byClass['Specificity'])

df_under <- data.frame(Logistic_Oversampled, SVM_Oversampled, RandomForest_Oversam
  pled)
kable(df_under, caption = "Accuracy, Sensitivity and Specificity for
      the Logistic Regression, SVM and Random Forest models realised on Oversample
  d data")
```

Accuracy, Sensitivity and Specificity for the Logistic Regression, SVM and Random Forest models realised on

Oversampled data

	Logistic_Oversampled	SVM_Oversampled	RandomForest_Oversampled
Accuracy	0.7994710	0.8278432	0.8829045
Sensitivity	0.8051788	0.8345253	0.8954377
Specificity	0.5769231	0.5673077	0.3942308

From the table above we see similar performace with regard to specificity for the logistic and SVM classifiers without having compromised too much the accuracy and sensitivity of the data. These scores are also in line with the undersampling case. On the contrary the random forest classifier arrived at a poor specificity score.

3.4 Logistic regression, SVM and Randomforest with Hybrid sampling techniques

In this section Logistic, SVM and Randomforest classifiers will be modeled on Hybrid sampled - datasets will undergo under and oversampling - datasets and their performaces will be recored and tablulated.

3.4.1 SMOTE TOMEK Hybrid sampling

I began with deploying a SMOTE TOMEK hybrid sampling teching over the original data. Oversampling using SMOTE followed by under sampling using Tomek. Once the hybrid dataset is created Logistic, SVM and Randomforest classifiers are modeled upon it.

Smote picks an example form the minority class, finds its KNN (k is a hyper-parameter) from the neighbourhood, then goes on to randomly choose r points such that r <= k. A line is then drawn between each of these r points and the orignal selected point and instances of the minority classes are created on any random point on these line. A pair of examples is a **Tomek** link if they belong to different classes and they are each others nearest neighbour. Here we'll aim at removing all Tomek links form class 0.

Hide

```
### SmoteTomek hybrid sampling with logistic regression, SVMs and Randomforest
#----------------------------------------------------------------SmoteTomek_log
  istic_svm_rf

## Data preprocessing - SmoteTomek hybrid sampling

# Using numerical data for Tomek
inTrain <- createDataPartition(crowd_fund$contrib, p=0.7, list=FALSE)
my_train_num<- crowd_fund[inTrain, ]
my_test_num<- crowd_fund[-inTrain, ]

temp<- ubSMOTE(X= my_train_num[,-24],  Y= my_train$contrib,
               perc.over = 2000, k = 5, perc.under = 200, verbose = FALSE)
data_smote<-cbind(temp$X, temp$Y)
colnames(data_smote)[24]<- "contrib"

temp<- ubTomek(X= data_smote[,-24],  Y= data_smote$contrib, verbose = FALSE)
data_tomsmot<-cbind(temp$X, temp$Y)
colnames(data_tomsmot)[24]<- "contrib"

## Modeling

# Logistic Regresion
logreg_8<- glm(contrib~., data = data_tomsmot, family  = "binomial")
pred_logr_8<- predict(logreg_8, my_test_num,type = "response")
PredictedValue =  ifelse(pred_logr_8 > 0.5,1,0)
cm_st_log<- confusionMatrix(PredictedValue, my_test_num$contrib)

# SVM
svm_3 <- svm(contrib ~ ., data = data_tomsmot,kernel = "radial")
svm_pred_3  <- predict(svm_3, my_test_num, type = "class")
cm_st_svm<- confusionMatrix(svm_pred_3, my_test_num$contrib)

# Random Forest
rf_3<- randomForest(contrib ~ ., data = data_tomsmot, importance = TRUE, ntree = 1
  000)
rf_pred_3  <- predict(rf_3, my_test_num, type = "class")
cm_st_rf<- confusionMatrix(rf_pred_3, my_test_num$contrib)

#------------------------------------------------dataframe with SmoteTomek hybr
  id sampling results

## Storing Accuracy, Sensitivity and Specificity resutls after above modelling

Logistic_SmoteTomek <-  c(cm_st_log$overall['Accuracy'], cm_st_log$byClass['Sensit
  ivity'],
                          cm_st_log$byClass['Specificity'])
SVM_SmoteTomek <- c(cm_st_svm$overall['Accuracy'], cm_st_svm$byClass['Sensitivity'
  ],
                     cm_st_svm$byClass['Specificity'])
```

```
RandomForest_SmoteTomek <- c(cm_st_rf$overall['Accuracy'], cm_st_rf$byClass['Sensi
  tivity'],
                             cm_st_rf$byClass['Specificity'])

df_st_hybrid <- data.frame(Logistic_SmoteTomek, SVM_SmoteTomek, RandomForest_Smote
  Tomek)
kable(df_st_hybrid, caption = "Accuracy, Sensitivity and Specificity for
      the Logistic Regression, SVM and Random Forest models realised on SmoteTomek
   sampled data")
```

Accuracy, Sensitivity and Specificity for the Logistic Regression, SVM and Random Forest models realised on SmoteTomek sampled data

	Logistic_SmoteTomek	SVM_SmoteTomek	RandomForest_SmoteTomek
Accuracy	0.8153402	0.8687184	0.9148834
Sensitivity	0.8304541	0.8822804	0.9365745
Specificity	0.2429907	0.3551402	0.0934579

As seen in the table above, unfortunately, the SMOTE TOMEK hybrid sampling doesn't render high specificity scores. While the specificity score for logistic regression was low, the scores for SVM and random forest were worse off.

3.4.2 SMOTE ENN Hybrid sampling

I then moved onto SMOTE ENN sampling of the original data. TOversampling using SMOTE followed by under sampling using ENN. Once the oversampled dataset is created Logistic, SVM and Randomforest classifiers are modeled upon it.

ENN removes any example whose class label differs from the class of at least two of its three nearest neighbors.

Hide

```
### SmoteEnn hybrid sampling with logistic regression, SVMs and Randomforest
#------------------------------------------------------------------SmoteEnn_logis
  tic_svm_rf
# Using numerical data for enn
inTrain <- createDataPartition(crowd_fund$contrib, p=0.7, list=FALSE)
my_train_num<- crowd_fund[inTrain, ]
my_test_num<- crowd_fund[-inTrain, ]

## Data preprocessing - SmoteEnn hybrid sampling
temp<- ubSMOTE(X= my_train_num[,-24],  Y= my_train$contrib,
               perc.over = 2000, k = 5, perc.under = 100, verbose = FALSE)
data_smote<-cbind(temp$X, temp$Y)
colnames(data_smote)[24]<- "contrib"

temp<- ubENN(X= data_smote[,-24],  Y= data_smote$contrib, k = 3, verbose = FALSE)
data_ennsmot<-cbind(temp$X, temp$Y)
colnames(data_ennsmot)[24]<- "contrib"

## Modeling

# Logistic Regression
logreg_9<- glm(contrib~., data = data_ennsmot, family  = "binomial")
pred_logr_9<- predict(logreg_9, my_test_num,type = "response")
PredictedValue =  ifelse(pred_logr_9 > 0.5,1,0)
cm_se_log<- confusionMatrix(PredictedValue, my_test_num$contrib)

# SVM
svm_4 <- svm(contrib ~ ., data = data_ennsmot,kernel = "radial")
svm_pred_4  <- predict(svm_4, my_test_num, type = "class")
cm_se_svm<- confusionMatrix(svm_pred_4, my_test$contrib)

# Random Forest
rf_4<- randomForest(contrib ~ ., data = data_ennsmot, importance = TRUE, ntree = 1
  000)
rf_pred_4  <- predict(rf_4, my_test_num, type = "class")
cm_se_rf<- confusionMatrix(rf_pred_4, my_test_num$contrib)

#----------------------------------------------------dataframe with SmoteEnn hybrid
   sampling results

## Storing Accuracy, Sensitivity and Specificity resutls after above modelling

Logistic_SmoteEnn <-  c(cm_se_log$overall['Accuracy'], cm_se_log$byClass['Sensitiv
  ity'],
                          cm_se_log$byClass['Specificity'])
SVM_SmoteEnn <- c(cm_se_svm$overall['Accuracy'], cm_se_svm$byClass['Sensitivity'],

                    cm_se_svm$byClass['Specificity'])
RandomForest_SmoteEnn <- c(cm_se_rf$overall['Accuracy'], cm_se_rf$byClass['Sensiti
  vity'],
```

```
cm_se_rf$byClass['Specificity'])

df_se_hybrid <- data.frame(Logistic_SmoteEnn, SVM_SmoteEnn, RandomForest_SmoteEnn)
kable(df_se_hybrid, caption = "Accuracy, Sensitivity and Specificity for
      the Logistic Regression, SVM and Random Forest models realised on SmoteEnn s
  ampled data")
```

Accuracy, Sensitivity and Specificity for the Logistic Regression, SVM and Random Forest models realised on SmoteEnn sampled data

	Logistic_SmoteEnn	SVM_SmoteEnn	RandomForest_SmoteEnn
Accuracy	0.8030777	0.8547728	0.9139216
Sensitivity	0.8158612	0.8727497	0.9395291
Specificity	0.3870968	0.1538462	0.0806452

Concerning the results from the final modeling, we still see a rather poor performance from the classifiers with regard to specificity. Once again, while the specificity score for logistic regression was low, the scores for SVM and random forest were worse off.

4 Deployment

Off the classifiers and sampling methods tested one could safely attest to the fact that, on a whole the classifiers performed best with undersampled data. While this is true, good performance was seen by weighted logitic regression too. So let’s put all of these models to a final test.

Hide

```
# Logistic Regression weighted
pred_logr_2<- predict(logreg_2, crowd_fund_test,type = "response")
PredictedValue =  ifelse(pred_logr_2 > 0.5,1,0)
cm_wt_log<- confusionMatrix(PredictedValue, crowd_fund_test$contrib)

# Logistic Regression undersampled
#logreg_3<- glm(contrib~., data = data_under, family  = "binomial")
pred_logr_3<- predict(logreg_3, crowd_fund_test,type = "response")
PredictedValue =  ifelse(pred_logr_3 > 0.5,1,0)
cm_u_log<- confusionMatrix(PredictedValue, crowd_fund_test$contrib)

# SVM undersampled
#svm_1 <- svm(contrib ~ ., data = data_under,kernel = "radial")
svm_pred_1  <- predict(svm_1, crowd_fund_test, type = "class")
cm_u_svm<- confusionMatrix(svm_pred_1, crowd_fund_test$contrib)

# Random Forest undersampled
#rf_1<- randomForest(contrib ~ ., data = data_under, importance = TRUE, ntree = 10
  00)
rf_pred_1  <- predict(rf_1, crowd_fund_test, type = "class")
cm_u_rf<- confusionMatrix(rf_pred_1, crowd_fund_test$contrib)

#---------------------------------------------------dataframe with final modelling
   results

## Storing Accuracy, Sensitivity and Specificity resutls after above modelling

Logistic_Weighted <-  c(cm_wt_log$overall['Accuracy'], cm_wt_log$byClass['Sensitiv
  ity'],
                          cm_wt_log$byClass['Specificity'])
Logistic_Undersampled <-  c(cm_u_log$overall['Accuracy'], cm_u_log$byClass['Sensit
  ivity'],
                          cm_u_log$byClass['Specificity'])
SVM_Undersampled <- c(cm_u_svm$overall['Accuracy'], cm_u_svm$byClass['Sensitivity'
  ],
                    cm_u_svm$byClass['Specificity'])
RandomForest_Undersampled <- c(cm_u_rf$overall['Accuracy'], cm_u_rf$byClass['Sensi
  tivity'],
                             cm_u_rf$byClass['Specificity'])

df_final <- data.frame(Logistic_Weighted,Logistic_Undersampled,
                           SVM_Undersampled, RandomForest_Undersampled)
kable(df_final, caption = "Accuracy, Sensitivity and Specificity for the final set
   of
      choosen models ")
```

Accuracy, Sensitivity and Specificity for the final set of choosen models

	Logistic_Weighted	Logistic_Undersampled	SVM_Undersampled	RandomForest_Undersampled

	Logistic_Weighted	Logistic_Undersampled	SVM_Undersampled	RandomForest_Undersampled
Accuracy	0.8749730	0.8617781	0.8751893	0.7728748
Sensitivity	0.8785699	0.8654894	0.8790059	0.7737083
Specificity	0.4166667	0.3888889	0.3888889	0.6666667

From the final simulation it is observed that the random forest model performed best. But on must keep in mind that the random forest model is complex and computationally intensive. With some fine tuning and feature engineering simpler models such has logistic regression can be set to work quite well for classifcation problems on imbalanced datasets. This is imperative as datasets get bigger and bigger, the longer ensemble models take to fit. At the end of the day, it boils down to how much computational power one has at his or her disposal and what levels of accuracy/specificity is expected.