DSU MLM 1812 Project by Richa Singh

Table of Contents

Introduction

Context

The providers use the Payment Integrity Compass or PIC application to build workflows for recovering revenue from payers. The revenue can be in the form of Denials or Underpayments. The revenue recovery specialists use PIC to open appeal for the identified underpayments/denials. They send the information over to the payer and then add information to the appeal record in the system so that other revenue recovery specialists can also follow up. The Account Managers from PIC help the providers build the workflows.

Objective

The Payment Integrity Compass (PIC) system allows the users to create appeals in an automated way based on some triggers like Denial code in EOB. It does not track whether the appeal was successful or not. The hospital executives using PIC are looking for ways to increase the success rate of the appeal process.
The hospital executives define an appeal to be successful when a payment is received as a result of the appeal or the denial is reversed. A payment directly reduces the Accounts Receivable and helps balance their balance sheet.

The executives want to identify factors that will increase the probability of a denial getting reversed after a denial is appealed.

Data Collection

Dataset is based on appeal data in Payment Integrity Compass. The data is used by providers to recover underpayments and denials from the payers.
The provider stores the patient data in various systems including patient accounting system. Some providers collect all the data in a warehouse and then extract it to send to Payment Integrity Compass via flat/EDI files. The Appeal data is entered in the Payment Integrity Compass by the end users who work underpayments/denials.

Data Lineage

When the patient goes to a provider, an account is created for that encounter in the patient accounting system. The account has patient demographic (age, gender) and visit related details (inpatient or outpatient). The clinical information is coded into ICD/HCPCS/Revenue codes and charges are applied to the service lines. The coded data is sent as claims to payers. The account information that includes payer, clinical, service lines charge and procedure information is extracted and sent to Payment Integrity Compass. The reimbursement expected from the payer is calculated based on the payer-provider contract. The payer sends Explanation of Benefits, payment/adjustment transactions to provider who forwards the same to Payment Integrity Compass. The Payment Integrity software applies rules and creates appeals. Revenue recovery specialists also create appeals to overturn denials or get account paid correctly. Payer sends payments or denies the appeal. The user closes the account.

Data Storage/Retrieval

The data sent by providers to Payment Integrity compass in received in flat files. The claim and EOB data are received as EDI. The data is sent to Spark where validations are performed. Then the data is stored in Oracle tables.
The UI is linked to the Oracle database where the user creates and closes appeals. The data is also extracted for reporting using MicroStrategy with Postgres being the data store.
The data is accessed programmatically or using SQLs.

Denominator

The appeals can be created for both denials and underpayments. The workflow for appealing denials is much better defined as compared to underpayments. Hence for this Investigational Design I am studying the factors that can predict the success of appeal process.

The denominator is accounts for appeal was created after a denial was received from payer. These only include primary payers.

Events: Denials for which denial was reversed as a result of appeal, in other words appeal was successful.
Non-Events: Accounts which were denied again even after appeal.
Not-Events: Appeals that were created for reasons other than denial, i.e. opened in error.
Positive Control: Accounts that are closed by revenue recovery specialists. We can consider that as expert opinion.
Negative Control: Randomize data before applying statistical models.
Value Model: The appeal data for 500 days had:
Appeal Amount: $517 M (34496 accounts), Denial Amount: $350M (26146 accounts), Denial Reversal: $187M (13014 account). Increasing the probability of denial reversal will be a gain of $12K per account

Baseline

Baseline is appeals that were successful in 2017 i.e. denials were reversed. In other words, to what extent historical success rate of appeals predicts the success rate of new appeals.

Hypothesis

Null Hypothesis: The success of appeal (reversal of denial) does **not** depend on type of denial, payer, appeal amount, patient type (inpatient/outpatient), length of stay, days appeal opened after discharge, payment/adjustment amount received after appeal, Contractual adjustment received after appeal, first and last payment/adjustment days.
Alternate Hypothesis: The success of appeal depends on type of denial, payer, appeal amount, patient type (inpatient/outpatient), length of stay, days appeal opened after discharge, payment/adjustment amount received after appeal, Contractual adjustment received after appeal, first and last payment/adjustment days

Preliminary Data Analysis & Cleanse

Initial dataset for 500 days of appeals – 34496 records

Field	Type	Description
APL_DISPOSITION_CODE_DESC	VARCHAR2	User entered reason for the appeal
PAYER_ORG	VARCHAR2	Payer (insurance companies)
IP_OP_IND	VARCHAR2	Inpatientor Outpatient
APPEALED_AMT	NUMERIC	Amount appealed
AMOUNT_DUE	NUMERIC	Amount due from payer
COVERED_CHARGES	NUMERIC	Covered Charges for the hospital visit
TOTAL_PAYMENTS	NUMERIC	Total payments received from payer

LENGTH_OF_STAY	NUMERIC	Length of stay for the patient, calculated as Discharge - Admit days
TOTAL_ADJUSTMENTS	NUMERIC	Total adjustments received from payer
PAYMENTS_AFTER_APPEAL	NUMERIC	Payments received from payer after appeal was filed
ADJUSTMENTS_AFTER_APPEAL	NUMERIC	Adjustments received from payer after appeal was filed
DAYS_TO_FIRST_PAYMENT_AFT_APPEAL	NUMERIC	Days after appeal first payment transaction was received. Calculated as (First Payment Date - Appeal date)
DAYS_TO_LAST_PAYMENT_AFT_APPEAL	NUMERIC	Days after appeal first adjustment transaction was received. Calculated as (First Adjustment Date - Appeal date)
DAYS_TO_FIRST_ADJUSTMENT_AFT_APPEAL	NUMERIC	Days after appeal last payment transaction was received. Calculated as (Last Payment Date - Appeal date)
DAYS_TO_LAST_ADJUSTMENT_AFT_APPEAL	NUMERIC	Days after appeal last adjustment transaction was received. Calculated as (Last Adjustment Date - Appeal date)
DAYS_APPEAL_AFTER_DISCHARGE	NUMERIC	Days appeal opened after discharge. Calculated as (Appeal Open Day - Discharge Date)
PRE_APPEAL_DENIAL_AMOUNT	NUMERIC	Denial amount from EOB before appeal opened. This is the main cause of why appeal was opened.
POST_APPEAL_CA_AMOUNT	NUMERIC	Contractual Adjustment sent as part of EOB after appeal was opened.
POST_APPEAL_DENIAL_AMOUNT	NUMERIC	Denial amount from EOB after appeal opened. This is negative when the appeal is successful and denial is reversed
DAYS_TO_CLOSE_ACCOUNT	NUMERIC	Days after discharge account is closed. Calculated as (Close Date – Discharge Date)
Derived Fields		
TOTAL_PAY_ADJ_AMT	NUMERIC	Total Payment + Total Adjustment (as both as financial transactions)
PAY_ADJ_AMT_AFTER_APPEAL	NUMERIC	Payment + adjustment amount after appeal
FINAL_DENIAL_AMOUNT	NUMERIC	Denial Amount before appeal + Denial Amount after appeal
DAYS_TO_FIRST_PAY_ADJ_AFT_APPEAL	NUMERIC	Earliest day payment or adjustment was received after appeal
DAYS_TO_LAST_PAY_ADJ_AFT_APPEAL	NUMERIC	Latest day payment or adjustment was received after appeal
SUCCESS	Boolean	1 if FINAL_DENIAL_AMOUNT < 0 (denial reversed) 0 otherwise

Data Summary

Appeal_prelim_datacleanse.R

```
library(xda)
numSummary(appeal_all_df)
```

	n	mean	sd	max	min	range	miss%
APPEALED_AMT	34496	14990.13	338025.65	61518143	0.01	61518143	0
COVERED_CHARGES	34496	19682.31	42488.02	1766021	0	1766021	0
TOTAL_PAYMENTS	29860	10729.09	28931.19	1101652	-8256.51	1109908	13.44
LENGTH_OF_STAY	34496	4.33	6.45	127	1	126	0
TOTAL_ADJUSTMENTS	25212	8525.8	20827.85	1421495	-95334.46	1516829	26.91
PAYMENTS_AFTER_APPEAL	24775	8797.03	27163.76	1101652	-184962.36	1286614	28.18
ADJUSTMENTS_AFTER_APPEAL	23270	6685.76	19586.36	1421495	-124613.31	1546108	32.54
DAYS_TO_FIRST_PAYMENT_AFT_APPEAL	24775	51.75	52.84	471	1	470	28.18
DAYS_TO_LAST_PAYMENT_AFT_APPEAL	24775	68.33	66.56	493	1	492	28.18
DAYS_TO_FIRST_ADJUSTMENT_AFT_APPEAL	23270	60.66	62.13	471	1	470	32.54
DAYS_TO_LAST_ADJUSTMENT_AFT_APPEAL	23270	70.04	70.06	493	1	492	32.54
DAYS_APPEAL_AFTER_DISCHARGE	34496	79.55	68.46	985	-220	1205	0
PRE_APPEAL_DENIAL_AMOUNT	26145	13360.07	39300.65	2184056	-130473.41	2314530	24.21

POST_APPEAL_CA_AMOUNT	17951	7286.56	24711.23	1447666	-733419.13	2181085	47.96
POST_APPEAL_DENIAL_AMOUNT	18303	-4019.12	50855.25	2546271	-734814.07	3281085	46.94
DAYS_TO_CLOSE_ACCOUNT	31060	196.48	86.37	498	1	497	9.96
TOTAL_PAY_ADJ_AMT	31254	17128.15	38085.33	1765805	-27950	1793755	9.4
PAY_ADJ_AMT_AFTER_APPEAL	27805	11190.62	36174.88	2842989	-249226.62	3092216	19.4
FINAL_DENIAL_AMOUNT	34496	7993.31	46225.32	3162111	-646050.38	3808161	0
DAYS_TO_FIRST_PAY_ADJ_AFT_APPEAL	27805	53.74	56.24	456	1	455	19.4
DAYS_TO_LAST_PAY_ADJ_AFT_APPEAL	27805	72.93	71.11	493	1	492	19.4

categorical data

table(appeal_all_df$APL_DISPOSITION_CODE_DESC

```
DNL01 - First level appeal/med nec
3418
DNL02 - Second level appeal/ med nec
26
DNL03 - External review/med nec
55
DNL04 - P2P
10
DNL05 - First level appeal/ Coding
368
DNL06 - Retro Authorization
483
DNL07 - Coding Correction
2850
DNL08 - Charge correction
99
DNL09 - Medical Records Sent
10125
DNL10 - Itemized Statement/EOB Sent
244
DNL11 - Requested reprocessing payer error
5828
DNL12 - Requested reprocessing provided additional detail
4961
DNL13 - Not Appealable
3688
DNL14 - Rebilled Account
1595
DNL15 - LOMN/MR Sent
379
DNL16 - Modifier Added
169
DNL17 - Appeal Denied
188
DNL18 - Peer to Peer Denied
10
```

table(appeal_all_df$PAYER_ORG)

```
Aetna - Institutional    Aetna - Professional Blue Cross Blue Shield - 
Institutional 
2081    2880    6626
Blue Cross Blue Shield - Professional    Cigna - Institutional    Cigna - Professional
12351    1201    1027
Commercial - Institutional    Commercial - Professional    Coventry - Institutional
1659    3366    23
Coventry - Professional    Humana - Institutional    Humana - Professional
75    91    226
Medicaid - Institutional    Medicaid - Professional    Medicare - Institutional
2    6    1
Medicare - Professional    Self Pay - Institutional    TRICARE - Institutional
2    1    20
TRICARE - Professional    Unicare - Institutional    United - Institutional
30    1    639
United - Professional    Workers Comp - Institutional    Workers Comp - 
Professional 
2140    26    22
```

table(appeal_all_df$IP_OP_IND)

```
  I     O
4068 30428
```

Data Cleansing

1. Remove columns that were used to generate other columns like 'PAYMENTS_AFTER_APPEAL','ADJUSTMENTS_AFTER_APPEAL'

```
colsToBeDropped <- c('DAYS_TO_FIRST_PAYMENT_AFT_APPEAL','DAYS_TO_LAST_PAYMENT_AFT_APPEAL',
                'DAYS_TO_FIRST_ADJUSTMENT_AFT_APPEAL','TOTAL_PAY_ADJ_AMT',
                'DAYS_TO_LAST_ADJUSTMENT_AFT_APPEAL','APL_DISPOSITION_DESC','PAYER_DESC',
               'PAYMENTS_AFTER_APPEAL','ADJUSTMENTS_AFTER_APPEAL'
               ,'POST_APPEAL_DENIAL_AMOUNT','DAYS_TO_CLOSE_ACCOUNT','PRE_APPEAL_DENIAL_AMOUNT' )

appeal_all_df <- appeal_all_df %>%
  select(-one_of(colsToBeDropped))
```

2. Split Disposition code_desc and Payer org by '-' to get the codes and drop the descriptions. The codes can then be used as factors.

```
appeal_all_df <- appeal_all_df %>% separate(APL_DISPOSITION_CODE_DESC, " - ",
           into = c("APL_DISPOSITION_CODE", "APL_DISPOSITION_DESC"),
           remove = TRUE)

appeal_all_df <- appeal_all_df %>% separate(PAYER_ORG, " - ",
                  into = c("PAYER", "PAYER_DESC"),
                  remove = TRUE)
```

3. Drop Denial and Payer code rows that have very sparse data.

```
dropDenial <- c("DNL02", "DNL03", "DNL04","DNL05","DNL08","DNL16","DNL17","DNL18")
dropDenial
dropPayer <- c("TRICARE","Humana","Unicare","Workers Comp","Coventry",  "Self Pay","Medicaid", "Medicare")
appeal_all_df <- appeal_all_df %>% filter(!APL_DISPOSITION_CODE %in% dropDenial)
appeal_all_df <- appeal_all_df %>% filter(!PAYER %in% dropPayer)
```

4. Drop rows with NA values.

```
appeal_no_na_df <- appeal_all_df %>% drop_na
```

5. Drop rows with extremely high values. This left with around 10K rows

```
appeal_all_df <- appeal_all_df %>% filter(   between (FINAL_DENIAL_AMOUNT, -10000, 10000))
appeal_all_df <- appeal_all_df %>% filter(COVERED_CHARGES <80000)
```

6. Take a subset of 25%rows, as some models were very slow, using stratified sampling by columns "SUCCESS', and 'I-O-Ind'.

```
appeal_subset_df <- appeal_renamed_df %>% group_by(success,io) %>% sample_frac(size = 0.25)
```

7. Rename the columns to have smaller names to fit in plots.

```
appeal_renamed_df <- setnames(appeal_no_na_df, old=c("APL_DISPOSITION_CODE","IP_OP_IND","APPEALED_AMT",
'COVERED_CHARGES', "TOTAL_PAYMENTS","LENGTH_OF_STAY",
              "TOTAL_ADJUSTMENTS","DAYS_APPEAL_AFTER_DISCHARGE",
              "POST_APPEAL_CA_AMOUNT","PAY_ADJ_AMT_AFTER_APPEAL",
             "FINAL_DENIAL_AMOUNT","DAYS_TO_FIRST_PAY_ADJ_AFT_APPEAL","DAYS_TO_LAST_PAY_ADJ_AFT_APPEAL",
"SUCCESS", "PAYER"),
      new=c("code","io", "aplamt","covchg","totpay","los", "totadj","openday","postca",
"aplapayadj","finden","fpadj","lpadj","success","payer"),skip_absent=TRUE)
```

Deeper Dive into Data

Appeal_xda_plots.R

Summary

```
> summary(appeal_df)
    success          code            payper            io             aplamt            covchg           totpay
 Min.   :0.000   DNL09  :1040   Aetna     : 125   Min.   :0.000   Min.   :      9   Min.   :   61   Min.   :   -3
 1st Qu.:0.000   DNL11  : 706   BCBS      :2738   1st Qu.:1.000   1st Qu.:    395   1st Qu.:  424   1st Qu.:   81
 Median :1.000   DNL12  : 680   Cigna     : 141   Median :1.000   Median :    895   Median : 1145   Median :  177
 Mean   :0.649   DNL01  : 268   Commercial: 378   Mean   :0.889   Mean   :   6395   Mean   : 9151   Mean   : 3868
 3rd Qu.:1.000   DNL07  : 238   United    :  39   3rd Qu.:1.000   3rd Qu.:   3396   3rd Qu.: 7735   3rd Qu.: 1352
 Max.   :1.000   DNL13  : 205                     Max.   :1.000   Max.   :4953147   Max.   :79658   Max.   :73616
                 (Other): 284
      los           totadj          openday        postca          aplapayadj         fpadj          lpadj
 Min.   : 1.0   Min.   :-9752   Min.   :  8   Min.   :-60820   Min.   :-19505   Min.   :  1   Min.   :  2
 1st Qu.: 1.0   1st Qu.:  298   1st Qu.: 32   1st Qu.:   340   1st Qu.:   574   1st Qu.: 16   1st Qu.: 21
 Median : 1.0   Median :  828   Median : 50   Median :   909   Median :  1282   Median : 30   Median : 38
 Mean   : 3.1   Mean   : 4888   Mean   : 65   Mean   :  3797   Mean   :  7084   Mean   : 47   Mean   : 61
 3rd Qu.: 3.0   3rd Qu.: 4480   3rd Qu.: 79   3rd Qu.:  2986   3rd Qu.:  5098   3rd Qu.: 58   3rd Qu.: 77
 Max.   :62.0   Max.   :74807   Max.   :430   Max.   :149615   Max.   :149615   Max.   :396   Max.   :461
```

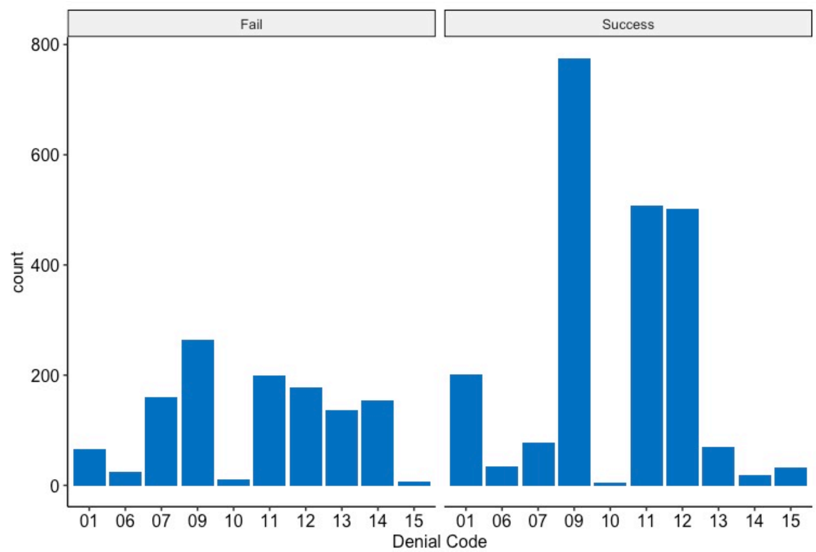
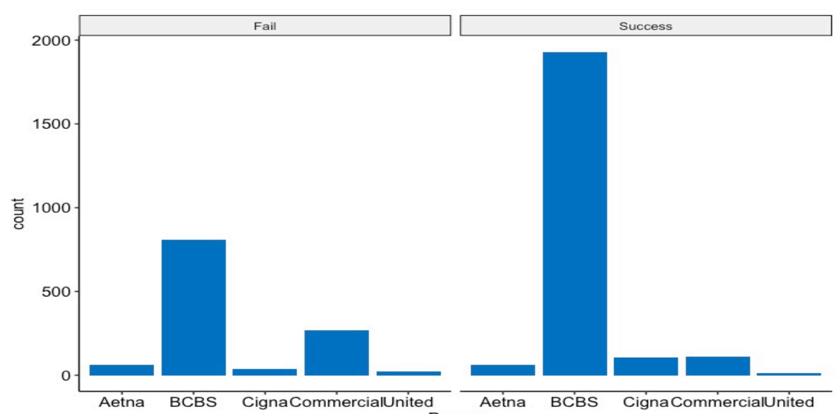
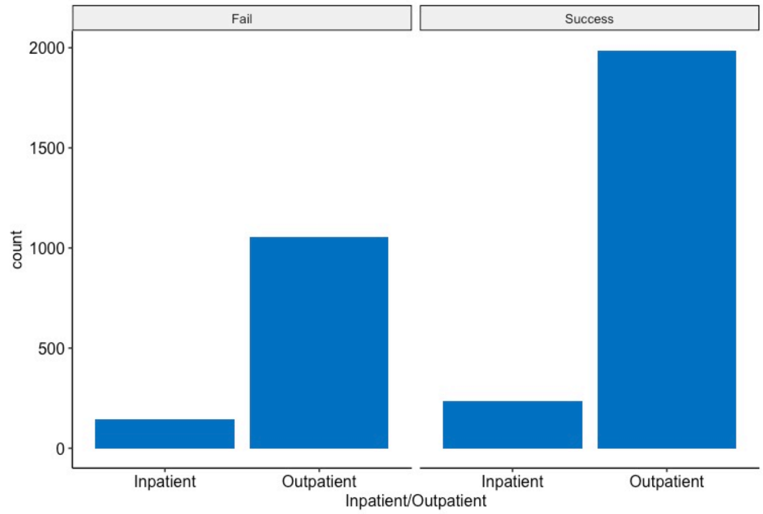
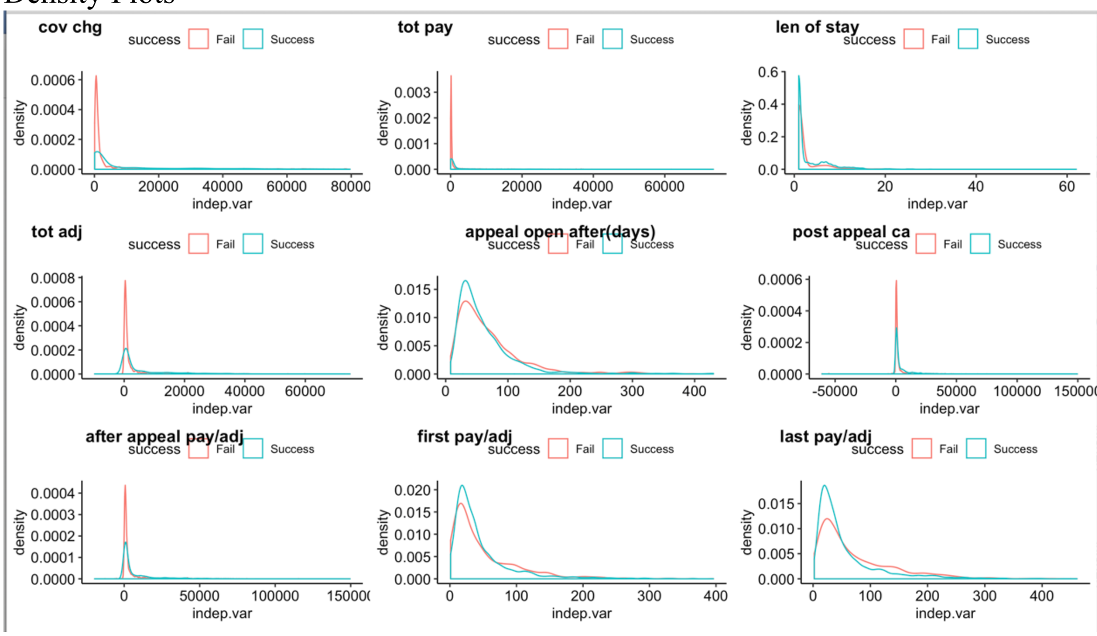


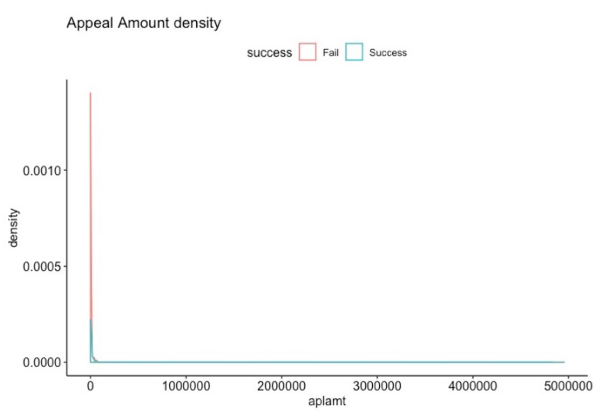
Figure 1:Denial Codes



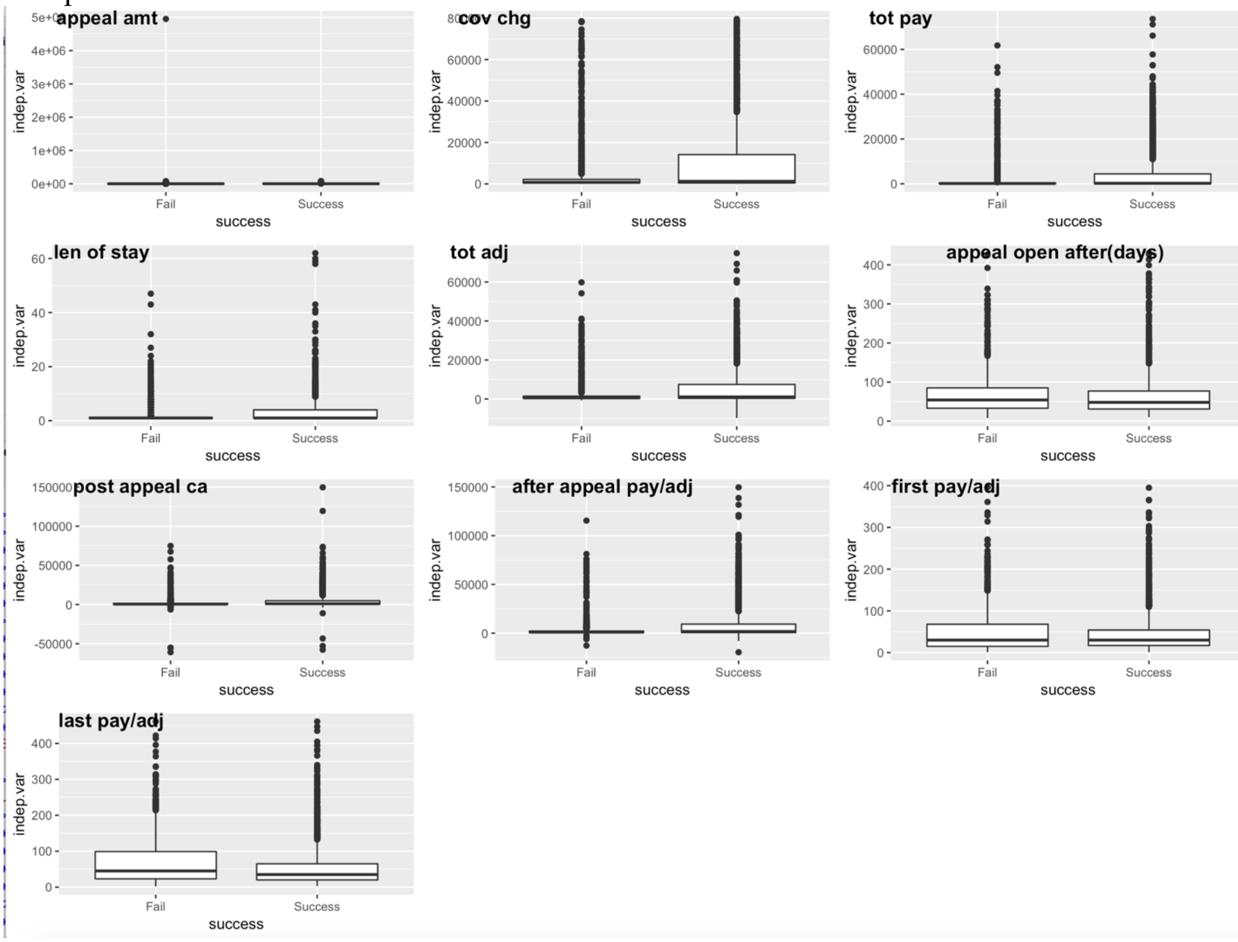


Density Plots





Box plots of continuous variables



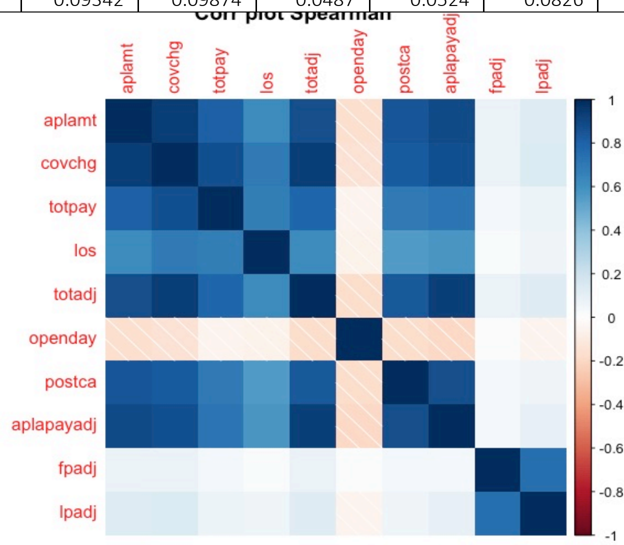
Finding Correlations

Appeal_featuresel_corr.R

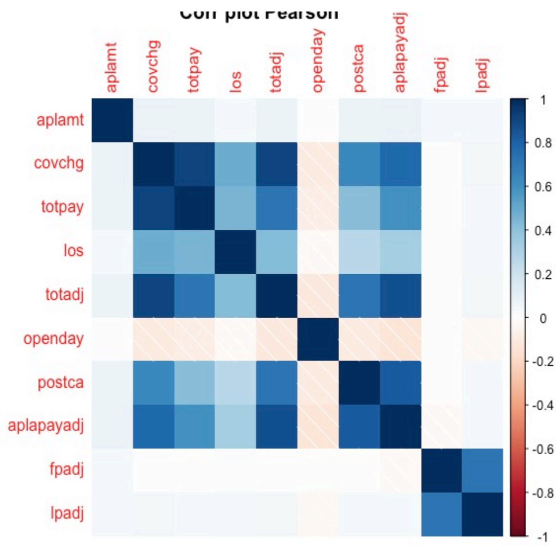
```
##Pearson's correlation
corr_vals <- cor(appeal_df[,5:14])
corr_vals
corrplot(corr_vals, method = "shade")
#Spearman Correaltion
corr_vals_spearman <- cor(appeal_df[,5:14],method ="spearman")
corr_vals_spearman
corrplot(corr_vals_spearman, method = "shade")

#Kendall Correlation
corr_vals_kendal <- cor(appeal_df[,5:14],method ="kendall")
corr_vals_kendal
corrplot(corr_vals_kendal, method = "shade")
```

Spearman Correlation Values

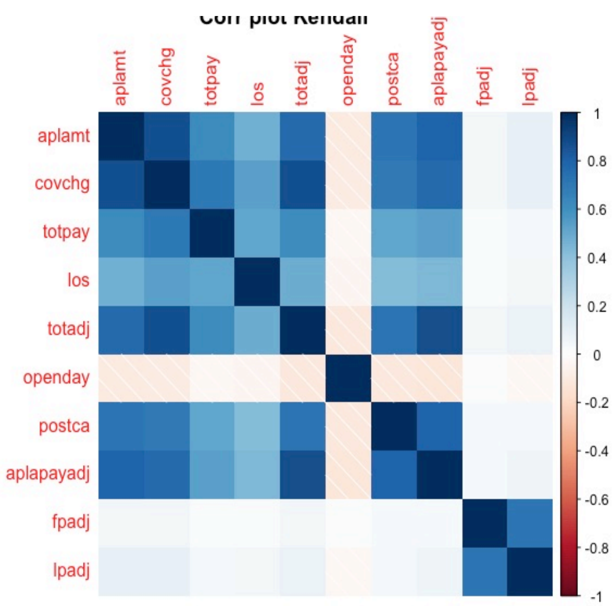
	aplamt	covchg	totpay	los	totadj	openday	postca	aplapayadj	fpadj	lpadj
aplamt	1	0.86008	0.6212	0.4793	0.7637	-0.1088	0.7192	0.7902	0.0549	0.0934
covchg	0.86008	1	0.7056	0.5389	0.8677	-0.0947	0.6953	0.7676	0.0545	0.0987
totpay	0.6212	0.70565	1	0.515	0.6118	-0.0355	0.5195	0.5393	0.0261	0.0487
los	0.47926	0.53889	0.515	1	0.4893	-0.0425	0.4262	0.446	0.0205	0.0524
totadj	0.76373	0.86765	0.6118	0.4893	1	-0.1112	0.7223	0.874	0.0536	0.0826
openday	-0.10881	-0.09471	-0.0355	-0.0425	-0.1112	1	-0.1127	-0.13	0.0124	-0.0385
postca	0.71919	0.69526	0.5195	0.4262	0.7223	-0.1127	1	0.7978	0.0317	0.0411
aplapayadj	0.79018	0.7676	0.5393	0.446	0.874	-0.13	0.7978	1	0.0332	0.0677
fpadj	0.05488	0.05448	0.0261	0.0205	0.0536	0.0124	0.0317	0.0332	1	0.7109
lpadj	0.09342	0.09874	0.0487	0.0524	0.0826	-0.0385	0.0411	0.0677	0.7109	1



Corr Plot 1:Spearman



Corr Plot 2: Pearson



Corr Plot 3:Kendall

Dimension Reduction

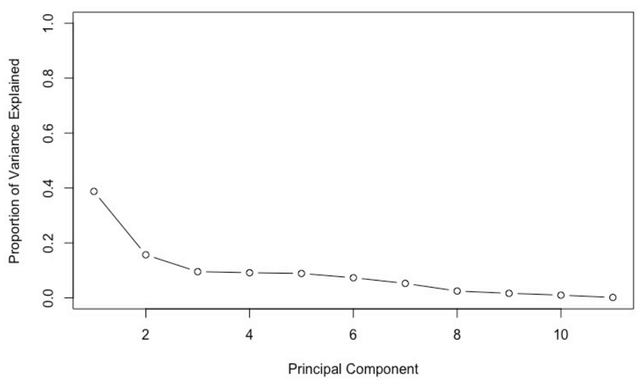
As there were some correlations between the numeric variables, tried PCA on numeric data.

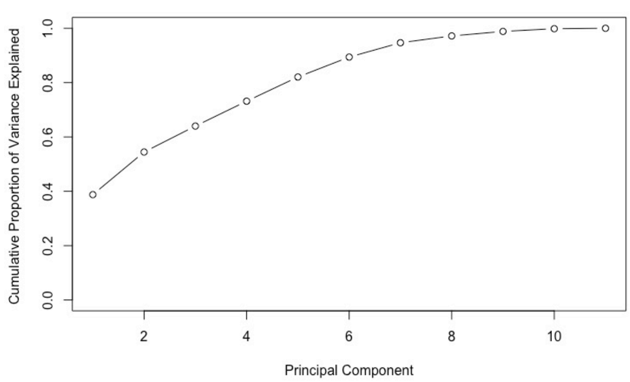
1. PCA on numerical data

appeal.pca = prcomp(appeal_df[,4:14], scale = TRUE)

x<-summary(appeal.pca)
x

```
Importance of components:
                         PC1   PC2    PC3    PC4   PC5    PC6    PC7   PC8    PC9    PC10   PC11
Standard deviation     2.065 1.314 1.0251 1.0039 0.989 0.8983 0.7618 0.525 0.4277 0.33152 0.1325
Proportion of Variance 0.388 0.157 0.0955 0.0916 0.089 0.0734 0.0528 0.025 0.0166 0.00999 0.0016
Cumulative Proportion  0.388 0.545 0.6401 0.7317 0.821 0.8940 0.9467 0.972 0.9884 0.99840 1.0000
```

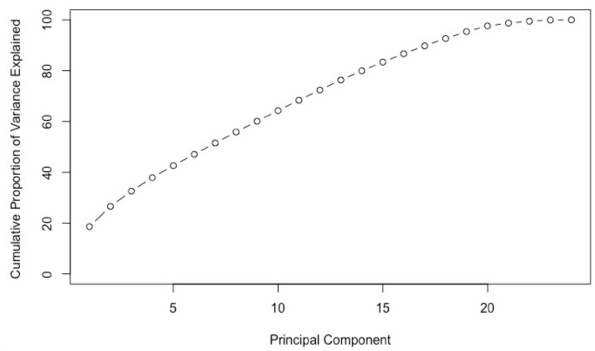




2. PCAMix on all data

library(PCAmixdata)
#pca with mix data
split.appeal <- splitmix(appeal_df2[,2:14])
X1 <- split.appeal$X.quanti
X2 <- split.appeal$X.quali
res.pcamix <- PCAmix(X.quanti=X1, X.quali=X2,rename.level=TRUE,
graph=FALSE)
res.pcamix

```
       Eigenvalue Proportion Cumulative
dim 1      4.4737    18.6406       18.6
dim 2      1.9123     7.9680       26.6
dim 3      1.4356     5.9815       32.6
dim 4      1.2774     5.3225       37.9
dim 5      1.1297     4.7071       42.6
dim 6      1.0728     4.4700       47.1
dim 7      1.0660     4.4418       51.5
dim 8      1.0420     4.3418       55.9
dim 9      1.0189     4.2452       60.1
dim 10     0.9980     4.1585       64.3
dim 11     0.9842     4.1010       68.4
dim 12     0.9629     4.0122       72.4
dim 13     0.9474     3.9477       76.3
dim 14     0.8681     3.6173       80.0
dim 15     0.8285     3.4523       83.4
dim 16     0.7845     3.2686       86.7
dim 17     0.7460     3.1082       89.8
dim 18     0.6901     2.8755       92.7
dim 19     0.6547     2.7281       95.4
dim 20     0.5413     2.2556       97.6
dim 21     0.2632     1.0966       98.7
dim 22     0.1798     0.7494       99.5
dim 23     0.1062     0.4423       99.9
dim 24     0.0164     0.0683      100.0
```



As none of the dimensions have captured a sizeable variance, will not use the dimensions generated by PCA.

3. LDA on all data

```
> appeal.lda
Call:
lda(success ~ ., data = appeal_scaled_df)

Prior probabilities of groups:
    0     1 
0.351 0.649 

Group means:
  codeDNL06 codeDNL07 codeDNL09 codeDNL10 codeDNL11 codeDNL12 codeDNL13 codeDNL14 codeDNL15 payerBCBS payerCigna
0    0.0208    0.1333     0.221   0.00833     0.166     0.148    0.1133    0.1283   0.00583     0.674     0.0292
1    0.0153    0.0351     0.349   0.00180     0.228     0.226    0.0311    0.0081   0.01441     0.869     0.0477
  payerCommercial payerUnited    io  aplamt covchg totpay    los totadj openday postca aplapayadj fpadj lpadj
0          0.2242     0.02000 0.878 0.00148 0.0609 0.0239 0.0219  0.145   0.145  0.299      0.137 0.123 0.152
1          0.0491     0.00675 0.894 0.00118 0.1430 0.0681 0.0403  0.188   0.129  0.311      0.168 0.112 0.117

Coefficients of linear discriminants:
                    LD1
codeDNL06       -0.4183
codeDNL07       -1.5180
codeDNL09       -0.1030
codeDNL10       -2.0293
codeDNL11        0.1202
codeDNL12       -0.0611
codeDNL13       -1.7806
codeDNL14       -2.7945
codeDNL15        0.1684
payerBCBS        0.9446
payerCigna       1.1132
payerCommercial -0.7606
payerUnited     -0.2457
io              -0.3185
aplamt          -2.0117
covchg          -3.0664
totpay           3.6549
los              0.6822
totadj           5.1983
openday          0.1381
```

This showed the categorical variables were more important. Also splitting the data and running sometimes gave errors of collinearity in date. Hence proceeded with some more feature selection and collinearity tests.

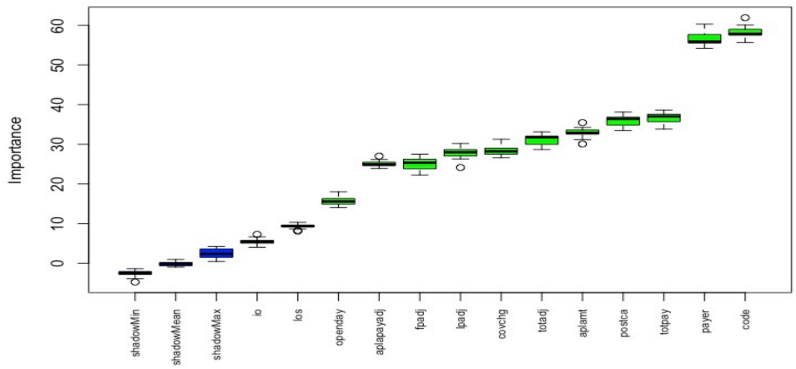
Feature Selection

The following automatic feature selection methods were used:

1. BORUTA

library(Boruta)
boruta.appeal_data <- Boruta(success~., data = appeal_subset_df, doTrace = 2)
print(boruta.appeal_data)
Boruta performed 15 iterations in 9.67 secs.

13 attributes confirmed important: aplamt, aplapayadj, code, covchg, fpadj and 8 more;
No attributes deemed unimportant.



2. Recursive Feature Elimination using random forest based rfFuncs (Caret p[ackage)

```
> ####################################
> set.seed(100)
> options(warn=-1)
> subsets <- c(1:5, 10, 15, 18)
> ctrl <- rfeControl(functions = rfFuncs,
+                    method = "repeatedcv",
+                    repeats = 5,
+                    verbose = FALSE)
> lmProfile <- rfe(x=appeal_subset_df[, 2:14], y=appeal_subset_df$success,
+                  sizes = subsets,
+                  rfeControl = ctrl, metric="Accuracy")
> lmProfile

Recursive feature selection

Outer resampling method: Cross-Validated (10 fold, repeated 5 times)

Resampling performance over subset size:

 Variables  RMSE Rsquared   MAE RMSESD RsquaredSD   MAESD Selected
         1 0.440    0.153 0.386 0.0122     0.0362 0.00976
         2 0.422    0.221 0.362 0.0119     0.0386 0.00957
         3 0.407    0.275 0.343 0.0115     0.0393 0.00952
         4 0.393    0.328 0.325 0.0114     0.0379 0.01030
         5 0.387    0.349 0.319 0.0120     0.0422 0.01006
        10 0.375    0.383 0.284 0.0141     0.0458 0.01182
        13 0.374    0.388 0.287 0.0139     0.0462 0.01181        *

The top 5 variables (out of 13):
   code, payer, postca, totpay, lpadj
```

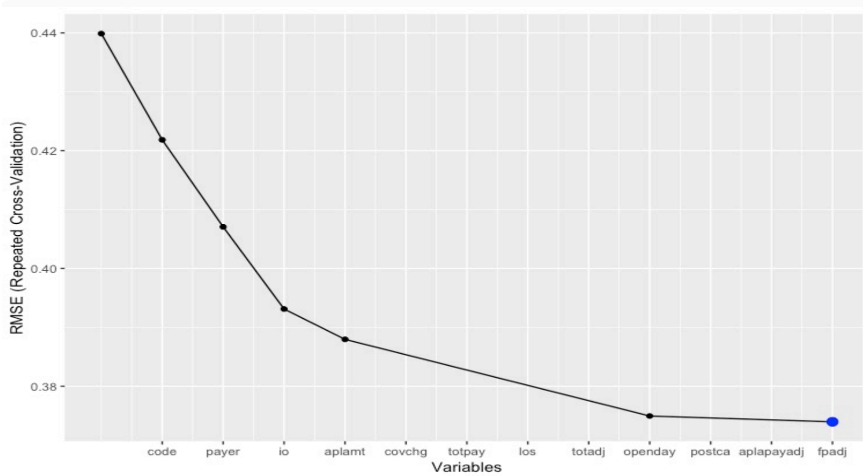


Figure 2:Feature Importance from RFE

3. STEP AIC

```
> model.step

Call:  glm(formula = success ~ code + payer + totadj + lpadj + fpadj + 
    io + totpay + covchg + postca, family = binomial(link = "logit"), 
    data = appeal_subset_df)

Coefficients:
    (Intercept)        codeDNL06        codeDNL07        codeDNL09        codeDNL10        codeDNL11  
         -1.412           -0.376           -1.470           -0.161           -2.354            0.154  
      codeDNL12        codeDNL13        codeDNL14        codeDNL15        payerBCBS       payerCigna  
         -0.106           -1.707           -3.140            0.314            0.955            1.180  
payerCommercial      payerUnited           totadj            lpadj            fpadj               io  
         -0.857           -0.231            6.601           -2.586            1.909           -0.415  
         totpay           covchg           postca  
          5.071           -3.806            4.357  

Degrees of Freedom: 3420 Total (i.e. Null);  3400 Residual
Null Deviance:      4430 
Residual Deviance: 3530         AIC: 3570
> # get the shortlisted variable
> shortlistedVars <- names(unlist(model.step[[1]]))
> #shortlistedVars
> # remove intercept
> shortlistedVars <- shortlistedVars[!shortlistedVars %in% "(Intercept)"]
> print(shortlistedVars)
 [1] "codeDNL06"         "codeDNL07"         "codeDNL09"         "codeDNL10"         "codeDNL11"         "codeDNL12"        
 [7] "codeDNL13"         "codeDNL14"         "codeDNL15"         "payerBCBS"         "payerCigna"        "payerCommercial"  
[13] "payerUnited"       "totadj"            "lpadj"             "fpadj"             "io"                "totpay"           
[19] "covchg"            "postca"
```

```
model.full <- glm(success ~ . , data= appeal_subset_df,
            family=binomial)
coef(model.full)
model.full

library(MASS)
step.model <- model.full %>% stepAIC(trace = FALSE)
coef(step.model)
step.model
```

```
> step.model

Call:  glm(formula = success ~ code + payer + io + covchg + totpay + 
    totadj + postca + fpadj + lpadj, family = binomial(link = "logit"), 
    data = appeal_subset_df)

Coefficients:
    (Intercept)        codeDNL06        codeDNL07        codeDNL09        codeDNL10        codeDNL11  
         -1.412           -0.376           -1.470           -0.161           -2.354            0.154  
      codeDNL12        codeDNL13        codeDNL14        codeDNL15        payerBCBS       payerCigna  
         -0.106           -1.707           -3.140            0.314            0.955            1.180  
payerCommercial      payerUnited               io           covchg           totpay           totadj  
         -0.857           -0.231           -0.415           -3.806            5.071            6.601  
         postca            fpadj            lpadj  
          4.357            1.909           -2.586  

Degrees of Freedom: 3420 Total (i.e. Null);  3400 Residual
Null Deviance:      4430 
Residual Deviance: 3530         AIC: 3570
```

Collinearity Check

Use vif (variance inflation factors) to figure out collinearity

```
model.full <- lm(success~., data=appeal_df)
summary(model.full)

vif(model.full) # variance inflation factors

sqrt(vif(model.full)) > 2 # problem?
```

```
> sqrt(vif(model.full)) > 2 # problem?
           GVIF    Df GVIF^(1/(2*Df))
code       FALSE  TRUE           FALSE
payer      FALSE FALSE           FALSE
io         FALSE FALSE           FALSE
aplamt     FALSE FALSE           FALSE
covchg      TRUE FALSE            TRUE
totpay      TRUE FALSE           FALSE
los        FALSE FALSE           FALSE
totadj      TRUE FALSE           FALSE
openday    FALSE FALSE           FALSE
postca     FALSE FALSE           FALSE
aplapayadj  TRUE FALSE           FALSE
fpadj      FALSE FALSE           FALSE
lpadj      FALSE FALSE           FALSE
```

Drop the Use vif (variance inflation factors) to figure out impact on collinearity

####After removing collinear variables

model.noncollinear <- lm(formula=success ~ code + payer + postca +io+los + aplamt +aplapayadj + openday + fpadj, data=appeal_df)

 # Evaluate Collinearity

vif(model.noncollinear) # variance inflation factors

sqrt(vif(model.noncollinear)) > 2 # problem?

```
> # Evaluate Collinearity
> vif(model.noncollinear) # variance inflation factors
               GVIF Df GVIF^(1/(2*Df))
code       1.615013  9        1.026988
payer      1.237160  4        1.026959
postca     3.177379  1        1.782520
io         1.069571  1        1.034201
los        1.226553  1        1.107499
aplamt     1.013341  1        1.006649
aplapayadj 3.349079  1        1.830049
openday    1.134470  1        1.065115
fpadj      1.078700  1        1.038605
> sqrt(vif(model.noncollinear)) > 2 # problem?
            GVIF    Df GVIF^(1/(2*Df))
code       FALSE  TRUE           FALSE
payer      FALSE FALSE           FALSE
postca     FALSE FALSE           FALSE
io         FALSE FALSE           FALSE
los        FALSE FALSE           FALSE
aplamt     FALSE FALSE           FALSE
aplapayadj FALSE FALSE           FALSE
openday    FALSE FALSE           FALSE
fpadj      FALSE FALSE           FALSE
```

 # Test for Autocorrelated Errors

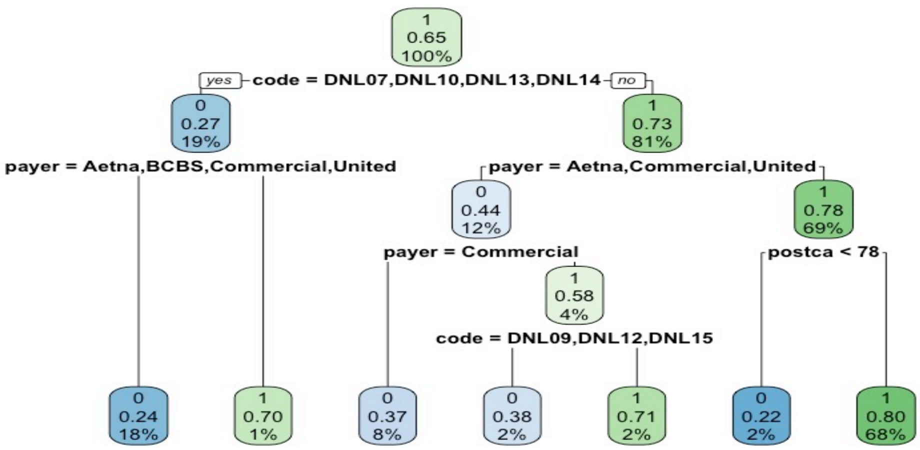
durbinWatsonTest(model.noncollinear)
library(gvlma)
gvmodel <- gvlma(model.noncollinear)
summary(gvmodel)

```
ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
Level of Significance =  0.05

Call:
 gvlma(x = model.noncollinear)

                    Value   p-value                   Decision
Global Stat        714.99 0.000e+00 Assumptions NOT satisfied!
Skewness           248.97 0.000e+00 Assumptions NOT satisfied!
Kurtosis            52.36 4.614e-13 Assumptions NOT satisfied!
Link Function       10.12 1.469e-03 Assumptions NOT satisfied!
Heteroscedasticity 403.54 0.000e+00 Assumptions NOT satisfied!
```

As the assumptions are not satisfied only linear models will not be best option. So will be trying a mix of models

Decision Tree

Plotted decision tree. After pruning that also showed that the categorical factors like denial code and payer contribute more to the prediction.

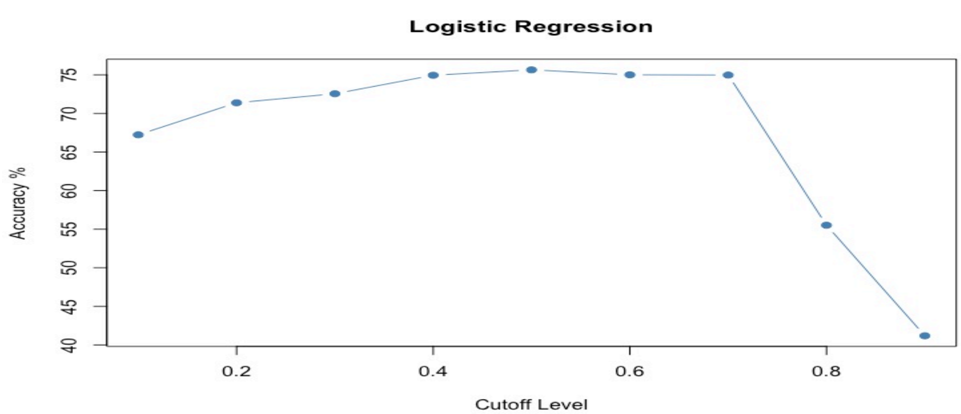


Modeling

As the problem statement is to identify factors that can predict if an appeal is successful or not, I decided to use classification models. As the data had collinearity issues I wanted to compare the results of the linear models with tree based models.
Appeal_modeling.R

Identify the cutoff point for prediction values

```
cutoffs <- seq(0.1,0.9,0.1)
accuracy <- NULL
for (i in seq(along = cutoffs)){
  prediction <- ifelse(model.selected.rfe$fitted.values >= cutoffs[i], 1, 0) #Predicting for cut-off
  accuracy <- c(accuracy,length(which(appeal_scaled_df$success ==prediction))/length(prediction)*100)}
plot(cutoffs, accuracy, pch =19,type='b',col= "steelblue",
     main ="Logistic Regression", xlab="Cutoff Level", ylab = "Accuracy %")
```



Data Split

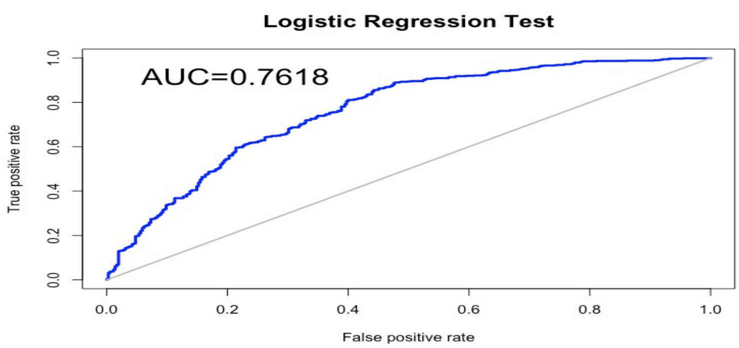
Used the caret package to split the data 70, 30 into training and test data sets. The createDataPartition() keeps the proportion of categories of the "Y" column in both training and test datasets.

Models Considered

1. Logistic Regression
2. LDA
3. Multivariate Adaptive Regression Splines (MARS)
4. SVM
5. Decision Trees
6. Random Forest
7. Artificial Neural Network

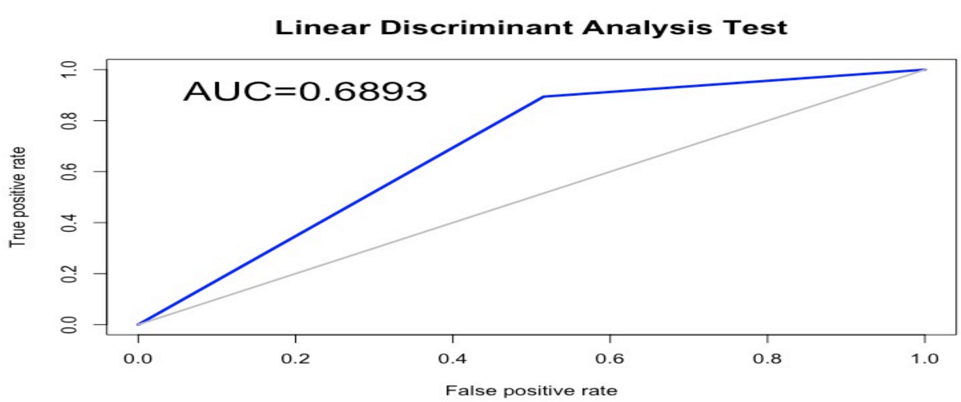
Logistic Regression

```
model.logit <- glm(success ~ . , data= train.data,
           family=binomial)
model.logit
```



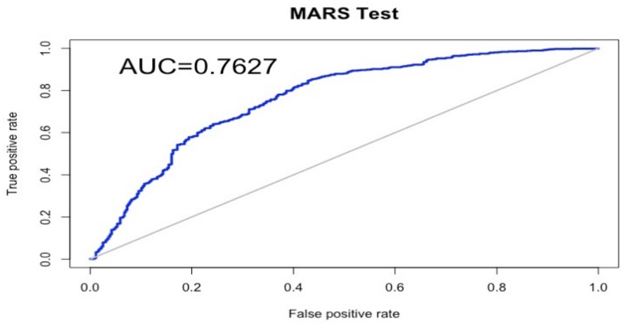
Linear Discriminant Analysis

```
library("MASS")
model.Lda = lda(formula = success ~ .,
          data = train.data )
summary(model.Lda)
```



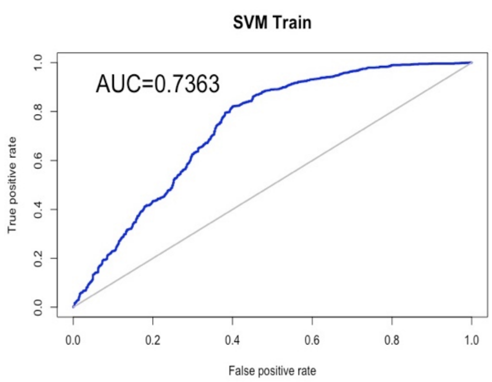
Multivariate Adaptive Regression Splines (MARS)

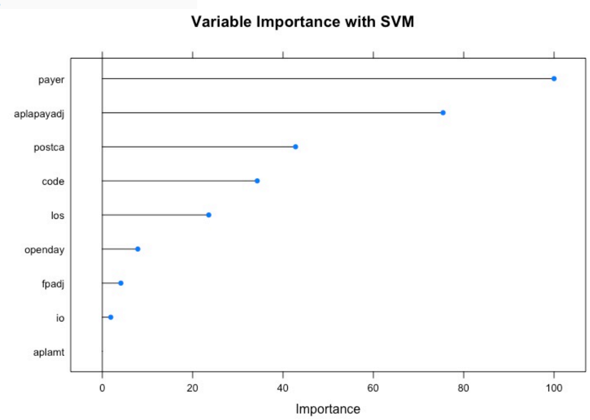
model_mars = train(success ~ ., data=train.data, method='earth',tuneLength = 5,trControl=fitControl)
fitted <- predict(model_mars)



Support Vector Machines

```
> model.svm <- train(success ~., data = train.data, method = "svmLinear",
+                    trControl=fitControl,
+                    preProcess = c("center", "scale"),
+                    tuneLength = 10)
> model.svm
Support Vector Machines with Linear Kernel

2395 samples
   9 predictor

Pre-processing: centered (20), scaled (20)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2155, 2156, 2155, 2156, 2155, 2155, ...
Resampling results:

  RMSE   Rsquared  MAE
  0.489  0.171     0.292

Tuning parameter 'C' was held constant at a value of 1
```

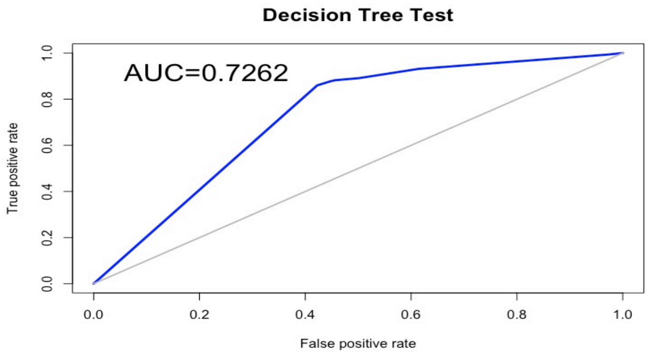




Decision Trees

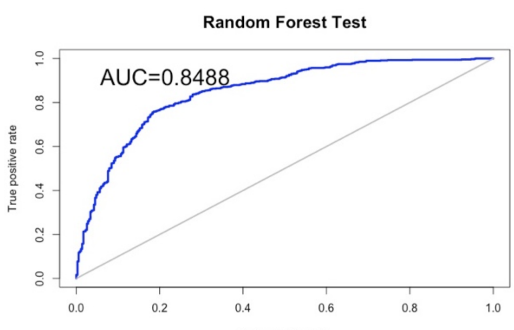
model.tree <- rpart(formula = success ~ .,
data = train.data, control=rpart.control(cp=.01))

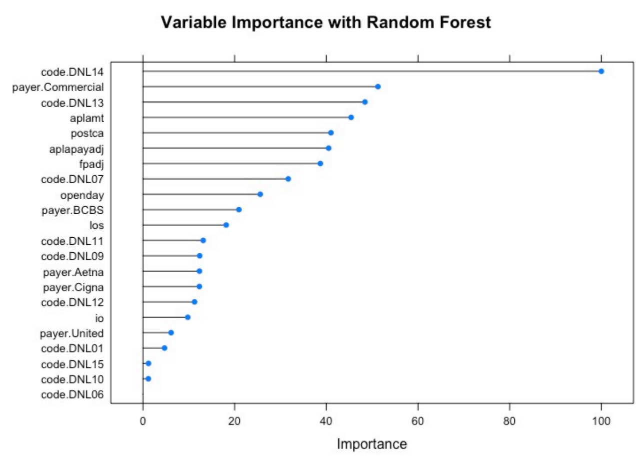
rpart.plot(model.tree)



Random Forest

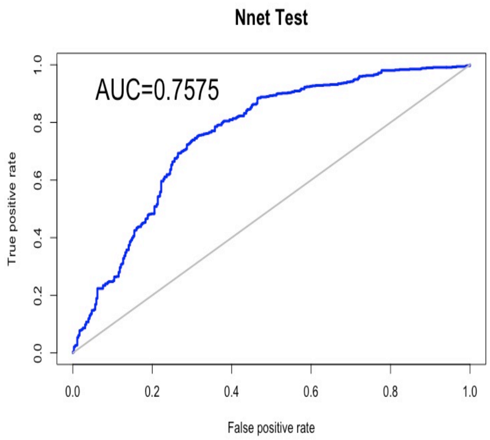
```
> model.rf = train(success ~ ., data=train.data, method='rf', tuneLength=5, trControl = fitControl)
> model.rf
Random Forest

2395 samples
   9 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2156, 2155, 2155, 2155, 2156, 2156, ...
Resampling results across tuning parameters:

  mtry  RMSE   Rsquared  MAE
   2    0.406  0.302     0.357
   6    0.387  0.346     0.304
  11    0.389  0.340     0.297
  15    0.390  0.336     0.295
  20    0.392  0.331     0.295

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 6.
```

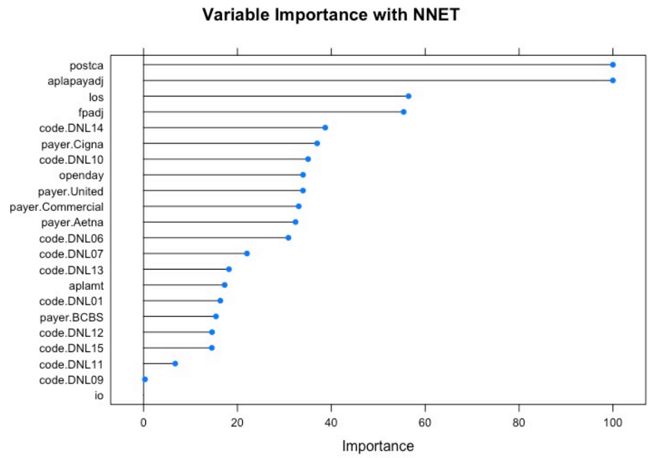




Artificial Neural Network

model.nnet = train(success ~ ., data=train.data, method='nnet', tuneLength=5, trControl = fitControl,linout = TRUE)
model.nnet





Comparison of models

Train	GLM	LDA	MARS	SVM	Decision Trees	Random Forest	Nnet
F1 Score	0.812	0.824	0.825	0.825	0.831	0.986	0.83
Accuracy	0.747	0.756	0.755	0.749	0.771	0.981	0.764
95% CI	(0.729, 0.764)	(0.738, 0.773)	(0.738, 0.772)	(0.731, 0.766)	(0.753, 0.787)	(0.975, 0.986)	(0.747, 0.781)
No Information Rate	0.647	0.647	0.647	0.647	0.647	0.647	0.647
P-Value [Acc > NIR]	<0.0000000000000002	<0.0000000000000002	<0.0000000000000002	<0.0000000000000002	< 0.0000000000000002	< 0.0000000000000002	<0.0000000000000002
Kappa	0.425	0.432	0.426	0.396	0.476	0.959	0.449
Mcnemar's Test P-Value	0.000000132	<0.0000000000000002	<0.0000000000000002	<0.0000000000000002	2.78E-11	0.000347	<0.0000000000000002
Sensitivity	0.846	0.884	0.891	0.914	0.874	0.994	0.892
Specificity	0.563	0.522	0.507	0.446	0.582	0.959	0.53
Pos Pred Value	0.78	0.772	0.768	0.752	0.793	0.978	0.777
Neg Pred Value	0.667	0.71	0.717	0.739	0.715	0.988	0.727
Prevalence	0.647	0.647	0.647	0.647	0.647	0.647	0.647
Detection Rate	0.548	0.572	0.577	0.592	0.565	0.643	0.577
Detection Prevalence	0.702	0.741	0.751	0.787	0.713	0.658	0.743
Balanced Accuracy	0.705	0.703	0.699	0.68	0.728	0.976	0.711
Test							
F1 Score	0.822	0.825	0.821	0.818	0.831	0.844	0.823
Accuracy	0.753	0.752	0.749	0.733	0.765	0.786	0.755
95% CI	(0.726, 0.78)	(0.725, 0.779)	(0.721, 0.775)	(0.705, 0.76)	(0.738, 0.791)	(0.759, 0.81)	(0.728, 0.781)
No Information Rate	0.654	0.654	0.654	0.654	0.654	0.654	0.654
P-Value [Acc > NIR]	3.84E-12	6.24E-12	4.12E-11	3.5E-08	7.64E-15	< 0.0000000000000002	1.44E-12
Kappa	0.425	0.409	0.404	0.339	0.451	0.504	0.43
Mcnemar's Test P-Value	9.39916E-07	3.29E-12	3.216E-10	< 0.0000000000000002	1.27722E-07	0.00000627	8.50897E-07
Sensitivity	0.87	0.894	0.884	0.917	0.882	0.887	0.872
Specificity	0.532	0.485	0.493	0.386	0.544	0.594	0.535
Pos Pred Value	0.779	0.766	0.767	0.738	0.785	0.805	0.78
Neg Pred Value	0.685	0.708	0.692	0.71	0.71	0.735	0.688
Prevalence	0.654	0.654	0.654	0.654	0.654	0.654	0.654
Detection Rate	0.569	0.585	0.578	0.599	0.577	0.58	0.57
Detection Prevalence	0.731	0.763	0.753	0.812	0.735	0.72	0.731
Balanced Accuracy	0.701	0.689	0.688	0.651	0.713	0.741	0.704

Conclusion

The tests during exploratory data analysis had shown that the data did not meet the assumptions for linearity. Comparing the results, the non-linear classifiers performed slightly better than linear classifiers for training set but were comparable for test set. Random Forest performed as the best model.
The success of the appeal depends on Type of denial, payer, appeal amount, payment/adjustment amount received after appeal, Contractual adjustment received after appeal. The other features patient type (inpatient/outpatient), length of stay, days appeal opened after discharge, first and last payment/adjustment days are not strong predictors of success of appeal.
Based on the above, I will have to reject my null hypothesis that these features do not play a role in prediction. Further analysis with different combinations of features is required to better access the prediction accuracy.

Files attached

I started with creating one R script but looking at dependencies between R packages being imported, I decided to go for modular code. The following files are attached:

1. Preliminary data cleansing (dataset not attached as it is big) - Appeal_prelim_datacleanse
2. Appeal_final_data.csv -The output of preliminary data cleansing (Attached)
3. Exploratory data analysis - Appeal_xda_plots.R
4. Feature selection -- Appeal_featuresel_corr.R
5. Modeling – Appeal_modeling.R

References

1. https://www.r-bloggers.com/introducing-xda-r-package-for-exploratory-data-analysis/
2. https://www.statmethods.net/stats/rdiagnostics.html
3. https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html
4. https://www.datacamp.com/community/tutorials/feature-selection-R-boruta
5. Caret package - A complete guide to machine learning in r
6. http://r-statistics.co/Variable-Selection-and-Importance-With-R.html
7. http://www.sthda.com/english/wiki/ggplot2-density-plot-quick-start-guide-r-software-and-data-visualization
8. http://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/81-ggplot2-easy-way-to-mix-multiple-graphs-on-the-same-page/
9. https://datascienceplus.com/perform-logistic-regression-in-r/
10. https://www.kaggle.com/sindhuee/r-caret-example
11. http://dataaspirant.com/2017/01/19/support-vector-machine-classifier-implementation-r-caret-package/

Books :

R In Action by Robert Kabacoff
An Introduction to Statistical Learning With Applications in R. by Robert Tibshirani, Trevor Hastie, Daniela Witten, Gareth James
Practical Statistics for Data Scientists by Peter Bruce, Andrew Bruce