Churn Prediction - PowerCo.

Fasih Atif

Define what is your (broad) research question! What you want to learn from the data or what you want to use it? (a) What is the main aim: prediction or causality?

Understanding our Data

The raw data consisted of 33 columns and 16096 observations in each column. The data represents the several characteristics of the company that could play an important part in their churning and retention. The data contains variables such as dates, forecasted consumption and prices, net/gross margins and churn status. It is often said that 80% of the time is spent in data cleaning and processing while just 20% is spent on running the analytics on the clean data. This saying was applicable in our case as well. There were several data quality issues such as missing values, outliers, wrong negative values, multicollinearity, and skewed variables. Our dependent variable would be the binary churn variable while the rest of the variables would act as the independent variables.

Exploring Data Analysis

The missing data (**Table 1**) was very small for majority of the variables, so we were able to easily replace the missing values with mean and median approximations.Any column that had more than 30% missing observations were removed. We then conducted Feature Engineering. We converted dates into durations to make them more useful. After getting done with our initial data cleaning and feature Engineering, we checked the distributions of the variables through histograms (**Figure 1-7**).

Our exploratory data analysis shows that around 10% of the of total customers have churned (**Figure 1**). All Consumption related variables (cons_12m ,cons_last_month,imp_cons,cons_gas_12m) and 2 of the forecast variables (forecast_cons_12m,forecast_meter_rent_12m) are extremely right skewed (**Figure 2,3**). The values on the higher end and lower ends of the distribution are potentially outliers. Next we checked a non-parametric estimator loess smoother to have a general idea about the functional form between a variable and churn. We will specifically focus on those variables who had skewed distributions (Figure). As expected, the functional forms are very non linear and hence we took log of the 6 variables(4 consumption and 2 forecast variables) to give the loess line a more linear shape (**Figure 7**). The transformation of the log variables can be seen in (**Figure 8**).

We used boxplots to tell us more about the outliers and what their values are. We then removed these values and replaced them with the mean of the respective columns values excluding the outliers.

Multicollinearity and Dummy Variables

When you have two independent variables that are very highly correlated, you definitely should remove one of them because you run into the multicollinearity conundrum and your regression model's regression coefficients related to the two highly correlated variables will be unreliable. We checked for multicollinearity between the explanatory variables. We drew a correlated matrix (**Figure 9**) to observe which pair of variables were highly correlated. Once we found the pairs, we calculated the Variance Inflation Factor (VIF) (**Table 2**) to confirm the correlation and remove one variable from the pairs.This was done to reduce multicollinearity. For this, we first created the correlation matrix. For pairs that have very high correlations, we will use

Variance Inflation Factor to determine which variable to remove. Categorical columns ('channel_sales' and 'has_gas') were converted into dummy columns and the reference columns were removed.

Machine Learning and Predictions

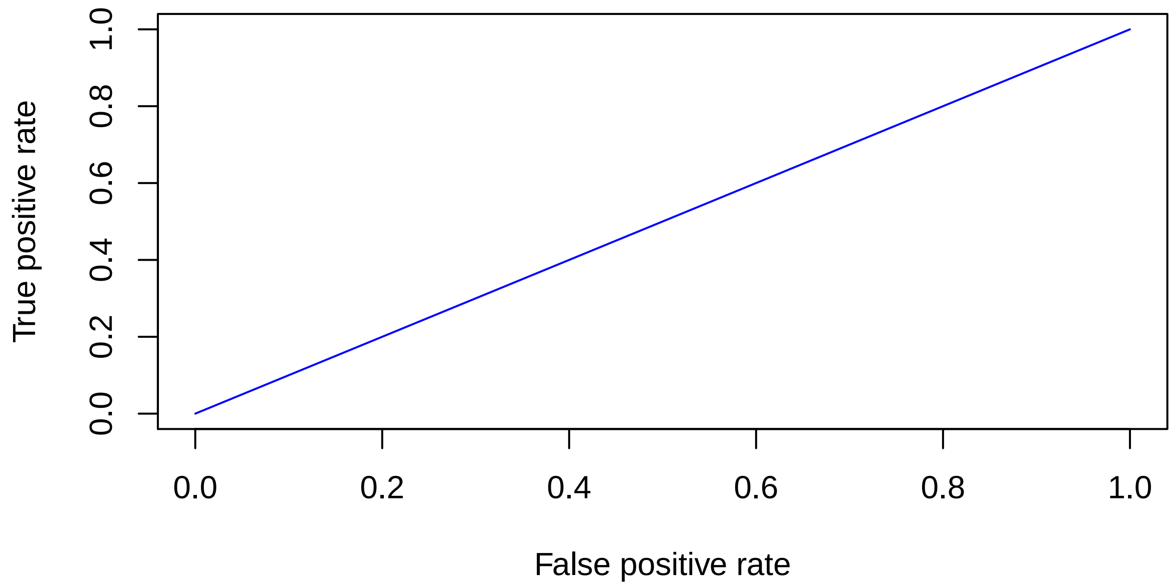
```r
##### SPLIT DATA INTO TEST/TRAIN DATASET #####
set.seed(2017)
intrain<- createDataPartition(df$churn,p=0.75,list=FALSE)
train <- df[intrain,]
test <- df[-intrain,]
```

```r
df_xgb <- df
df$churn <- as.factor(df$churn)

##### SPLIT DATA INTO TEST/TRAIN DATASET #####
set.seed(1111)
intrain<- createDataPartition(df$churn,p=0.75,list=FALSE)
train <- df[intrain,]
test <- df[-intrain,]

###########################################
##### LOGISTIC REGRESSION VIA GLM()  ####
###########################################

glm_model1 <- glm(churn ~ ., data = df, family = binomial("logit"))
summary(glm_model1)
```

```
##
## Call:
## glm(formula = churn ~ ., family = binomial("logit"), data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9887  -0.4976  -0.4219  -0.3448   2.7905
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -1.629e+00  1.427e+00  -1.142  0.25353
## forecast_discount_energy  -2.057e-03  5.527e-03  -0.372  0.70982
## nb_prod_act               -3.423e-02  4.459e-02  -0.768  0.44270
## months_active             -1.108e-02  1.986e-03  -5.578 2.43e-08 ***
## months_end                -1.139e-03  1.612e-02  -0.071  0.94367
## months_modif              -1.722e-03  1.118e-03  -1.541  0.12329
## months_renewal            -1.816e-02  1.543e-02  -1.176  0.23947
## channel_epum              -1.175e+01  2.621e+02  -0.045  0.96426
## channel_ewpa              -2.935e-01  1.404e-01  -2.091  0.03655 *
## channel_fixd              -1.179e+01  3.767e+02  -0.031  0.97502
## channel_foos               1.766e-01  8.370e-02   2.110  0.03483 *
## channel_lmke              -5.425e-01  1.192e-01  -4.550 5.36e-06 ***
## channel_sddi              -1.163e+01  1.529e+02  -0.076  0.93937
## channel_usil              -4.908e-02  1.178e-01  -0.417  0.67682
## has_gas_1                 -2.276e-01  9.669e-02  -2.354  0.01856 *
## forecast_price_energy_p1  -7.901e+00  2.602e+00  -3.036  0.00239 **
## forecast_price_energy_p2   2.230e+00  9.846e-01   2.265  0.02353 *
```

```
## forecast_price_pow_p1       1.667e-02  3.277e-02   0.509  0.61096
## margin_net_pow_ele          2.337e-02  2.483e-03   9.413  < 2e-16 ***
## net_margin                  3.890e-04  2.672e-04   1.456  0.14536
## pow_max                    -1.273e-02  9.500e-03  -1.340  0.18024
## ln_cons_12m                 1.665e-02  2.727e-02   0.611  0.54147
## ln_cons_last_month         -2.513e-02  1.634e-02  -1.538  0.12415
## ln_imp_cons                -4.625e-03  2.374e-02  -0.195  0.84553
## ln_forecast_cons_12m        1.243e-02  3.135e-02   0.397  0.69171
## ln_forecast_meter_rent_12m  1.058e-02  3.134e-02   0.337  0.73577
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 10401  on 16095  degrees of freedom
## Residual deviance: 10073  on 16070  degrees of freedom
## AIC: 10125
##
## Number of Fisher Scoring iterations: 12
```

```r
glm_model2 <- glm(as.factor(churn)~. -channel_sddi -channel_epum -net_margin -ln_forecast_meter_rent_12m

glm_model3 <- glm(as.factor(churn)~. -channel_sddi -channel_epum -net_margin -ln_forecast_meter_rent_12m



# df_test <- subset(df, select = c(forecast_discount_energy,forecast_price_energy_p1,forecast_price_ene
# glm_model <- lm(churn~., data = df, family = binomial("logit"),control=glm.control(maxit=100))



#Caret Model  Prediction
pred_type_test <- predict(glm_model1, newdata = test, type = "response")
pred_type_test <- ifelse(pred_type_test > 0.5, 1, 0)

cm_glm <- confusionMatrix(as.factor(pred_type_test),as.factor(test$churn))


# Producing ROC curve of model
#pred_response <- predict(glm_model1, newdata = test, type = "response")
predictFull <- prediction(pred_type_test,as.factor(test$churn))

# Plot AUC
auc_roc <- performance(predictFull, measure = 'tpr',x.measure = 'fpr')
plot(auc_roc, col = "blue")
```




```
auc_score_glm <- auc(test$churn, pred_type_test) #0.6221063
rmse_score_glm <- rmse(test$churn, pred_type_test) # 0.2994171
```

We used 3 Machine Learning models to predict whether a customer would churn or not and their probabilities.

Logistic Regression:

Logistic Regression belongs to the family of generalized linear models. It is a binary classification algorithm used when the response variable is dichotomous (1 or 0). Inherently, it returns the set of probabilities of target class. But, we can also obtain response labels using a probability threshold value.

For our analysis, we first split the data 75/25 into train and test models respectively. We ran the Logistic Regression via the ML Caret Package. We used K fold Cross Validation on our training set in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset. For further robustness check, we ran 3 variants of the Logistic Regression model with varying $$B coefficients.

The equation for our first model is:

$$churn = \beta_0+\beta_1 cons\_12m_i+\beta_2 log(cons\_last\_month)_i+\beta_3 log(imp\_cons)_i+\beta_4 forecast\_price\_energy\_p1_i+\beta_5 forecast\_p$$

We ran the logistic regression with all the explanatory variables and received the results as shown in

```
set.seed(2018)
intrain<- createDataPartition(df_xgb$churn,p=0.75,list=FALSE)
ml_train <- df[intrain,]
ml_test <- df[-intrain,]

labels <- ml_train$churn
```

```r
ts_label <- ml_test$churn

labels <- as.numeric(as.character(labels))
ts_label <- as.numeric(as.character(ts_label))

#XGBoost takes matrix for data hence we convert dataframe to matrix
# df_xgb_backup<- df_xgb
xgb_train <- ml_train %>% select(-churn)
xgb_train <- xgb.DMatrix(data = as.matrix(xgb_train),label = labels)

xgb_test <- ml_test %>% select(-churn)
xgb_test <- xgb.DMatrix(data = as.matrix(xgb_test),label = ts_label)

#default parameters
xgb_params <- list(booster = "gbtree",
                   objective = "binary:logistic",
                   eta=0.3, gamma=0,
                   max_depth=6,
                   min_child_weight=1,
                   subsample=1,
                   colsample_bytree=1)

# Calculate the best nround for this model. In addition, this function also returns CV error, which is a
set.seed(2018)
xgbcv <- xgb.cv( params = xgb_params, data = xgb_train, nrounds = 500, nfold = 5, showsd = T, stratified
```

```
## [1]  train-error:0.095531+0.002493   test-error:0.103047+0.007512
## Multiple eval metrics are present. Will use test_error for early stopping.
## Will train until test_error hasn't improved in 20 rounds.
##
## [11] train-error:0.091700+0.003118   test-error:0.098740+0.007256
## [21] train-error:0.084452+0.002687   test-error:0.097829+0.008136
## [31] train-error:0.077017+0.002225   test-error:0.096752+0.008367
## [41] train-error:0.069065+0.002436   test-error:0.097001+0.006765
## [51] train-error:0.062334+0.001860   test-error:0.097084+0.006316
## [61] train-error:0.056018+0.001637   test-error:0.096504+0.007065
## Stopping. Best iteration:
## [43] train-error:0.067947+0.002275   test-error:0.096421+0.006211
```

```r
# Iteration 47 gave lowest test error

elog <- as.data.frame(xgbcv$evaluation_log)
nround <- which.min(elog$test_error_mean)
##best iteration = 47
## The model returned lowest error at the 47th (nround) iteration.
# CV accuracy is 1-0.0968 = 90.32%

#first default - model training
xgb_model <- xgb.train(params = xgb_params,
                       data = xgb_train,
                       nrounds = nround,
                       watchlist = list(train=xgb_train,test=xgb_test),
                       print_every_n = 10, early_stop_round = 10,
```

```r
                        maximize = F ,
                        eval_metric = "error")
```

```
## [02:26:26] WARNING: amalgamation/../src/learner.cc:516:
## Parameters: { early_stop_round } might not be used.
##
##   This may not be accurate due to some parameters are only used in language bindings but
##   passed down to XGBoost core.  Or some parameters are not used but slip through this
##   verification. Please open an issue if you find above cases.
##
##
## [1]  train-error:0.095427    test-error:0.100895
## [11] train-error:0.091534    test-error:0.095924
## [21] train-error:0.084990    test-error:0.095427
## [31] train-error:0.078446    test-error:0.092197
## [41] train-error:0.071405    test-error:0.091451
## [43] train-error:0.069583    test-error:0.091700
```

```r
summary(xgb_model)
```

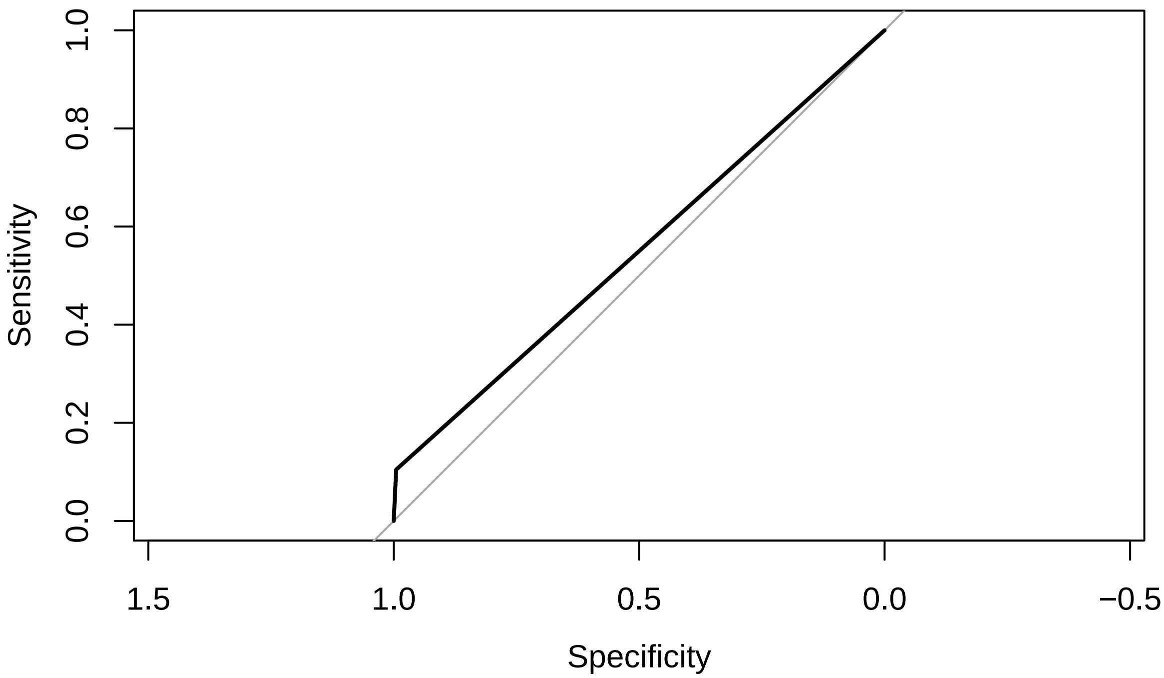
```
##                 Length Class              Mode
## handle               1 xgb.Booster.handle externalptr
## raw             123667 -none-             raw
## niter                1 -none-             numeric
## evaluation_log       3 data.table         list
## call                 9 -none-             call
## params              11 -none-             list
## callbacks            2 -none-             list
## feature_names       25 -none-             character
## nfeatures            1 -none-             numeric
```

```r
#model prediction
xgbpred <- predict(xgb_model,xgb_test)

summary(xgbpred)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.001651 0.041295 0.065449 0.094780 0.109142 0.947142
```

```r
# The objective function binary:logistic returns output probabilities rather than labels. To convert it
# manually use a cutoff value. As seen above, I've used 0.5 as my cutoff value for predictions. We can
# accuracy using confusionMatrix() function from caret package.

#confusion matrix
xgbpred <- ifelse(xgbpred > 0.5,1,0)
cm_xgb <- confusionMatrix (as.factor(xgbpred), as.factor(ts_label))
#Accuracy - 86.54%`

#view variable importance plot
mat <- xgb.importance (feature_names = colnames(df_xgb),model = xgb_model)
```

```r
# The ggplot-backend method also performs 1-D clustering of the importance values, with bar colors
# corresponding to different clusters that have somewhat similar importance values.
xgb_feature_plot <- xgb.ggplot.importance (importance_matrix = mat)

library(Metrics)
roc_test <- roc(ts_label, xgbpred, algorithm = 2)
roc_xgb <- plot(roc_test)
```




```r
auc_xgb <- auc(ts_label, xgbpred) #0.5684948
rmse_xgb <- rmse(ts_label, xgbpred) # 0.2953
```

$$

APPENDIX A

APPENDIX B

Table 1 - Missing Values

Table 1: Missing Values in each Variable

	na_count	na_percent
id	0	0.00
activity_new	9545	59.30
campaign_disc_ele	16096	100.00
channel_sales	4218	26.21
cons_12m	0	0.00
cons_gas_12m	0	0.00
cons_last_month	0	0.00
date_activ	0	0.00
date_end	2	0.01
date_first_activ	12588	78.21
date_modif_prod	157	0.98
date_renewal	40	0.25
forecast_base_bill_ele	12588	78.21
forecast_base_bill_year	12588	78.21
forecast_bill_12m	12588	78.21
forecast_cons	12588	78.21
forecast_cons_12m	0	0.00
forecast_cons_year	0	0.00
forecast_discount_energy	126	0.78
forecast_meter_rent_12m	0	0.00
forecast_price_energy_p1	126	0.78
forecast_price_energy_p2	126	0.78
forecast_price_pow_p1	126	0.78
has_gas	0	0.00
imp_cons	0	0.00
margin_gross_pow_ele	13	0.08
margin_net_pow_ele	13	0.08
nb_prod_act	0	0.00
net_margin	15	0.09
num_years_antig	0	0.00
origin_up	87	0.54
pow_max	3	0.02
churn	0	0.00

Table 2 - Variance Inflation Factor (VIF)

Table 2: Variance Inflation Factor (VIF)

	VIF
forecast_discount_energy	1.256417
nb_prod_act	1.242218
num_years_antig	41.724615
contract_duration	2733.747467
months_active	2579.186182
months_end	90.275012
months_modif	1.410913
months_renewal	3.979559
channel_epum	1.005128
channel_ewpa	1.378587
channel_fixd	1.002336
channel_foos	2.301143
channel_lmke	1.574024
channel_sddi	1.011849
channel_usil	1.569362
has_gas_1	15.877874
forecast_price_energy_p1	3.285631
forecast_price_energy_p2	2.965002
forecast_price_pow_p1	5.430790
margin_gross_pow_ele	151.607239
margin_net_pow_ele	151.419021
net_margin	2.024534
pow_max	2.807251
ln_cons_12m	3.056839
ln_cons_gas_12m	15.732625
ln_cons_last_month	4.963743
ln_imp_cons	4.182487
ln_forecast_cons_12m	2.632955
ln_forecast_meter_rent_12m	2.326036

Table 3: Variance Inflation Factor (VIF)

	VIF
forecast_discount_energy	1.256417
nb_prod_act	1.242218
num_years_antig	41.724615
contract_duration	2733.747467
months_active	2579.186182
months_end	90.275012
months_modif	1.410913
months_renewal	3.979559
channel_epum	1.005128
channel_ewpa	1.378587
channel_fixd	1.002336
channel_foos	2.301143
channel_lmke	1.574024
channel_sddi	1.011849

	VIF
channel_usil	1.569362
has_gas_1	15.877874
forecast_price_energy_p1	3.285631
forecast_price_energy_p2	2.965002
forecast_price_pow_p1	5.430790
margin_gross_pow_ele	151.607239
margin_net_pow_ele	151.419021
net_margin	2.024534
pow_max	2.807251
ln_cons_12m	3.056839
ln_cons_gas_12m	15.732625
ln_cons_last_month	4.963743
ln_imp_cons	4.182487
ln_forecast_cons_12m	2.632955
ln_forecast_meter_rent_12m	2.326036

APPENDIX C

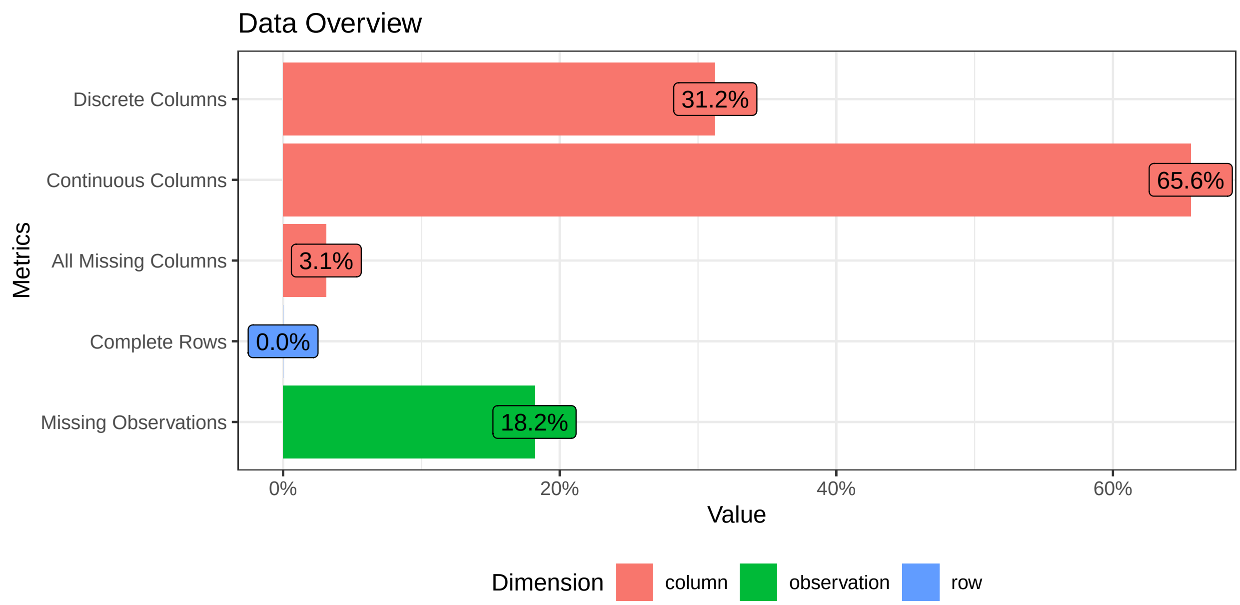


Figure: 1 - Churn Rate

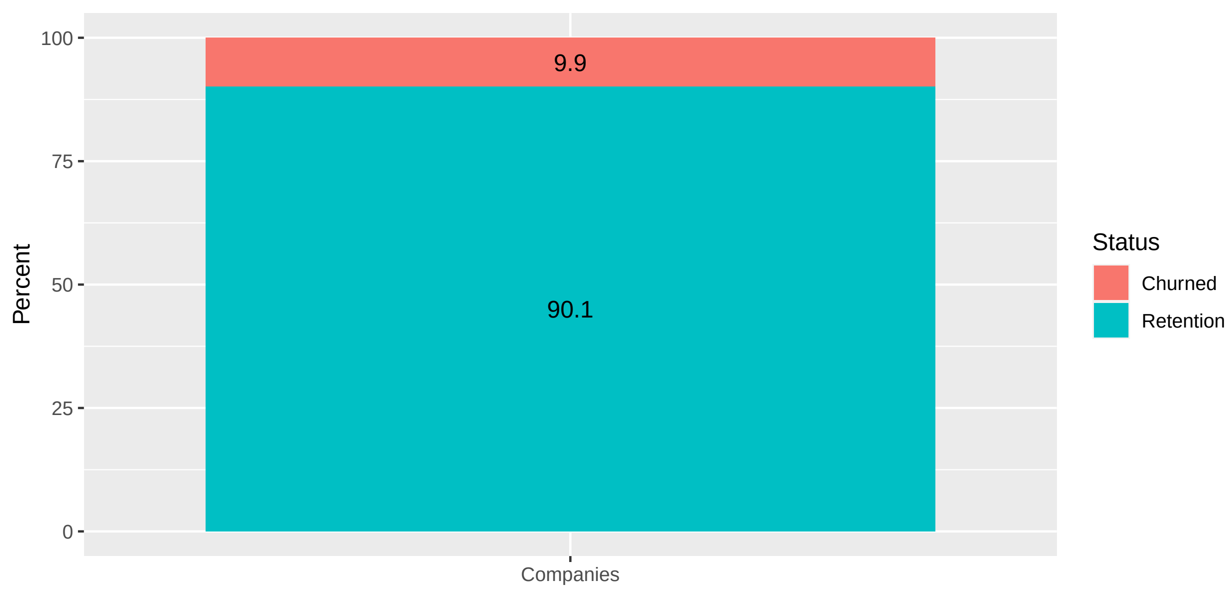
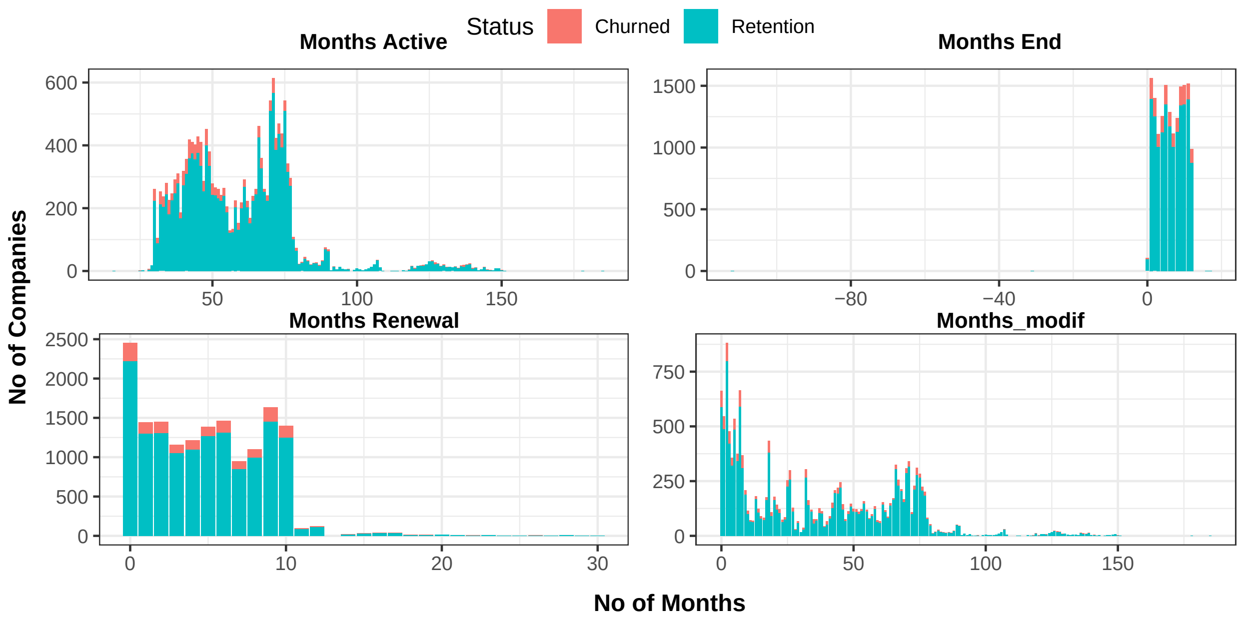


Figure: 2 - Month Duration charts



Status
Churned
Retention
Months Active
Months End
Months Renewal
Months_modif
No of Companies
No of Months
600
400
200
0
50
100
150
1500
1000
500
0
−80
−40
0
2500
2000
1500
1000
500
0
10
20
30
750
500
250
0
50
100
150

Figure: 3 - Consumption Variables Exploration

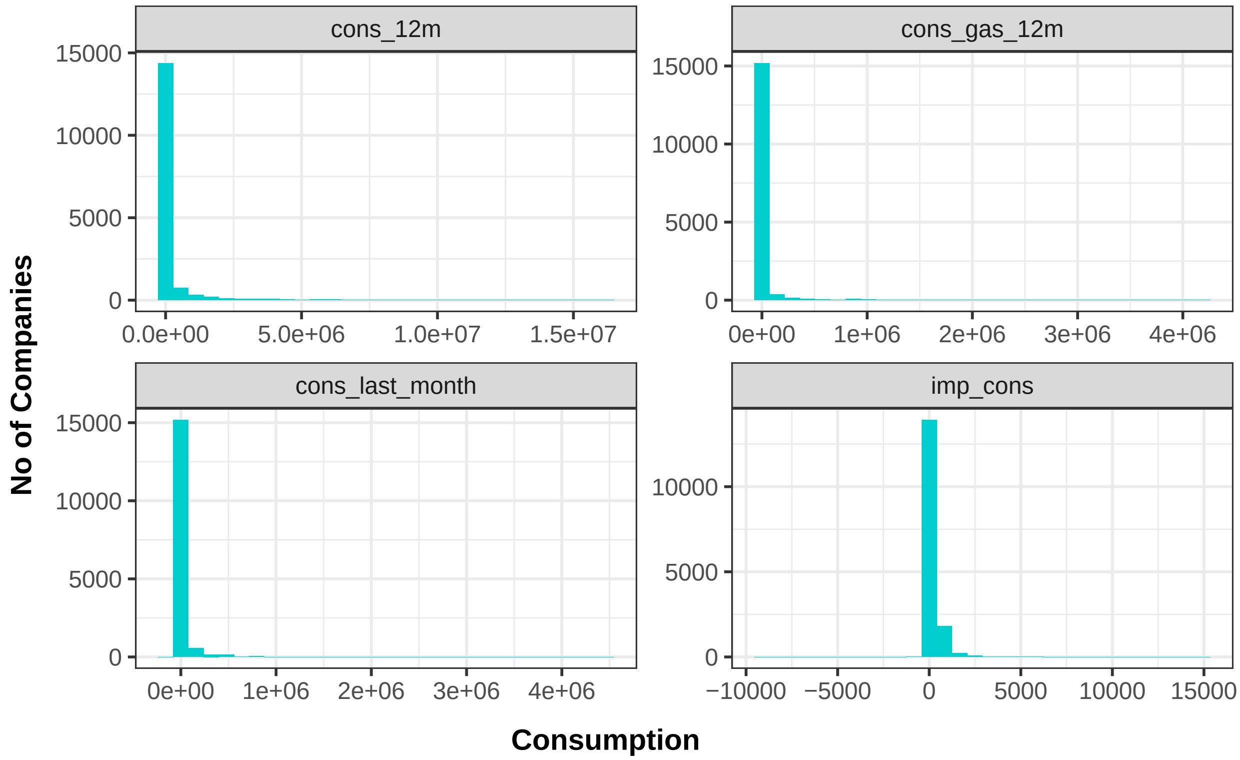


Figure: 4 - Forecast Variables Exploration

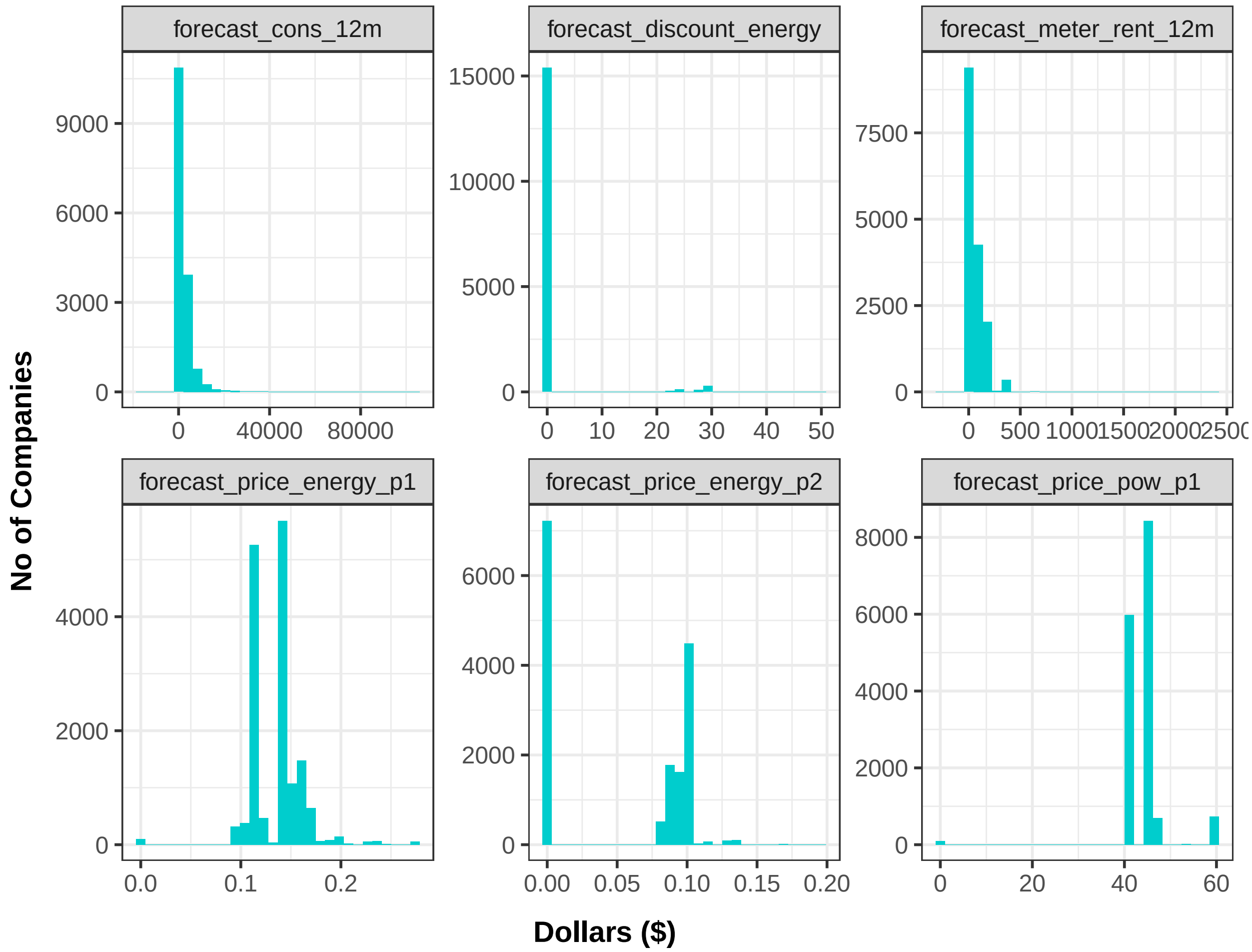

forecast_cons_12m
forecast_discount_energy
forecast_meter_rent_12m
forecast_price_energy_p1
forecast_price_energy_p2
forecast_price_pow_p1
No of Companies
Dollars ($)

Figure: 5 - Margin Variables Exploration

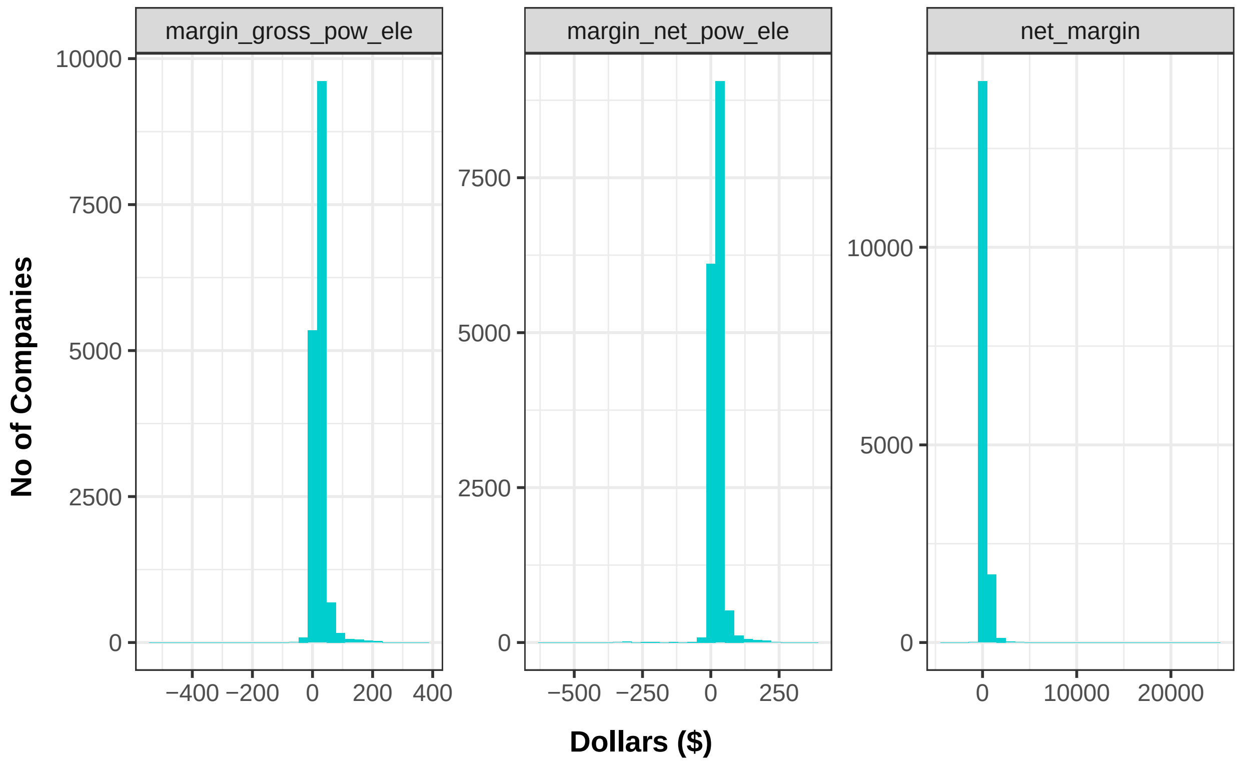


Figure: 6 - Other Variables Exploration

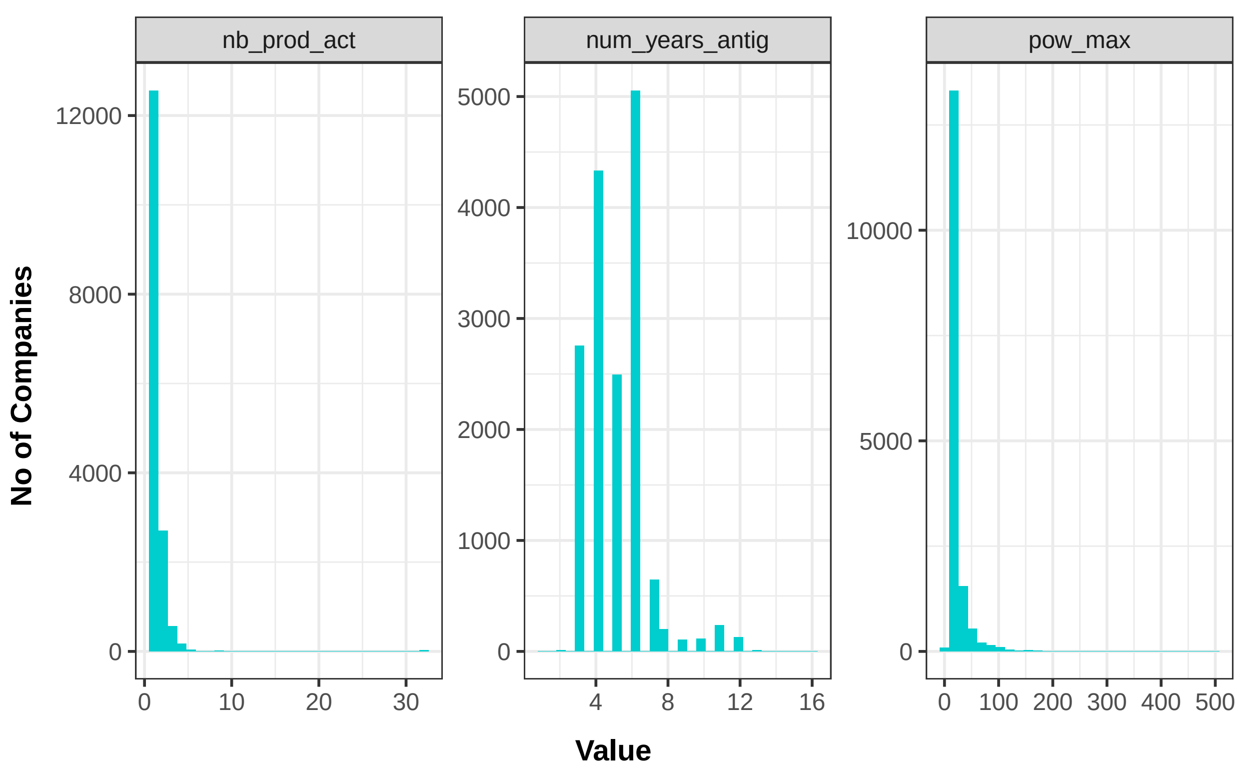
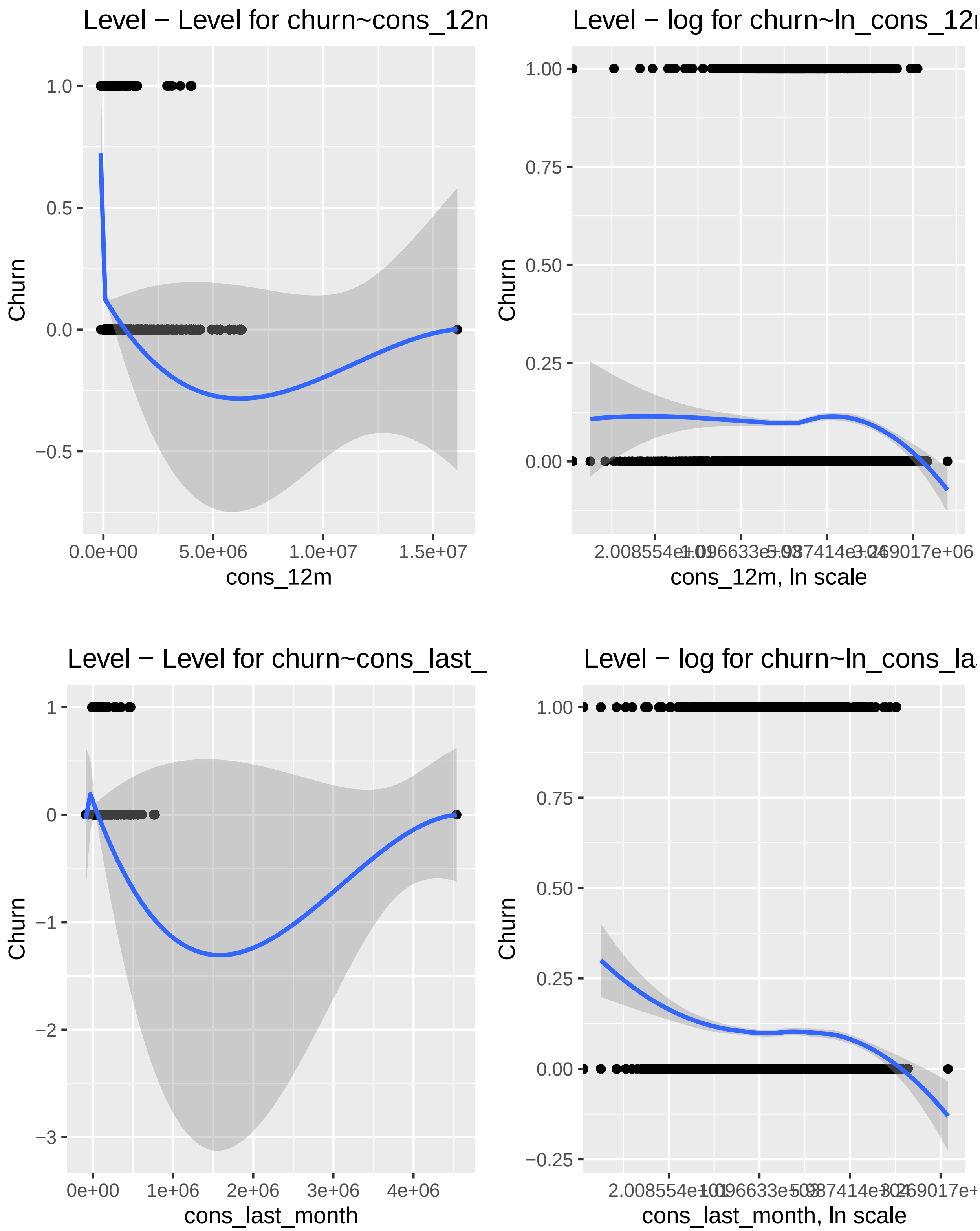
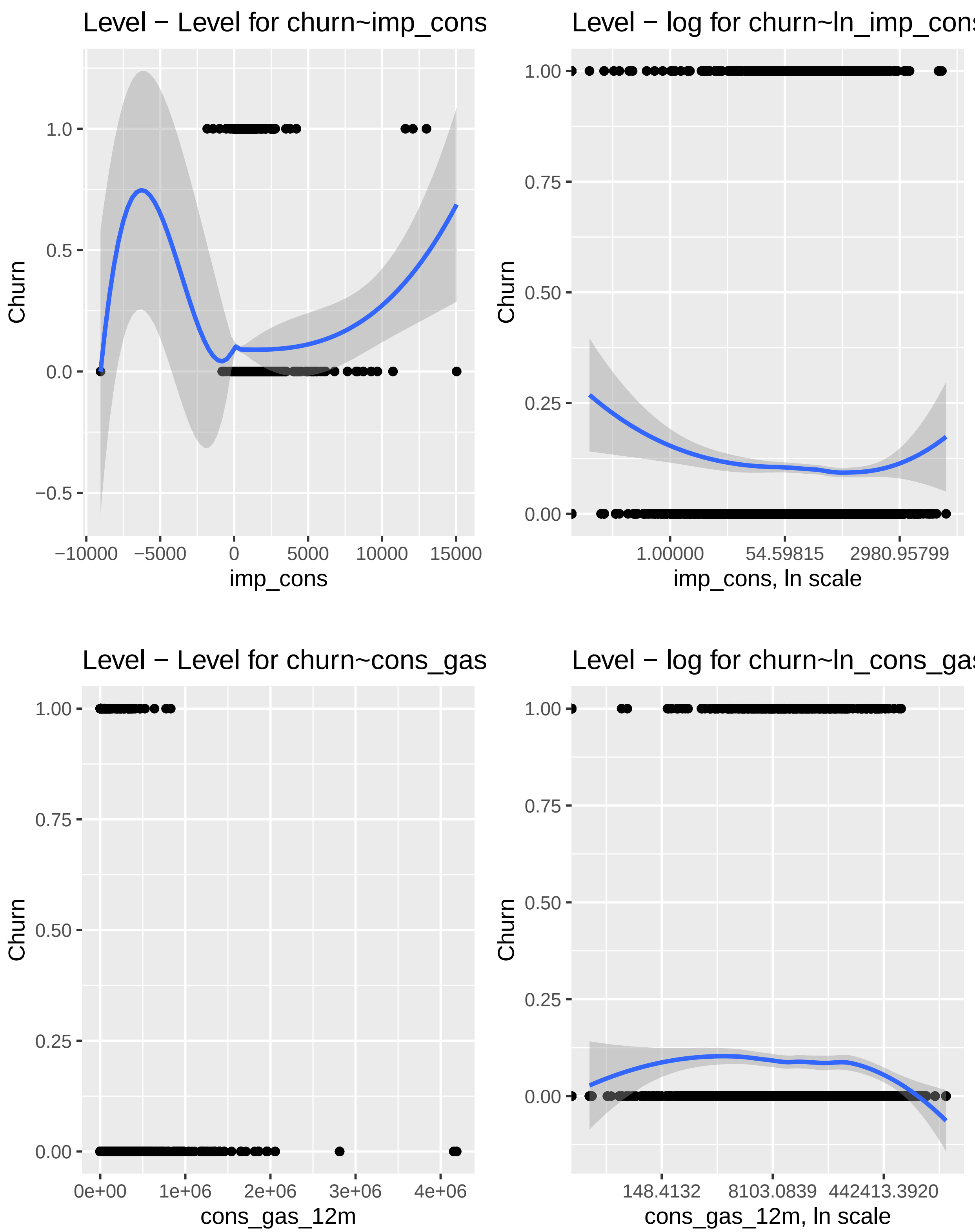


Figure: 7 - Loess/Scatterplot of Variables




Level − Level for churn~imp_cons
Churn
1.0
0.5
0.0
−0.5
−10000
−5000
0
5000
10000
15000
imp_cons
Level − log for churn~ln_imp_cons
1.00
0.75
0.50
0.25
0.00
1.00000
54.59815
2980.95799
imp_cons, ln scale
Level − Level for churn~cons_gas
0.00
0e+00
1e+06
2e+06
3e+06
4e+06
cons_gas_12m
Level − log for churn~ln_cons_gas
148.4132
8103.0839
442413.3920
cons_gas_12m, ln scale

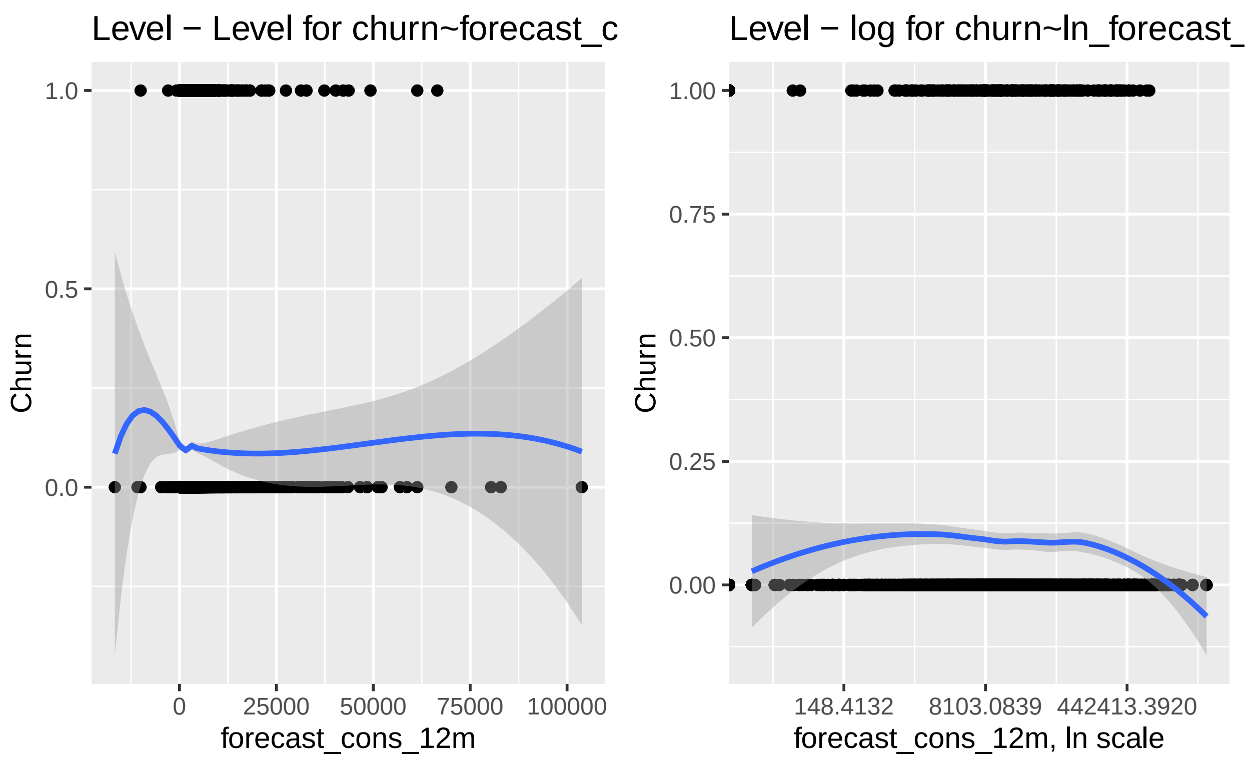


Figure: 8 - Transformed Variables Exploration

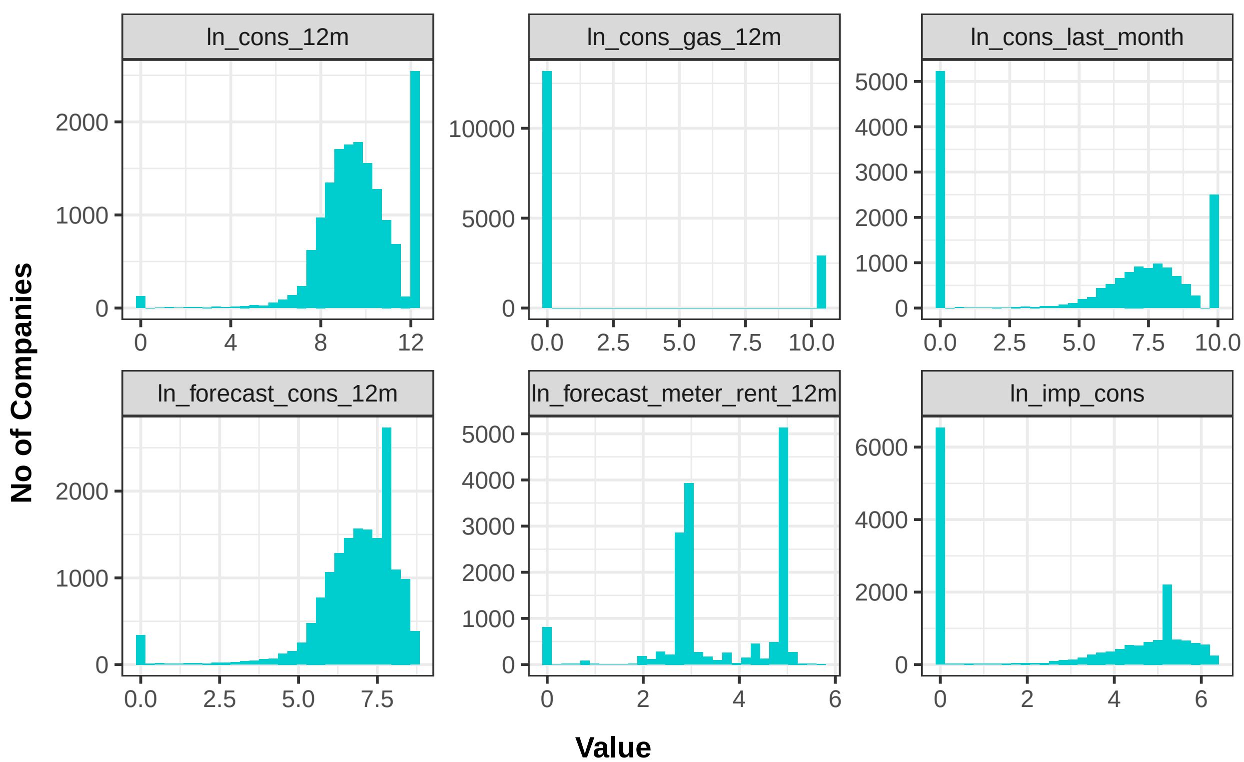
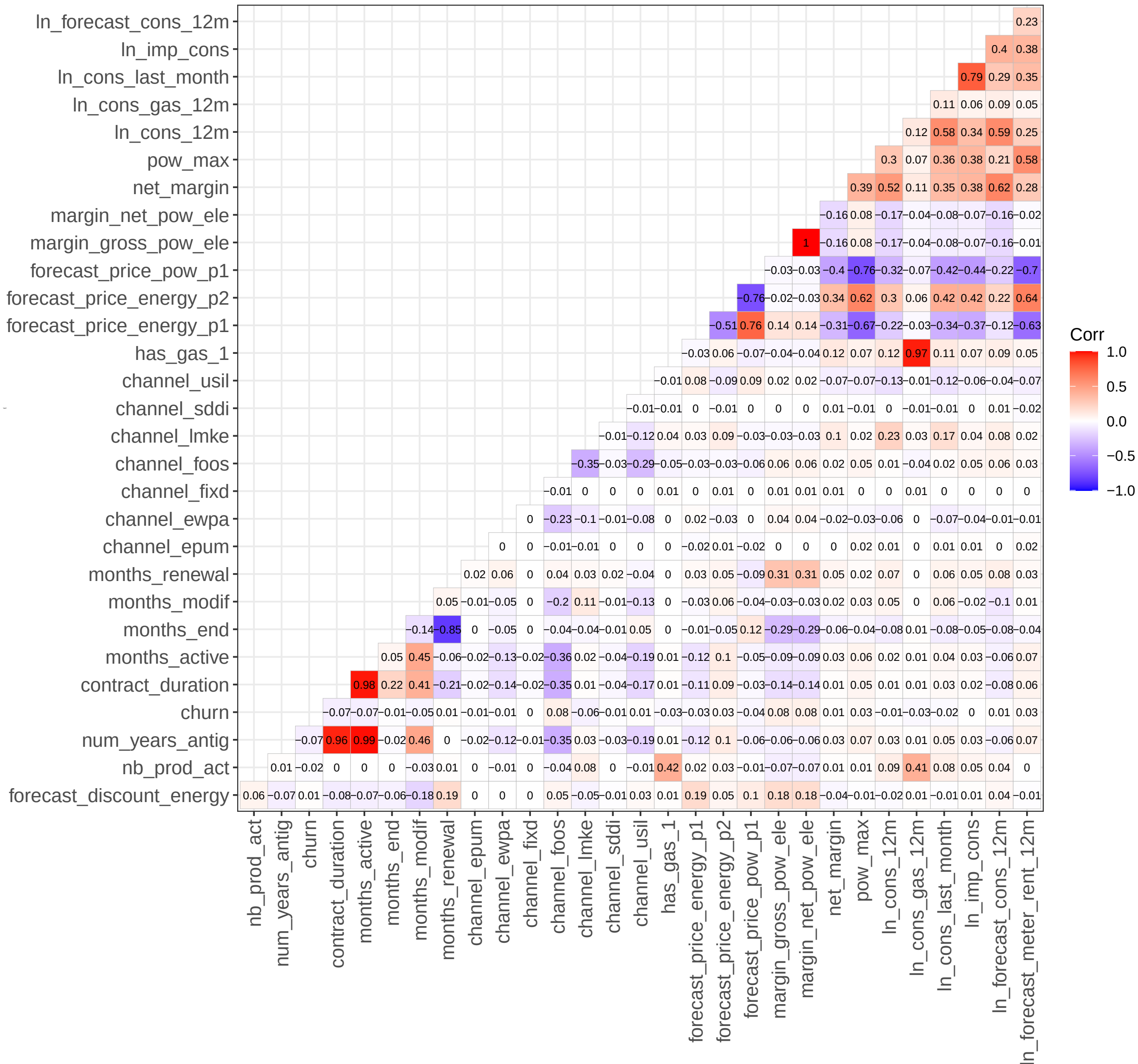


Figure 9 - Correlation Matrix


ln_forecast_cons_12m
ln_imp_cons
ln_cons_last_month
ln_cons_gas_12m
ln_cons_12m
pow_max
net_margin
margin_net_pow_ele
margin_gross_pow_ele
forecast_price_pow_p1
forecast_price_energy_p2
forecast_price_energy_p1
has_gas_1
channel_usil
channel_sddi
channel_lmke
channel_foos
channel_fixd
channel_ewpa
channel_epum
months_renewal
months_modif
months_end
months_active
contract_duration
churn
num_years_antig
nb_prod_act
forecast_discount_energy
ln_forecast_meter_rent_12m
Corr
1.0
0.5
0.0
−0.5
−1.0