Probability of Recession

William Chiu

2022-12-10

Summary

Forecast the probability of a recession in the next 126 trading days using the following predictors:

1. Spread between 10Y CMT and Effective Federal Funds Rate
2. Lags of the spread
3. Adstock transformations of the spread
4. Moving averages of the spread

There are between 250 and 253 trading days in a year.

Extract Historical Data

Refer to this vignette for FRED data access.

```r
library(tidyverse)
library(lubridate)
library(fredr)
library(car)
library(MLmetrics)
library(caret)
library(pdp)
library(gridExtra)
library(mboost)
library(gbm)
library(randomForest)
library(glmnet)
library(gtsummary)

randSeed <- 1983

startTestDate <- "1978-01-01"
startTrainDate <- "1988-01-01"
```

```r
# series_id <- c("FEDFUNDS", "GS10", "USREC", "UNRATE", "CPIAUCSL")

series_id <- c("DFF", "DGS10")  # daily

response_id <- "USREC" # monthly
```

```r
full_data <- map_dfr(series_id, function(x) {
  fredr(
    series_id = x,
    observation_start = as.Date("1950-01-01"),
    observation_end = as.Date("2023-12-01")
  )
})

recession_dates <- map_dfr(response_id, function(x) {
  fredr(
    series_id = x,
    observation_start = as.Date("1950-01-01"),
    observation_end = as.Date("2023-12-01")
  )
})
```

Pivot Wider

```r
full_data_wide_raw <- full_data %>%
  arrange(date) %>%
  select(date, series_id, value) %>%
  pivot_wider(id_cols=date, names_from = series_id,
              values_from = value)%>%
  drop_na()
```

Calculate Features/Predictors

```r
full_data_wide_features <- full_data_wide_raw %>%
  arrange(date) %>%
  mutate(SPRD_10YCMT_FEDFUNDS = DGS10 - DFF,
         D_SPRD = SPRD_10YCMT_FEDFUNDS -
           lag(SPRD_10YCMT_FEDFUNDS, 21)
         ) %>%
  mutate(across(
    .cols=c(SPRD_10YCMT_FEDFUNDS),
    .fns=list(lag1m = ~lag(.x, 1*21),
         lag3m = ~lag(.x, 3*21),
         lag6m = ~lag(.x, 6*21),
         lag9m = ~lag(.x, 9*21),
         lag12m = ~lag(.x, 12*21),
         lag5d = ~lag(.x, 5),
         lag10d = ~lag(.x, 10),
         lag15d = ~lag(.x, 15)
         )
  )) %>%
  drop_na()
```

Calculate Adstock

The adstock transformation is an auto-regressive transformation of a time series. The transformation takes into account past values of the time series. The intuition is that past values of the time series has a contemporaneous effect on the outcome.

$$AdStock(x_t) = x_t + \theta AdStock(x_{t-1})$$

where

$$0 < \theta < 1$$

.

The parameters cannot be estimated easily with least squares or logistic regression. Instead, we assume a range of potential values.

```
full_data_wide_features_adstock <- full_data_wide_features %>%
  arrange(date) %>%
    mutate(across(
    .cols=c(SPRD_10YCMT_FEDFUNDS, D_SPRD),
    .fns=list(adstk001 = ~stats::filter(.x,
                                  filter=0.001,
                                  method="recursive") ,
         adstk10 = ~stats::filter(.x,
                                  filter=0.10,
                                  method="recursive") ,
         adstk20 = ~stats::filter(.x,
                                  filter=0.20,
                                  method="recursive"),
         adstk40 = ~stats::filter(.x,
                                  filter=0.40,
                                  method="recursive"),
         adstk75 = ~stats::filter(.x,
                                  filter=0.75,
                                  method="recursive"),
         adstk95 = ~stats::filter(.x,
                                  filter=0.95,
                                  method="recursive")
  ))) %>%
  mutate(constant=1)
```

Calculate Moving Average

```
ma_fun <- function(k_param){
  rep(1/k_param, k_param)
}

full_data_wide_features_adstock <- full_data_wide_features_adstock %>%
  arrange(date) %>%
    mutate(across(
    .cols=c(SPRD_10YCMT_FEDFUNDS, D_SPRD),
    .fns=list(ma5d = ~stats::filter(.x,
```

```
                                    filter=ma_fun(5),
                                   method="convolution",
                                   sides=1) ,
         ma10d = ~stats::filter(.x,
                                   filter=ma_fun(10),
                                   method="convolution",
                                   sides=1) ,
         ma15d = ~stats::filter(.x,
                                    filter=ma_fun(15),
                                   method="convolution",
                                   sides=1),
         ma20d = ~stats::filter(.x,
                                     filter=ma_fun(20),
                                   method="convolution",
                                   sides=1),
         ma25d = ~stats::filter(.x,
                                    filter=ma_fun(25),
                                   method="convolution",
                                   sides=1),
         ma2m = ~stats::filter(.x,
                                     filter=ma_fun(2*21),
                                   method="convolution",
                                   sides=1),
         ma3m = ~stats::filter(.x,
                                     filter=ma_fun(3*21),
                                   method="convolution",
                                   sides=1),
         ma6m = ~stats::filter(.x,
                                     filter=ma_fun(6*21),
                                   method="convolution",
                                   sides=1),
         ma9m = ~stats::filter(.x,
                                     filter=ma_fun(9*21),
                                   method="convolution",
                                   sides=1),
         ma12m = ~stats::filter(.x,
                                     filter=ma_fun(12*21),
                                   method="convolution",
                                   sides=1)
  )))
```

Recession in next 6 months

```
full_data_wide <- full_data_wide_features_adstock %>%
  arrange(date) %>%
  mutate(date_month = month(date),
         date_year = year(date))

recession_df <- recession_dates %>%
  select(date, value) %>%
  arrange(date) %>%
```

```r
  mutate(date_month = month(date),
         date_year = year(date))


full_data_wide <- full_data_wide %>%
  left_join(recession_df,
            by = c("date_month" = "date_month",
                   "date_year" = "date_year")) %>%
  mutate(USREC = value)


df_FUTREC = as.data.frame(
  data.table::shift(
    full_data_wide$USREC,
    n = 1:(6 * 21),
    type = "lead",
    give.names = TRUE,
    fill = NA
  )
) %>%
  rowwise() %>%
  mutate(FUTREC = max(c_across(V1_lead_1:V1_lead_126)))

full_data_wide$FUTREC <- df_FUTREC$FUTREC

full_data_wide <- full_data_wide %>%
  select(date=date.x, everything(), -date_month,
         -date_year, -date.y,
         -value)

full_data_wide$constant <- 1

full_data_wide_noUSREC <- full_data_wide %>%
  select(-USREC)
```

Remove the last 12 months of historical data

Since the NBER often dates recessions after they have already occurred (and sometimes ended), remove the last 12 months of historical data from both the training and test data sets.

```r
recent_data <- tail(full_data_wide_noUSREC, 12*21)

train_test <- head(full_data_wide_noUSREC, -12*21) %>%
  drop_na()
```

Split Train/Test

```r
train_data <- train_test %>%
  filter(date >= startTrainDate)
```

```r
test_data <- train_test %>%
  filter(date >= startTestDate) %>%
  filter(date < startTrainDate)

train_yes_no <- train_data %>%
  mutate(FUTREC = case_when(FUTREC == 1 ~ "yes",
                            TRUE ~ "no"))

train_yes_no$FUTREC <- factor(train_yes_no$FUTREC,
                              levels=c("yes","no"))



tbl_summary(train_data)
```

Characteristic	**N = 8,489**
date	1988-01-04 to 2021-12-06
DFF	2.44 (0.20, 5.28)
DGS10	4.41 (2.58, 6.18)
SPRD_10YCMT_FEDFUNDS	1.48 (0.50, 2.51)
D_SPRD	-0.02 (-0.21, 0.18)
SPRD_10YCMT_FEDFUNDS_lag1m	1.48 (0.50, 2.51)
SPRD_10YCMT_FEDFUNDS_lag3m	1.49 (0.50, 2.52)
SPRD_10YCMT_FEDFUNDS_lag6m	1.51 (0.50, 2.52)
SPRD_10YCMT_FEDFUNDS_lag9m	1.51 (0.50, 2.52)
SPRD_10YCMT_FEDFUNDS_lag12m	1.51 (0.50, 2.52)
SPRD_10YCMT_FEDFUNDS_lag5d	1.48 (0.50, 2.51)
SPRD_10YCMT_FEDFUNDS_lag10d	1.48 (0.50, 2.51)
SPRD_10YCMT_FEDFUNDS_lag15d	1.48 (0.50, 2.51)
SPRD_10YCMT_FEDFUNDS_adstk001	1.48 (0.50, 2.51)
SPRD_10YCMT_FEDFUNDS_adstk10	1.64 (0.56, 2.79)
SPRD_10YCMT_FEDFUNDS_adstk20	1.84 (0.63, 3.14)
SPRD_10YCMT_FEDFUNDS_adstk40	2.46 (0.84, 4.19)
SPRD_10YCMT_FEDFUNDS_adstk75	5.9 (2.1, 10.1)
SPRD_10YCMT_FEDFUNDS_adstk95	29 (10, 51)
D_SPRD_adstk001	-0.02 (-0.21, 0.18)
D_SPRD_adstk10	-0.03 (-0.23, 0.20)
D_SPRD_adstk20	-0.03 (-0.26, 0.23)
D_SPRD_adstk40	-0.04 (-0.34, 0.29)
D_SPRD_adstk75	-0.08 (-0.76, 0.63)
D_SPRD_adstk95	-0.4 (-2.6, 2.2)
constant	8,489 (100%)
SPRD_10YCMT_FEDFUNDS_ma5d	1.47 (0.51, 2.52)
SPRD_10YCMT_FEDFUNDS_ma10d	1.47 (0.52, 2.51)
SPRD_10YCMT_FEDFUNDS_ma15d	1.46 (0.53, 2.51)
SPRD_10YCMT_FEDFUNDS_ma20d	1.47 (0.53, 2.52)
SPRD_10YCMT_FEDFUNDS_ma25d	1.46 (0.53, 2.52)
SPRD_10YCMT_FEDFUNDS_ma2m	1.46 (0.53, 2.53)
SPRD_10YCMT_FEDFUNDS_ma3m	1.47 (0.49, 2.53)
SPRD_10YCMT_FEDFUNDS_ma6m	1.46 (0.49, 2.57)
SPRD_10YCMT_FEDFUNDS_ma9m	1.46 (0.49, 2.59)
SPRD_10YCMT_FEDFUNDS_ma12m	1.47 (0.51, 2.58)

Characteristic	N = 8,489
D_SPRD_ma5d	-0.02 (-0.20, 0.17)
D_SPRD_ma10d	-0.02 (-0.19, 0.16)
D_SPRD_ma15d	-0.02 (-0.18, 0.15)
D_SPRD_ma20d	-0.02 (-0.17, 0.14)
D_SPRD_ma25d	-0.02 (-0.16, 0.13)
D_SPRD_ma2m	-0.03 (-0.13, 0.11)
D_SPRD_ma3m	-0.02 (-0.12, 0.10)
D_SPRD_ma6m	-0.01 (-0.10, 0.08)
D_SPRD_ma9m	-0.01 (-0.09, 0.08)
D_SPRD_ma12m	-0.01 (-0.07, 0.08)
FUTREC	1,248 (15%)

Remove stale data from test set

Exclude historical data prior to 1978-01-01 because the economy changed dramatically (due to computational innovation).

```
summary(test_data$date)
```

```
##          Min.      1st Qu.       Median         Mean      3rd Qu.          Max.
## "1978-01-03" "1980-07-02" "1983-01-04" "1983-01-01" "1985-07-02" "1987-12-31"
```

```
test_data <- test_data %>%
  filter(date >= startTestDate)

summary(test_data$date)
```

```
##          Min.      1st Qu.       Median         Mean      3rd Qu.          Max.
## "1978-01-03" "1980-07-02" "1983-01-04" "1983-01-01" "1985-07-02" "1987-12-31"
```

Setup Parallel Processing

```
library(doParallel)

cl <- makePSOCKcluster(3)
registerDoParallel(cl)
```

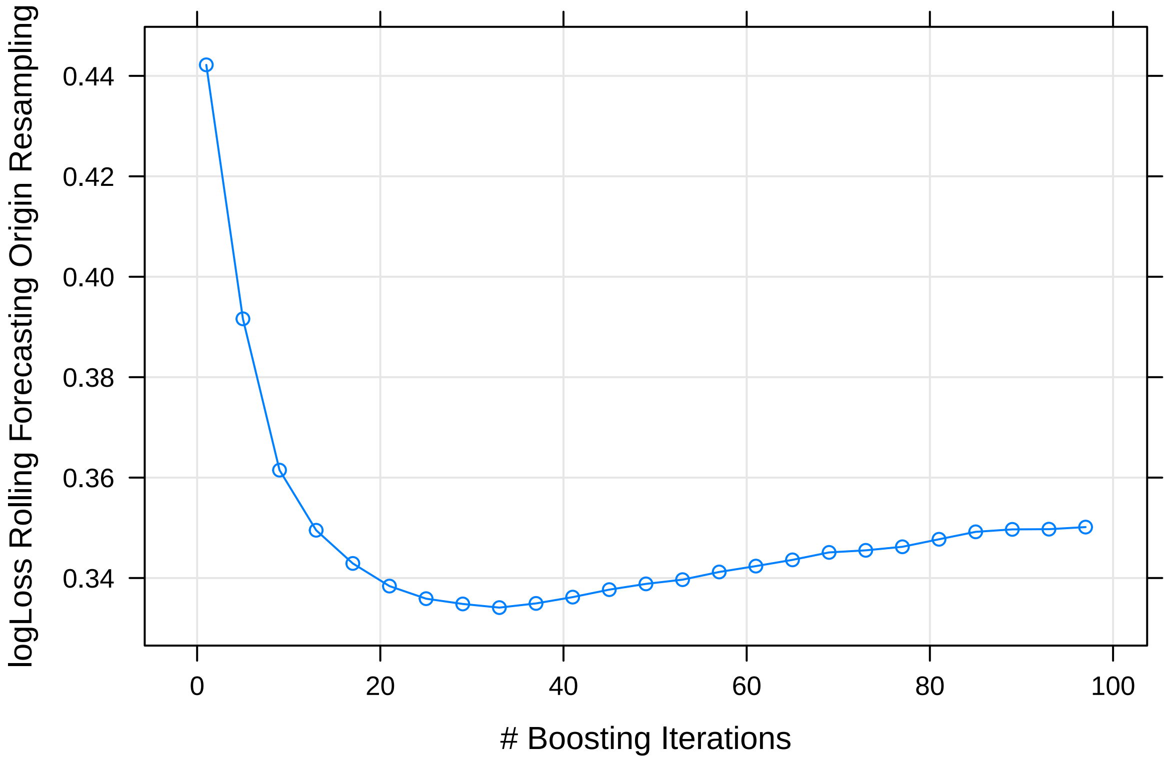
Cross-Validation Framework

```
fcstHorizon <- 6*21
initWindow <- 120*21
param_skip <- fcstHorizon - 1

if(initWindow < 100){
  stop("Too few observations.")
```

```r
}

fitControl_oneSE <- trainControl(method = "timeslice",
                           initialWindow=initWindow,
                           horizon=fcstHorizon,
                           fixedWindow=FALSE,
                           skip=param_skip,
                           ## Estimate class probabilities
                           classProbs = TRUE,
                           ## Evaluate performance using
                           ## the following function
                           summaryFunction = mnLogLoss,
                           selectionFunction="oneSE")

fitControl_best <- trainControl(method = "timeslice",
                           initialWindow=initWindow,
                           horizon=fcstHorizon,
                           fixedWindow=FALSE,
                           skip=param_skip,
                           ## Estimate class probabilities
                           classProbs = TRUE,
                           ## Evaluate performance using
                           ## the following function
                           summaryFunction = mnLogLoss,
                           selectionFunction="best")
```

Gradient Boosting for Additive Models

```r
grid_gam <- expand.grid(mstop=seq(1,100,4),
                        prune="no")

set.seed(randSeed)

gam_mod <- train(
  FUTREC ~ . - date - constant,
  data = train_yes_no,
  method = "gamboost",
  trControl = fitControl_oneSE,
  metric = "logLoss",
  tuneGrid = grid_gam,
  family = Binomial(),
  dfbase =3

)

plot(gam_mod)
```



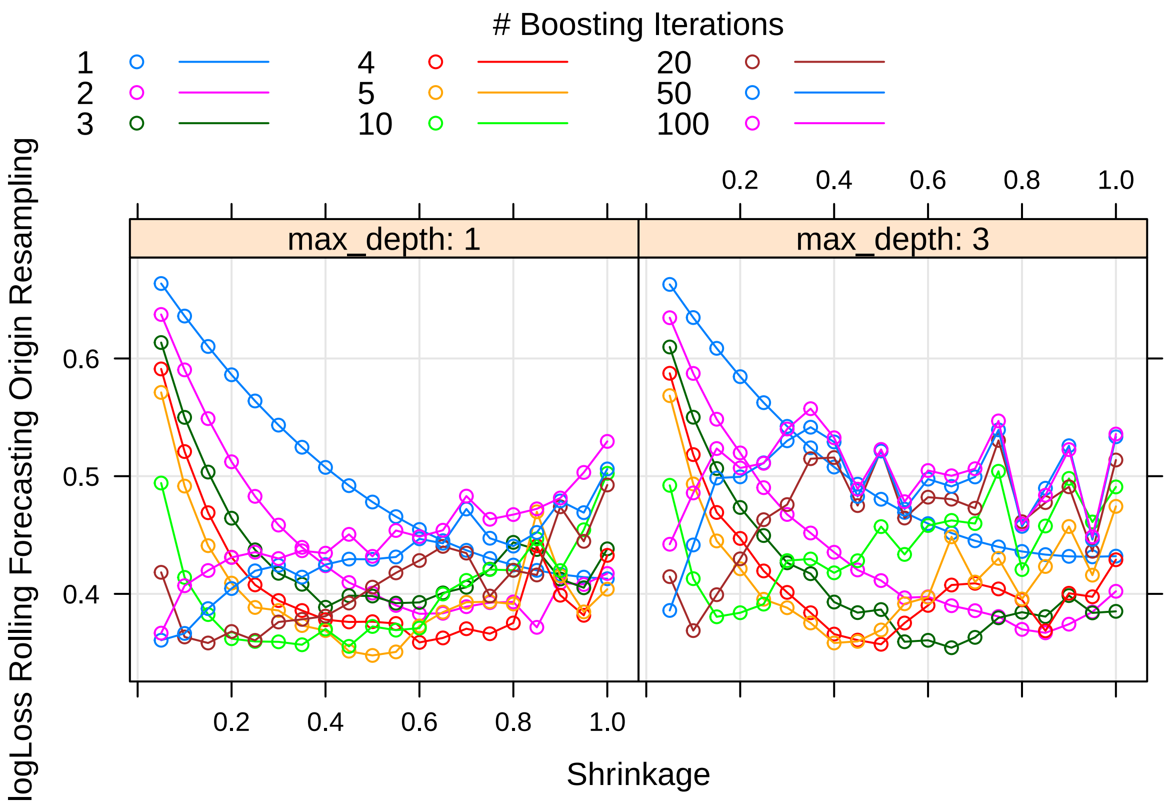

```
gam_mod$bestTune
```

```
##   mstop prune
## 2     5    no
```

eXtreme Gradient Boosting Trees

```
grid_xgb <- expand.grid(nrounds=c(1,2,3,4,5,10,20,
                                  50,100),
                        max_depth=c(1,3),
                        eta=seq(0.05,1,0.05),
                        gamma=0,
                        colsample_bytree=1,
                        min_child_weight=10,
                        subsample=1
                        )

set.seed(randSeed)

xgb_mod <- train(
  FUTREC ~ . - date - constant,
  data = train_yes_no,
  method = "xgbTree",
```

```r
  trControl = fitControl_oneSE,
  metric = "logLoss",
  tuneGrid = grid_xgb,
  objective  = "binary:logistic"
)

plot(xgb_mod)
```



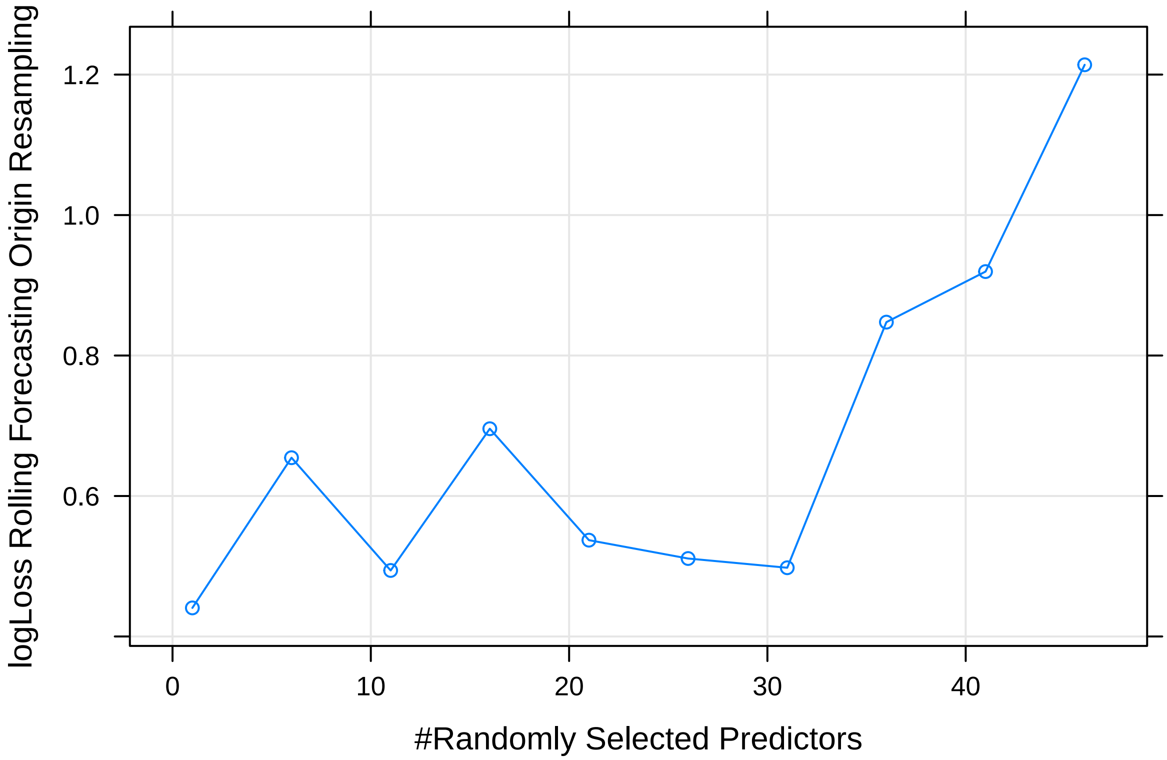

```r
xgb_mod$bestTune
```

```
##     nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
## 289       1         1 0.85     0                1               10         1
```

Random Forest

```r
grid_rf <- data.frame(mtry=seq.int(1,50,5))

set.seed(randSeed)

rf_mod <- train(
  FUTREC ~ . - date - constant,
  data = train_yes_no,
```

```
  method = "rf",
  trControl = fitControl_oneSE,
  metric = "logLoss",
  tuneGrid = grid_rf,
  importance = TRUE
)

plot(rf_mod)
```




```
rf_mod$bestTune
```

```
##   mtry
## 1    1
```

Stepwise Regression

The `glmStepAIC` method uses the `glm()` function from the `stats` package. The documentation for `glm()` says:

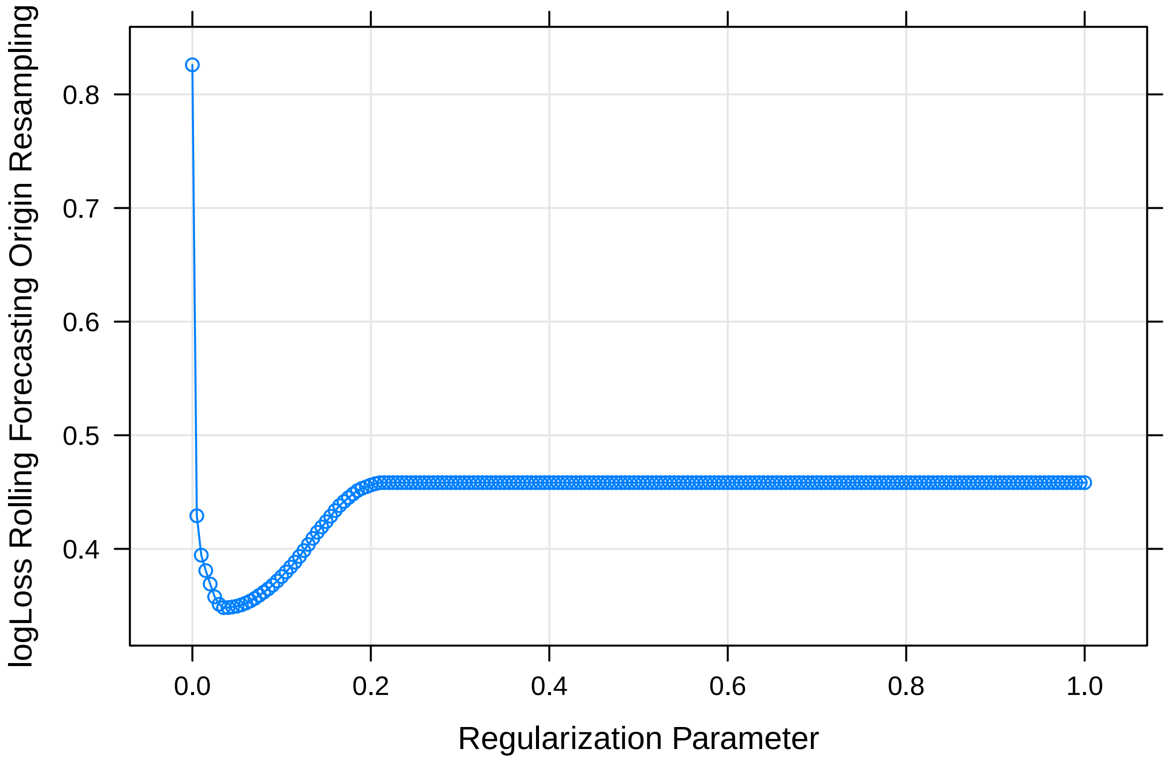
> For binomial and quasibinomial families the response can also be specified as a factor (when the first level denotes failure and all others success) or as a two-column matrix with the columns giving the numbers of successes and failures.

However, for most methods (that do not invoke `glm()`) in `train`, the first level denotes the success (the opposite of `glm()`). This behavior causes the coefficient signs to flip. Be highly suspicious when interpreting coefficients from models that are fit using `train`.

```
set.seed(randSeed)

stepwise_mod <- train(
  FUTREC ~ . - date - constant,
  data = train_yes_no,
  method = "glmStepAIC",
  trControl = fitControl_oneSE,
  metric = "logLoss",
  tuneLength = 10,
  family = binomial,
  trace = 0,
  k = 10*log(nrow(train_yes_no)),
  direction = "forward"
)
```

Elastic Net (Lasso)

```
grid_glmnet <- expand.grid(
  alpha = 1,
  lambda = seq(0, 1, 0.005)
)

set.seed(randSeed)

glmnet_mod <- train(
  FUTREC ~ . - date - constant,
  data = train_yes_no,
  method = "glmnet",
  trControl = fitControl_best,
  metric = "logLoss",
  tuneGrid = grid_glmnet,
  family = "binomial"
)

plot(glmnet_mod)
```

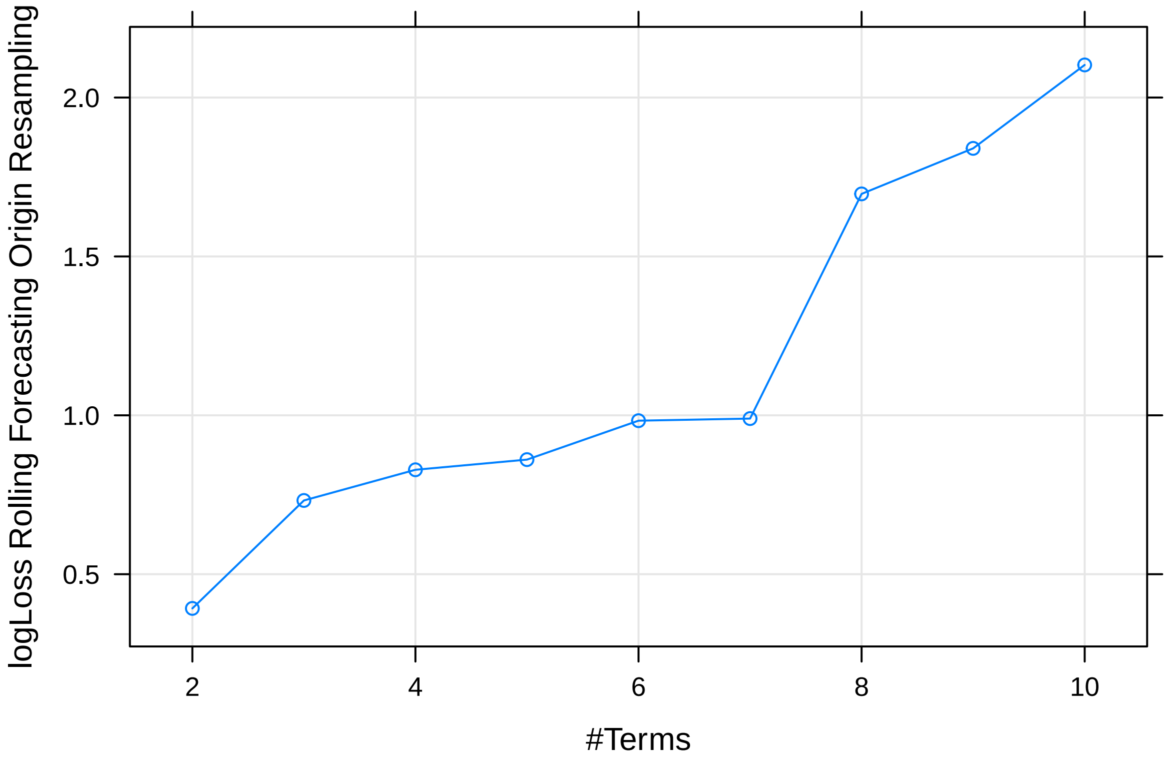



```
glmnet_mod$bestTune
```

```
##   alpha lambda
## 8     1  0.035
```

Multivariate Adaptive Regression Splines

```
grid_mars <- expand.grid(nprune=seq(2,10,1),
                         degree=1)

set.seed(randSeed)

earth_mod <- train(
  FUTREC ~ . - date - constant,
  data = train_yes_no,
  method = "earth",
  trControl = fitControl_oneSE,
  metric = "logLoss",
  tuneGrid = grid_mars,
  glm = list(family = binomial)
)

plot(earth_mod)
```




```r
earth_mod$bestTune
```

```
##   nprune degree
## 1      2      1
```

Null Model: Intercept-only Model

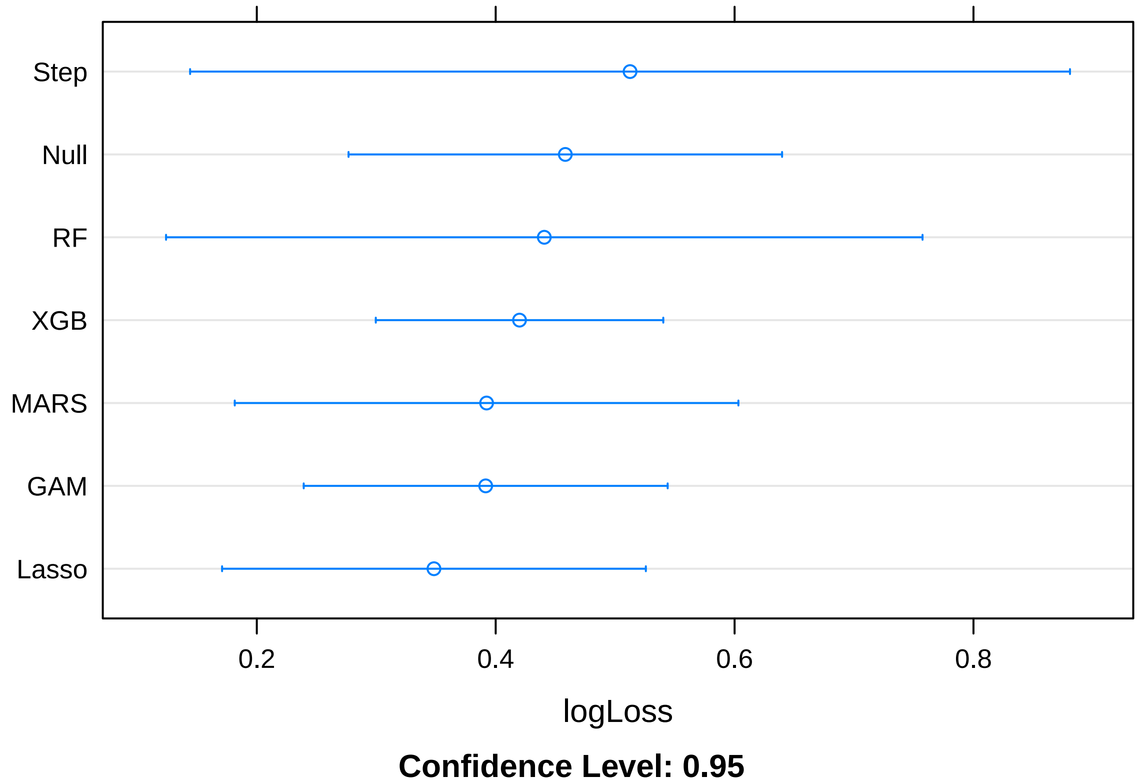
```r
set.seed(randSeed)

null_mod <- train(
  FUTREC ~ constant,
  data = train_yes_no,
  method = "glm",
  trControl = fitControl_oneSE,
  metric = "logLoss",
  family = binomial
)
```

Compare Models

```r
resamps <- resamples(list(XGB = xgb_mod,
                          GAM = gam_mod,
                          RF = rf_mod,
                          Step = stepwise_mod,
                          Lasso = glmnet_mod,
                          MARS = earth_mod,
                          Null = null_mod)
                     )
summary(resamps)
```

```
##
## Call:
## summary.resamples(object = resamps)
##
## Models: XGB, GAM, RF, Step, Lasso, MARS, Null
## Number of resamples: 47
##
## logLoss
##               Min.     1st Qu.     Median      Mean   3rd Qu.     Max. NA's
## XGB   1.846234e-01 0.200712912 0.20922462 0.4199587 0.4124256 1.722716    0
## GAM   9.649594e-02 0.126199466 0.14959695 0.3916179 0.2183111 1.762089    0
## RF    1.192593e-03 0.021711417 0.06454622 0.4406862 0.4236321 6.824860    0
## Step  1.454773e-06 0.007158461 0.03391915 0.5124784 0.2773649 6.880424    0
## Lasso 3.896341e-03 0.035041496 0.07488620 0.3483172 0.3323924 2.687742    0
## MARS  1.679652e-03 0.055271610 0.07262942 0.3923739 0.4797078 3.418330    0
## Null  1.004490e-01 0.154729719 0.17284281 0.4582848 0.2194922 2.167452    0
```

```r
dotplot(resamps, metric = "logLoss", conf.level=0.95)
```



Explore XGB Model

```
xgb_mod$bestTune
```

```
##     nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
## 289       1         1 0.85     0                1               10         1
```

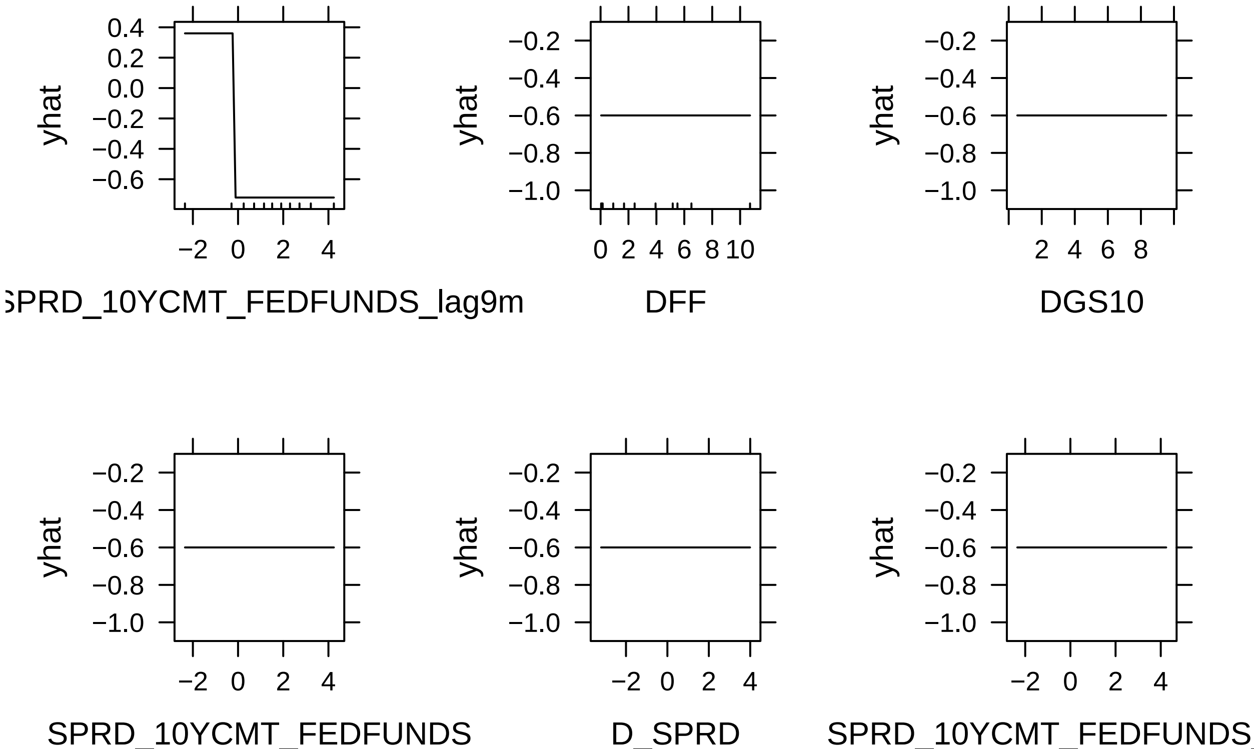
```
df_imp <- varImp(xgb_mod)$importance %>%
  arrange(desc(Overall))

df_imp$variable <- rownames(df_imp)

df_imp <- df_imp %>%
  select(variable, Overall)

row.names(df_imp) <- NULL

knitr::kable(df_imp)
```

variable	Overall
SPRD_10YCMT_FEDFUNDS_lag9m	100

variable	Overall
DFF	0
DGS10	0
SPRD_10YCMT_FEDFUNDS	0
D_SPRD	0
SPRD_10YCMT_FEDFUNDS_lag1m	0
SPRD_10YCMT_FEDFUNDS_lag3m	0
SPRD_10YCMT_FEDFUNDS_lag6m	0
SPRD_10YCMT_FEDFUNDS_lag12m	0
SPRD_10YCMT_FEDFUNDS_lag5d	0
SPRD_10YCMT_FEDFUNDS_lag10d	0
SPRD_10YCMT_FEDFUNDS_lag15d	0
SPRD_10YCMT_FEDFUNDS_adstk001	0
SPRD_10YCMT_FEDFUNDS_adstk10	0
SPRD_10YCMT_FEDFUNDS_adstk20	0
SPRD_10YCMT_FEDFUNDS_adstk40	0
SPRD_10YCMT_FEDFUNDS_adstk75	0
SPRD_10YCMT_FEDFUNDS_adstk95	0
D_SPRD_adstk001	0
D_SPRD_adstk10	0
D_SPRD_adstk20	0
D_SPRD_adstk40	0
D_SPRD_adstk75	0
D_SPRD_adstk95	0
SPRD_10YCMT_FEDFUNDS_ma5d	0
SPRD_10YCMT_FEDFUNDS_ma10d	0
SPRD_10YCMT_FEDFUNDS_ma15d	0
SPRD_10YCMT_FEDFUNDS_ma20d	0
SPRD_10YCMT_FEDFUNDS_ma25d	0
SPRD_10YCMT_FEDFUNDS_ma2m	0
SPRD_10YCMT_FEDFUNDS_ma3m	0
SPRD_10YCMT_FEDFUNDS_ma6m	0
SPRD_10YCMT_FEDFUNDS_ma9m	0
SPRD_10YCMT_FEDFUNDS_ma12m	0
D_SPRD_ma5d	0
D_SPRD_ma10d	0
D_SPRD_ma15d	0
D_SPRD_ma20d	0
D_SPRD_ma25d	0
D_SPRD_ma2m	0
D_SPRD_ma3m	0
D_SPRD_ma6m	0
D_SPRD_ma9m	0
D_SPRD_ma12m	0

```
pdp.top1 <- partial(xgb_mod,
          pred.var = df_imp$variable[1],
          plot = TRUE,
          rug = TRUE)

pdp.top2 <- partial(xgb_mod,
          pred.var = df_imp$variable[2],
```

```
          plot = TRUE,
          rug = TRUE)

pdp.top3 <- partial(xgb_mod,
    pred.var = df_imp$variable[3],
    plot = TRUE,
    chull = TRUE
  )

pdp.top4 <- partial(xgb_mod,
    pred.var = df_imp$variable[4],
    plot = TRUE,
    chull = TRUE
  )

pdp.top5 <- partial(xgb_mod,
    pred.var = df_imp$variable[5],
    plot = TRUE,
    chull = TRUE
  )

pdp.top6 <- partial(xgb_mod,
    pred.var = df_imp$variable[6],
    plot = TRUE,
    chull = TRUE
  )

grid.arrange(pdp.top1, pdp.top2, pdp.top3,
             pdp.top4, pdp.top5, pdp.top6, ncol = 3)
```



Peeking

Peeking means we use the insights from the automated models to choose variables in subsequent models. This is technically cheating and causes the cross-validation errors to be artificially low. This is addressed in the test set which does not have peeking bias.

```
top_predictors <- head(df_imp$variable)

best_predictor <- head(top_predictors, 1)

top_fmla <- as.formula(paste0("FUTREC ~",
                              paste0(top_predictors,
                                     collapse=" + ")))

top1_fmla <- as.formula(paste0("FUTREC ~",
                               paste0(best_predictor,
                                      collapse=" + ")))
```

Logistic Regression (with peeking)

As mentioned early, `train` and `glm` treat the reference level differently for binary outcomes. Hence, the coefficients are flipped when training a logistic regression inside `train`.

```r
logit_mod <- train(
  top1_fmla,
  data = train_yes_no,
  method = "glm",
  trControl = fitControl_oneSE,
  metric = "logLoss",
  family=binomial
)

summary(logit_mod)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1826   0.0671   0.1789   0.4300   2.7138
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   0.47223    0.04009   11.78   <2e-16 ***
## SPRD_10YCMT_FEDFUNDS_lag9m  1.75702    0.04861   36.14   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7088.2  on 8488  degrees of freedom
## Residual deviance: 4489.6  on 8487  degrees of freedom
## AIC: 4493.6
##
## Number of Fisher Scoring iterations: 6
```

Compare Models

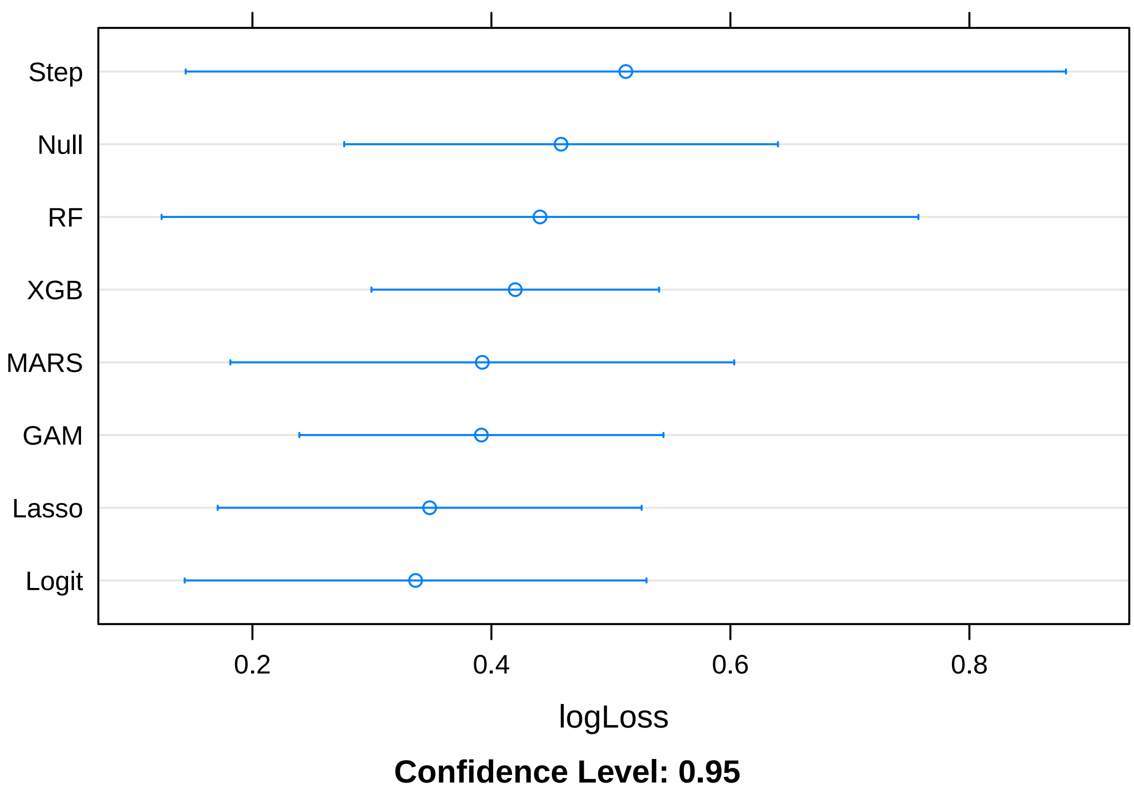
CV errors for models with peeking are misleadingly low. This will be addressed with a test set.

```r
mymods <- list(XGB = xgb_mod,
                          GAM = gam_mod,
                          RF = rf_mod,
                          Step = stepwise_mod,
                          Lasso = glmnet_mod,
                          MARS = earth_mod,
                          Null = null_mod,
                          Logit = logit_mod)  ## peeking

resamps <- resamples(mymods)
summary(resamps)
```

```
##
## Call:
```

```
## summary.resamples(object = resamps)
##
## Models: XGB, GAM, RF, Step, Lasso, MARS, Null, Logit
## Number of resamples: 47
##
## logLoss
##               Min.     1st Qu.     Median      Mean   3rd Qu.     Max. NA's
## XGB   1.846234e-01 0.200712912 0.20922462 0.4199587 0.4124256 1.722716    0
## GAM   9.649594e-02 0.126199466 0.14959695 0.3916179 0.2183111 1.762089    0
## RF    1.192593e-03 0.021711417 0.06454622 0.4406862 0.4236321 6.824860    0
## Step  1.454773e-06 0.007158461 0.03391915 0.5124784 0.2773649 6.880424    0
## Lasso 3.896341e-03 0.035041496 0.07488620 0.3483172 0.3323924 2.687742    0
## MARS  1.679652e-03 0.055271610 0.07262942 0.3923739 0.4797078 3.418330    0
## Null  1.004490e-01 0.154729719 0.17284281 0.4582848 0.2194922 2.167452    0
## Logit 4.929317e-04 0.012132627 0.05237117 0.3365571 0.3580587 2.966334    0
```

```
dotplot(resamps, metric = "logLoss", conf.level=0.95)
```



Test Set Performance

```
perf <-
  function(lst_mods,
           f_metric = caTools::colAUC,
```

```r
           metricname = "ROC-AUC",
           dat=test_data,
           response="FUTREC") {
    lst_preds <- map(
      .x = lst_mods,
      .f = function(x) {
        if (class(x)[1] != "train") {
          predict(x, newdata = dat, type = "response")
        } else
          (
            predict(x, newdata = dat, type = "prob")[, "yes"]

          )
      }
    )

    map_dfr(lst_preds, function(x) {
      f_metric(x, dat[,response, drop=TRUE])
    }) %>%
      pivot_longer(everything(), names_to = "model", values_to = metricname)
  }

perf(mymods, caTools::colAUC, "ROC-AUC") %>%
  arrange(desc(`ROC-AUC`)) %>%
      knitr::kable()
```

model	ROC-AUC
MARS	0.9468407
RF	0.9230453
Lasso	0.9123974
Step	0.8927061
Logit	0.8677468
GAM	0.8677424
XGB	0.8227510
Null	0.5000000

```r
perf(mymods, MLmetrics::LogLoss, "LogLoss") %>%
  arrange(LogLoss) %>%
      knitr::kable()
```

model	LogLoss
MARS	0.2857263
RF	0.3538911
Lasso	0.3627565
XGB	0.4242359
Step	0.4353394
GAM	0.4604357
Logit	0.5947867
Null	0.6566205

Probability of Recession (Most Recent 12 months)

```r
curr_data <- recent_data

curr_data$date
```

```
##   [1] "2021-12-07" "2021-12-08" "2021-12-09" "2021-12-10" "2021-12-13"
##   [6] "2021-12-14" "2021-12-15" "2021-12-16" "2021-12-17" "2021-12-20"
##  [11] "2021-12-21" "2021-12-22" "2021-12-23" "2021-12-27" "2021-12-28"
##  [16] "2021-12-29" "2021-12-30" "2021-12-31" "2022-01-03" "2022-01-04"
##  [21] "2022-01-05" "2022-01-06" "2022-01-07" "2022-01-10" "2022-01-11"
##  [26] "2022-01-12" "2022-01-13" "2022-01-14" "2022-01-18" "2022-01-19"
##  [31] "2022-01-20" "2022-01-21" "2022-01-24" "2022-01-25" "2022-01-26"
##  [36] "2022-01-27" "2022-01-28" "2022-01-31" "2022-02-01" "2022-02-02"
##  [41] "2022-02-03" "2022-02-04" "2022-02-07" "2022-02-08" "2022-02-09"
##  [46] "2022-02-10" "2022-02-11" "2022-02-14" "2022-02-15" "2022-02-16"
##  [51] "2022-02-17" "2022-02-18" "2022-02-22" "2022-02-23" "2022-02-24"
##  [56] "2022-02-25" "2022-02-28" "2022-03-01" "2022-03-02" "2022-03-03"
##  [61] "2022-03-04" "2022-03-07" "2022-03-08" "2022-03-09" "2022-03-10"
##  [66] "2022-03-11" "2022-03-14" "2022-03-15" "2022-03-16" "2022-03-17"
##  [71] "2022-03-18" "2022-03-21" "2022-03-22" "2022-03-23" "2022-03-24"
##  [76] "2022-03-25" "2022-03-28" "2022-03-29" "2022-03-30" "2022-03-31"
##  [81] "2022-04-01" "2022-04-04" "2022-04-05" "2022-04-06" "2022-04-07"
##  [86] "2022-04-08" "2022-04-11" "2022-04-12" "2022-04-13" "2022-04-14"
##  [91] "2022-04-18" "2022-04-19" "2022-04-20" "2022-04-21" "2022-04-22"
##  [96] "2022-04-25" "2022-04-26" "2022-04-27" "2022-04-28" "2022-04-29"
## [101] "2022-05-02" "2022-05-03" "2022-05-04" "2022-05-05" "2022-05-06"
## [106] "2022-05-09" "2022-05-10" "2022-05-11" "2022-05-12" "2022-05-13"
## [111] "2022-05-16" "2022-05-17" "2022-05-18" "2022-05-19" "2022-05-20"
## [116] "2022-05-23" "2022-05-24" "2022-05-25" "2022-05-26" "2022-05-27"
## [121] "2022-05-31" "2022-06-01" "2022-06-02" "2022-06-03" "2022-06-06"
## [126] "2022-06-07" "2022-06-08" "2022-06-09" "2022-06-10" "2022-06-13"
## [131] "2022-06-14" "2022-06-15" "2022-06-16" "2022-06-17" "2022-06-21"
## [136] "2022-06-22" "2022-06-23" "2022-06-24" "2022-06-27" "2022-06-28"
## [141] "2022-06-29" "2022-06-30" "2022-07-01" "2022-07-05" "2022-07-06"
## [146] "2022-07-07" "2022-07-08" "2022-07-11" "2022-07-12" "2022-07-13"
## [151] "2022-07-14" "2022-07-15" "2022-07-18" "2022-07-19" "2022-07-20"
## [156] "2022-07-21" "2022-07-22" "2022-07-25" "2022-07-26" "2022-07-27"
## [161] "2022-07-28" "2022-07-29" "2022-08-01" "2022-08-02" "2022-08-03"
## [166] "2022-08-04" "2022-08-05" "2022-08-08" "2022-08-09" "2022-08-10"
## [171] "2022-08-11" "2022-08-12" "2022-08-15" "2022-08-16" "2022-08-17"
## [176] "2022-08-18" "2022-08-19" "2022-08-22" "2022-08-23" "2022-08-24"
## [181] "2022-08-25" "2022-08-26" "2022-08-29" "2022-08-30" "2022-08-31"
## [186] "2022-09-01" "2022-09-02" "2022-09-06" "2022-09-07" "2022-09-08"
## [191] "2022-09-09" "2022-09-12" "2022-09-13" "2022-09-14" "2022-09-15"
## [196] "2022-09-16" "2022-09-19" "2022-09-20" "2022-09-21" "2022-09-22"
## [201] "2022-09-23" "2022-09-26" "2022-09-27" "2022-09-28" "2022-09-29"
## [206] "2022-09-30" "2022-10-03" "2022-10-04" "2022-10-05" "2022-10-06"
## [211] "2022-10-07" "2022-10-11" "2022-10-12" "2022-10-13" "2022-10-14"
## [216] "2022-10-17" "2022-10-18" "2022-10-19" "2022-10-20" "2022-10-21"
## [221] "2022-10-24" "2022-10-25" "2022-10-26" "2022-10-27" "2022-10-28"
## [226] "2022-10-31" "2022-11-01" "2022-11-02" "2022-11-03" "2022-11-04"
## [231] "2022-11-07" "2022-11-08" "2022-11-09" "2022-11-10" "2022-11-14"
```

```
## [236] "2022-11-15" "2022-11-16" "2022-11-17" "2022-11-18" "2022-11-21"
## [241] "2022-11-22" "2022-11-23" "2022-11-25" "2022-11-28" "2022-11-29"
## [246] "2022-11-30" "2022-12-01" "2022-12-02" "2022-12-05" "2022-12-06"
## [251] "2022-12-07" "2022-12-08"
```

Probability of Recession (the 12 most recent months)

```
curr_data <- recent_data

curr_data$date
```

```
##   [1] "2021-12-07" "2021-12-08" "2021-12-09" "2021-12-10" "2021-12-13"
##   [6] "2021-12-14" "2021-12-15" "2021-12-16" "2021-12-17" "2021-12-20"
##  [11] "2021-12-21" "2021-12-22" "2021-12-23" "2021-12-27" "2021-12-28"
##  [16] "2021-12-29" "2021-12-30" "2021-12-31" "2022-01-03" "2022-01-04"
##  [21] "2022-01-05" "2022-01-06" "2022-01-07" "2022-01-10" "2022-01-11"
##  [26] "2022-01-12" "2022-01-13" "2022-01-14" "2022-01-18" "2022-01-19"
##  [31] "2022-01-20" "2022-01-21" "2022-01-24" "2022-01-25" "2022-01-26"
##  [36] "2022-01-27" "2022-01-28" "2022-01-31" "2022-02-01" "2022-02-02"
##  [41] "2022-02-03" "2022-02-04" "2022-02-07" "2022-02-08" "2022-02-09"
##  [46] "2022-02-10" "2022-02-11" "2022-02-14" "2022-02-15" "2022-02-16"
##  [51] "2022-02-17" "2022-02-18" "2022-02-22" "2022-02-23" "2022-02-24"
##  [56] "2022-02-25" "2022-02-28" "2022-03-01" "2022-03-02" "2022-03-03"
##  [61] "2022-03-04" "2022-03-07" "2022-03-08" "2022-03-09" "2022-03-10"
##  [66] "2022-03-11" "2022-03-14" "2022-03-15" "2022-03-16" "2022-03-17"
##  [71] "2022-03-18" "2022-03-21" "2022-03-22" "2022-03-23" "2022-03-24"
##  [76] "2022-03-25" "2022-03-28" "2022-03-29" "2022-03-30" "2022-03-31"
##  [81] "2022-04-01" "2022-04-04" "2022-04-05" "2022-04-06" "2022-04-07"
##  [86] "2022-04-08" "2022-04-11" "2022-04-12" "2022-04-13" "2022-04-14"
##  [91] "2022-04-18" "2022-04-19" "2022-04-20" "2022-04-21" "2022-04-22"
##  [96] "2022-04-25" "2022-04-26" "2022-04-27" "2022-04-28" "2022-04-29"
## [101] "2022-05-02" "2022-05-03" "2022-05-04" "2022-05-05" "2022-05-06"
## [106] "2022-05-09" "2022-05-10" "2022-05-11" "2022-05-12" "2022-05-13"
## [111] "2022-05-16" "2022-05-17" "2022-05-18" "2022-05-19" "2022-05-20"
## [116] "2022-05-23" "2022-05-24" "2022-05-25" "2022-05-26" "2022-05-27"
## [121] "2022-05-31" "2022-06-01" "2022-06-02" "2022-06-03" "2022-06-06"
## [126] "2022-06-07" "2022-06-08" "2022-06-09" "2022-06-10" "2022-06-13"
## [131] "2022-06-14" "2022-06-15" "2022-06-16" "2022-06-17" "2022-06-21"
## [136] "2022-06-22" "2022-06-23" "2022-06-24" "2022-06-27" "2022-06-28"
## [141] "2022-06-29" "2022-06-30" "2022-07-01" "2022-07-05" "2022-07-06"
## [146] "2022-07-07" "2022-07-08" "2022-07-11" "2022-07-12" "2022-07-13"
## [151] "2022-07-14" "2022-07-15" "2022-07-18" "2022-07-19" "2022-07-20"
## [156] "2022-07-21" "2022-07-22" "2022-07-25" "2022-07-26" "2022-07-27"
## [161] "2022-07-28" "2022-07-29" "2022-08-01" "2022-08-02" "2022-08-03"
## [166] "2022-08-04" "2022-08-05" "2022-08-08" "2022-08-09" "2022-08-10"
## [171] "2022-08-11" "2022-08-12" "2022-08-15" "2022-08-16" "2022-08-17"
## [176] "2022-08-18" "2022-08-19" "2022-08-22" "2022-08-23" "2022-08-24"
## [181] "2022-08-25" "2022-08-26" "2022-08-29" "2022-08-30" "2022-08-31"
## [186] "2022-09-01" "2022-09-02" "2022-09-06" "2022-09-07" "2022-09-08"
## [191] "2022-09-09" "2022-09-12" "2022-09-13" "2022-09-14" "2022-09-15"
## [196] "2022-09-16" "2022-09-19" "2022-09-20" "2022-09-21" "2022-09-22"
## [201] "2022-09-23" "2022-09-26" "2022-09-27" "2022-09-28" "2022-09-29"
```

```
## [206] "2022-09-30" "2022-10-03" "2022-10-04" "2022-10-05" "2022-10-06"
## [211] "2022-10-07" "2022-10-11" "2022-10-12" "2022-10-13" "2022-10-14"
## [216] "2022-10-17" "2022-10-18" "2022-10-19" "2022-10-20" "2022-10-21"
## [221] "2022-10-24" "2022-10-25" "2022-10-26" "2022-10-27" "2022-10-28"
## [226] "2022-10-31" "2022-11-01" "2022-11-02" "2022-11-03" "2022-11-04"
## [231] "2022-11-07" "2022-11-08" "2022-11-09" "2022-11-10" "2022-11-14"
## [236] "2022-11-15" "2022-11-16" "2022-11-17" "2022-11-18" "2022-11-21"
## [241] "2022-11-22" "2022-11-23" "2022-11-25" "2022-11-28" "2022-11-29"
## [246] "2022-11-30" "2022-12-01" "2022-12-02" "2022-12-05" "2022-12-06"
## [251] "2022-12-07" "2022-12-08"
```

```
score_fun <- function(mods, dat) {
  output <- map_dfc(.x = mods, .f = function(x) {
    if(class(x)[1] != "train"){
      predict(x, newdata = dat, type = "response")
    } else(
       predict(x, newdata = dat, type = "prob")[,"yes"]

    )

  })

  output$date <- dat$date


  output <- output %>%
    pivot_longer(-date, names_to = "model",
                 values_to = "prob_rec")

  return(output)
}


recent_prob <- score_fun(mymods, curr_data)

knitr::kable(recent_prob %>% filter(
  date >= "2022-10-01"
))
```

date	model	prob_rec
2022-10-03	XGB	0.1914026
2022-10-03	GAM	0.1308002
2022-10-03	RF	0.0420000
2022-10-03	Step	0.0376653
2022-10-03	Lasso	0.0741688
2022-10-03	MARS	0.0515608
2022-10-03	Null	0.1470138
2022-10-03	Logit	0.0465343
2022-10-04	XGB	0.1914026
2022-10-04	GAM	0.1274175
2022-10-04	RF	0.0400000
2022-10-04	Step	0.0337861

date	model	prob_rec
2022-10-04	Lasso	0.0696312
2022-10-04	MARS	0.0518495
2022-10-04	Null	0.1470138
2022-10-04	Logit	0.0393310
2022-10-05	XGB	0.1914026
2022-10-05	GAM	0.1264605
2022-10-05	RF	0.0240000
2022-10-05	Step	0.0337305
2022-10-05	Lasso	0.0696119
2022-10-05	MARS	0.0520765
2022-10-05	Null	0.1470138
2022-10-05	Logit	0.0373871
2022-10-06	XGB	0.1914026
2022-10-06	GAM	0.1249212
2022-10-06	RF	0.0200000
2022-10-06	Step	0.0315127
2022-10-06	Lasso	0.0668149
2022-10-06	MARS	0.0522854
2022-10-06	Null	0.1470138
2022-10-06	Logit	0.0343507
2022-10-07	XGB	0.1914026
2022-10-07	GAM	0.1243243
2022-10-07	RF	0.0160000
2022-10-07	Step	0.0292138
2022-10-07	Lasso	0.0639525
2022-10-07	MARS	0.0524919
2022-10-07	Null	0.1470138
2022-10-07	Logit	0.0332040
2022-10-11	XGB	0.1914026
2022-10-11	GAM	0.1234485
2022-10-11	RF	0.0080000
2022-10-11	Step	0.0284327
2022-10-11	Lasso	0.0628115
2022-10-11	MARS	0.0526833
2022-10-11	Null	0.1470138
2022-10-11	Logit	0.0315529
2022-10-12	XGB	0.1914026
2022-10-12	GAM	0.1228772
2022-10-12	RF	0.0040000
2022-10-12	Step	0.0262126
2022-10-12	Lasso	0.0599617
2022-10-12	MARS	0.0528977
2022-10-12	Null	0.1470138
2022-10-12	Logit	0.0304966
2022-10-13	XGB	0.1914026
2022-10-13	GAM	0.1237379
2022-10-13	RF	0.0080000
2022-10-13	Step	0.0257166
2022-10-13	Lasso	0.0595425
2022-10-13	MARS	0.0531033
2022-10-13	Null	0.1470138
2022-10-13	Logit	0.0320942

date	model	prob_rec
2022-10-14	XGB	0.1914026
2022-10-14	GAM	0.1240298
2022-10-14	RF	0.0040000
2022-10-14	Step	0.0261471
2022-10-14	Lasso	0.0601496
2022-10-14	MARS	0.0532935
2022-10-14	Null	0.1470138
2022-10-14	Logit	0.0326445
2022-10-17	XGB	0.1914026
2022-10-17	GAM	0.1252236
2022-10-17	RF	0.0000000
2022-10-17	Step	0.0274664
2022-10-17	Lasso	0.0619570
2022-10-17	MARS	0.0534681
2022-10-17	Null	0.1470138
2022-10-17	Logit	0.0349383
2022-10-18	XGB	0.1914026
2022-10-18	GAM	0.1228772
2022-10-18	RF	0.0040000
2022-10-18	Step	0.0263558
2022-10-18	Lasso	0.0601111
2022-10-18	MARS	0.0536336
2022-10-18	Null	0.1470138
2022-10-18	Logit	0.0304966
2022-10-19	XGB	0.1914026
2022-10-19	GAM	0.1204258
2022-10-19	RF	0.0060000
2022-10-19	Step	0.0228406
2022-10-19	Lasso	0.0549219
2022-10-19	MARS	0.0537799
2022-10-19	Null	0.1470138
2022-10-19	Logit	0.0261527
2022-10-20	XGB	0.1914026
2022-10-20	GAM	0.1214919
2022-10-20	RF	0.0040000
2022-10-20	Step	0.0236139
2022-10-20	Lasso	0.0559656
2022-10-20	MARS	0.0538940
2022-10-20	Null	0.1470138
2022-10-20	Logit	0.0280036
2022-10-21	XGB	0.1914026
2022-10-21	GAM	0.1214919
2022-10-21	RF	0.0020000
2022-10-21	Step	0.0223843
2022-10-21	Lasso	0.0541277
2022-10-21	MARS	0.0540378
2022-10-21	Null	0.1470138
2022-10-21	Logit	0.0280036
2022-10-24	XGB	0.1914026
2022-10-24	GAM	0.1237379
2022-10-24	RF	0.0280000
2022-10-24	Step	0.0256188

date	model	prob_rec
2022-10-24	Lasso	0.0577729
2022-10-24	MARS	0.0541688
2022-10-24	Null	0.1470138
2022-10-24	Logit	0.0320942
2022-10-25	XGB	0.1914026
2022-10-25	GAM	0.1237379
2022-10-25	RF	0.0120000
2022-10-25	Step	0.0261730
2022-10-25	Lasso	0.0578126
2022-10-25	MARS	0.0543593
2022-10-25	Null	0.1470138
2022-10-25	Logit	0.0320942
2022-10-26	XGB	0.1914026
2022-10-26	GAM	0.1228772
2022-10-26	RF	0.0020000
2022-10-26	Step	0.0265424
2022-10-26	Lasso	0.0578392
2022-10-26	MARS	0.0545637
2022-10-26	Null	0.1470138
2022-10-26	Logit	0.0304966
2022-10-27	XGB	0.1914026
2022-10-27	GAM	0.1209542
2022-10-27	RF	0.0100000
2022-10-27	Step	0.0255364
2022-10-27	Lasso	0.0561891
2022-10-27	MARS	0.0547886
2022-10-27	Null	0.1470138
2022-10-27	Logit	0.0270628
2022-10-28	XGB	0.1914026
2022-10-28	GAM	0.1220388
2022-10-28	RF	0.0120000
2022-10-28	Step	0.0275747
2022-10-28	Lasso	0.0583091
2022-10-28	MARS	0.0549912
2022-10-28	Null	0.1470138
2022-10-28	Logit	0.0289761
2022-10-31	XGB	0.1914026
2022-10-31	GAM	0.1228772
2022-10-31	RF	0.0180000
2022-10-31	Step	0.0319116
2022-10-31	Lasso	0.0627711
2022-10-31	MARS	0.0551377
2022-10-31	Null	0.1470138
2022-10-31	Logit	0.0304966
2022-11-01	XGB	0.1914026
2022-11-01	GAM	0.1225953
2022-11-01	RF	0.0100000
2022-11-01	Step	0.0315194
2022-11-01	Lasso	0.0617983
2022-11-01	MARS	0.0553046
2022-11-01	Null	0.1470138
2022-11-01	Logit	0.0299813

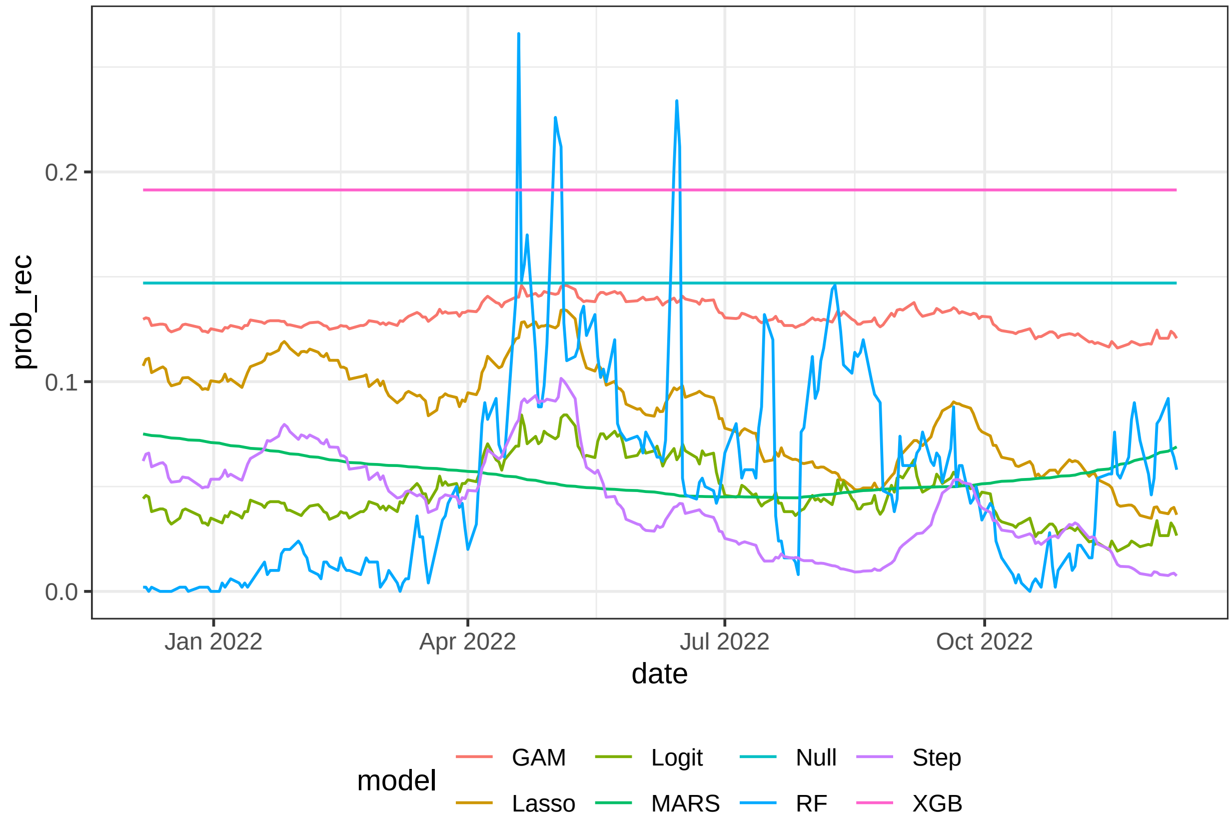
date	model	prob_rec
2022-11-02	XGB	0.1914026
2022-11-02	GAM	0.1220388
2022-11-02	RF	0.0120000
2022-11-02	Step	0.0326816
2022-11-02	Lasso	0.0622992
2022-11-02	MARS	0.0554587
2022-11-02	Null	0.1470138
2022-11-02	Logit	0.0289761
2022-11-03	XGB	0.1914026
2022-11-03	GAM	0.1228772
2022-11-03	RF	0.0220000
2022-11-03	Step	0.0319020
2022-11-03	Lasso	0.0618197
2022-11-03	MARS	0.0558590
2022-11-03	Null	0.1470138
2022-11-03	Logit	0.0304966
2022-11-04	XGB	0.1914026
2022-11-04	GAM	0.1217641
2022-11-04	RF	0.0220000
2022-11-04	Step	0.0302011
2022-11-04	Lasso	0.0603378
2022-11-04	MARS	0.0562450
2022-11-04	Null	0.1470138
2022-11-04	Logit	0.0284858
2022-11-07	XGB	0.1914026
2022-11-07	GAM	0.1188933
2022-11-07	RF	0.0160000
2022-11-07	Step	0.0254911
2022-11-07	Lasso	0.0548508
2022-11-07	MARS	0.0566302
2022-11-07	Null	0.1470138
2022-11-07	Logit	0.0235977
2022-11-08	XGB	0.1914026
2022-11-08	GAM	0.1191433
2022-11-08	RF	0.0160000
2022-11-08	Step	0.0260127
2022-11-08	Lasso	0.0561086
2022-11-08	MARS	0.0570212
2022-11-08	Null	0.1470138
2022-11-08	Logit	0.0240060
2022-11-09	XGB	0.1914026
2022-11-09	GAM	0.1181555
2022-11-09	RF	0.0300000
2022-11-09	Step	0.0256097
2022-11-09	Lasso	0.0561120
2022-11-09	MARS	0.0573940
2022-11-09	Null	0.1470138
2022-11-09	Logit	0.0224133
2022-11-10	XGB	0.1914026
2022-11-10	GAM	0.1186453
2022-11-10	RF	0.0540000
2022-11-10	Step	0.0225853

date	model	prob_rec
2022-11-10	Lasso	0.0533421
2022-11-10	MARS	0.0578947
2022-11-10	Null	0.1470138
2022-11-10	Logit	0.0231963
2022-11-14	XGB	0.1914026
2022-11-14	GAM	0.1165037
2022-11-14	RF	0.0560000
2022-11-14	Step	0.0202497
2022-11-14	Lasso	0.0506947
2022-11-14	MARS	0.0583607
2022-11-14	Null	0.1470138
2022-11-14	Logit	0.0198709
2022-11-15	XGB	0.1914026
2022-11-15	GAM	0.1191433
2022-11-15	RF	0.0560000
2022-11-15	Step	0.0184559
2022-11-15	Lasso	0.0493750
2022-11-15	MARS	0.0588941
2022-11-15	Null	0.1470138
2022-11-15	Logit	0.0240060
2022-11-16	XGB	0.1914026
2022-11-16	GAM	0.1176738
2022-11-16	RF	0.0760000
2022-11-16	Step	0.0155535
2022-11-16	Lasso	0.0456053
2022-11-16	MARS	0.0594857
2022-11-16	Null	0.1470138
2022-11-16	Logit	0.0216561
2022-11-17	XGB	0.1914026
2022-11-17	GAM	0.1160488
2022-11-17	RF	0.0560000
2022-11-17	Step	0.0128439
2022-11-17	Lasso	0.0414318
2022-11-17	MARS	0.0600648
2022-11-17	Null	0.1470138
2022-11-17	Logit	0.0191980
2022-11-18	XGB	0.1914026
2022-11-18	GAM	0.1165037
2022-11-18	RF	0.0540000
2022-11-18	Step	0.0119419
2022-11-18	Lasso	0.0405803
2022-11-18	MARS	0.0606310
2022-11-18	Null	0.1470138
2022-11-18	Logit	0.0198709
2022-11-21	XGB	0.1914026
2022-11-21	GAM	0.1179137
2022-11-21	RF	0.0640000
2022-11-21	Step	0.0117532
2022-11-21	Lasso	0.0411697
2022-11-21	MARS	0.0611875
2022-11-21	Null	0.1470138
2022-11-21	Logit	0.0220315

date	model	prob_rec
2022-11-22	XGB	0.1914026
2022-11-22	GAM	0.1191433
2022-11-22	RF	0.0820000
2022-11-22	Step	0.0112568
2022-11-22	Lasso	0.0411222
2022-11-22	MARS	0.0617709
2022-11-22	Null	0.1470138
2022-11-22	Logit	0.0240060
2022-11-23	XGB	0.1914026
2022-11-23	GAM	0.1186453
2022-11-23	RF	0.0900000
2022-11-23	Step	0.0105061
2022-11-23	Lasso	0.0402573
2022-11-23	MARS	0.0623596
2022-11-23	Null	0.1470138
2022-11-23	Logit	0.0231963
2022-11-25	XGB	0.1914026
2022-11-25	GAM	0.1174359
2022-11-25	RF	0.0720000
2022-11-25	Step	0.0085125
2022-11-25	Lasso	0.0362495
2022-11-25	MARS	0.0629988
2022-11-25	Null	0.1470138
2022-11-25	Logit	0.0212869
2022-11-28	XGB	0.1914026
2022-11-28	GAM	0.1181555
2022-11-28	RF	0.0560000
2022-11-28	Step	0.0078324
2022-11-28	Lasso	0.0351473
2022-11-28	MARS	0.0636556
2022-11-28	Null	0.1470138
2022-11-28	Logit	0.0224133
2022-11-29	XGB	0.1914026
2022-11-29	GAM	0.1179137
2022-11-29	RF	0.0460000
2022-11-29	Step	0.0075946
2022-11-29	Lasso	0.0348587
2022-11-29	MARS	0.0642841
2022-11-29	Null	0.1470138
2022-11-29	Logit	0.0220315
2022-11-30	XGB	0.1914026
2022-11-30	GAM	0.1214919
2022-11-30	RF	0.0540000
2022-11-30	Step	0.0093584
2022-11-30	Lasso	0.0400541
2022-11-30	MARS	0.0648834
2022-11-30	Null	0.1470138
2022-11-30	Logit	0.0280036
2022-12-01	XGB	0.1914026
2022-12-01	GAM	0.1246215
2022-12-01	RF	0.0800000
2022-12-01	Step	0.0090746

date	model	prob_rec
2022-12-01	Lasso	0.0403123
2022-12-01	MARS	0.0655624
2022-12-01	Null	0.1470138
2022-12-01	Logit	0.0337726
2022-12-02	XGB	0.1914026
2022-12-02	GAM	0.1206889
2022-12-02	RF	0.0820000
2022-12-02	Step	0.0080379
2022-12-02	Lasso	0.0378019
2022-12-02	MARS	0.0662242
2022-12-02	Null	0.1470138
2022-12-02	Logit	0.0266039
2022-12-05	XGB	0.1914026
2022-12-05	GAM	0.1206889
2022-12-05	RF	0.0920000
2022-12-05	Step	0.0075719
2022-12-05	Lasso	0.0370224
2022-12-05	MARS	0.0668523
2022-12-05	Null	0.1470138
2022-12-05	Logit	0.0266039
2022-12-06	XGB	0.1914026
2022-12-06	GAM	0.1240298
2022-12-06	RF	0.0680000
2022-12-06	Step	0.0083228
2022-12-06	Lasso	0.0393201
2022-12-06	MARS	0.0675262
2022-12-06	Null	0.1470138
2022-12-06	Logit	0.0326445
2022-12-07	XGB	0.1914026
2022-12-07	GAM	0.1228772
2022-12-07	RF	0.0640000
2022-12-07	Step	0.0086798
2022-12-07	Lasso	0.0400539
2022-12-07	MARS	0.0682064
2022-12-07	Null	0.1470138
2022-12-07	Logit	0.0304966
2022-12-08	XGB	0.1914026
2022-12-08	GAM	0.1206889
2022-12-08	RF	0.0580000
2022-12-08	Step	0.0074452
2022-12-08	Lasso	0.0365303
2022-12-08	MARS	0.0689012
2022-12-08	Null	0.1470138
2022-12-08	Logit	0.0266039

```r
ggplot(recent_prob, aes(x=date, y=prob_rec,
                        group=model, color=model)) +
  geom_line() + theme_bw() +
  theme(legend.position = "bottom")
```



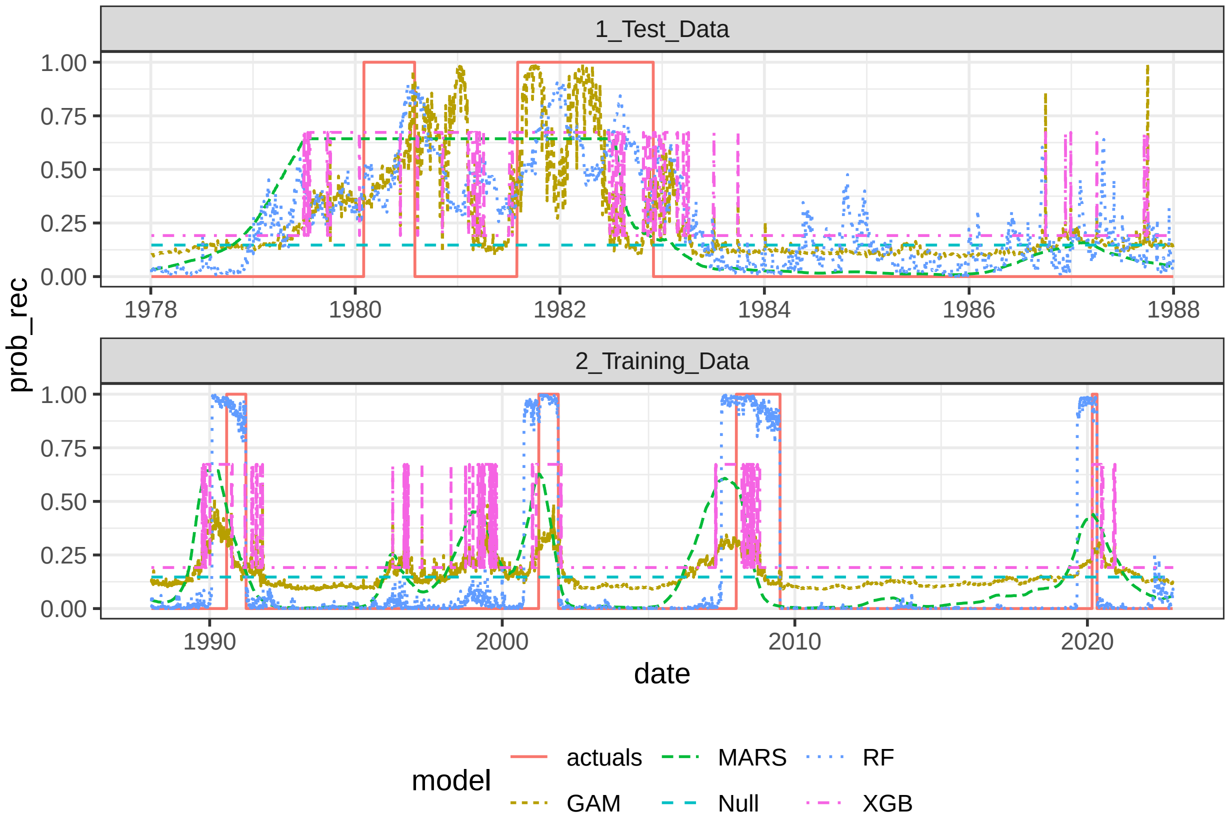
Backtesting

```
full_data_bktst <- full_data_wide %>%
  filter(date >= startTestDate)

bkst_fun <- function(mods, dat) {
  output <- map_dfc(.x = mods, .f = function(x) {
    if(class(x)[1] != "train"){
      predict(x, newdata = dat, type = "response")
    } else(
       predict(x, newdata = dat, type = "prob")[,"yes"]

    )

  })

  output$date <- dat$date

  output <- output%>%
    pivot_longer(-date, names_to = "model",
                 values_to = "prob_rec")

  return(output)
}
```

```r
df_plot <- bkst_fun(mymods, full_data_bktst)

actuals <- full_data_bktst %>%
  mutate(model="actuals") %>%
  select(date, model, prob_rec=USREC)

df_plot_final <- bind_rows(df_plot, actuals)

end_test_date <- max(test_data$date)

df_plot_final <- df_plot_final %>%
  mutate(epoc = case_when(date <= end_test_date ~ "1_Test_Data",
                          TRUE ~ "2_Training_Data")
  )

df_plot_logit_scam <- df_plot_final %>%
  filter(model %in% c('actuals', 'Null',
                      'Logit', 'Step', 'Lasso',
                      'LogitKnot'))

df_plot_knots_gbm <- df_plot_final %>%
  filter(model %in% c('actuals', 'Null',
                      'XGB', 'RF',
                      'GAM',
                      'MARS'))
```

```r
ggplot(df_plot_logit_scam, aes(x=date, y=prob_rec, group=model,
                          linetype=model, color=model)) +
  geom_line() +
  theme_bw() +
  theme(legend.position = "bottom") +
  facet_wrap(vars(epoc), scales="free", nrow=2)
```


```r
ggplot(df_plot_knots_gbm, aes(x=date, y=prob_rec, group=model,
                              linetype=model, color=model)) +
  geom_line() +
  theme_bw() +
  theme(legend.position = "bottom") +
  facet_wrap(vars(epoc), scales="free", nrow=2)
```




```
stopCluster(cl)
```