Probability of Recession

William Chiu

2022-12-17

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.

```
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"
```

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

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

response_id <- "USREC" # monthly
```

```
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

```
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

```
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) %>%
```

```
  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.

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

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

Split Train/Test

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

```
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 = 8,494**
date	1988-01-04 to 2021-12-13
DFF	2.44 (0.20, 5.28)
DGS10	4.41 (2.58, 6.18)
SPRD_10YCMT_FEDFUNDS	1.47 (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, 2.51)
SPRD_10YCMT_FEDFUNDS_lag6m	1.51 (0.50, 2.51)
SPRD_10YCMT_FEDFUNDS_lag9m	1.51 (0.50, 2.51)
SPRD_10YCMT_FEDFUNDS_lag12m	1.51 (0.50, 2.51)
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	1.47 (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, 4.18)
SPRD_10YCMT_FEDFUNDS_adstk75	5.9 (2.1, 10.0)
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	8,494 (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, 2.51)
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	1.45 (0.49, 2.57)
SPRD_10YCMT_FEDFUNDS_ma9m	1.46 (0.50, 2.59)
SPRD_10YCMT_FEDFUNDS_ma12m	1.47 (0.51, 2.58)

Characteristic	N = 8,494
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 (-0.08, 0.07)
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)
```

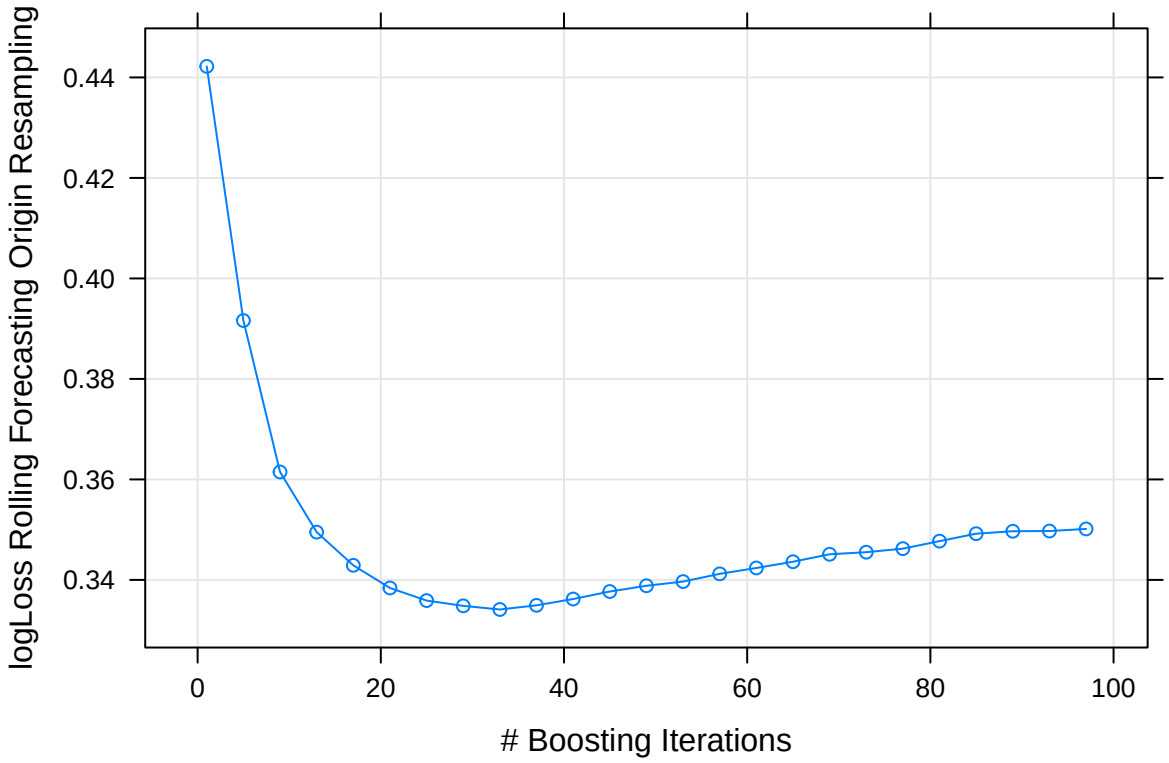
Cross-Validation Framework

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

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

```
}

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

```
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)
```



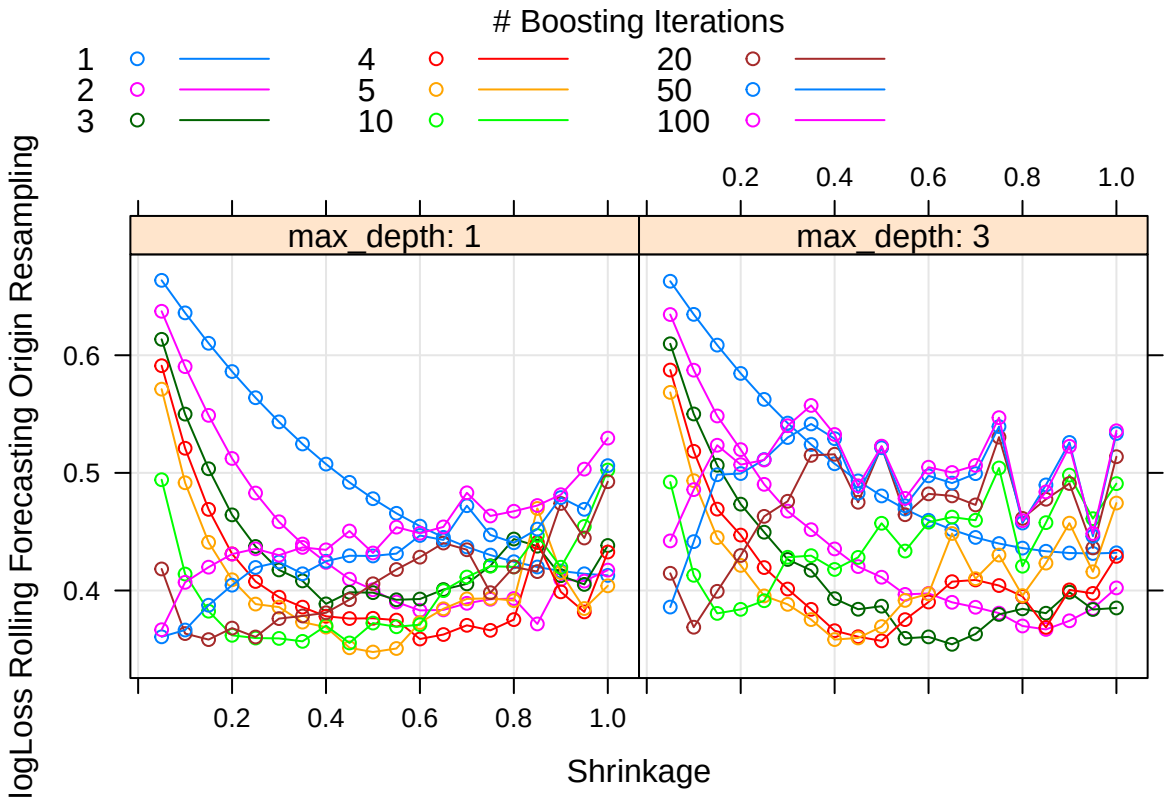

```
gam_mod$bestTune
```

```
##   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",
```

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

plot(xgb_mod)
```



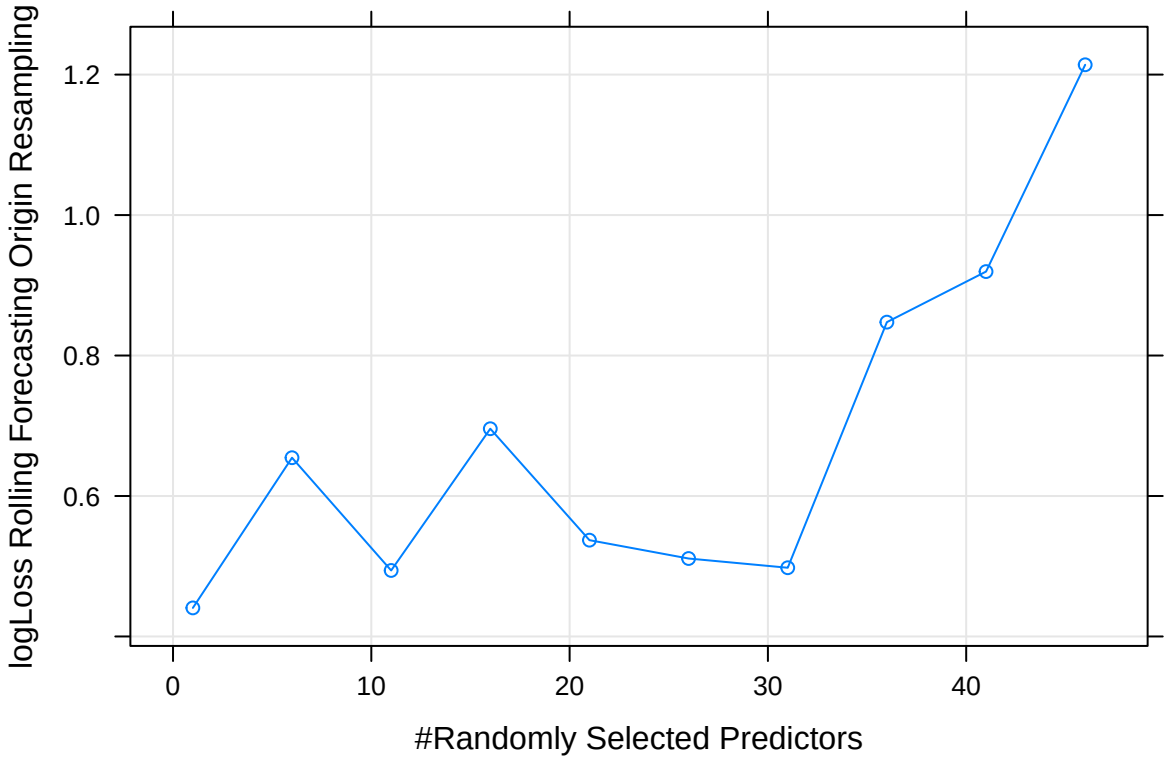

```
xgb_mod$bestTune
```

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

Random Forest

```
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:

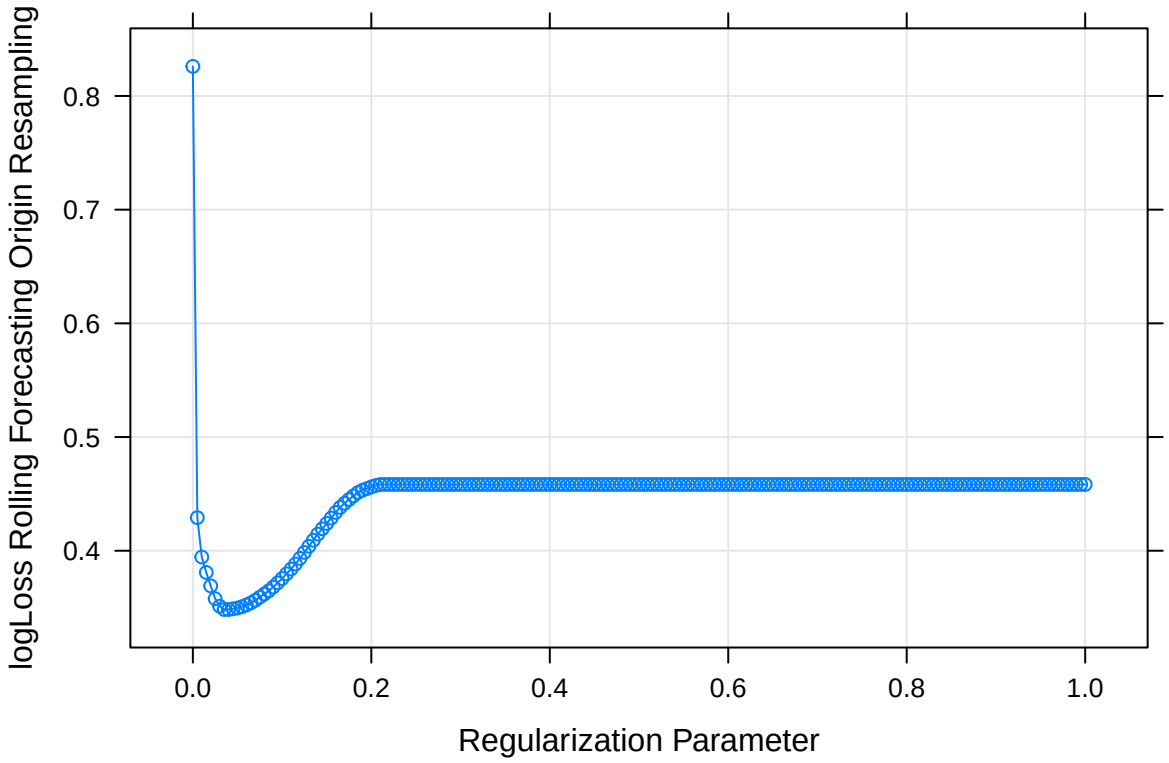
> For binomial and quasibinomial families the response can also be specified as a factor (when the first level denotes failure and all others success) or as a two-column matrix with the columns giving the numbers of successes and failures.

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)
```

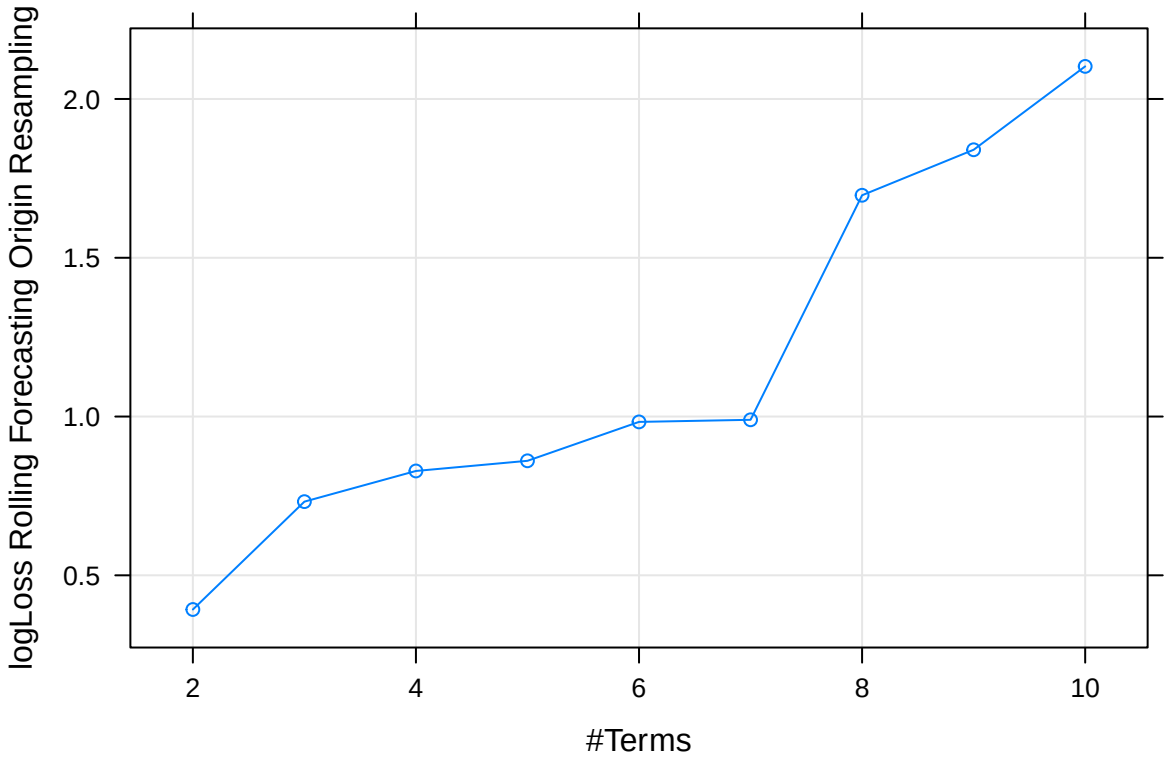



```
glmnet_mod$bestTune
```

```
##   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)
```




```
earth_mod$bestTune
```

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

Null Model: Intercept-only Model

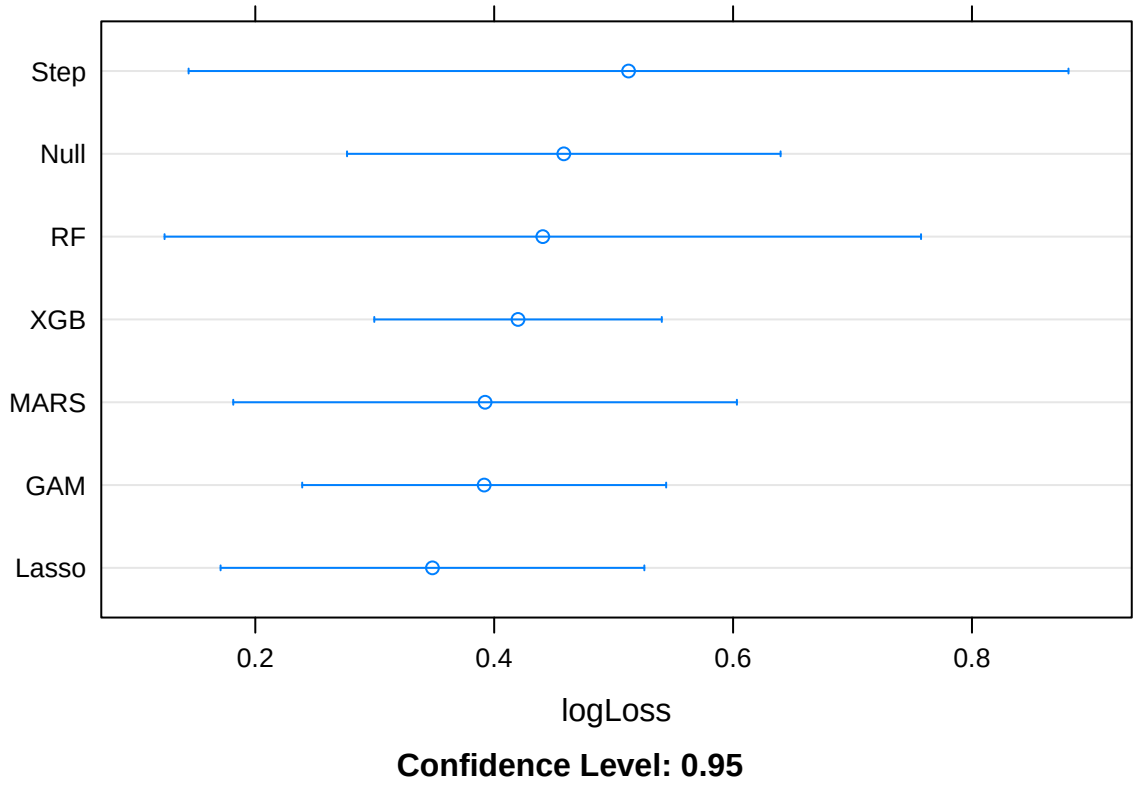
```
set.seed(randSeed)

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
```

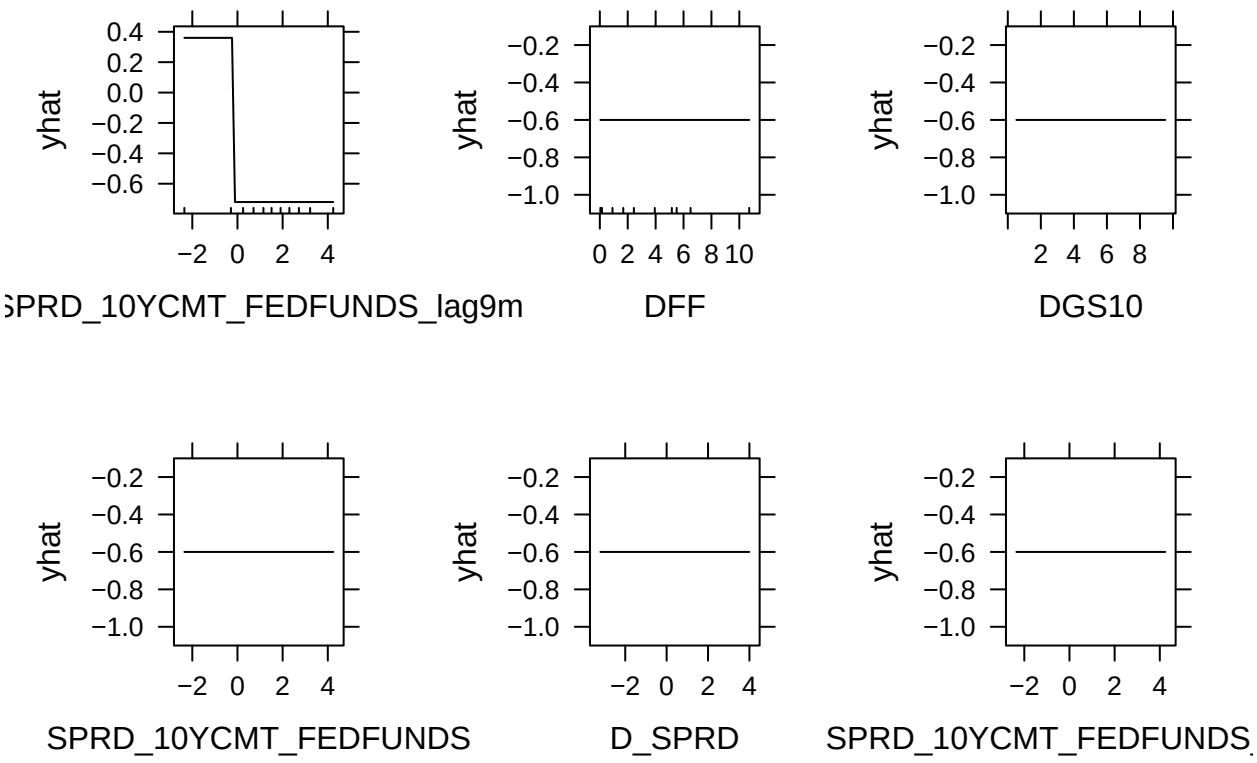
```
df_imp <- varImp(xgb_mod)$importance %>%
  arrange(desc(Overall))

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`.

```
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
## -3.1831   0.0670   0.1788   0.4298   2.7142
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  0.47237    0.04009   11.78   <2e-16 ***
## SPRD_10YCMT_FEDFUNDS_lag9m  1.75764    0.04861   36.16   <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: 7089.8  on 8493  degrees of freedom
## Residual deviance: 4490.1  on 8492  degrees of freedom
## AIC: 4494.1
##
## Number of Fisher Scoring iterations: 6
```

Compare Models

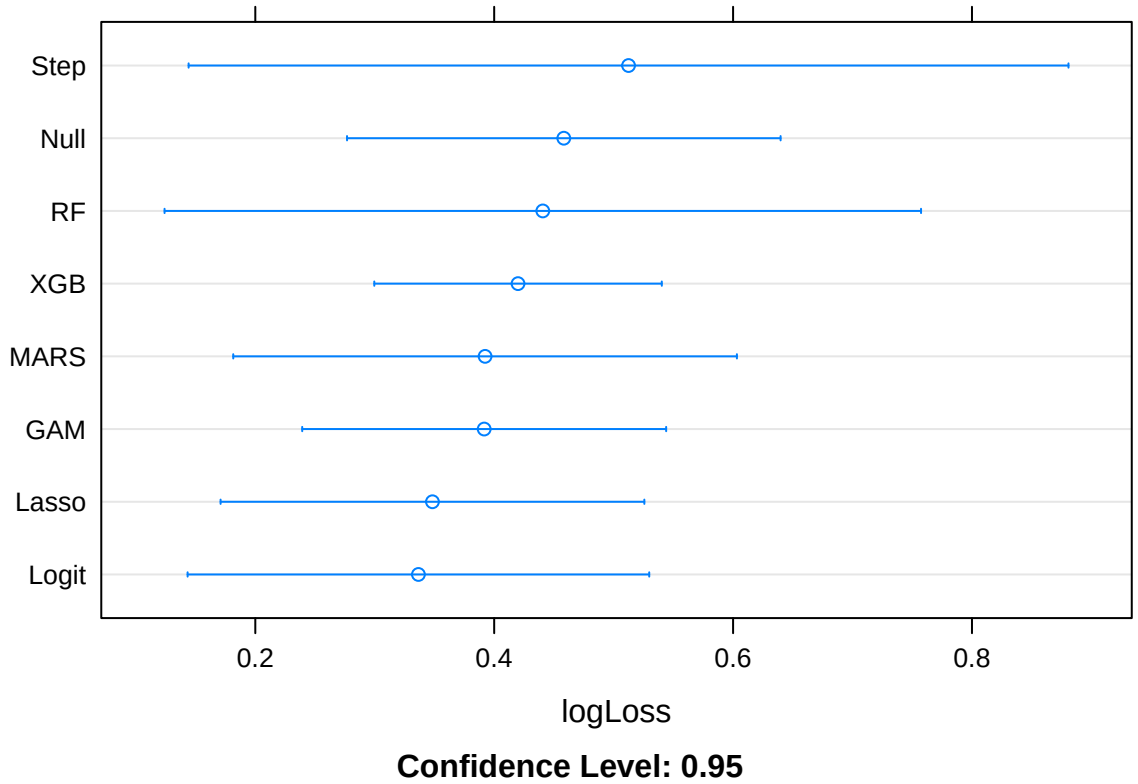
CV errors for models with peeking are misleadingly low. This will be addressed with a test set.

```
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,
```

```
           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
RF	0.9201238
Lasso	0.9123442
MARS	0.8977965
Step	0.8927124
GAM	0.8677476
Logit	0.8677416
XGB	0.8227510
Null	0.5000000

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

model	LogLoss
MARS	0.3486103
RF	0.3564049
Lasso	0.3629046
XGB	0.4242245
Step	0.4354626
GAM	0.4604459
Logit	0.5949134
Null	0.6567144

Probability of Recession (Most Recent 12 months)

```
curr_data <- recent_data

curr_data$date
```

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

```
## [236] "2022-11-22" "2022-11-23" "2022-11-25" "2022-11-28" "2022-11-29"
## [241] "2022-11-30" "2022-12-01" "2022-12-02" "2022-12-05" "2022-12-06"
## [246] "2022-12-07" "2022-12-08" "2022-12-09" "2022-12-12" "2022-12-13"
## [251] "2022-12-14" "2022-12-15"
```

Probability of Recession (the 12 most recent months)

```
curr_data <- recent_data

curr_data$date
```

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

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

```r
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	0.1913760
2022-10-03	GAM	0.1307263
2022-10-03	RF	0.0440000
2022-10-03	Step	0.0376113
2022-10-03	Lasso	0.0740821
2022-10-03	MARS	0.0397330
2022-10-03	Null	0.1469272
2022-10-03	Logit	0.0464885
2022-10-04	XGB	0.1913760
2022-10-04	GAM	0.1273448
2022-10-04	RF	0.0360000
2022-10-04	Step	0.0337332

date	model	prob_rec
2022-10-04	Lasso	0.0695389
2022-10-04	MARS	0.0404610
2022-10-04	Null	0.1469272
2022-10-04	Logit	0.0392897
2022-10-05	XGB	0.1913760
2022-10-05	GAM	0.1263882
2022-10-05	RF	0.0320000
2022-10-05	Step	0.0336755
2022-10-05	Lasso	0.0695127
2022-10-05	MARS	0.0451040
2022-10-05	Null	0.1469272
2022-10-05	Logit	0.0373470
2022-10-06	XGB	0.1913760
2022-10-06	GAM	0.1248496
2022-10-06	RF	0.0360000
2022-10-06	Step	0.0314594
2022-10-06	Lasso	0.0667155
2022-10-06	MARS	0.0442964
2022-10-06	Null	0.1469272
2022-10-06	Logit	0.0343128
2022-10-07	XGB	0.1913760
2022-10-07	GAM	0.1242529
2022-10-07	RF	0.0260000
2022-10-07	Step	0.0291640
2022-10-07	Lasso	0.0638588
2022-10-07	MARS	0.0404610
2022-10-07	Null	0.1469272
2022-10-07	Logit	0.0331669
2022-10-11	XGB	0.1913760
2022-10-11	GAM	0.1233775
2022-10-11	RF	0.0140000
2022-10-11	Step	0.0283831
2022-10-11	Lasso	0.0627160
2022-10-11	MARS	0.0412017
2022-10-11	Null	0.1469272
2022-10-11	Logit	0.0315170
2022-10-12	XGB	0.1913760
2022-10-12	GAM	0.1228065
2022-10-12	RF	0.0040000
2022-10-12	Step	0.0261666
2022-10-12	Lasso	0.0598720
2022-10-12	MARS	0.0376237
2022-10-12	Null	0.1469272
2022-10-12	Logit	0.0304615
2022-10-13	XGB	0.1913760
2022-10-13	GAM	0.1236667
2022-10-13	RF	0.0160000
2022-10-13	Step	0.0256725
2022-10-13	Lasso	0.0594572
2022-10-13	MARS	0.0356223
2022-10-13	Null	0.1469272
2022-10-13	Logit	0.0320579

date	model	prob_rec
2022-10-14	XGB	0.1913760
2022-10-14	GAM	0.1239586
2022-10-14	RF	0.0060000
2022-10-14	Step	0.0261023
2022-10-14	Lasso	0.0600625
2022-10-14	MARS	0.0369449
2022-10-14	Null	0.1469272
2022-10-14	Logit	0.0326078
2022-10-17	XGB	0.1913760
2022-10-17	GAM	0.1251519
2022-10-17	RF	0.0020000
2022-10-17	Step	0.0274203
2022-10-17	Lasso	0.0618695
2022-10-17	MARS	0.0390176
2022-10-17	Null	0.1469272
2022-10-17	Logit	0.0349000
2022-10-18	XGB	0.1913760
2022-10-18	GAM	0.1228065
2022-10-18	RF	0.0020000
2022-10-18	Step	0.0263087
2022-10-18	Lasso	0.0600159
2022-10-18	MARS	0.0419554
2022-10-18	Null	0.1469272
2022-10-18	Logit	0.0304615
2022-10-19	XGB	0.1913760
2022-10-19	GAM	0.1203562
2022-10-19	RF	0.0040000
2022-10-19	Step	0.0227978
2022-10-19	Lasso	0.0548313
2022-10-19	MARS	0.0369449
2022-10-19	Null	0.1469272
2022-10-19	Logit	0.0261211
2022-10-20	XGB	0.1913760
2022-10-20	GAM	0.1214218
2022-10-20	RF	0.0060000
2022-10-20	Step	0.0235707
2022-10-20	Lasso	0.0558754
2022-10-20	MARS	0.0369449
2022-10-20	Null	0.1469272
2022-10-20	Logit	0.0279704
2022-10-21	XGB	0.1913760
2022-10-21	GAM	0.1214218
2022-10-21	RF	0.0060000
2022-10-21	Step	0.0223439
2022-10-21	Lasso	0.0540438
2022-10-21	MARS	0.0331126
2022-10-21	Null	0.1469272
2022-10-21	Logit	0.0279704
2022-10-24	XGB	0.1913760
2022-10-24	GAM	0.1236667
2022-10-24	RF	0.0200000
2022-10-24	Step	0.0255754

date	model	prob_rec
2022-10-24	Lasso	0.0576888
2022-10-24	MARS	0.0331126
2022-10-24	Null	0.1469272
2022-10-24	Logit	0.0320579
2022-10-25	XGB	0.1913760
2022-10-25	GAM	0.1236667
2022-10-25	RF	0.0140000
2022-10-25	Step	0.0261292
2022-10-25	Lasso	0.0577286
2022-10-25	MARS	0.0313433
2022-10-25	Null	0.1469272
2022-10-25	Logit	0.0320579
2022-10-26	XGB	0.1913760
2022-10-26	GAM	0.1228065
2022-10-26	RF	0.0080000
2022-10-26	Step	0.0264970
2022-10-26	Lasso	0.0577505
2022-10-26	MARS	0.0325124
2022-10-26	Null	0.1469272
2022-10-26	Logit	0.0304615
2022-10-27	XGB	0.1913760
2022-10-27	GAM	0.1208844
2022-10-27	RF	0.0060000
2022-10-27	Step	0.0254904
2022-10-27	Lasso	0.0560957
2022-10-27	MARS	0.0337235
2022-10-27	Null	0.1469272
2022-10-27	Logit	0.0270304
2022-10-28	XGB	0.1913760
2022-10-28	GAM	0.1219685
2022-10-28	RF	0.0100000
2022-10-28	Step	0.0275265
2022-10-28	Lasso	0.0582144
2022-10-28	MARS	0.0343453
2022-10-28	Null	0.1469272
2022-10-28	Logit	0.0289422
2022-10-31	XGB	0.1913760
2022-10-31	GAM	0.1228065
2022-10-31	RF	0.0200000
2022-10-31	Step	0.0318564
2022-10-31	Lasso	0.0626651
2022-10-31	MARS	0.0404610
2022-10-31	Null	0.1469272
2022-10-31	Logit	0.0304615
2022-11-01	XGB	0.1913760
2022-11-01	GAM	0.1225247
2022-11-01	RF	0.0000000
2022-11-01	Step	0.0314650
2022-11-01	Lasso	0.0616948
2022-11-01	MARS	0.0383146
2022-11-01	Null	0.1469272
2022-11-01	Logit	0.0299467

date	model	prob_rec
2022-11-02	XGB	0.1913760
2022-11-02	GAM	0.1219685
2022-11-02	RF	0.0100000
2022-11-02	Step	0.0326246
2022-11-02	Lasso	0.0621910
2022-11-02	MARS	0.0390176
2022-11-02	Null	0.1469272
2022-11-02	Logit	0.0289422
2022-11-03	XGB	0.1913760
2022-11-03	GAM	0.1228065
2022-11-03	RF	0.0140000
2022-11-03	Step	0.0318475
2022-11-03	Lasso	0.0617159
2022-11-03	MARS	0.0376237
2022-11-03	Null	0.1469272
2022-11-03	Logit	0.0304615
2022-11-04	XGB	0.1913760
2022-11-04	GAM	0.1216939
2022-11-04	RF	0.0080000
2022-11-04	Step	0.0301476
2022-11-04	Lasso	0.0602292
2022-11-04	MARS	0.0390176
2022-11-04	Null	0.1469272
2022-11-04	Logit	0.0284523
2022-11-07	XGB	0.1913760
2022-11-07	GAM	0.1188245
2022-11-07	RF	0.0140000
2022-11-07	Step	0.0254426
2022-11-07	Lasso	0.0547462
2022-11-07	MARS	0.0362778
2022-11-07	Null	0.1469272
2022-11-07	Logit	0.0235683
2022-11-08	XGB	0.1913760
2022-11-08	GAM	0.1190745
2022-11-08	RF	0.0060000
2022-11-08	Step	0.0259625
2022-11-08	Lasso	0.0559983
2022-11-08	MARS	0.0412017
2022-11-08	Null	0.1469272
2022-11-08	Logit	0.0239762
2022-11-09	XGB	0.1913760
2022-11-09	GAM	0.1180872
2022-11-09	RF	0.0140000
2022-11-09	Step	0.0255581
2022-11-09	Lasso	0.0559937
2022-11-09	MARS	0.0476119
2022-11-09	Null	0.1469272
2022-11-09	Logit	0.0223848
2022-11-10	XGB	0.1913760
2022-11-10	GAM	0.1185767
2022-11-10	RF	0.0400000
2022-11-10	Step	0.0225403

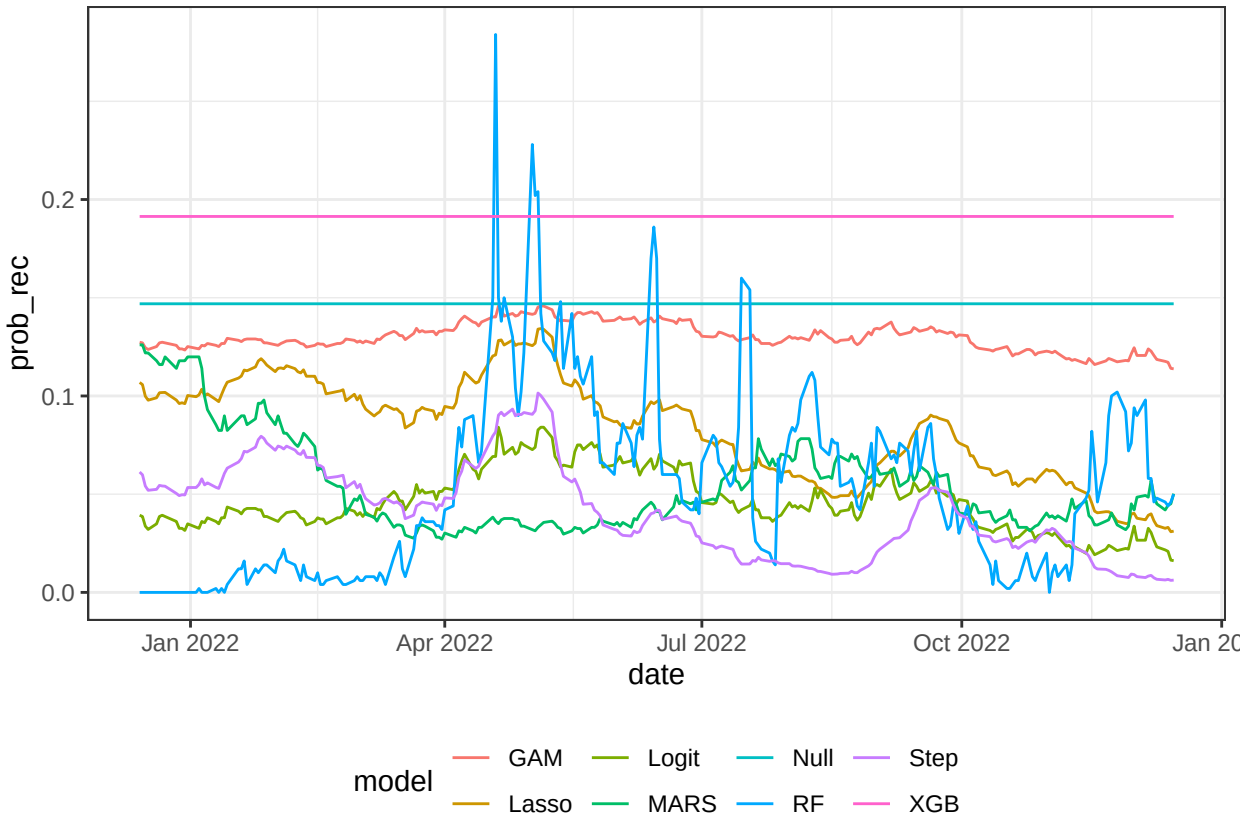
date	model	prob_rec
2022-11-10	Lasso	0.0532353
2022-11-10	MARS	0.0427223
2022-11-10	Null	0.1469272
2022-11-10	Logit	0.0231672
2022-11-14	XGB	0.1913760
2022-11-14	GAM	0.1164363
2022-11-14	RF	0.0480000
2022-11-14	Step	0.0202064
2022-11-14	Lasso	0.0505840
2022-11-14	MARS	0.0467616
2022-11-14	Null	0.1469272
2022-11-14	Logit	0.0198448
2022-11-15	XGB	0.1913760
2022-11-15	GAM	0.1190745
2022-11-15	RF	0.0520000
2022-11-15	Step	0.0184191
2022-11-15	Lasso	0.0492822
2022-11-15	MARS	0.0390176
2022-11-15	Null	0.1469272
2022-11-15	Logit	0.0239762
2022-11-16	XGB	0.1913760
2022-11-16	GAM	0.1176058
2022-11-16	RF	0.0820000
2022-11-16	Step	0.0155211
2022-11-16	Lasso	0.0455173
2022-11-16	MARS	0.0376237
2022-11-16	Null	0.1469272
2022-11-16	Logit	0.0216283
2022-11-17	XGB	0.1913760
2022-11-17	GAM	0.1159816
2022-11-17	RF	0.0660000
2022-11-17	Step	0.0128160
2022-11-17	Lasso	0.0413510
2022-11-17	MARS	0.0343453
2022-11-17	Null	0.1469272
2022-11-17	Logit	0.0191725
2022-11-18	XGB	0.1913760
2022-11-18	GAM	0.1164363
2022-11-18	RF	0.0460000
2022-11-18	Step	0.0119160
2022-11-18	Lasso	0.0405027
2022-11-18	MARS	0.0343453
2022-11-18	Null	0.1469272
2022-11-18	Logit	0.0198448
2022-11-21	XGB	0.1913760
2022-11-21	GAM	0.1178455
2022-11-21	RF	0.0660000
2022-11-21	Step	0.0117282
2022-11-21	Lasso	0.0410939
2022-11-21	MARS	0.0362778
2022-11-21	Null	0.1469272
2022-11-21	Logit	0.0220034

date	model	prob_rec
2022-11-22	XGB	0.1913760
2022-11-22	GAM	0.1190745
2022-11-22	RF	0.0900000
2022-11-22	Step	0.0112333
2022-11-22	Lasso	0.0410498
2022-11-22	MARS	0.0369449
2022-11-22	Null	0.1469272
2022-11-22	Logit	0.0239762
2022-11-23	XGB	0.1913760
2022-11-23	GAM	0.1185767
2022-11-23	RF	0.1000000
2022-11-23	Step	0.0104835
2022-11-23	Lasso	0.0401842
2022-11-23	MARS	0.0404610
2022-11-23	Null	0.1469272
2022-11-23	Logit	0.0231672
2022-11-25	XGB	0.1913760
2022-11-25	GAM	0.1173680
2022-11-25	RF	0.1020000
2022-11-25	Step	0.0084937
2022-11-25	Lasso	0.0361847
2022-11-25	MARS	0.0343453
2022-11-25	Null	0.1469272
2022-11-25	Logit	0.0212595
2022-11-28	XGB	0.1913760
2022-11-28	GAM	0.1180872
2022-11-28	RF	0.0920000
2022-11-28	Step	0.0078154
2022-11-28	Lasso	0.0350882
2022-11-28	MARS	0.0319227
2022-11-28	Null	0.1469272
2022-11-28	Logit	0.0223848
2022-11-29	XGB	0.1913760
2022-11-29	GAM	0.1178455
2022-11-29	RF	0.0720000
2022-11-29	Step	0.0075779
2022-11-29	Lasso	0.0347987
2022-11-29	MARS	0.0337235
2022-11-29	Null	0.1469272
2022-11-29	Logit	0.0220034
2022-11-30	XGB	0.1913760
2022-11-30	GAM	0.1214218
2022-11-30	RF	0.0760000
2022-11-30	Step	0.0093388
2022-11-30	Lasso	0.0399867
2022-11-30	MARS	0.0451040
2022-11-30	Null	0.1469272
2022-11-30	Logit	0.0279704
2022-12-01	XGB	0.1913760
2022-12-01	GAM	0.1245499
2022-12-01	RF	0.0940000
2022-12-01	Step	0.0090567

date	model	prob_rec
2022-12-01	Lasso	0.0402541
2022-12-01	MARS	0.0419554
2022-12-01	Null	0.1469272
2022-12-01	Logit	0.0337351
2022-12-02	XGB	0.1913760
2022-12-02	GAM	0.1206192
2022-12-02	RF	0.0900000
2022-12-02	Step	0.0080204
2022-12-02	Lasso	0.0377369
2022-12-02	MARS	0.0484770
2022-12-02	Null	0.1469272
2022-12-02	Logit	0.0265719
2022-12-05	XGB	0.1913760
2022-12-05	GAM	0.1206192
2022-12-05	RF	0.0980000
2022-12-05	Step	0.0075552
2022-12-05	Lasso	0.0369586
2022-12-05	MARS	0.0493569
2022-12-05	Null	0.1469272
2022-12-05	Logit	0.0265719
2022-12-06	XGB	0.1913760
2022-12-06	GAM	0.1239586
2022-12-06	RF	0.0580000
2022-12-06	Step	0.0083057
2022-12-06	Lasso	0.0392599
2022-12-06	MARS	0.0484770
2022-12-06	Null	0.1469272
2022-12-06	Logit	0.0326078
2022-12-07	XGB	0.1913760
2022-12-07	GAM	0.1228065
2022-12-07	RF	0.0580000
2022-12-07	Step	0.0086612
2022-12-07	Lasso	0.0399860
2022-12-07	MARS	0.0569606
2022-12-07	Null	0.1469272
2022-12-07	Logit	0.0304615
2022-12-08	XGB	0.1913760
2022-12-08	GAM	0.1206192
2022-12-08	RF	0.0460000
2022-12-08	Step	0.0074287
2022-12-08	Lasso	0.0364668
2022-12-08	MARS	0.0493569
2022-12-08	Null	0.1469272
2022-12-08	Logit	0.0265719
2022-12-09	XGB	0.1913760
2022-12-09	GAM	0.1185767
2022-12-09	RF	0.0480000
2022-12-09	Step	0.0066848
2022-12-09	Lasso	0.0339906
2022-12-09	MARS	0.0451040
2022-12-09	Null	0.1469272
2022-12-09	Logit	0.0231672

date	model	prob_rec
2022-12-12	XGB	0.1913760
2022-12-12	GAM	0.1176058
2022-12-12	RF	0.0460000
2022-12-12	Step	0.0063211
2022-12-12	Lasso	0.0326276
2022-12-12	MARS	0.0419554
2022-12-12	Null	0.1469272
2022-12-12	Logit	0.0216283
2022-12-13	XGB	0.1913760
2022-12-13	GAM	0.1171322
2022-12-13	RF	0.0440000
2022-12-13	Step	0.0066431
2022-12-13	Lasso	0.0330923
2022-12-13	MARS	0.0442964
2022-12-13	Null	0.1469272
2022-12-13	Logit	0.0208969
2022-12-14	XGB	0.1913760
2022-12-14	GAM	0.1140230
2022-12-14	RF	0.0460000
2022-12-14	Step	0.0061201
2022-12-14	Lasso	0.0308781
2022-12-14	MARS	0.0451040
2022-12-14	Null	0.1469272
2022-12-14	Logit	0.0164134
2022-12-15	XGB	0.1913760
2022-12-15	GAM	0.1138138
2022-12-15	RF	0.0500000
2022-12-15	Step	0.0062612
2022-12-15	Lasso	0.0312013
2022-12-15	MARS	0.0502520
2022-12-15	Null	0.1469272
2022-12-15	Logit	0.0161320

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

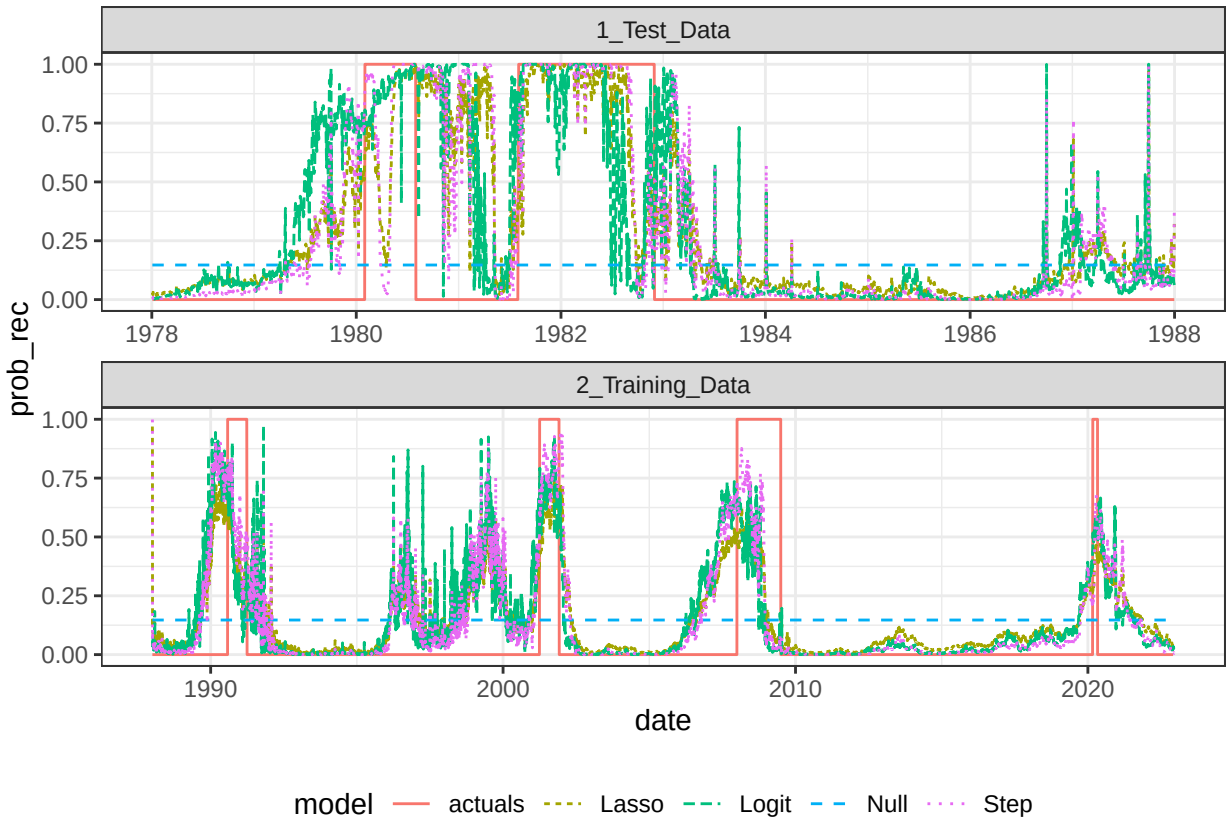


Backtesting

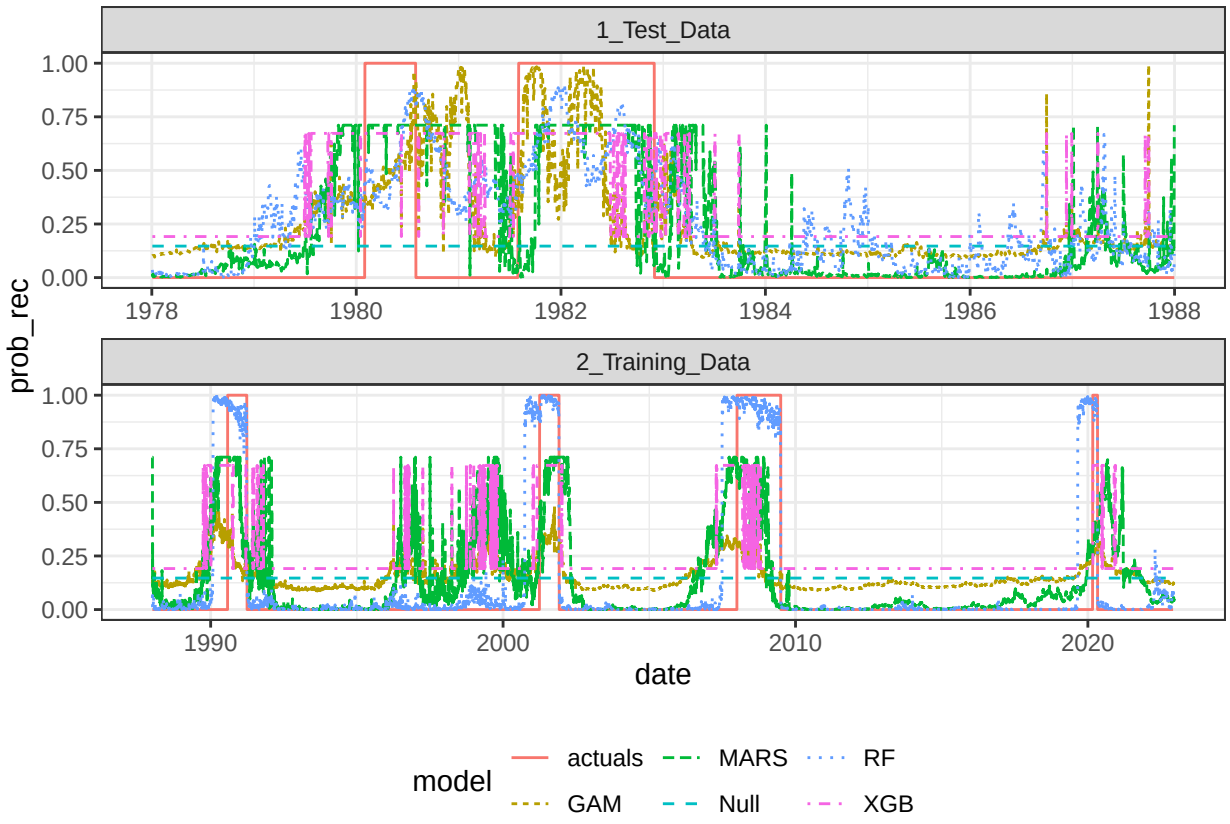
```
full_data_bktst <- full_data_wide %>%
  filter(date >= startTestDate)

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)
```




```
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)
```