Probability of Recession

William Chiu

2022-12-10

Summary

Forecast the probability of a recession in the next 6 months using the spread between 10Y CMT and Effective Federal Funds Rate and its transformations:

1. Lags
2. Adstock
3. Moving average

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")

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")
  )
})
```

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)
```

Calculate Features/Predictors

```
full_data_wide_features <- full_data_wide_raw %>%
  arrange(date) %>%
  mutate(SPRD_10YCMT_FEDFUNDS = GS10 - FEDFUNDS,
         D_SPRD = SPRD_10YCMT_FEDFUNDS -
           lag(SPRD_10YCMT_FEDFUNDS, 1)
         ) %>%
  mutate(across(
    .cols=c(SPRD_10YCMT_FEDFUNDS, D_SPRD),
    .fns=list(lag1 = ~lag(.x, 1),
         lag3 = ~lag(.x, 3),
         lag6 = ~lag(.x, 6),
         lag9 = ~lag(.x, 9),
         lag12 = ~lag(.x, 12))
  ))  %>%
  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(adstk85 = ~stats::filter(.x,
                                filter=0.85,
                                method="recursive") ,
         adstk91 = ~stats::filter(.x,
```

```
                                    filter=0.91,
                                    method="recursive") ,
        adstk92 = ~stats::filter(.x,
                                    filter=0.92,
                                    method="recursive"),
        adstk93 = ~stats::filter(.x,
                                    filter=0.93,
                                    method="recursive"),
        adstk94 = ~stats::filter(.x,
                                    filter=0.94,
                                    method="recursive"),
        adstk95 = ~stats::filter(.x,
                                    filter=0.95,
                                    method="recursive"),
        adstk99 = ~stats::filter(.x,
                                    filter=0.99,
                                    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(
        ma2m = ~stats::filter(.x,
                                    filter=ma_fun(2),
                                  method="convolution",
                                  sides=1),
        ma3m = ~stats::filter(.x,
                                    filter=ma_fun(3),
                                  method="convolution",
                                  sides=1),
        ma6m = ~stats::filter(.x,
                                    filter=ma_fun(6),
                                  method="convolution",
                                  sides=1),
        ma9m = ~stats::filter(.x,
                                    filter=ma_fun(9),
                                  method="convolution",
                                  sides=1),
        ma12m = ~stats::filter(.x,
                                    filter=ma_fun(12),
                                  method="convolution",
                                  sides=1)
  )))
```

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_features_adstock, 12)

train_test <- head(full_data_wide_features_adstock, -12) %>%
  drop_na()
```

Recession in next 6 months

```
full_data_wide <- train_test %>%
  arrange(date) %>%
  mutate(USREC_LEAD1 = lead(USREC, 1),
         USREC_LEAD2 = lead(USREC, 2),
         USREC_LEAD3 = lead(USREC, 3),
         USREC_LEAD4 = lead(USREC, 4),
         USREC_LEAD5 = lead(USREC, 5),
         USREC_LEAD6 = lead(USREC, 6),
         FUTREC = pmax(USREC_LEAD1, USREC_LEAD2, USREC_LEAD3,
                              USREC_LEAD4, USREC_LEAD5, USREC_LEAD6)) %>%
  drop_na() %>%
  select( -USREC_LEAD1, -USREC_LEAD2, -USREC_LEAD3,
         -USREC_LEAD4, -USREC_LEAD5, -USREC_LEAD6)
```

Split Train/Test

```
full_data_wide$constant <- 1

train_data <- full_data_wide %>%
  filter(date >= startTrainDate)

test_data <- full_data_wide %>%
  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 = 401
date	1988-01-01 to 2021-05-01
USREC	36 (9.0%)
GS10	4.50 (2.65, 6.22)
FEDFUNDS	2.63 (0.36, 5.29)
SPRD_10YCMT_FEDFUNDS	1.49 (0.53, 2.56)
D_SPRD	-0.02 (-0.16, 0.14)
SPRD_10YCMT_FEDFUNDS_lag1	1.49 (0.53, 2.56)
SPRD_10YCMT_FEDFUNDS_lag3	1.49 (0.53, 2.56)
SPRD_10YCMT_FEDFUNDS_lag6	1.53 (0.53, 2.56)
SPRD_10YCMT_FEDFUNDS_lag9	1.55 (0.53, 2.56)
SPRD_10YCMT_FEDFUNDS_lag12	1.55 (0.53, 2.56)
D_SPRD_lag1	-0.02 (-0.16, 0.14)
D_SPRD_lag3	-0.02 (-0.16, 0.14)
D_SPRD_lag6	-0.02 (-0.16, 0.14)
D_SPRD_lag9	-0.02 (-0.16, 0.15)
D_SPRD_lag12	-0.02 (-0.16, 0.15)
SPRD_10YCMT_FEDFUNDS_adstk85	10 (4, 17)
SPRD_10YCMT_FEDFUNDS_adstk91	17 (7, 26)
SPRD_10YCMT_FEDFUNDS_adstk92	19 (8, 29)
SPRD_10YCMT_FEDFUNDS_adstk93	22 (9, 32)
SPRD_10YCMT_FEDFUNDS_adstk94	26 (11, 37)
SPRD_10YCMT_FEDFUNDS_adstk95	31 (15, 44)
SPRD_10YCMT_FEDFUNDS_adstk99	137 (111, 161)
D_SPRD_adstk85	-0.06 (-0.47, 0.42)
D_SPRD_adstk91	-0.09 (-0.62, 0.48)
D_SPRD_adstk92	-0.10 (-0.65, 0.48)
D_SPRD_adstk93	-0.10 (-0.68, 0.51)
D_SPRD_adstk94	-0.10 (-0.69, 0.58)
D_SPRD_adstk95	-0.14 (-0.72, 0.59)
D_SPRD_adstk99	0.00 (-0.80, 1.20)
constant	401 (100%)
SPRD_10YCMT_FEDFUNDS_ma2m	1.49 (0.48, 2.54)
SPRD_10YCMT_FEDFUNDS_ma3m	1.49 (0.49, 2.56)
SPRD_10YCMT_FEDFUNDS_ma6m	1.51 (0.50, 2.59)
SPRD_10YCMT_FEDFUNDS_ma9m	1.50 (0.49, 2.62)
SPRD_10YCMT_FEDFUNDS_ma12m	1.49 (0.50, 2.60)
D_SPRD_ma2m	-0.03 (-0.12, 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, 0.07)
D_SPRD_ma12m	-0.01 (-0.08, 0.08)
FUTREC	56 (14%)

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.
```

```
## "1978-01-01" "1980-06-23" "1982-12-16" "1982-12-16" "1985-06-08" "1987-12-01"
```

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

summary(test_data$date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "1978-01-01" "1980-06-23" "1982-12-16" "1982-12-16" "1985-06-08" "1987-12-01"
```

Setup Parallel Processing

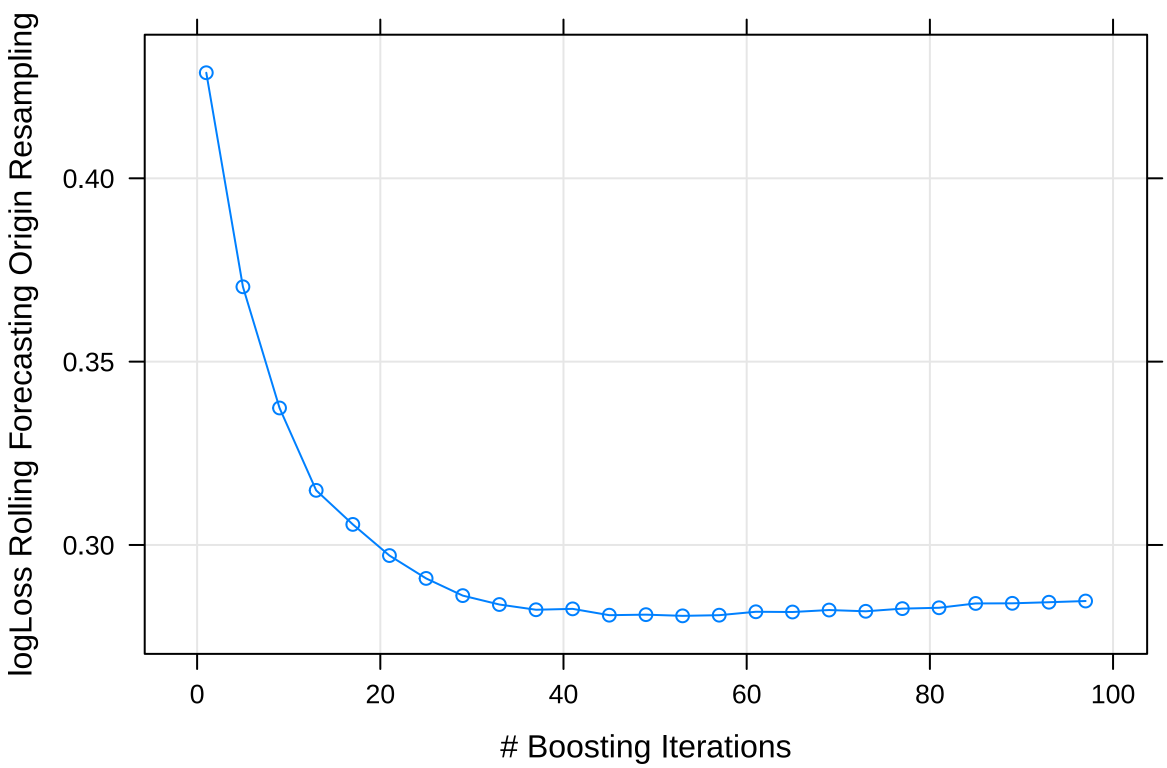
```
library(doParallel)

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

Cross-Validation Framework

```
fcstHorizon <- 6
initWindow <- 120
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 - USREC - constant,
  data = train_yes_no,
  method = "gamboost",
  trControl = fitControl_oneSE,
  metric = "logLoss",
  tuneGrid = grid_gam,
  family = Binomial(),
  dfbase =3
)

plot(gam_mod)
```

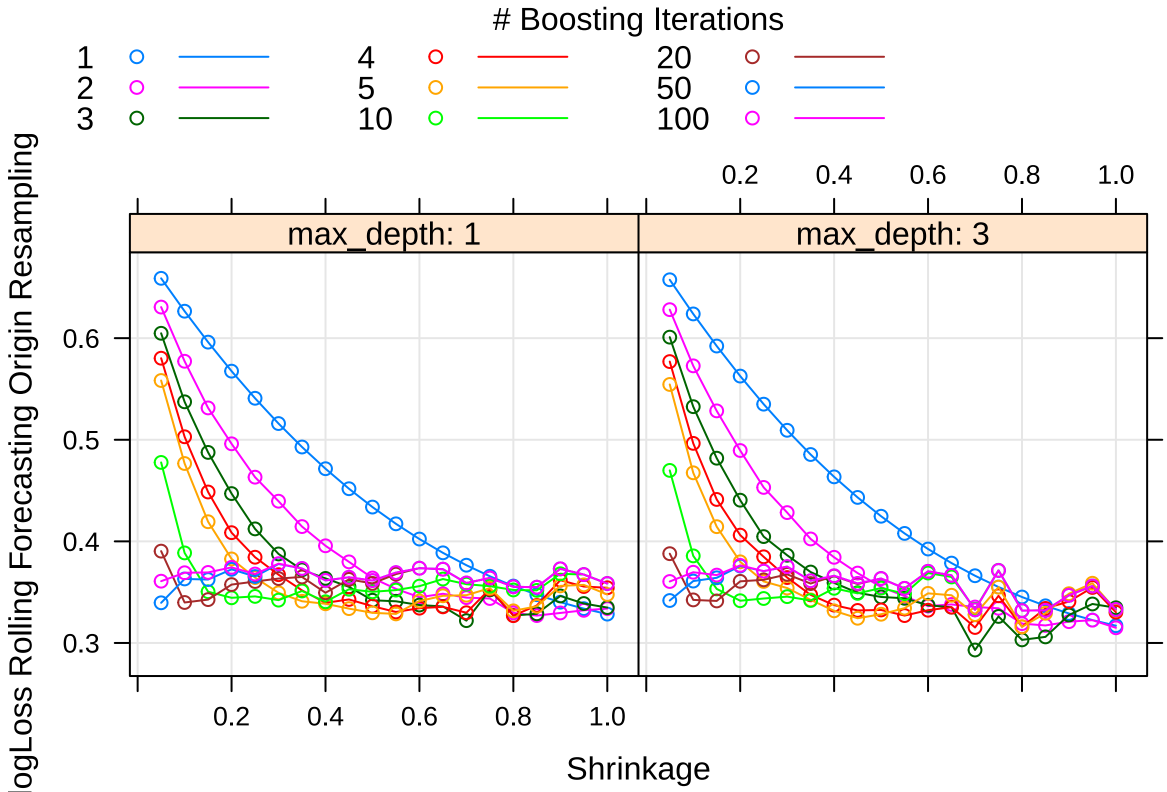



```
gam_mod$bestTune
```

```
##   mstop prune
## 3     9    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 - USREC - constant,
  data = train_yes_no,
  method = "xgbTree",
  trControl = fitControl_oneSE,
  metric = "logLoss",
  tuneGrid = grid_xgb,
  objective  = "binary:logistic"
)

plot(xgb_mod)
```

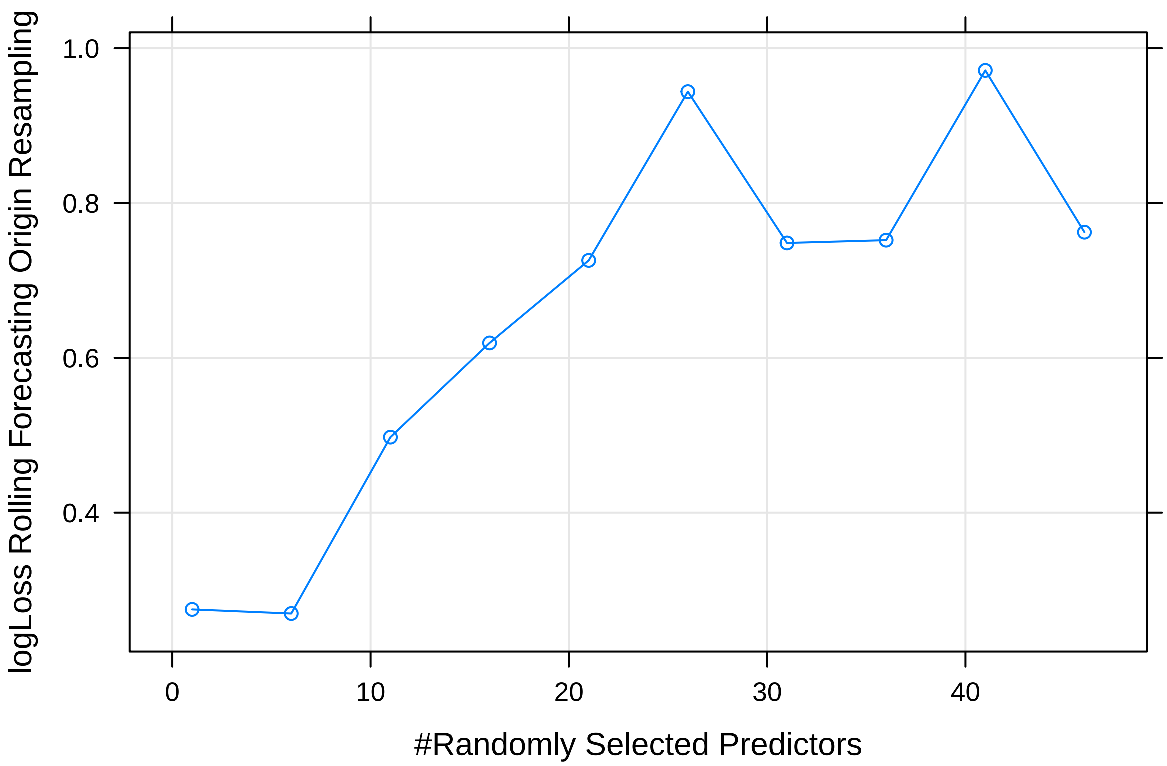


```
xgb_mod$bestTune
```

```
##     nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
## 253       1         1 0.75     0                1               10         1
```

Random Forest

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

set.seed(randSeed)

rf_mod <- train(
  FUTREC ~ . - date - USREC - constant,
  data = train_yes_no,
  method = "rf",
  trControl = fitControl_oneSE,
  metric = "logLoss",
  tuneGrid = grid_rf,
  importance = TRUE
)

plot(rf_mod)
```



```r
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.

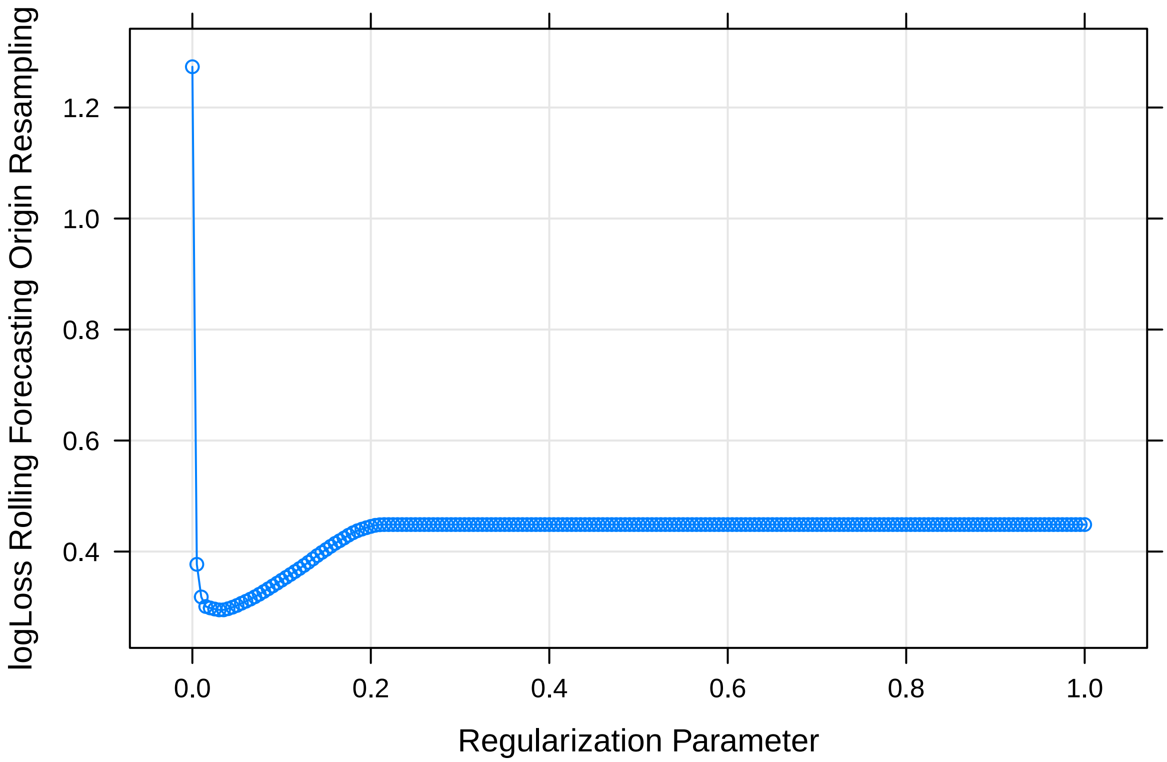
However, for most methods (that do not invoke `glm()`) in `train`, the first level denotes the success (the opposite of `glm()`). This behavior causes the coefficient signs to flip. Be highly suspicious when interpreting coefficients from models that are fit using `train`.

```r
set.seed(randSeed)

stepwise_mod <- train(
  FUTREC ~ . - date - USREC - 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)

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

set.seed(randSeed)

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

```r
  family = "binomial"
)

plot(glmnet_mod)
```



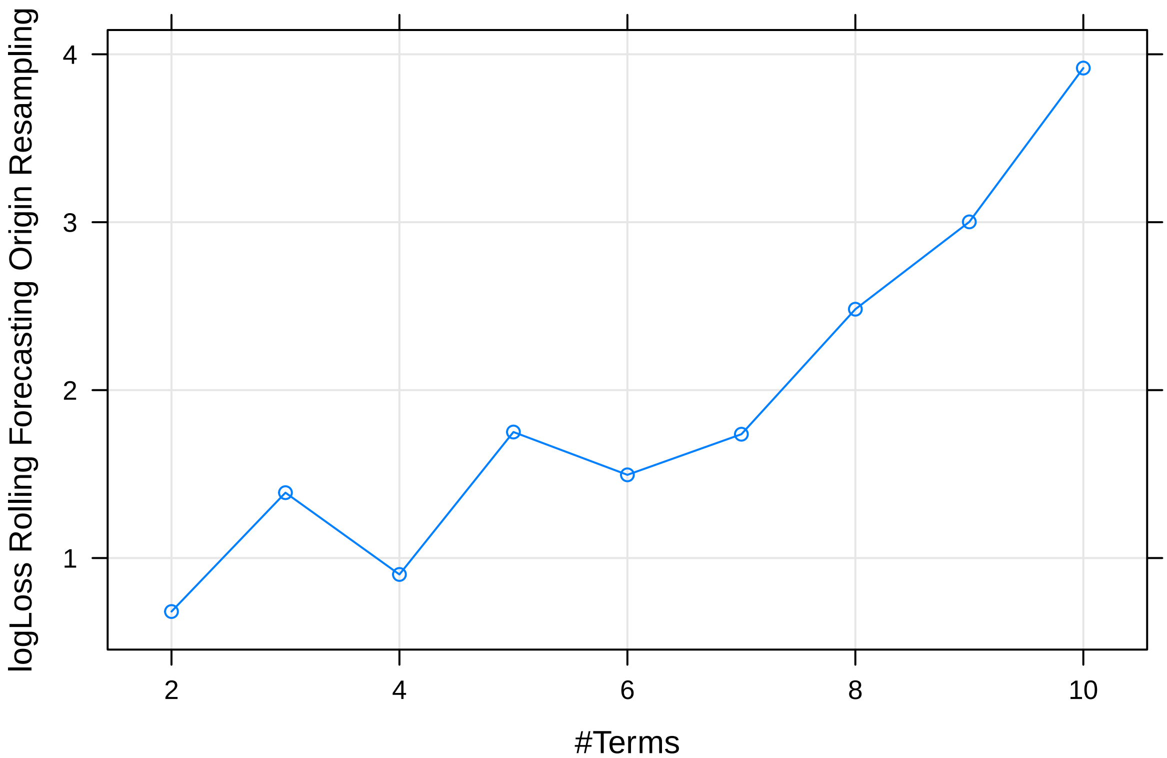

```r
glmnet_mod$bestTune
```

```
##   alpha lambda
## 7     1   0.03
```

Multivariate Adaptive Regression Splines

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

set.seed(randSeed)

earth_mod <- train(
  FUTREC ~ . - date - USREC - 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_best,
  metric = "logLoss",
```

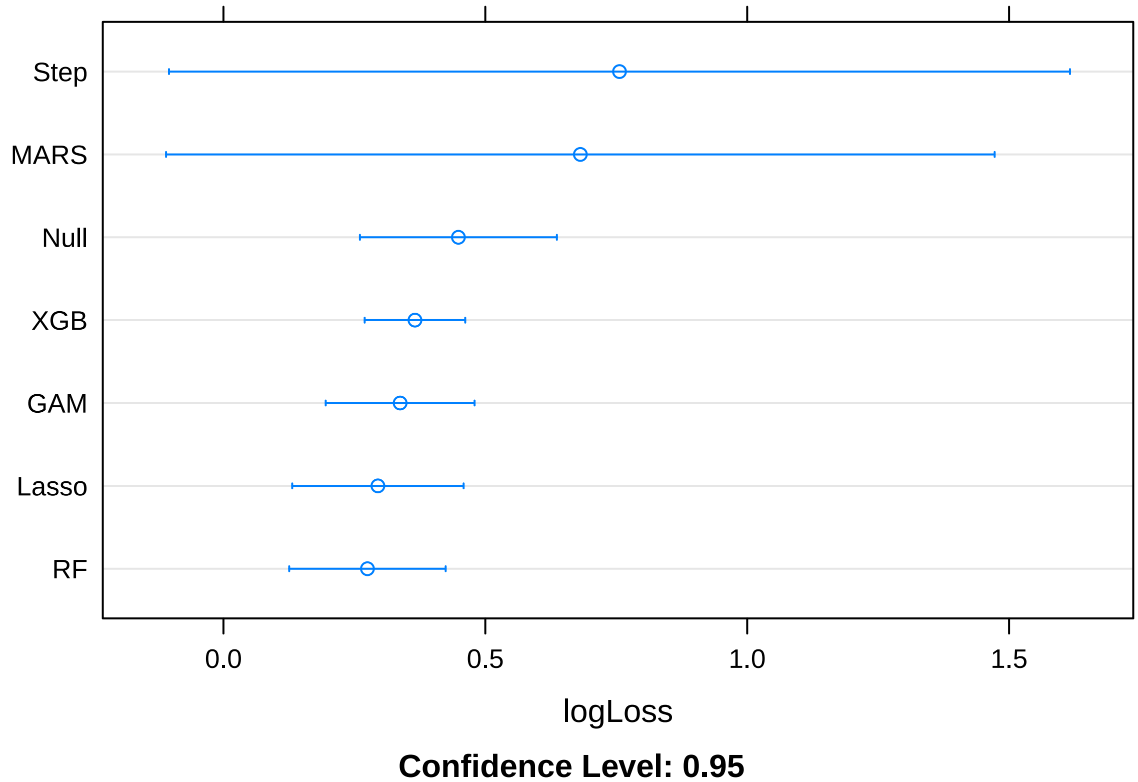
```r
  family = binomial
)
```

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: 46
##
## logLoss
##               Min.      1st Qu.       Median      Mean    3rd Qu.      Max.
## XGB   2.073761e-01 2.133103e-01 0.2322580636 0.3656524 0.24149644  1.654720
## GAM   7.048603e-02 8.713700e-02 0.0989558601 0.3373635 0.22842154  1.662805
## RF    2.003341e-03 9.816079e-03 0.0330654175 0.2749450 0.26223426  2.075257
## Step  9.992007e-16 3.654836e-12 0.0005376345 0.7561703 0.01570571 17.269388
## Lasso 2.021251e-04 8.571883e-03 0.0277020576 0.2949257 0.16335390  2.238882
## MARS  9.992007e-16 4.631315e-03 0.0288428833 0.6814256 0.05149754 17.269388
## Null  9.461598e-02 1.418640e-01 0.1612905053 0.4486199 0.19905716  2.277267
##       NA's
## XGB      0
## GAM      0
## RF       0
## Step     0
## Lasso    0
## MARS     0
## Null     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
## 253       1         1 0.75     0                1               10         1
```

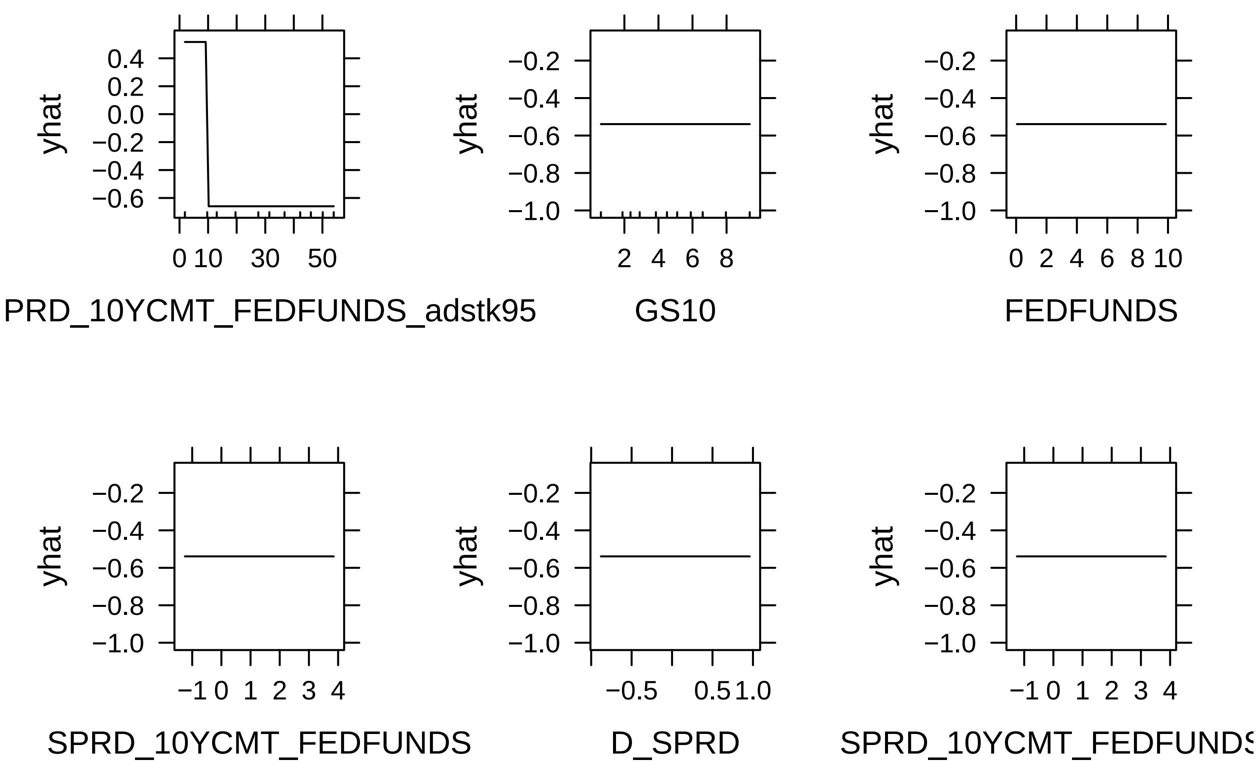
```
df_imp <- varImp(xgb_mod)$importance %>%
  arrange(desc(Overall))

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_adstk95	100

variable	Overall
GS10	0
FEDFUNDS	0
SPRD_10YCMT_FEDFUNDS	0
D_SPRD	0
SPRD_10YCMT_FEDFUNDS_lag1	0
SPRD_10YCMT_FEDFUNDS_lag3	0
SPRD_10YCMT_FEDFUNDS_lag6	0
SPRD_10YCMT_FEDFUNDS_lag9	0
SPRD_10YCMT_FEDFUNDS_lag12	0
D_SPRD_lag1	0
D_SPRD_lag3	0
D_SPRD_lag6	0
D_SPRD_lag9	0
D_SPRD_lag12	0
SPRD_10YCMT_FEDFUNDS_adstk85	0
SPRD_10YCMT_FEDFUNDS_adstk91	0
SPRD_10YCMT_FEDFUNDS_adstk92	0
SPRD_10YCMT_FEDFUNDS_adstk93	0
SPRD_10YCMT_FEDFUNDS_adstk94	0
SPRD_10YCMT_FEDFUNDS_adstk99	0
D_SPRD_adstk85	0
D_SPRD_adstk91	0
D_SPRD_adstk92	0
D_SPRD_adstk93	0
D_SPRD_adstk94	0
D_SPRD_adstk95	0
D_SPRD_adstk99	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_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
## -2.86620   0.01589   0.05037   0.21584   1.44313
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -2.35835    0.47437  -4.972 6.64e-07 ***
## SPRD_10YCMT_FEDFUNDS_adstk95  0.22847    0.03453   6.617 3.67e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 324.27  on 400  degrees of freedom
```

```
## Residual deviance: 169.90  on 399  degrees of freedom
## AIC: 173.9
##
## Number of Fisher Scoring iterations: 8
```

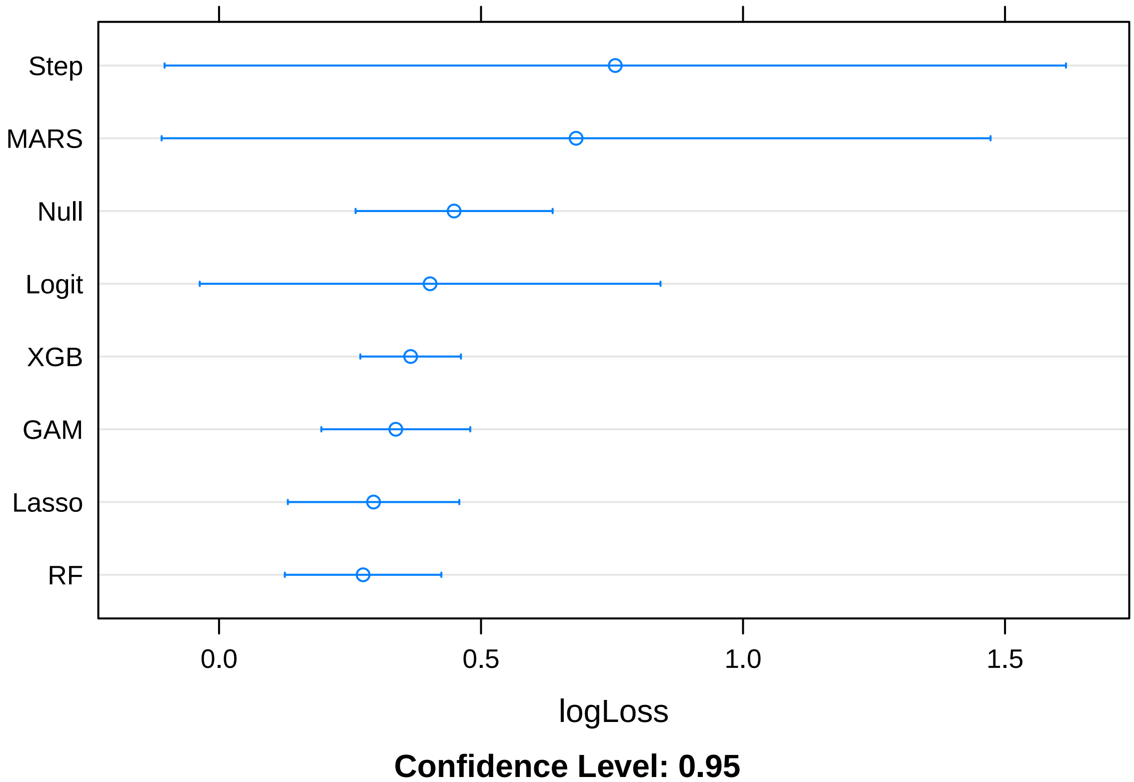
Compare Models

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: 46
##
## logLoss
##               Min.      1st Qu.       Median      Mean    3rd Qu.      Max.
## XGB   2.073761e-01 2.133103e-01 0.2322580636 0.3656524 0.24149644  1.654720
## GAM   7.048603e-02 8.713700e-02 0.0989558601 0.3373635 0.22842154  1.662805
## RF    2.003341e-03 9.816079e-03 0.0330654175 0.2749450 0.26223426  2.075257
## Step  9.992007e-16 3.654836e-12 0.0005376345 0.7561703 0.01570571 17.269388
## Lasso 2.021251e-04 8.571883e-03 0.0277020576 0.2949257 0.16335390  2.238882
## MARS  9.992007e-16 4.631315e-03 0.0288428833 0.6814256 0.05149754 17.269388
## Null  9.461598e-02 1.418640e-01 0.1612905053 0.4486199 0.19905716  2.277267
## Logit 9.992007e-16 1.102789e-04 0.0008839239 0.4028153 0.04970038  8.604311
##       NA's
## XGB      0
## GAM      0
## RF       0
## Step     0
## Lasso    0
## MARS     0
## Null     0
## Logit    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
Step	0.9634233
Logit	0.9634233
GAM	0.9623580
Lasso	0.9477983
MARS	0.9318182
RF	0.9131747
XGB	0.8863636
Null	0.5000000

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

model	LogLoss
MARS	0.2857389
GAM	0.3438593
RF	0.3544149
Lasso	0.3896416
XGB	0.4384522
Null	0.6352684
Step	0.6782045
Logit	0.6782045

Probability of Recession (the 12 most recent months)

```
curr_data <- recent_data

curr_data$date
```

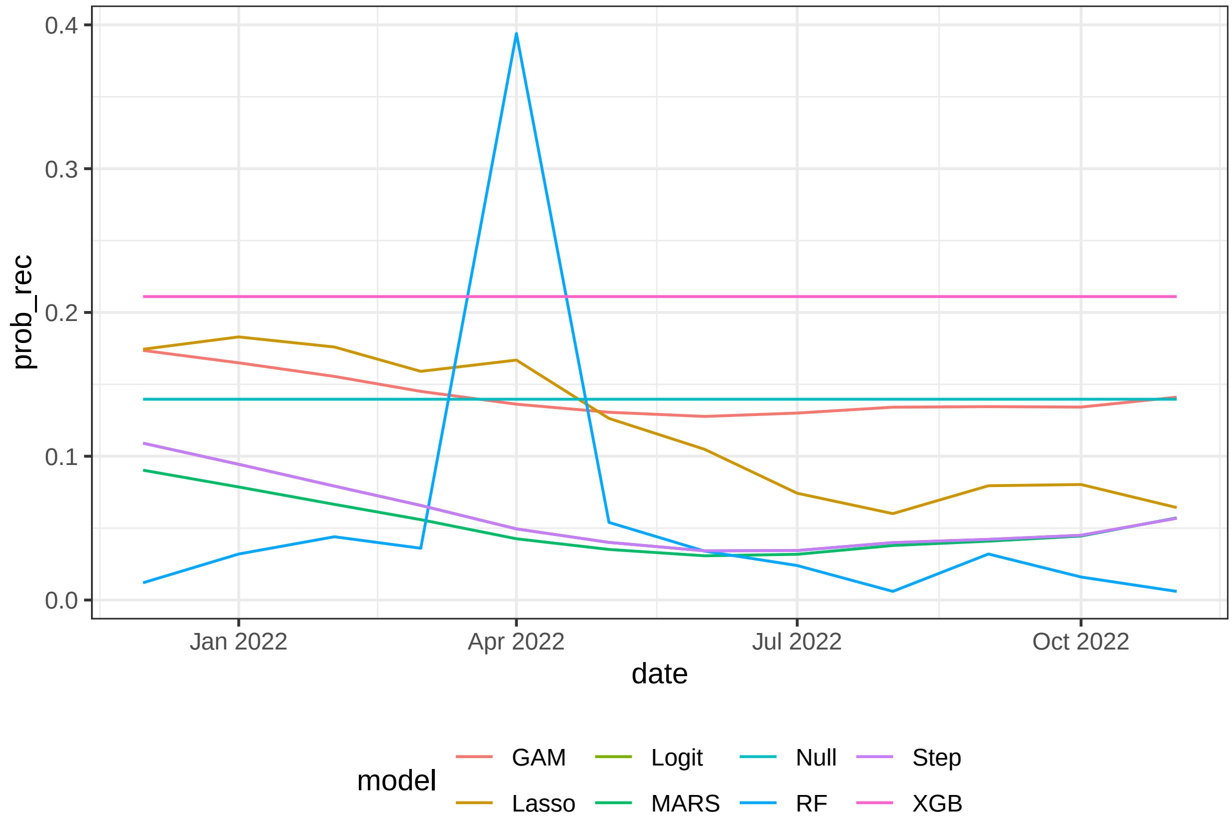
```
##  [1] "2021-12-01" "2022-01-01" "2022-02-01" "2022-03-01" "2022-04-01"
##  [6] "2022-05-01" "2022-06-01" "2022-07-01" "2022-08-01" "2022-09-01"
## [11] "2022-10-01" "2022-11-01"
```

```
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"]
```

```r
    )

  })

  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
2022-10-01	XGB	0.2110378
2022-10-01	GAM	0.1341967
2022-10-01	RF	0.0160000
2022-10-01	Step	0.0450450
2022-10-01	Lasso	0.0803249
2022-10-01	MARS	0.0444954
2022-10-01	Null	0.1396509
2022-10-01	Logit	0.0450450
2022-11-01	XGB	0.2110378
2022-11-01	GAM	0.1410906
2022-11-01	RF	0.0060000
2022-11-01	Step	0.0568657
2022-11-01	Lasso	0.0643143
2022-11-01	MARS	0.0571497
2022-11-01	Null	0.1396509
2022-11-01	Logit	0.0568657

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

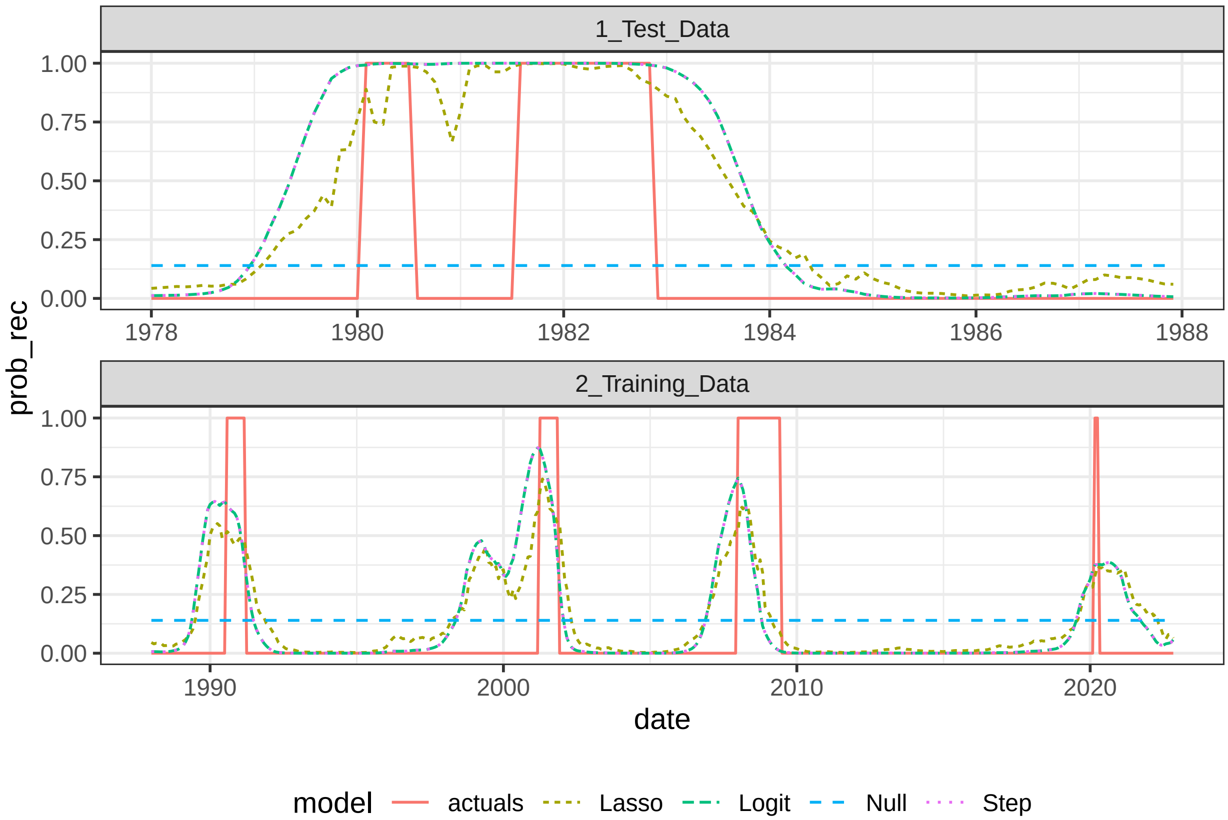


Backtesting

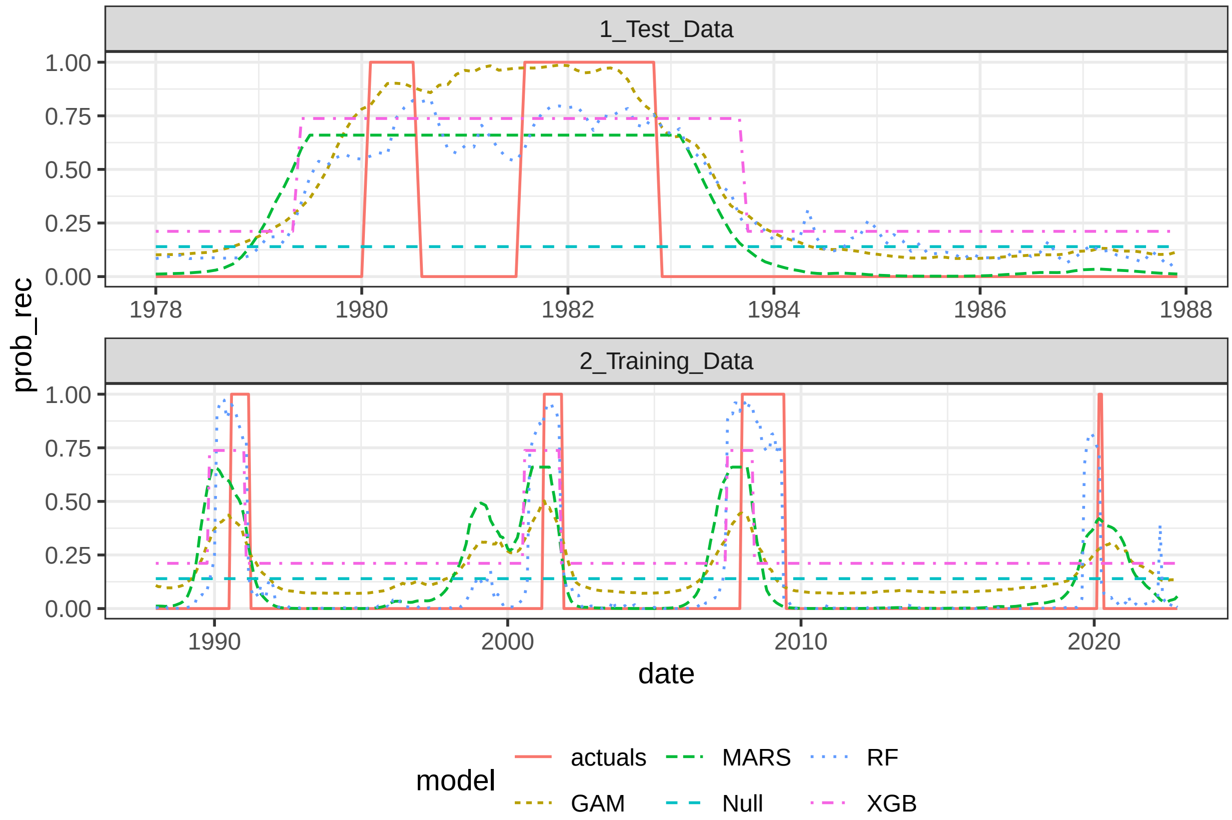
```
full_data_bktst <- full_data_wide_features_adstock %>%
  filter(date >= startTestDate)

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)
```