Probability of Recession

William Chiu

2022-11-29

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

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("2022-12-01")
  )
})

recession_dates <- map_dfr(response_id, function(x) {
  fredr(
    series_id = x,
    observation_start = as.Date("1950-01-01"),
    observation_end = as.Date("2022-12-01")
  )
})
```

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
         ) %>%
  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),
    .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),
    .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) %>%
  drop_na()

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)
```

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 = 8,337**
date	1988-01-04 to 2021-04-28
DFF	2.74 (0.36, 5.29)
DGS10	4.48 (2.65, 6.22)
SPRD_10YCMT_FEDFUNDS	1.48 (0.48, 2.53)
SPRD_10YCMT_FEDFUNDS_lag1m	1.48 (0.48, 2.53)
SPRD_10YCMT_FEDFUNDS_lag3m	1.49 (0.48, 2.54)
SPRD_10YCMT_FEDFUNDS_lag6m	1.52 (0.48, 2.54)
SPRD_10YCMT_FEDFUNDS_lag9m	1.54 (0.48, 2.54)
SPRD_10YCMT_FEDFUNDS_lag12m	1.54 (0.48, 2.54)
SPRD_10YCMT_FEDFUNDS_lag5d	1.48 (0.48, 2.53)
SPRD_10YCMT_FEDFUNDS_lag10d	1.48 (0.48, 2.53)
SPRD_10YCMT_FEDFUNDS_lag15d	1.48 (0.48, 2.53)
SPRD_10YCMT_FEDFUNDS_adstk001	1.48 (0.48, 2.53)
SPRD_10YCMT_FEDFUNDS_adstk10	1.65 (0.53, 2.81)
SPRD_10YCMT_FEDFUNDS_adstk20	1.85 (0.59, 3.16)
SPRD_10YCMT_FEDFUNDS_adstk40	2.47 (0.79, 4.22)
SPRD_10YCMT_FEDFUNDS_adstk75	5.9 (2.0, 10.1)
SPRD_10YCMT_FEDFUNDS_adstk95	30 (10, 51)
constant	8,337 (100%)
SPRD_10YCMT_FEDFUNDS_ma5d	1.48 (0.48, 2.53)
SPRD_10YCMT_FEDFUNDS_ma10d	1.47 (0.50, 2.53)
SPRD_10YCMT_FEDFUNDS_ma15d	1.47 (0.50, 2.53)
SPRD_10YCMT_FEDFUNDS_ma20d	1.47 (0.50, 2.54)
SPRD_10YCMT_FEDFUNDS_ma25d	1.47 (0.50, 2.54)
SPRD_10YCMT_FEDFUNDS_ma2m	1.47 (0.50, 2.56)
SPRD_10YCMT_FEDFUNDS_ma3m	1.47 (0.47, 2.55)
SPRD_10YCMT_FEDFUNDS_ma6m	1.51 (0.47, 2.58)
SPRD_10YCMT_FEDFUNDS_ma9m	1.52 (0.48, 2.61)
SPRD_10YCMT_FEDFUNDS_ma12m	1.49 (0.50, 2.60)
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

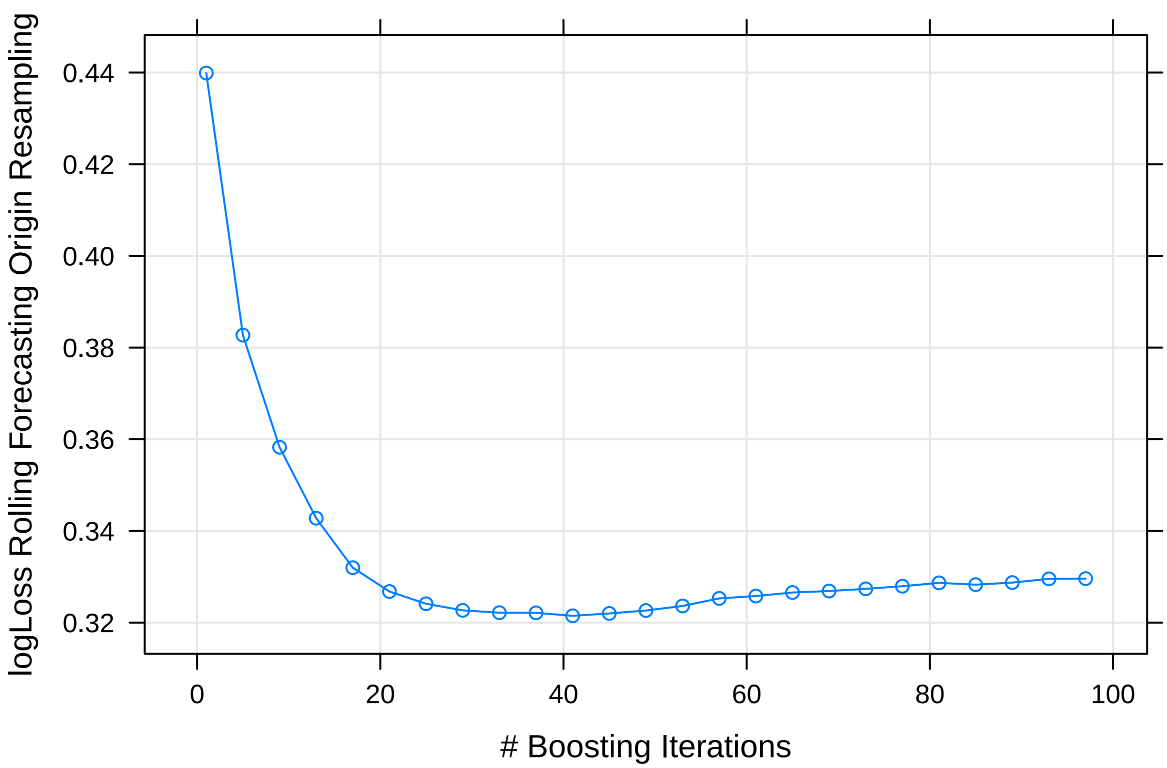
```
library(doParallel)

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

Cross-Validation Framework

```
fcstHorizon <- 3*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()
)

plot(gam_mod)
```

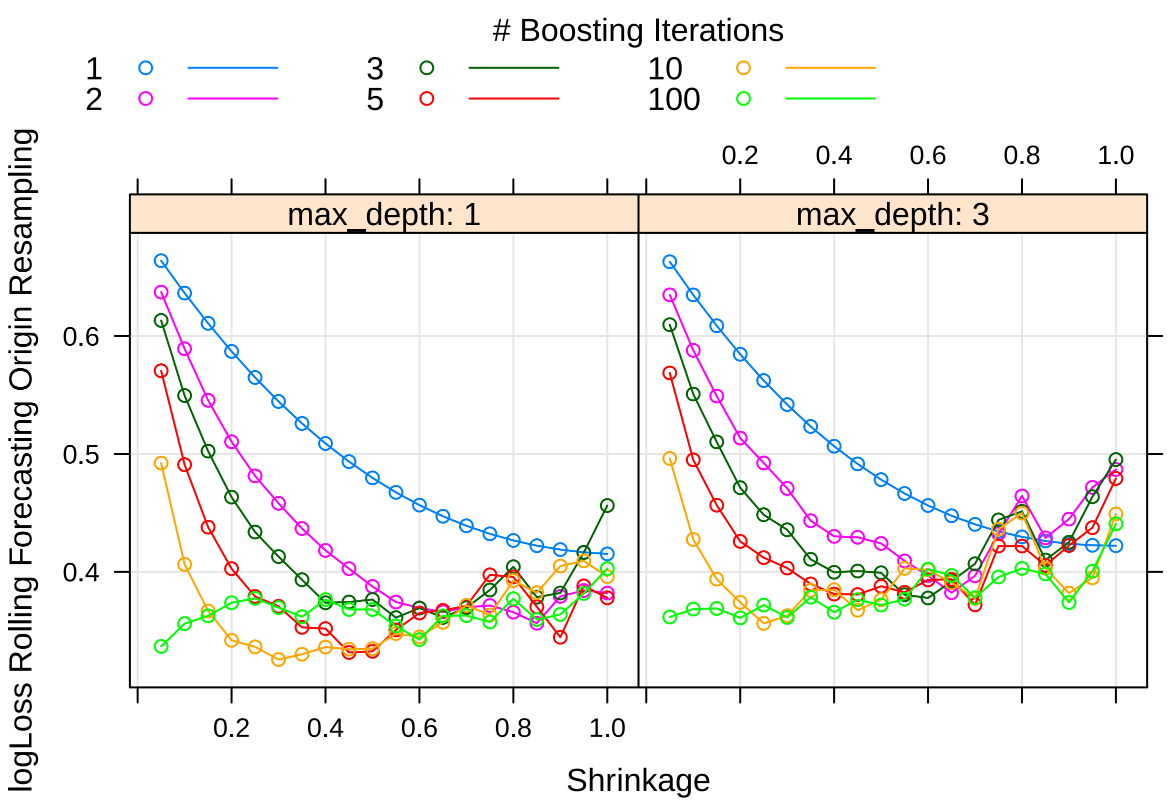



```
gam_mod$bestTune
```

```
##   mstop prune
## 3     9    no
```

eXtreme Gradient Boosting Trees

```
grid_xgb <- expand.grid(nrounds=c(1,2,3,5,10,100),
                        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)
```

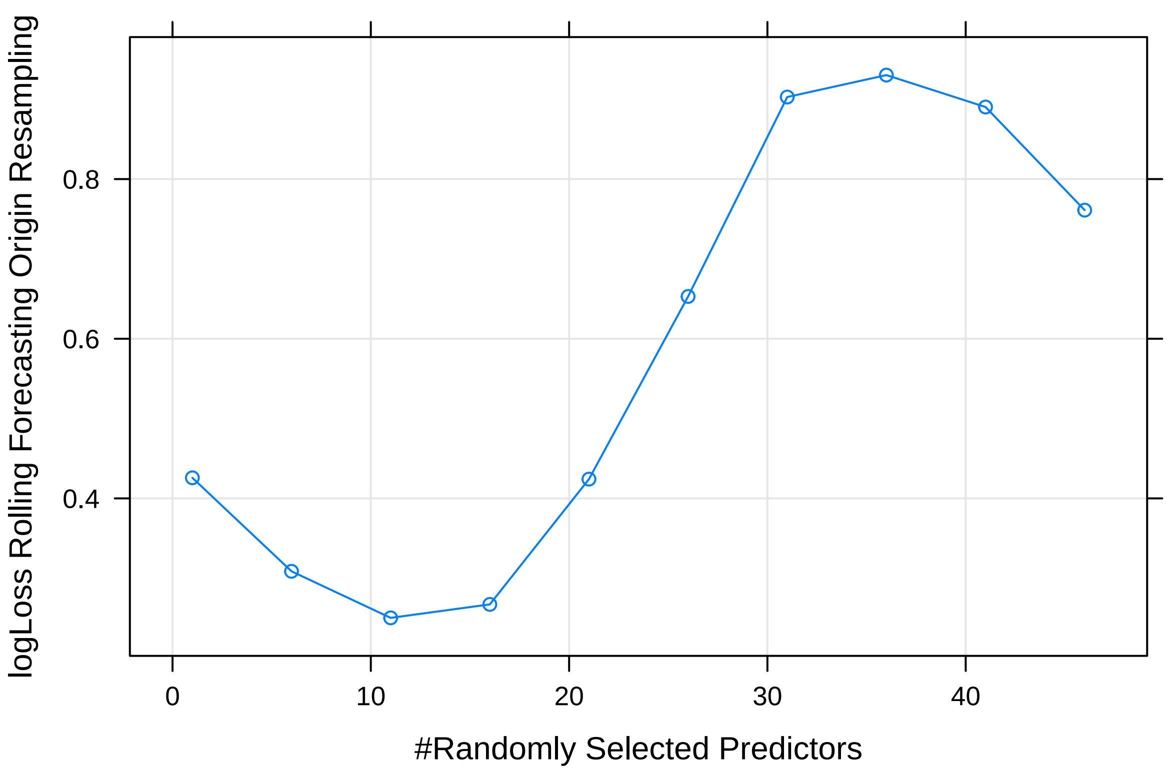


```
xgb_mod$bestTune
```

```
##     nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
## 122       2         1 0.55     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
## 3   11
```

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.

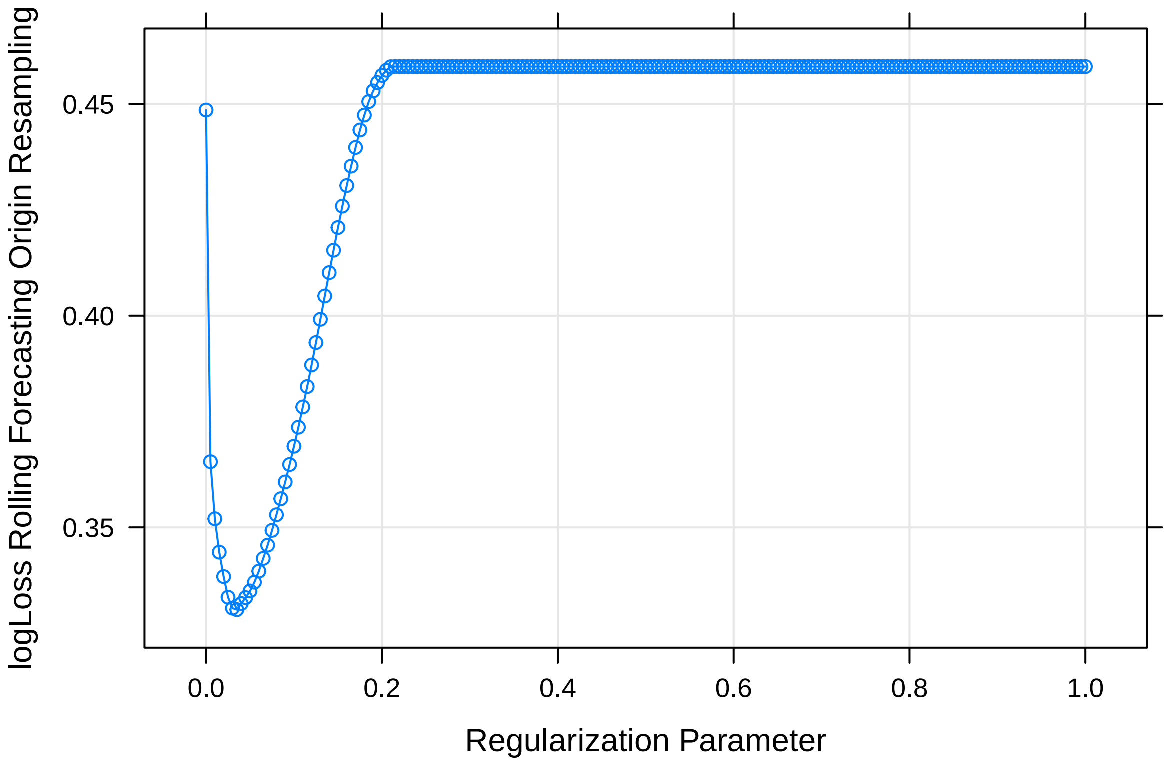
However, for most methods (that do not invoke `glm()`) in `train`, the first level denotes the success (the opposite of `glm()`). This behavior causes the coefficient signs to flip. Be highly suspicious when interpreting coefficients from models that are fit using `train`.

```
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)
```



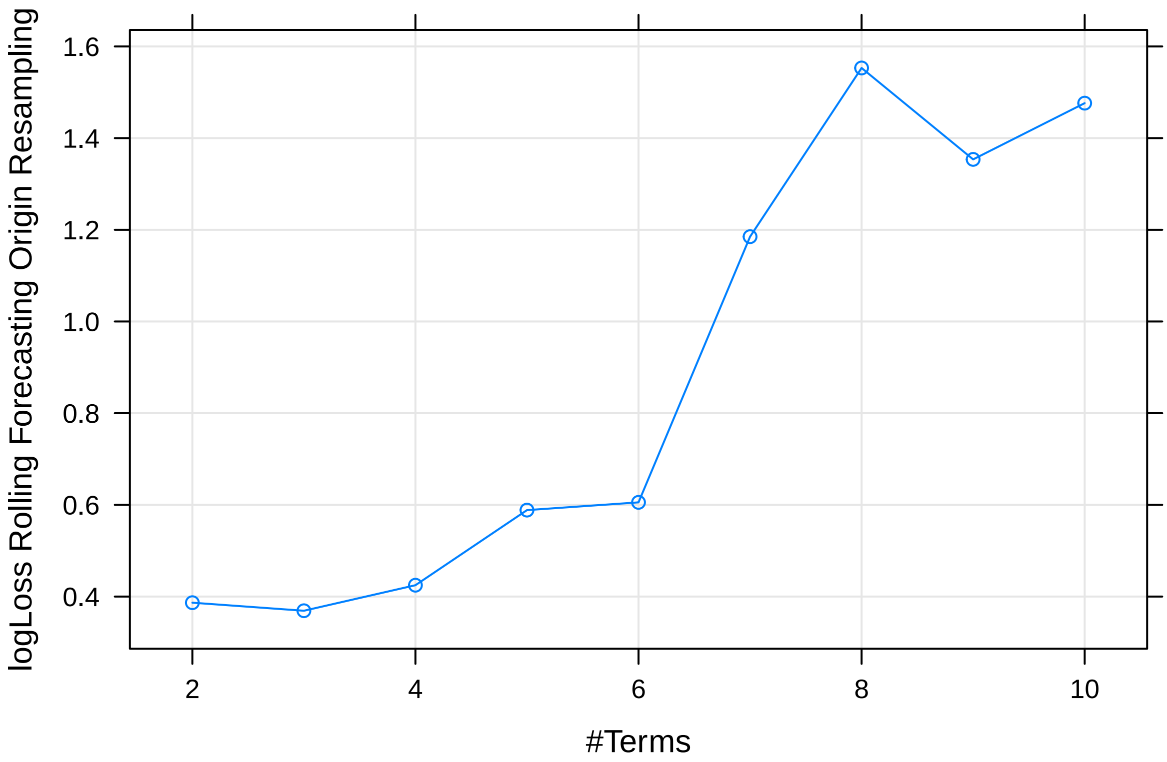

```
glmnet_mod$bestTune
```

```
##   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",
```

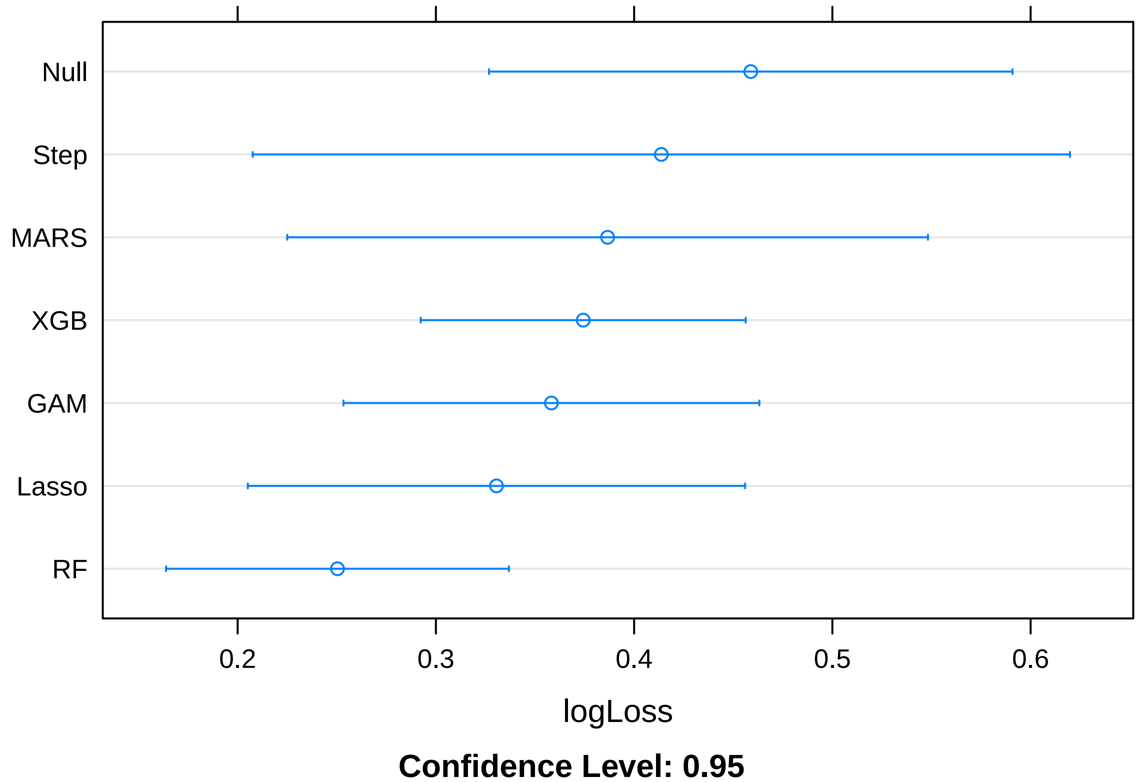
```r
  family = binomial
)
```

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: 92
##
## logLoss
##               Min.     1st Qu.     Median      Mean   3rd Qu.     Max. NA's
## XGB   1.619666e-01 0.183081639 0.19301715 0.3743187 0.3201883 1.885794    0
## GAM   7.922333e-02 0.099555336 0.12276169 0.3582712 0.2510666 2.394825    0
## RF    9.992007e-16 0.003909180 0.04892287 0.2503518 0.3065812 2.141862    0
## Step  1.962163e-06 0.003258413 0.03475042 0.4137270 0.2369905 6.402016    0
## Lasso 3.058111e-03 0.026365731 0.07397356 0.3305668 0.3175471 3.436418    0
## MARS  5.439199e-04 0.054539682 0.06656702 0.3866177 0.3742715 4.152559    0
## Null  9.829487e-02 0.151124175 0.17038163 0.4588389 0.2119709 2.331957    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
## 122       2         1 0.55     0                1               10         1
```

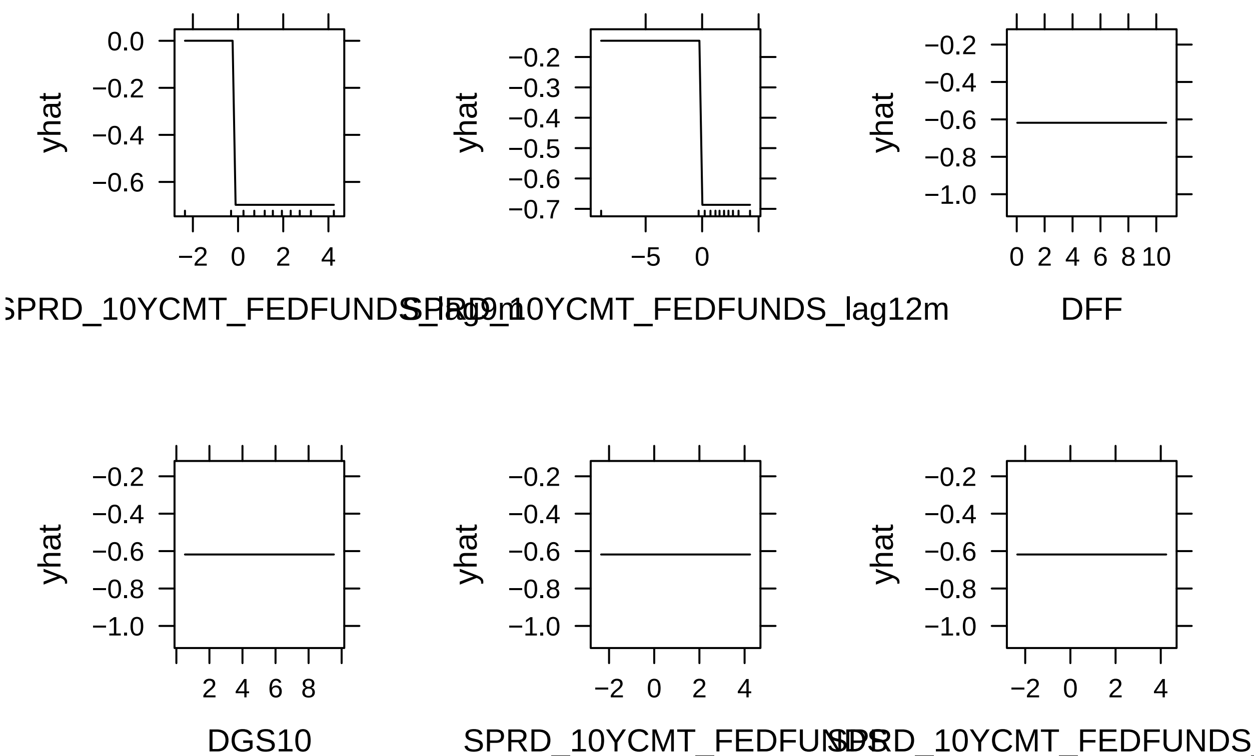
```
df_imp <- varImp(xgb_mod)$importance %>%
  arrange(desc(Overall))

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	100.00000

variable	Overall
SPRD_10YCMT_FEDFUNDS_lag12m	58.33536
DFF	0.00000
DGS10	0.00000
SPRD_10YCMT_FEDFUNDS	0.00000
SPRD_10YCMT_FEDFUNDS_lag1m	0.00000
SPRD_10YCMT_FEDFUNDS_lag3m	0.00000
SPRD_10YCMT_FEDFUNDS_lag6m	0.00000
SPRD_10YCMT_FEDFUNDS_lag5d	0.00000
SPRD_10YCMT_FEDFUNDS_lag10d	0.00000
SPRD_10YCMT_FEDFUNDS_lag15d	0.00000
SPRD_10YCMT_FEDFUNDS_adstk001	0.00000
SPRD_10YCMT_FEDFUNDS_adstk10	0.00000
SPRD_10YCMT_FEDFUNDS_adstk20	0.00000
SPRD_10YCMT_FEDFUNDS_adstk40	0.00000
SPRD_10YCMT_FEDFUNDS_adstk75	0.00000
SPRD_10YCMT_FEDFUNDS_adstk95	0.00000
SPRD_10YCMT_FEDFUNDS_ma5d	0.00000
SPRD_10YCMT_FEDFUNDS_ma10d	0.00000
SPRD_10YCMT_FEDFUNDS_ma15d	0.00000
SPRD_10YCMT_FEDFUNDS_ma20d	0.00000
SPRD_10YCMT_FEDFUNDS_ma25d	0.00000
SPRD_10YCMT_FEDFUNDS_ma2m	0.00000
SPRD_10YCMT_FEDFUNDS_ma3m	0.00000
SPRD_10YCMT_FEDFUNDS_ma6m	0.00000
SPRD_10YCMT_FEDFUNDS_ma9m	0.00000
SPRD_10YCMT_FEDFUNDS_ma12m	0.00000

```r
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 ~",
```

```r
                              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
## -3.15624   0.06926   0.18068   0.42219   2.70313
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 0.44798    0.04025   11.13   <2e-16 ***
## SPRD_10YCMT_FEDFUNDS_lag9m  1.73413    0.04799   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: 7039.4  on 8336  degrees of freedom
## Residual deviance: 4443.6  on 8335  degrees of freedom
## AIC: 4447.6
##
## Number of Fisher Scoring iterations: 6
```

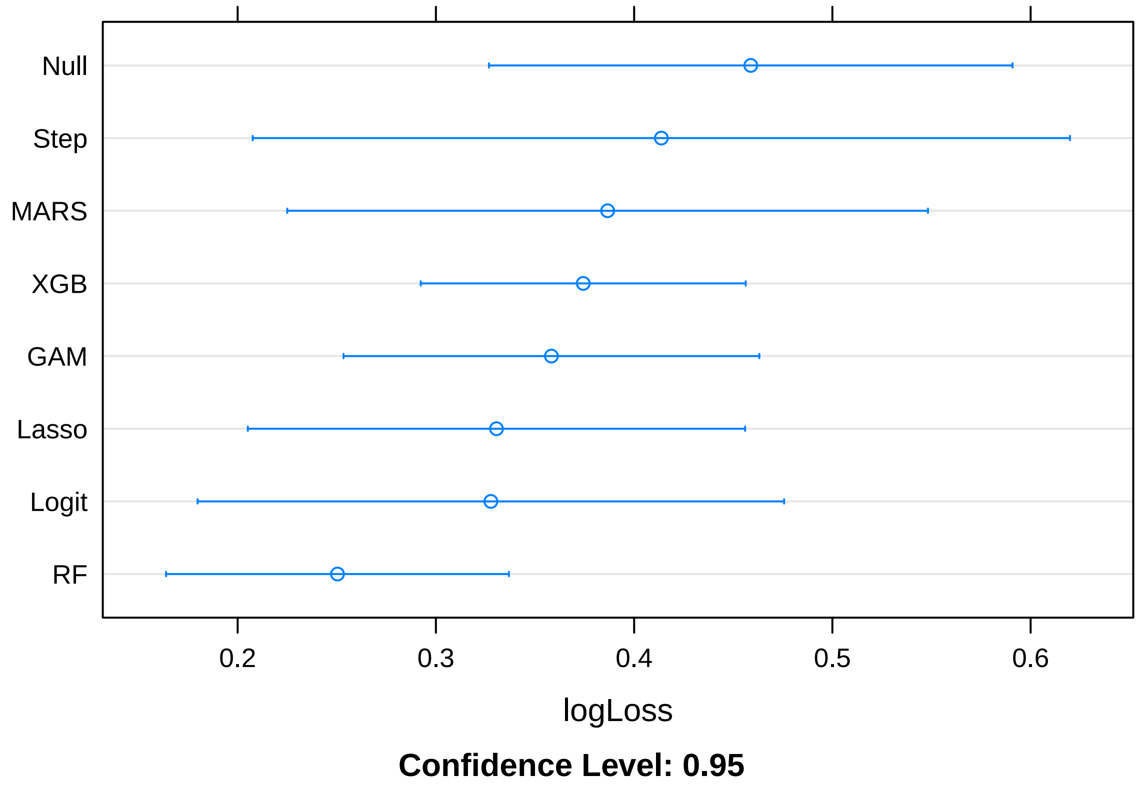
Compare Models

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: 92
##
## logLoss
##               Min.     1st Qu.     Median      Mean   3rd Qu.     Max. NA's
## XGB   1.619666e-01 0.183081639 0.19301715 0.3743187 0.3201883 1.885794    0
## GAM   7.922333e-02 0.099555336 0.12276169 0.3582712 0.2510666 2.394825    0
## RF    9.992007e-16 0.003909180 0.04892287 0.2503518 0.3065812 2.141862    0
## Step  1.962163e-06 0.003258413 0.03475042 0.4137270 0.2369905 6.402016    0
## Lasso 3.058111e-03 0.026365731 0.07397356 0.3305668 0.3175471 3.436418    0
## MARS  5.439199e-04 0.054539682 0.06656702 0.3866177 0.3742715 4.152559    0
## Null  9.829487e-02 0.151124175 0.17038163 0.4588389 0.2119709 2.331957    0
## Logit 2.979105e-04 0.010425233 0.04816905 0.3277418 0.3352069 4.530158    0
```

```r
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])
    }) %>%
```

```r
      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	0.9230540
Lasso	0.9229857
GAM	0.9187499
XGB	0.9009927
RF	0.8927922
Logit	0.8677504
MARS	0.8396945
Null	0.5000000

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

model	LogLoss
Lasso	0.3804360
XGB	0.3884594
GAM	0.4279373
RF	0.4409169
Step	0.4693547
Logit	0.5906803
Null	0.6537635
MARS	0.8911106

Probability of Recession (Most Recent Trading Day)

```r
curr_data <- tail(full_data_wide_features_adstock, 1)

curr_data$date
```

```
## [1] "2022-11-28"
```

```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"]

    )
```

```
  }) %>%
    pivot_longer(everything(), names_to = "model",
                 values_to = "prob_rec")

  output$prob_rec <- scales::percent(output$prob_rec)

  return(output)
}


knitr::kable(score_fun(mymods, curr_data))
```

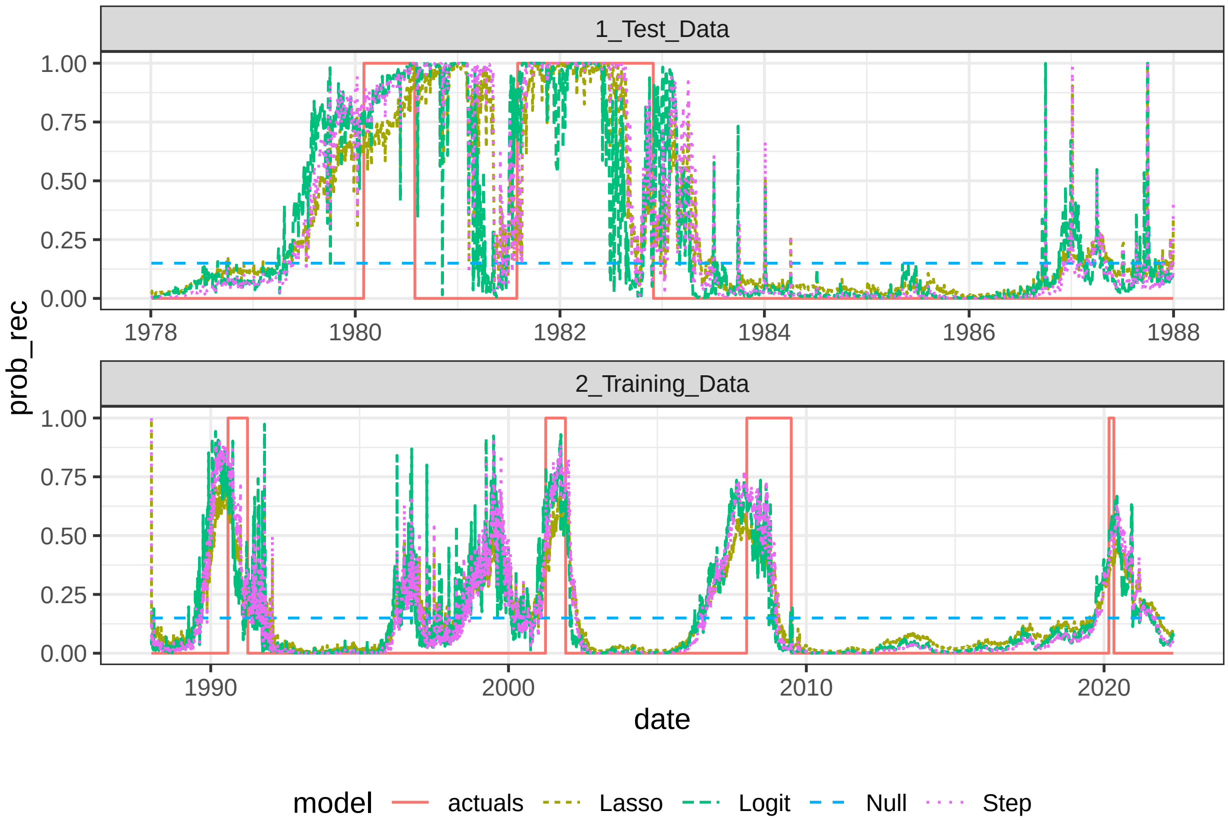
model	prob_rec
XGB	17.771%
GAM	10.131%
RF	0.000%
Step	2.327%
Lasso	6.142%
MARS	5.236%
Null	14.969%
Logit	2.394%

Backtesting

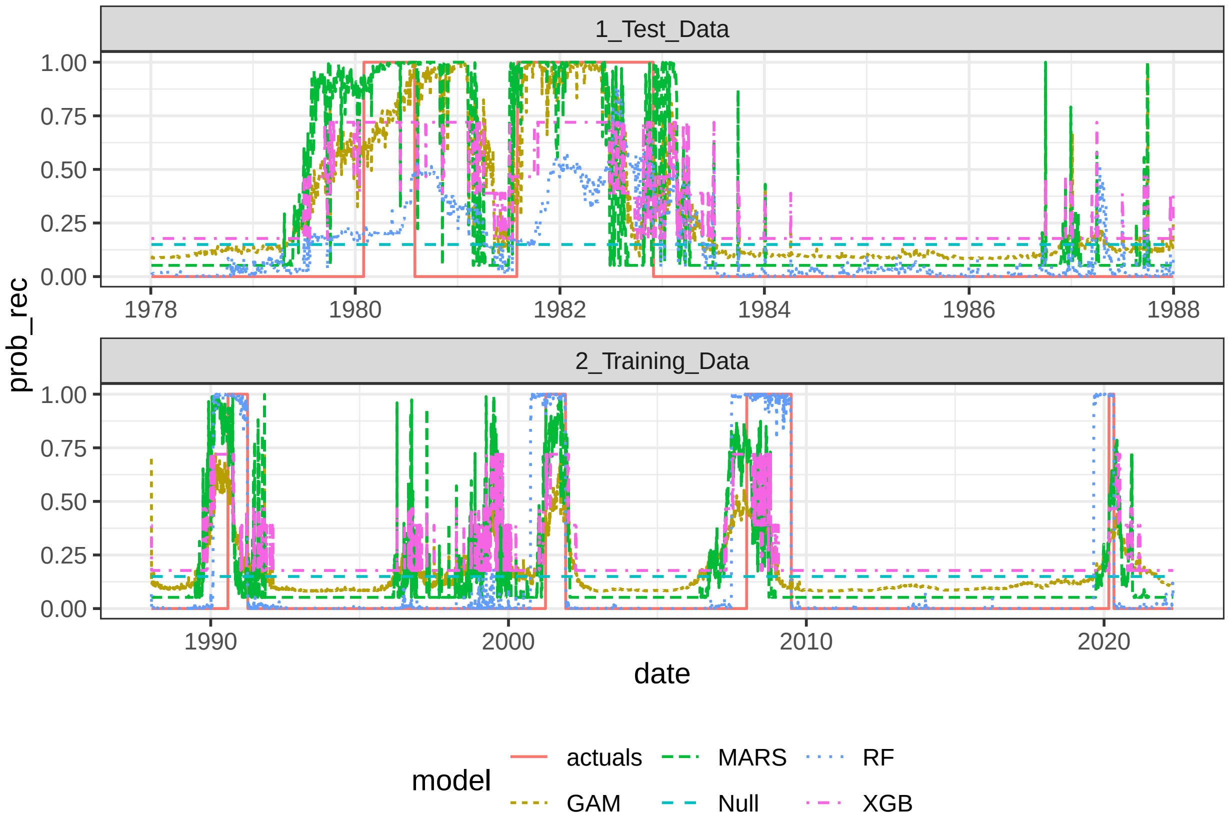
```
full_data_bktst <- full_data_wide %>%
  filter(date >= startTestDate)

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)
}

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

```r
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)
```