Probability of Recession

William Chiu

2022-11-23

Summary

Forecast the probability of a recession in the next 6 months using the following predictors:

1. Spread between 10Y CMT and Effective Federal Funds Rate
2. YOY change in Unemployment Rate
3. YOY growth in CPI-U
4. YOY change in Effective Federal Funds Rate
5. Adstock transformations of predictors

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 <- "1975-01-01"
```

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

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")
  )
})
```

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_UNRATE = UNRATE - lag(UNRATE, 12),
         G_CPIU = (CPIAUCSL / lag(CPIAUCSL, 12) - 1) * 100,
         D_EFFR = FEDFUNDS - lag(FEDFUNDS, 12),
         D_GS10 = GS10 - lag(GS10, 12)
         ) %>%
  mutate(across(
    .cols=c(SPRD_10YCMT_FEDFUNDS, D_UNRATE,
              G_CPIU, D_EFFR, GS10, D_GS10),
    .fns=list(lag1 = ~lag(.x, 1),
         lag3 = ~lag(.x, 3),
         lag6 = ~lag(.x, 6),
         lag9 = ~lag(.x, 9),
         lag12 = ~lag(.x, 12))
  )) %>%
  select(-CPIAUCSL) %>% ## index rises with time
  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 between 0.05 and 0.95.

```
full_data_wide_features_adstock <- full_data_wide_features %>%
  arrange(date) %>%
    mutate(across(
    .cols=c(UNRATE:D_GS10),
    .fns=list(adstkL = ~stats::filter(.x,
                                      filter=0.05,
```

```
                                    method="recursive") ,
        adstkM = ~stats::filter(.x,
                                    filter=0.45,
                                    method="recursive") ,
        adstkHL = ~stats::filter(.x,
                                    filter=0.65,
                                    method="recursive"),
        adstkHM = ~stats::filter(.x,
                                    filter=0.75,
                                    method="recursive"),
        adstkHH = ~stats::filter(.x,
                                    filter=0.85,
                                    method="recursive"),
        adstkHHH = ~stats::filter(.x,
                                    filter=0.95,
                                    method="recursive"),
        cumsum = ~stats::filter(.x,
                                    filter=1,
                                    method="recursive")
  ))) %>%
  mutate(constant=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)
```

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_size <- nrow(full_data_wide)

test_size <- floor(full_size*0.5)

test_id <- seq.int(1,test_size,1)

full_data_wide$constant <- 1

train_data <- full_data_wide[-test_id,]
test_data <- full_data_wide[test_id,]

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 = 389**
date	1988-12-01 to 2021-04-01
USREC	36 (9.3%)
UNRATE	5.50 (4.70, 6.80)
GS10	4.35 (2.60, 6.04)
FEDFUNDS	2.41 (0.24, 5.25)
SPRD_10YCMT_FEDFUNDS	1.49 (0.44, 2.61)
D_UNRATE	-0.30 (-0.60, 0.30)
G_CPIU	2.51 (1.69, 3.16)
D_EFFR	0.00 (-0.77, 0.59)
D_GS10	-0.31 (-0.86, 0.35)
SPRD_10YCMT_FEDFUNDS_lag1	1.47 (0.44, 2.61)
SPRD_10YCMT_FEDFUNDS_lag3	1.47 (0.44, 2.61)
SPRD_10YCMT_FEDFUNDS_lag6	1.47 (0.44, 2.61)
SPRD_10YCMT_FEDFUNDS_lag9	1.50 (0.44, 2.61)
SPRD_10YCMT_FEDFUNDS_lag12	1.55 (0.44, 2.61)
D_UNRATE_lag1	-0.30 (-0.60, 0.30)
D_UNRATE_lag3	-0.30 (-0.60, 0.30)
D_UNRATE_lag6	-0.30 (-0.60, 0.20)
D_UNRATE_lag9	-0.30 (-0.60, 0.20)
D_UNRATE_lag12	-0.30 (-0.60, 0.20)
G_CPIU_lag1	2.51 (1.69, 3.16)
G_CPIU_lag3	2.52 (1.69, 3.17)
G_CPIU_lag6	2.53 (1.70, 3.19)
G_CPIU_lag9	2.55 (1.72, 3.22)
G_CPIU_lag12	2.58 (1.73, 3.25)
D_EFFR_lag1	0.00 (-0.77, 0.60)

Characteristic	N = 389
D_EFFR_lag3	0.00 (-0.77, 0.62)
D_EFFR_lag6	0.01 (-0.75, 0.69)
D_EFFR_lag9	0.01 (-0.74, 0.69)
D_EFFR_lag12	0.02 (-0.73, 0.69)
GS10_lag1	4.42 (2.62, 6.05)
GS10_lag3	4.46 (2.68, 6.11)
GS10_lag6	4.50 (2.71, 6.20)
GS10_lag9	4.53 (2.72, 6.26)
GS10_lag12	4.56 (2.81, 6.28)
D_GS10_lag1	-0.31 (-0.86, 0.34)
D_GS10_lag3	-0.32 (-0.86, 0.34)
D_GS10_lag6	-0.31 (-0.85, 0.35)
D_GS10_lag9	-0.30 (-0.85, 0.38)
D_GS10_lag12	-0.29 (-0.83, 0.40)
UNRATE_adstkL	5.79 (4.94, 7.14)
UNRATE_adstkM	10.0 (8.5, 12.4)
UNRATE_adstkHL	15.8 (13.4, 19.5)
UNRATE_adstkHM	22 (19, 27)
UNRATE_adstkHH	37 (32, 46)
UNRATE_adstkHHH	114 (100, 132)
UNRATE_cumsum	3,431 (2,949, 4,115)
GS10_adstkL	4.58 (2.74, 6.37)
GS10_adstkM	7.9 (4.8, 11.2)
GS10_adstkHL	12.4 (7.6, 17.6)
GS10_adstkHM	17 (11, 25)
GS10_adstkHH	29 (17, 42)
GS10_adstkHHH	91 (51, 133)
GS10_cumsum	3,983 (3,487, 4,322)
FEDFUNDS_adstkL	2.54 (0.25, 5.53)
FEDFUNDS_adstkM	4.4 (0.6, 9.6)
FEDFUNDS_adstkHL	7 (1, 15)
FEDFUNDS_adstkHM	10 (2, 21)
FEDFUNDS_adstkHH	19 (4, 35)
FEDFUNDS_adstkHHH	67 (21, 103)
FEDFUNDS_cumsum	3,457 (3,096, 3,641)
SPRD_10YCMT_FEDFUNDS_adstkL	1.57 (0.46, 2.73)
SPRD_10YCMT_FEDFUNDS_adstkM	2.71 (0.88, 4.67)
SPRD_10YCMT_FEDFUNDS_adstkHL	4.2 (1.4, 7.4)
SPRD_10YCMT_FEDFUNDS_adstkHM	5.8 (1.9, 10.4)
SPRD_10YCMT_FEDFUNDS_adstkHH	10 (3, 17)
SPRD_10YCMT_FEDFUNDS_adstkHHH	31 (15, 44)
SPRD_10YCMT_FEDFUNDS_cumsum	527 (391, 681)
D_UNRATE_adstkL	-0.32 (-0.63, 0.31)
D_UNRATE_adstkM	-0.54 (-1.04, 0.42)
D_UNRATE_adstkHL	-0.9 (-1.6, 0.9)
D_UNRATE_adstkHM	-1.2 (-2.2, 1.2)
D_UNRATE_adstkHH	-2 (-3, 3)
D_UNRATE_adstkHHH	-6 (-9, 7)
D_UNRATE_cumsum	16 (7, 32)
G_CPIU_adstkL	2.62 (1.78, 3.32)
G_CPIU_adstkM	4.54 (3.17, 5.70)
G_CPIU_adstkHL	7.1 (5.0, 8.9)

Characteristic	N = 389
G_CPIU_adstkHM	10.0 (7.0, 12.5)
G_CPIU_adstkHH	16 (12, 20)
G_CPIU_adstkHHH	48 (38, 61)
G_CPIU_cumsum	2,416 (2,188, 2,654)
D_EFFR_adstkL	-0.01 (-0.82, 0.60)
D_EFFR_adstkM	-0.01 (-1.41, 1.10)
D_EFFR_adstkHL	-0.1 (-2.2, 1.7)
D_EFFR_adstkHM	-0.1 (-3.2, 2.4)
D_EFFR_adstkHH	0 (-5, 4)
D_EFFR_adstkHHH	-1 (-14, 6)
D_EFFR_cumsum	7 (-22, 35)
D_GS10_adstkL	-0.33 (-0.92, 0.36)
D_GS10_adstkM	-0.53 (-1.57, 0.49)
D_GS10_adstkHL	-0.91 (-2.36, 0.75)
D_GS10_adstkHM	-1.19 (-3.20, 1.03)
D_GS10_adstkHH	-1.9 (-4.6, 1.1)
D_GS10_adstkHHH	-4.2 (-8.5, -0.8)
D_GS10_cumsum	17 (-3, 39)
constant	389 (100%)
FUTREC	56 (14%)

Remove stale data from test set

Exclude historical data prior to 1975-01-01 because the economy changed dramatically (due to computational innovation).

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

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "1956-07-01" "1964-08-01" "1972-09-01" "1972-08-31" "1980-10-01" "1988-11-01"
```

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

Setup Parallel Processing

```
library(doParallel)

cl <- makePSOCKcluster(5)
registerDoParallel(cl)
```

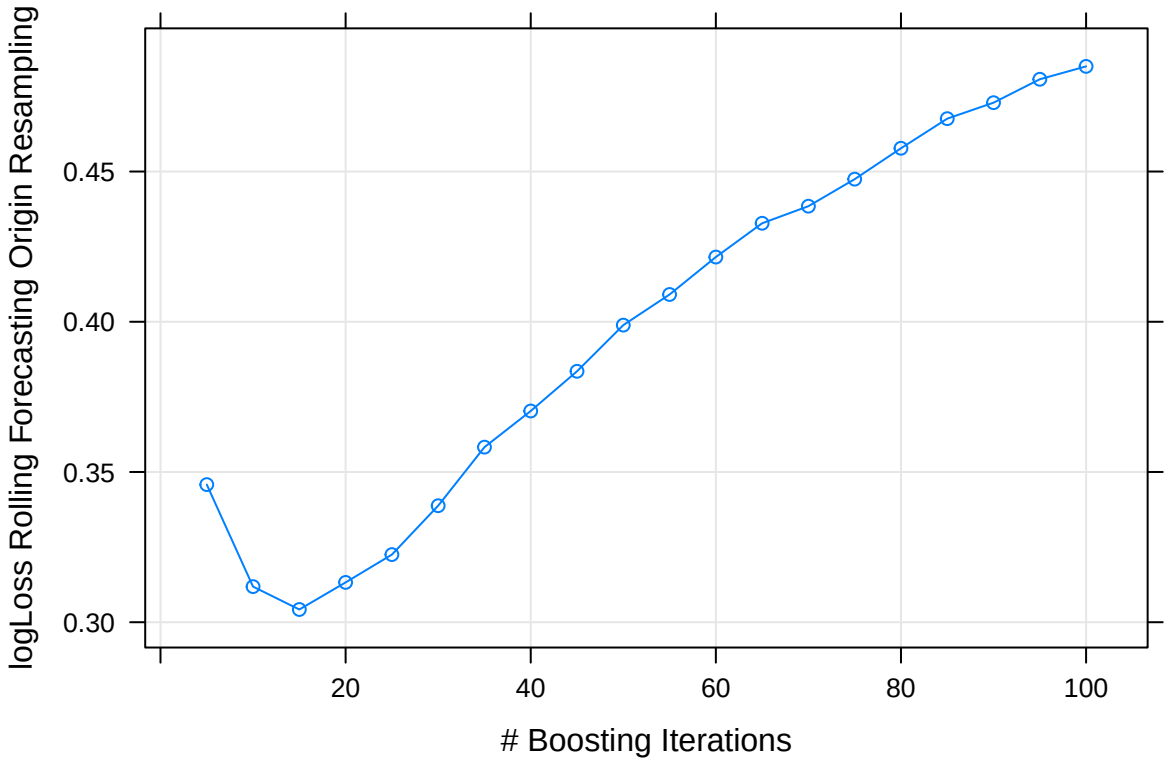
Cross-Validation Framework

```
folds <- 20
fcstHorizon <- 12
initWindow <- nrow(train_yes_no) - fcstHorizon * folds
```

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

fitControl_oneSE <- trainControl(method = "timeslice",
                           initialWindow=initWindow,
                           horizon=fcstHorizon,
                           fixedWindow=FALSE,
                           skip=fcstHorizon-1,
                           ## 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=fcstHorizon-1,
                           ## 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(5,100,5),
                        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()
)

plot(gam_mod)
```



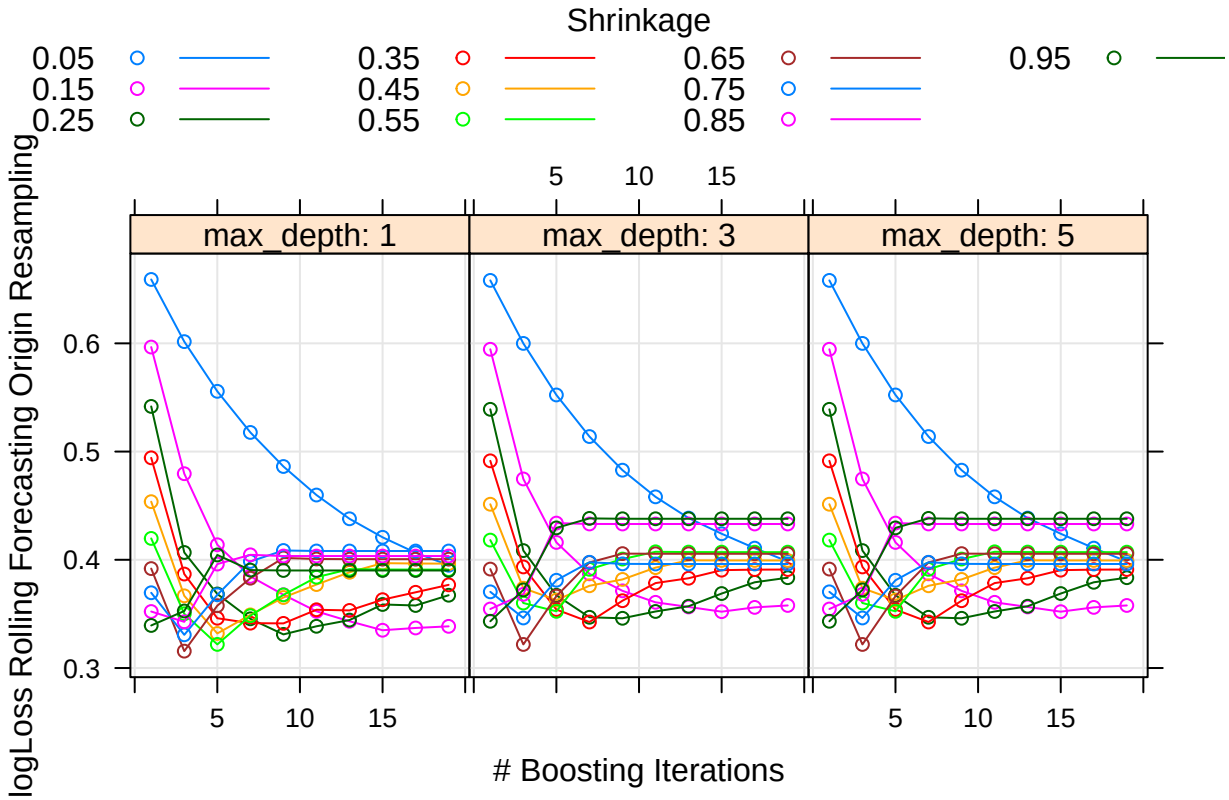

```
gam_mod$bestTune
```

```
##   mstop prune
## 1     5    no
```

eXtreme Gradient Boosting Trees

```
grid_xgb <- expand.grid(nrounds=seq(1, 20, 2),
                        max_depth=c(1,3,5),
                        eta=seq(0.05,1,0.1),
                        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_best,
```

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

plot(xgb_mod)
```



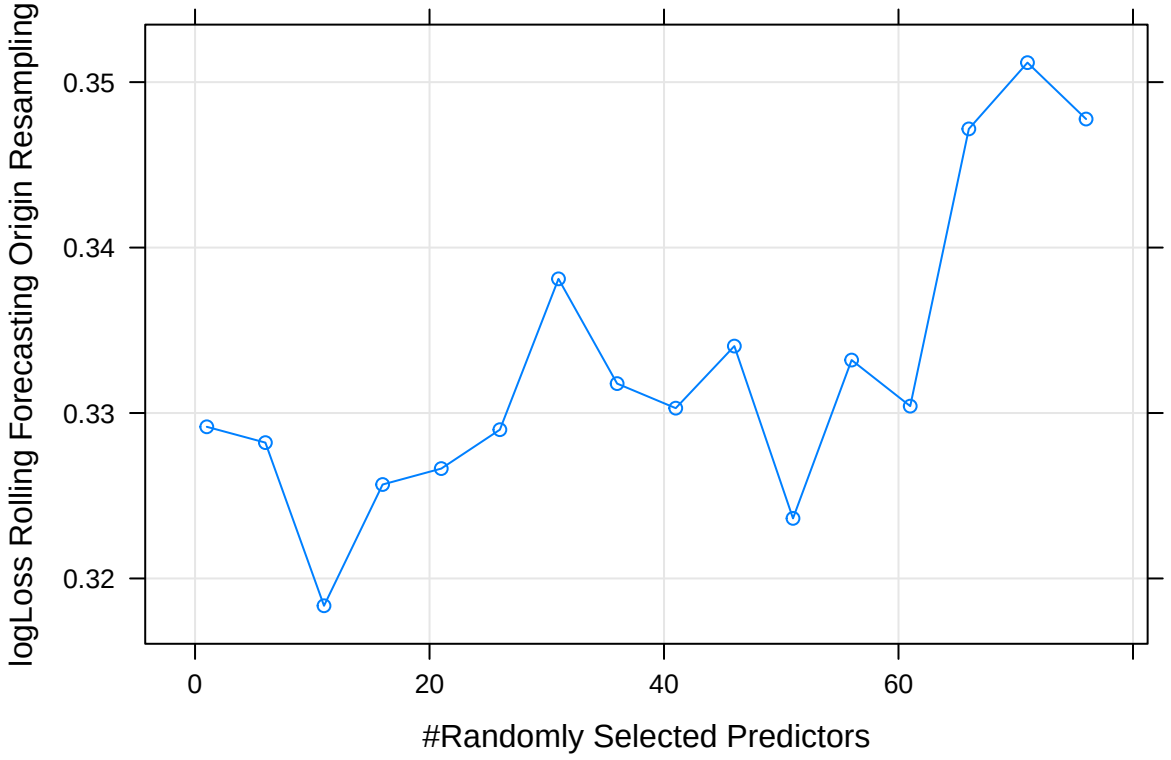

```
xgb_mod$bestTune
```

```
##     nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
## 182       3         1 0.65     0                1               10         1
```

Random Forest

```
grid_rf <- data.frame(mtry=seq.int(1,76,5))

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)
```




```
rf_mod$bestTune
```

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

Stepwise Regression

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

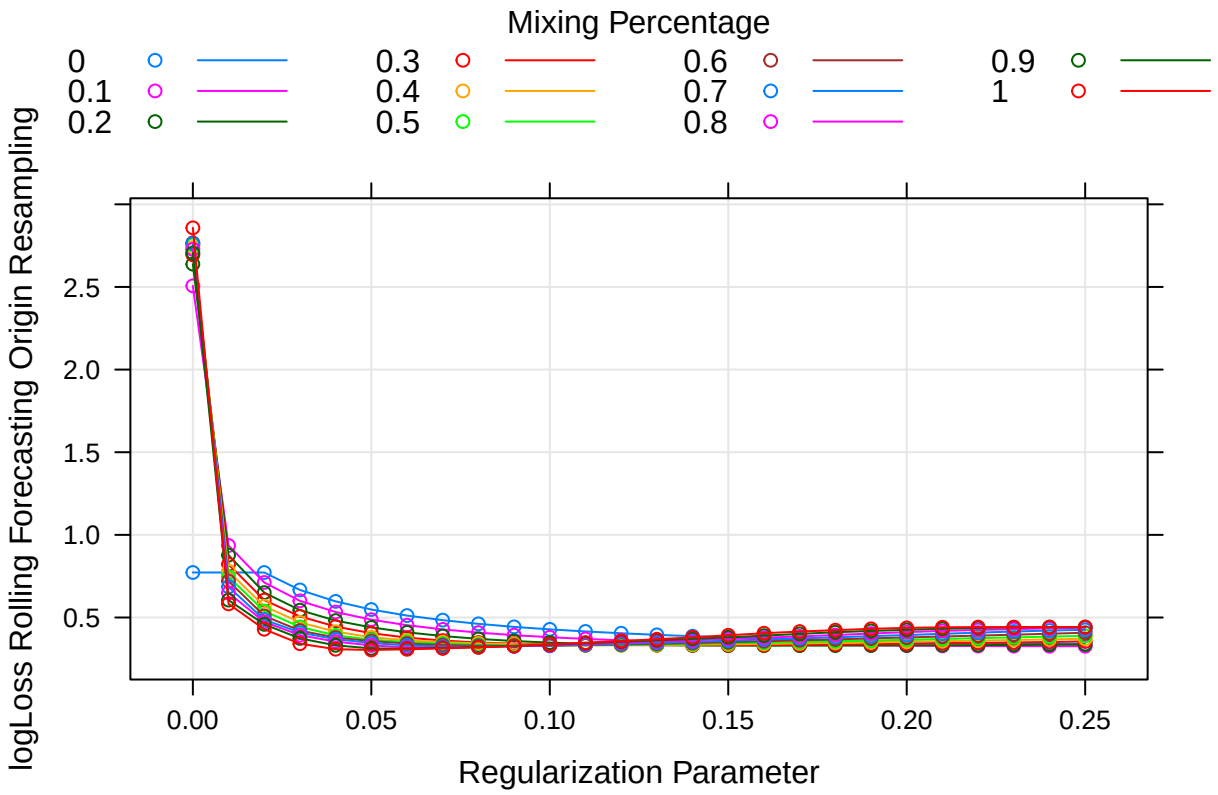
> For binomial and quasibinomial families the response can also be specified as a factor (when the first level denotes failure and all others success) or as a two-column matrix with the columns giving the numbers of successes and failures.

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 - USREC - constant,
  data = train_yes_no,
  method = "glmStepAIC",
  trControl = fitControl_best,
  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 = seq(0,1,0.1),
  lambda = seq(0, 0.25, 0.01)
)

set.seed(randSeed)

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

plot(glmnet_mod)
```

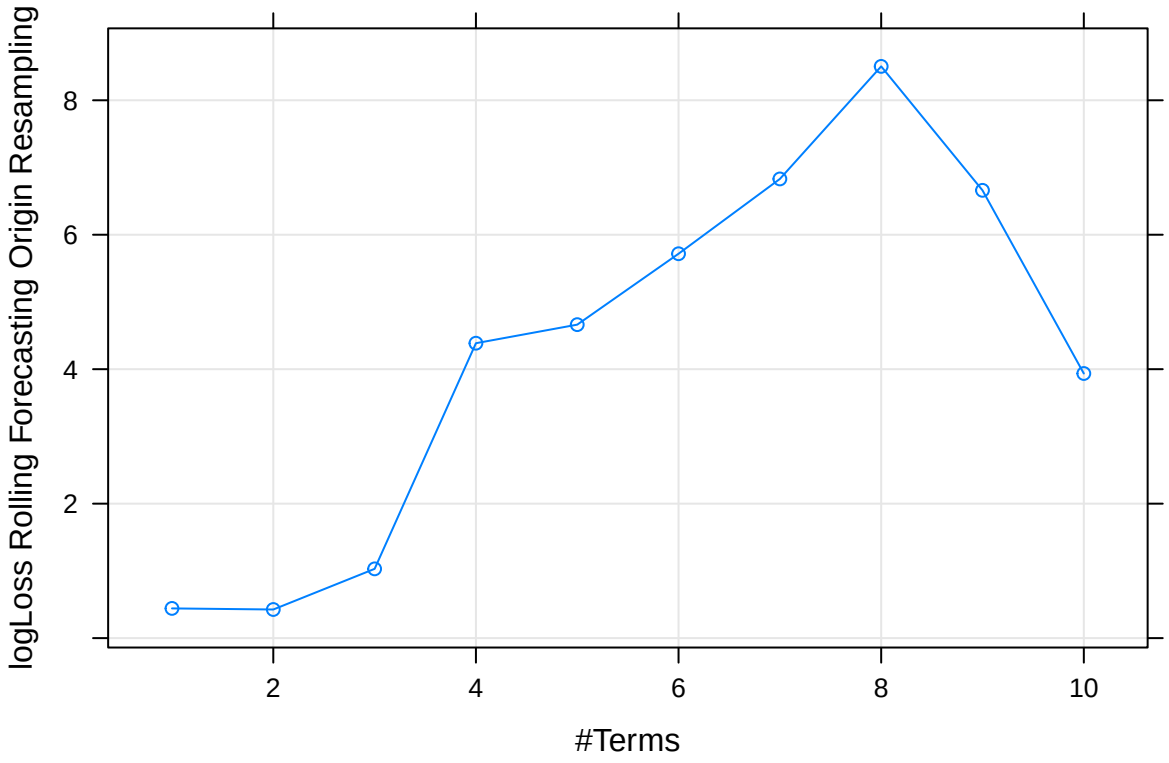



```
glmnet_mod$bestTune
```

```
##     alpha lambda
## 266     1   0.05
```

Multivariate Adaptive Regression Splines

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

set.seed(randSeed)

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

plot(earth_mod)
```




```
earth_mod$bestTune
```

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

Null Model: Intercept-only Model

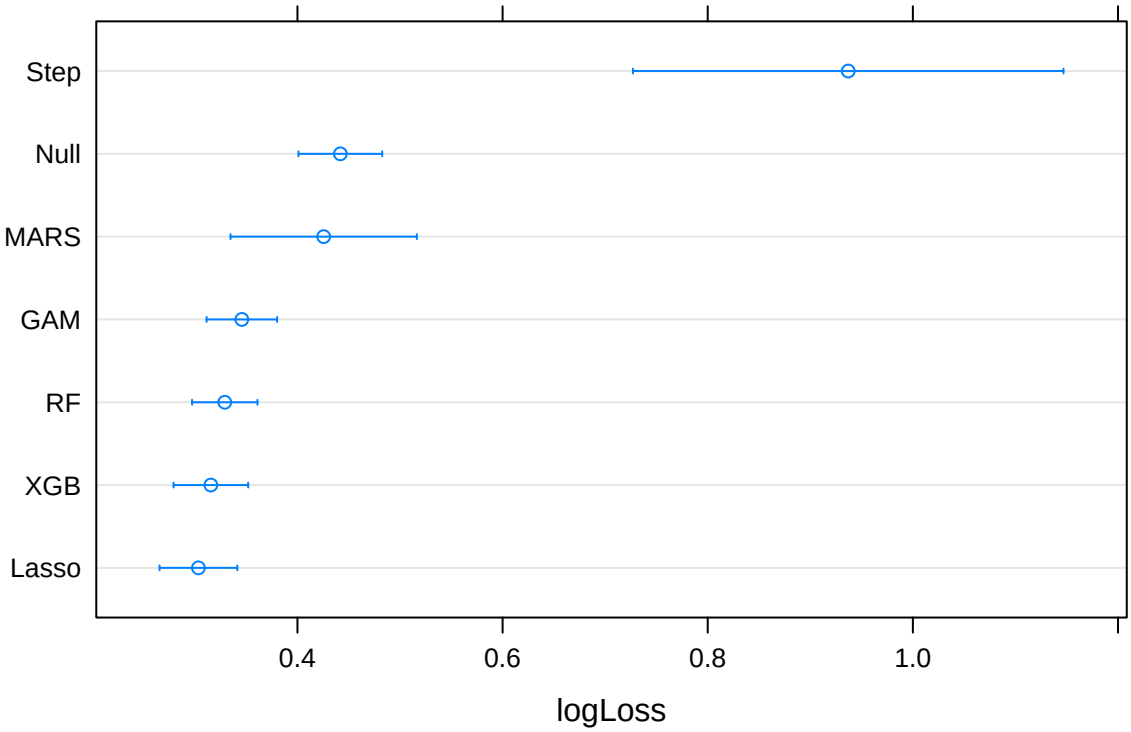
```
set.seed(randSeed)

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

Compare Models

```
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: 20
##
## logLoss
##                Min.      1st Qu.       Median      Mean    3rd Qu.       Max.
## XGB   5.532021e-02 7.098843e-02 7.507463e-02 0.3155920 0.33786337  1.870935
## GAM   9.620736e-02 1.145001e-01 1.235694e-01 0.3458054 0.28300292  1.892727
## RF    1.311735e-02 3.739762e-02 1.081287e-01 0.3291669 0.46164209  1.443193
## Step  9.992007e-16 1.490104e-13 5.306273e-11 0.9369846 0.07655798 11.921809
## Lasso 6.665714e-03 1.242178e-02 3.824509e-02 0.3034846 0.30825928  1.930786
## MARS  2.538100e-03 2.951890e-02 4.590833e-02 0.4255878 0.07711717  5.441623
## Null  1.328481e-01 1.593852e-01 1.756320e-01 0.4418435 0.24258385  1.867520
##       NA's
## XGB      0
## GAM      0
## RF       0
## Step     0
## Lasso    0
## MARS     0
## Null     0
```

```
dotplot(resamps, metric = "logLoss", conf.level=0.25)
```



Confidence Level: 0.25

Explore XGB Model

```
xgb_mod$bestTune
```

```
##     nrounds max_depth  eta gamma colsample_bytree min_child_weight subsample
## 182       3         1 0.65     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)
```

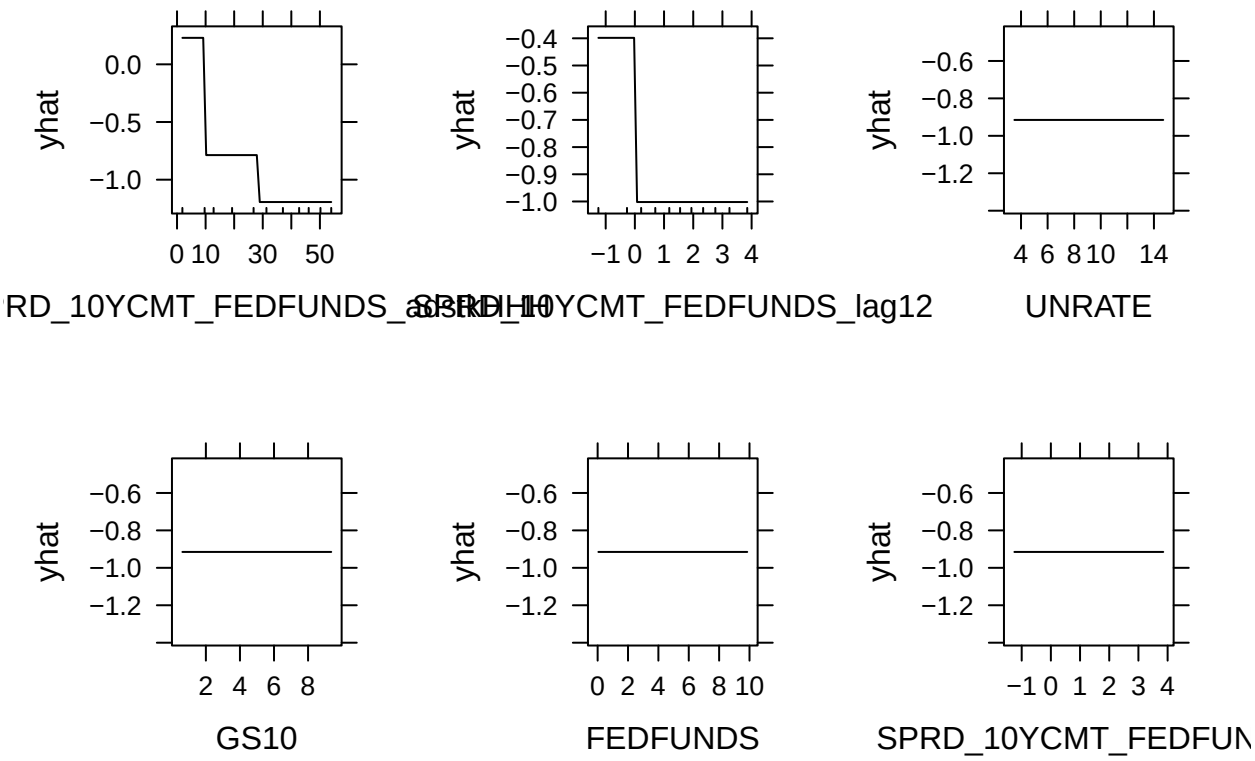
variable	Overall
SPRD_10YCMT_FEDFUNDS_adstkHHH	100.00000

variable	Overall
SPRD_10YCMT_FEDFUNDS_lag12	29.24978
UNRATE	0.00000
GS10	0.00000
FEDFUNDS	0.00000
SPRD_10YCMT_FEDFUNDS	0.00000
D_UNRATE	0.00000
G_CPIU	0.00000
D_EFFR	0.00000
D_GS10	0.00000
SPRD_10YCMT_FEDFUNDS_lag1	0.00000
SPRD_10YCMT_FEDFUNDS_lag3	0.00000
SPRD_10YCMT_FEDFUNDS_lag6	0.00000
SPRD_10YCMT_FEDFUNDS_lag9	0.00000
D_UNRATE_lag1	0.00000
D_UNRATE_lag3	0.00000
D_UNRATE_lag6	0.00000
D_UNRATE_lag9	0.00000
D_UNRATE_lag12	0.00000
G_CPIU_lag1	0.00000
G_CPIU_lag3	0.00000
G_CPIU_lag6	0.00000
G_CPIU_lag9	0.00000
G_CPIU_lag12	0.00000
D_EFFR_lag1	0.00000
D_EFFR_lag3	0.00000
D_EFFR_lag6	0.00000
D_EFFR_lag9	0.00000
D_EFFR_lag12	0.00000
GS10_lag1	0.00000
GS10_lag3	0.00000
GS10_lag6	0.00000
GS10_lag9	0.00000
GS10_lag12	0.00000
D_GS10_lag1	0.00000
D_GS10_lag3	0.00000
D_GS10_lag6	0.00000
D_GS10_lag9	0.00000
D_GS10_lag12	0.00000
UNRATE_adstkL	0.00000
UNRATE_adstkM	0.00000
UNRATE_adstkHL	0.00000
UNRATE_adstkHM	0.00000
UNRATE_adstkHH	0.00000
UNRATE_adstkHHH	0.00000
UNRATE_cumsum	0.00000
GS10_adstkL	0.00000
GS10_adstkM	0.00000
GS10_adstkHL	0.00000
GS10_adstkHM	0.00000
GS10_adstkHH	0.00000
GS10_adstkHHH	0.00000
GS10_cumsum	0.00000

variable	Overall
FEDFUNDS_adstkL	0.00000
FEDFUNDS_adstkM	0.00000
FEDFUNDS_adstkHL	0.00000
FEDFUNDS_adstkHM	0.00000
FEDFUNDS_adstkHH	0.00000
FEDFUNDS_adstkHHH	0.00000
FEDFUNDS_cumsum	0.00000
SPRD_10YCMT_FEDFUNDS_adstkL	0.00000
SPRD_10YCMT_FEDFUNDS_adstkM	0.00000
SPRD_10YCMT_FEDFUNDS_adstkHL	0.00000
SPRD_10YCMT_FEDFUNDS_adstkHM	0.00000
SPRD_10YCMT_FEDFUNDS_adstkHH	0.00000
SPRD_10YCMT_FEDFUNDS_cumsum	0.00000
D_UNRATE_adstkL	0.00000
D_UNRATE_adstkM	0.00000
D_UNRATE_adstkHL	0.00000
D_UNRATE_adstkHM	0.00000
D_UNRATE_adstkHH	0.00000
D_UNRATE_adstkHHH	0.00000
D_UNRATE_cumsum	0.00000
G_CPIU_adstkL	0.00000
G_CPIU_adstkM	0.00000
G_CPIU_adstkHL	0.00000
G_CPIU_adstkHM	0.00000
G_CPIU_adstkHH	0.00000
G_CPIU_adstkHHH	0.00000
G_CPIU_cumsum	0.00000
D_EFFR_adstkL	0.00000
D_EFFR_adstkM	0.00000
D_EFFR_adstkHL	0.00000
D_EFFR_adstkHM	0.00000
D_EFFR_adstkHH	0.00000
D_EFFR_adstkHHH	0.00000
D_EFFR_cumsum	0.00000
D_GS10_adstkL	0.00000
D_GS10_adstkM	0.00000
D_GS10_adstkHL	0.00000
D_GS10_adstkHM	0.00000
D_GS10_adstkHH	0.00000
D_GS10_adstkHHH	0.00000
D_GS10_cumsum	0.00000

```
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)
```

```r
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_best,
  metric = "logLoss",
  family=binomial
)

summary(logit_mod)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.84730   0.01632   0.04762   0.23358   1.44168
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    -2.33342    0.47449  -4.918 8.75e-07 ***
## SPRD_10YCMT_FEDFUNDS_adstkHHH   0.22564    0.03458   6.525 6.80e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 320.60  on 388  degrees of freedom
## Residual deviance: 169.25  on 387  degrees of freedom
## AIC: 173.25
##
## Number of Fisher Scoring iterations: 8
```

Compare Models

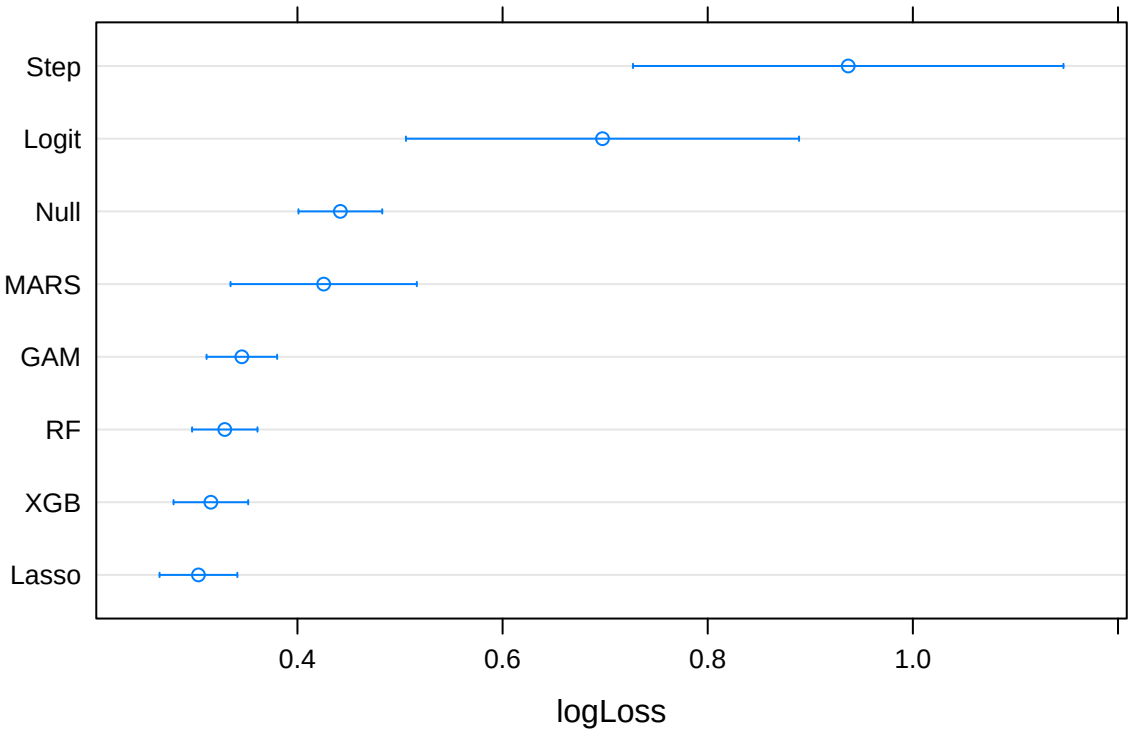
CV errors for models with peeking are misleadingly low. This will be addressed with a test set.

```
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: 20
##
## logLoss
##               Min.      1st Qu.       Median      Mean    3rd Qu.      Max.
## XGB   5.532021e-02 7.098843e-02 7.507463e-02 0.3155920 0.33786337  1.870935
## GAM   9.620736e-02 1.145001e-01 1.235694e-01 0.3458054 0.28300292  1.892727
## RF    1.311735e-02 3.739762e-02 1.081287e-01 0.3291669 0.46164209  1.443193
## Step  9.992007e-16 1.490104e-13 5.306273e-11 0.9369846 0.07655798 11.921809
## Lasso 6.665714e-03 1.242178e-02 3.824509e-02 0.3034846 0.30825928  1.930786
## MARS  2.538100e-03 2.951890e-02 4.590833e-02 0.4255878 0.07711717  5.441623
## Null  1.328481e-01 1.593852e-01 1.756320e-01 0.4418435 0.24258385  1.867520
## Logit 9.992007e-16 1.957131e-04 1.402309e-03 0.6974143 0.14182255 11.921809
##       NA's
## XGB      0
## GAM      0
## RF       0
## Step     0
## Lasso    0
## MARS     0
## Null     0
## Logit    0
```

```
dotplot(resamps, metric = "logLoss", conf.level=0.25)
```



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
GAM	0.9630694
Step	0.9630694
Logit	0.9630694
MARS	0.9542238
Lasso	0.9195046
RF	0.9002654
XGB	0.8998231
Null	0.5000000

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

model	LogLoss
GAM	0.3666134
XGB	0.3912503
RF	0.4067598

model	LogLoss
Lasso	0.5104607
Null	0.5184002
Step	0.7665525
Logit	0.7665525
MARS	3.4294889

Probability of Recession (Most Recent Month)

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

curr_data$date
```

```
## [1] "2022-10-01"
```

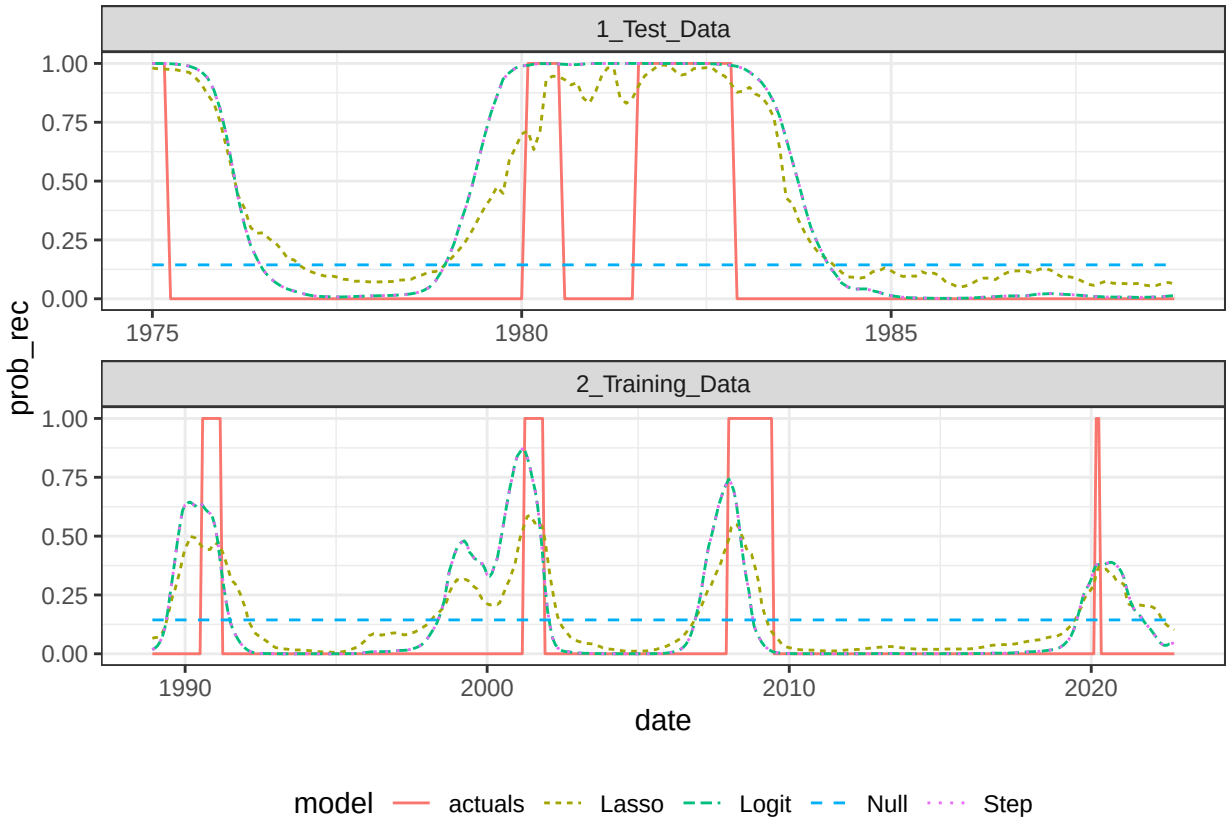
```
score_fun <- function(mods, dat) {
  output <- map_dfc(.x = mods, .f = function(x) {
    if(class(x)[1] != "train"){
      predict(x, newdata = dat, type = "response")
    } else(
       predict(x, newdata = dat, type = "prob")[,"yes"]

    )

  }) %>%
    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	14.83%
GAM	12.90%
RF	20.40%
Step	4.69%
Lasso	10.18%
MARS	5.43%
Null	14.40%
Logit	4.69%

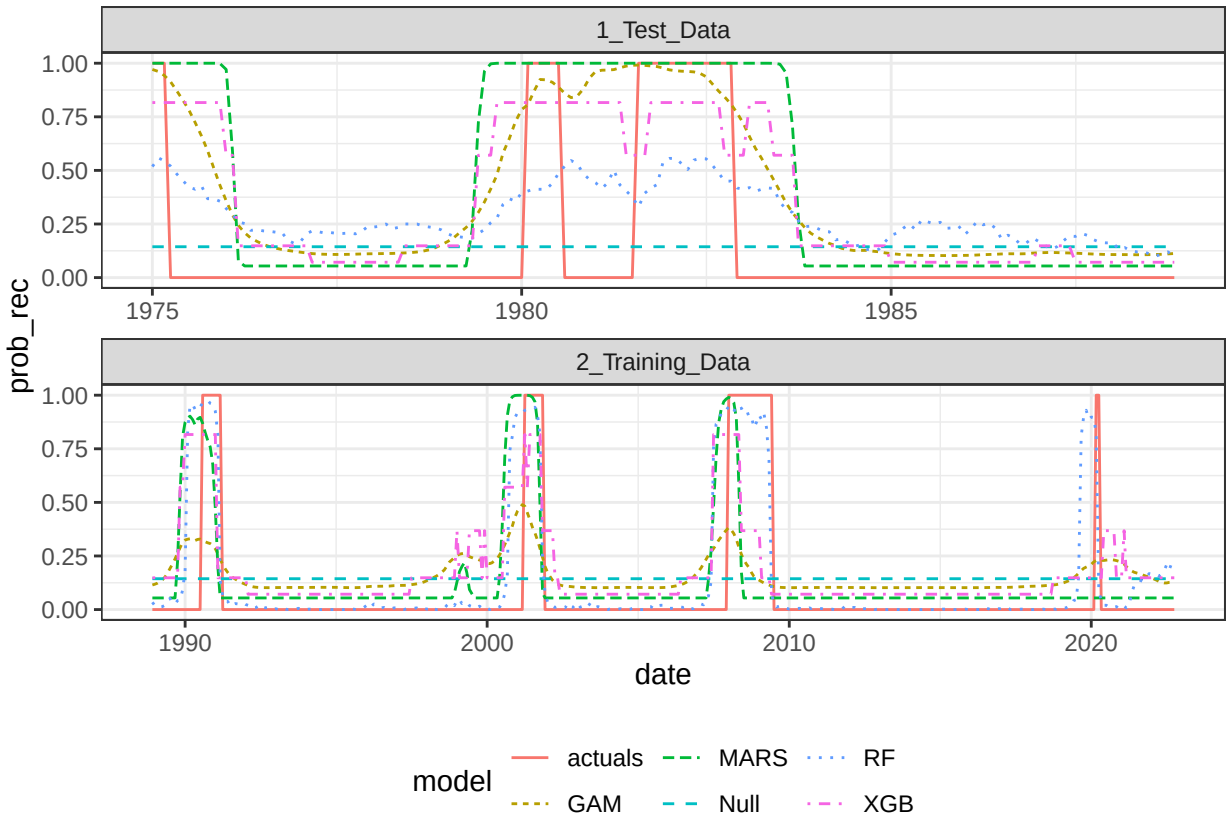
Backtesting

```
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)
}

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'))
```

```
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)
```