Probability of Recession

William Chiu

2022-11-27

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

```
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 1.

```r
full_data_wide_features_adstock <- full_data_wide_features %>%
  arrange(date) %>%
    mutate(across(
    .cols=c(UNRATE:D_GS10),
    .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

```r
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(
         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),
```

```r
         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.

```r
recent_data <- tail(full_data_wide_features_adstock, 12)

train_test <- head(full_data_wide_features_adstock, -12)
```

Recession in next 6 months

```r
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

```r
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",
```

```r
                            TRUE ~ "no"))

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



tbl_summary(train_data)
```

Characteristic	N = 400
date	1988-01-01 to 2021-04-01
USREC	36 (9.0%)
UNRATE	5.50 (4.70, 6.70)
GS10	4.50 (2.67, 6.23)
FEDFUNDS	2.71 (0.37, 5.29)
SPRD_10YCMT_FEDFUNDS	1.48 (0.53, 2.56)
D_UNRATE	-0.30 (-0.60, 0.23)
G_CPIU	2.54 (1.69, 3.22)
D_EFFR	0.01 (-0.75, 0.63)
D_GS10	-0.30 (-0.85, 0.39)
SPRD_10YCMT_FEDFUNDS_lag1	1.48 (0.53, 2.56)
SPRD_10YCMT_FEDFUNDS_lag3	1.50 (0.53, 2.56)
SPRD_10YCMT_FEDFUNDS_lag6	1.54 (0.53, 2.56)
SPRD_10YCMT_FEDFUNDS_lag9	1.56 (0.53, 2.56)
SPRD_10YCMT_FEDFUNDS_lag12	1.56 (0.53, 2.56)
D_UNRATE_lag1	-0.30 (-0.60, 0.23)
D_UNRATE_lag3	-0.30 (-0.60, 0.20)
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, 0.10)
G_CPIU_lag1	2.54 (1.69, 3.22)
G_CPIU_lag3	2.54 (1.70, 3.23)
G_CPIU_lag6	2.57 (1.72, 3.26)
G_CPIU_lag9	2.60 (1.73, 3.34)
G_CPIU_lag12	2.60 (1.74, 3.34)
D_EFFR_lag1	0.01 (-0.75, 0.63)
D_EFFR_lag3	0.01 (-0.75, 0.66)
D_EFFR_lag6	0.02 (-0.74, 0.68)
D_EFFR_lag9	0.02 (-0.73, 0.68)
D_EFFR_lag12	0.02 (-0.73, 0.68)
GS10_lag1	4.51 (2.70, 6.26)
GS10_lag3	4.53 (2.71, 6.27)
GS10_lag6	4.56 (2.72, 6.30)
GS10_lag9	4.58 (2.80, 6.42)
GS10_lag12	4.65 (2.84, 6.49)
D_GS10_lag1	-0.30 (-0.85, 0.39)
D_GS10_lag3	-0.30 (-0.85, 0.39)
D_GS10_lag6	-0.29 (-0.85, 0.41)
D_GS10_lag9	-0.27 (-0.83, 0.43)
D_GS10_lag12	-0.27 (-0.83, 0.43)
UNRATE_adstk85	37 (32, 45)
UNRATE_adstk91	62 (54, 74)

Characteristic	N = 400
UNRATE_adstk92	70 (61, 82)
UNRATE_adstk93	80 (71, 93)
UNRATE_adstk94	94 (83, 110)
UNRATE_adstk95	114 (101, 132)
UNRATE_adstk99	620 (557, 653)
GS10_adstk85	30 (17, 43)
GS10_adstk91	51 (29, 72)
GS10_adstk92	57 (33, 81)
GS10_adstk93	65 (37, 93)
GS10_adstk94	76 (43, 110)
GS10_adstk95	92 (51, 133)
GS10_adstk99	611 (434, 790)
FEDFUNDS_adstk85	21 (4, 36)
FEDFUNDS_adstk91	37 (9, 59)
FEDFUNDS_adstk92	43 (11, 66)
FEDFUNDS_adstk93	50 (14, 75)
FEDFUNDS_adstk94	59 (17, 87)
FEDFUNDS_adstk95	71 (21, 104)
FEDFUNDS_adstk99	457 (263, 652)
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_UNRATE_adstk85	-2 (-4, 2)
D_UNRATE_adstk91	-4 (-6, 4)
D_UNRATE_adstk92	-4 (-6, 4)
D_UNRATE_adstk93	-5 (-7, 5)
D_UNRATE_adstk94	-5 (-8, 6)
D_UNRATE_adstk95	-6 (-9, 6)
D_UNRATE_adstk99	-9 (-17, 5)
G_CPIU_adstk85	17 (12, 21)
G_CPIU_adstk91	28 (21, 35)
G_CPIU_adstk92	31 (24, 39)
G_CPIU_adstk93	35 (28, 44)
G_CPIU_adstk94	41 (32, 51)
G_CPIU_adstk95	49 (38, 61)
G_CPIU_adstk99	311 (261, 416)
D_EFFR_adstk85	0 (-5, 4)
D_EFFR_adstk91	0 (-9, 5)
D_EFFR_adstk92	-1 (-10, 5)
D_EFFR_adstk93	-1 (-11, 6)
D_EFFR_adstk94	-1 (-12, 6)
D_EFFR_adstk95	-2 (-13, 6)
D_EFFR_adstk99	-18 (-37, -8)
D_GS10_adstk85	-1.7 (-4.5, 1.4)
D_GS10_adstk91	-2.3 (-6.1, 1.2)
D_GS10_adstk92	-2.5 (-6.5, 1.0)
D_GS10_adstk93	-2.8 (-7.0, 0.6)
D_GS10_adstk94	-3.3 (-7.5, 0.1)

Characteristic	**N = 400**
D_GS10_adstk95	-4.2 (-8.4, -0.8)
D_GS10_adstk99	-20 (-23, -13)
constant	400 (100%)
SPRD_10YCMT_FEDFUNDS_ma2m	1.48 (0.48, 2.54)
SPRD_10YCMT_FEDFUNDS_ma3m	1.49 (0.49, 2.57)
SPRD_10YCMT_FEDFUNDS_ma6m	1.51 (0.50, 2.59)
SPRD_10YCMT_FEDFUNDS_ma9m	1.52 (0.49, 2.62)
SPRD_10YCMT_FEDFUNDS_ma12m	1.49 (0.50, 2.60)
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)
```

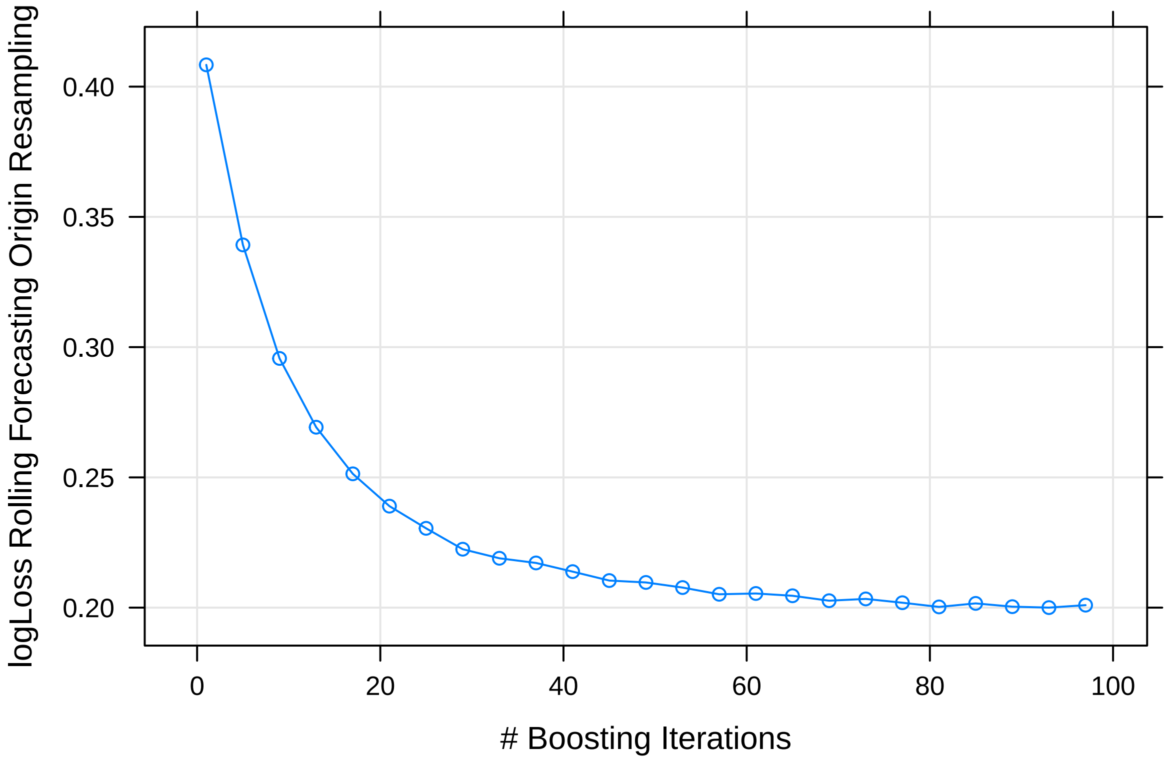
Cross-Validation Framework

```
fcstHorizon <- 1
initWindow <- 120
param_skip <- fcstHorizon - 1

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

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

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

Gradient Boosting for Additive Models

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

set.seed(randSeed)

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

plot(gam_mod)
```



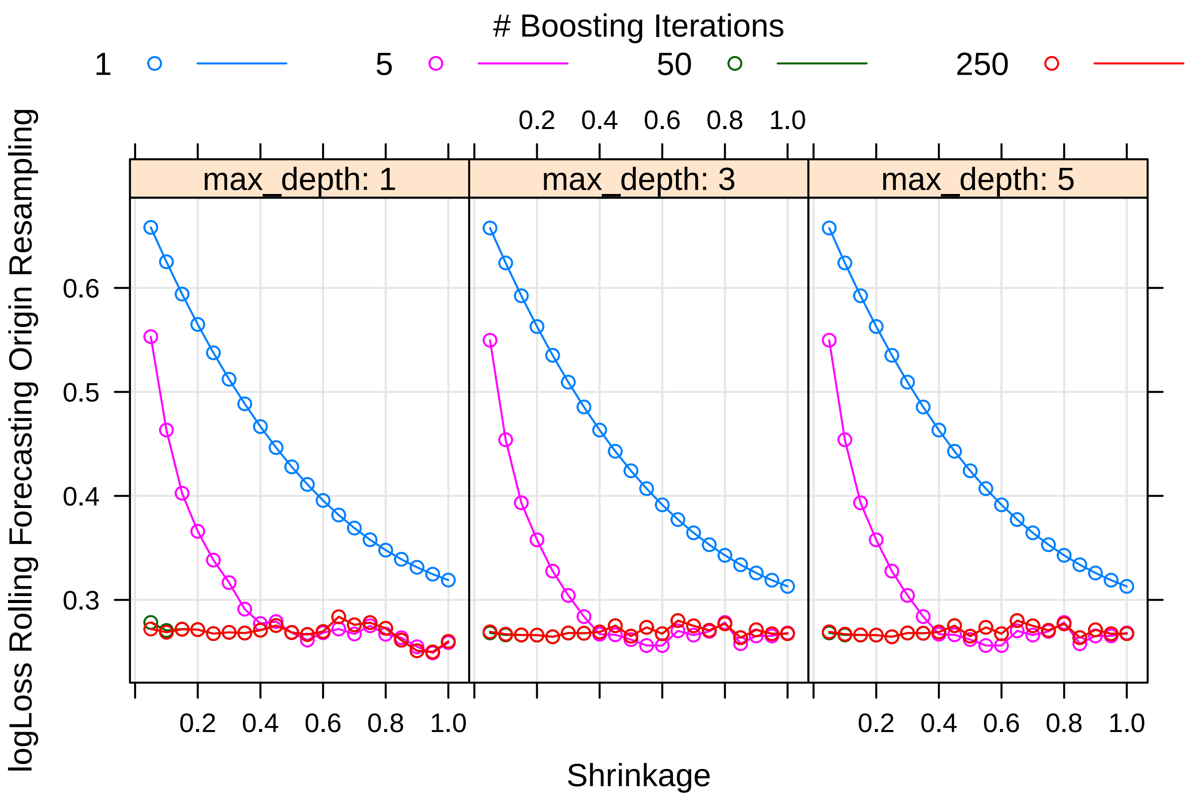

```
gam_mod$bestTune
```

```
##   mstop prune
## 7    25    no
```

eXtreme Gradient Boosting Trees

```
grid_xgb <- expand.grid(nrounds=c(1,5,50,250),
                        max_depth=c(1,3,5),
                        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,
```

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

plot(xgb_mod)
```



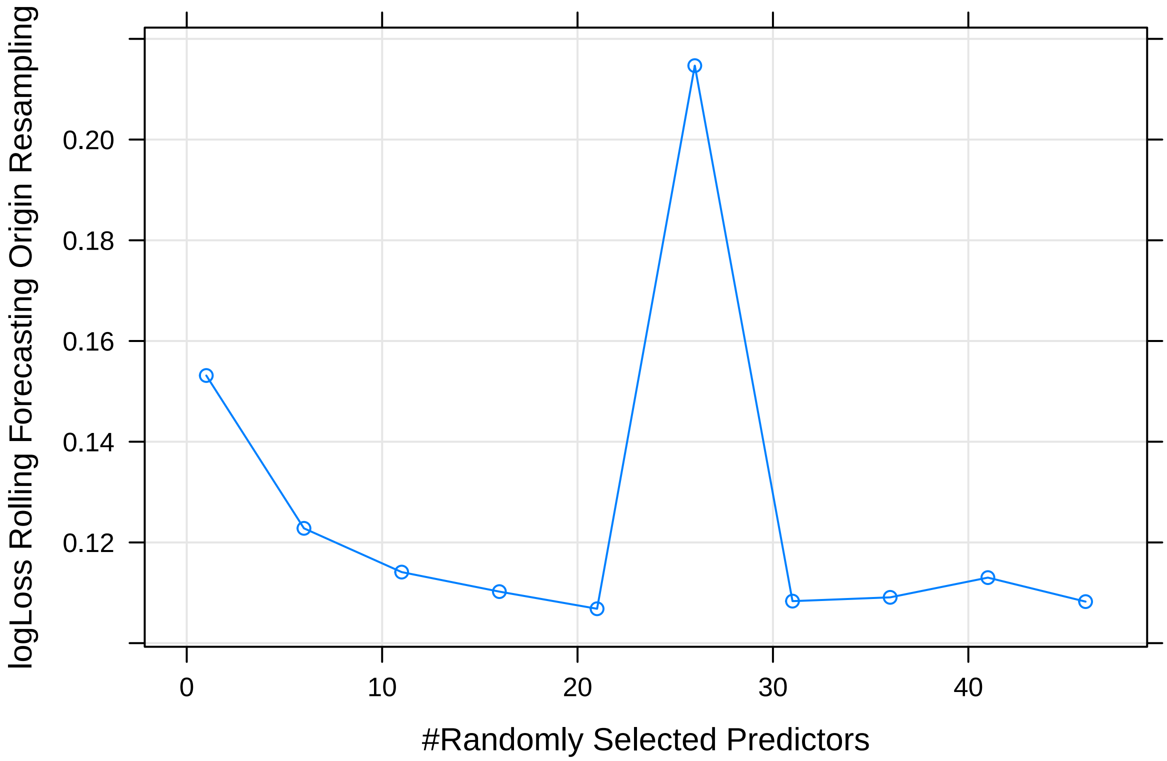

```r
xgb_mod$bestTune
```

```
##    nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 86       5         1 0.4     0                1               10         1
```

Random Forest

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

set.seed(randSeed)

rf_mod <- train(
  FUTREC ~ . - date - 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
## 2    6
```

Stepwise Regression

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

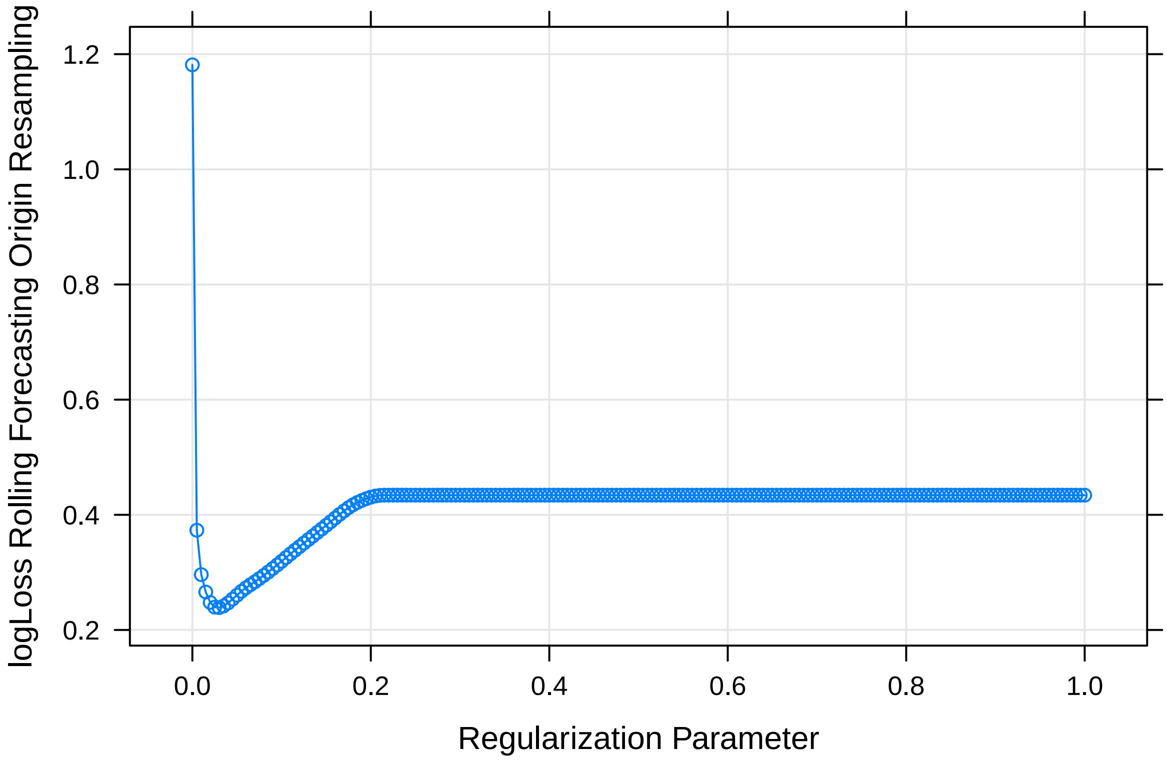
> For binomial and quasibinomial families the response can also be specified as a factor (when the first level denotes failure and all others success) or as a two-column matrix with the columns giving the numbers of successes and failures.

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,
  family = "binomial"
)

plot(glmnet_mod)
```

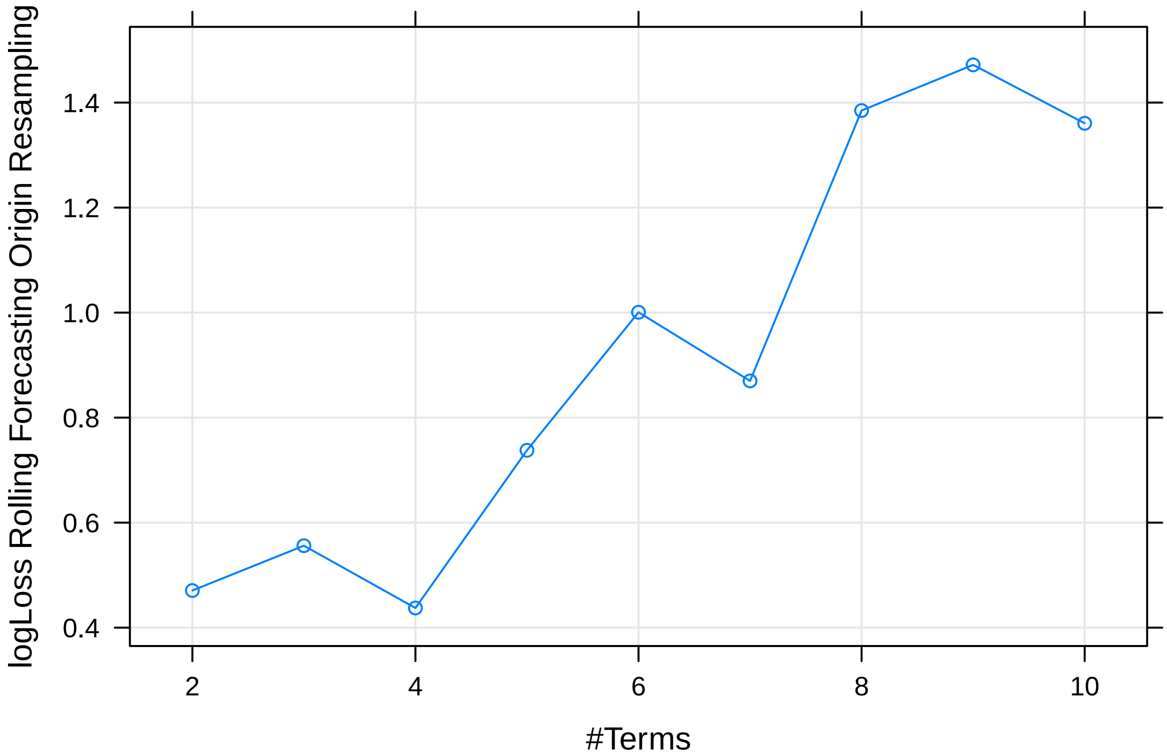



```
glmnet_mod$bestTune
```

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

Multivariate Adaptive Regression Splines

```
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

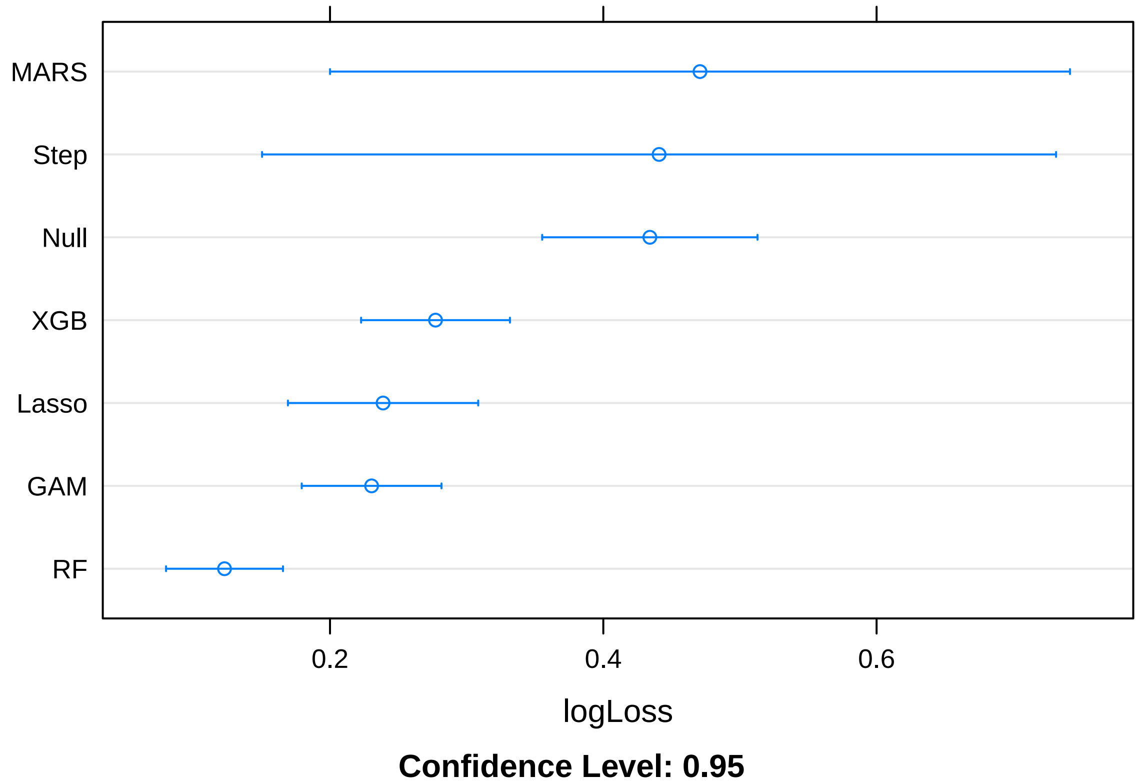
```
set.seed(randSeed)

null_mod <- train(
  FUTREC ~ constant,
  data = train_yes_no,
  method = "glm",
  trControl = fitControl_best,
  metric = "logLoss",
  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: 280
##
## logLoss
##               Min.      1st Qu.       Median      Mean      3rd Qu.      Max.
## XGB   5.602337e-02 7.602335e-02 9.952490e-02 0.2772170 0.1842396575  2.325495
## GAM   2.999232e-02 4.789796e-02 5.355077e-02 0.2304475 0.1707451150  3.358296
## RF    9.992007e-16 4.008021e-03 1.612938e-02 0.1228011 0.0882853626  3.729701
## Step  9.992007e-16 9.992007e-16 1.767475e-13 0.4408222 0.0004721395 34.538776
## Lasso 5.033888e-04 2.660753e-03 1.558069e-02 0.2388287 0.1083769170  4.758196
## MARS  9.992007e-16 4.364253e-03 3.867560e-02 0.4707717 0.0500827115 34.538776
## Null  8.940659e-02 1.395372e-01 1.562128e-01 0.4340732 0.1895729911  2.465489
##       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
## 86       5         1 0.4     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)
```

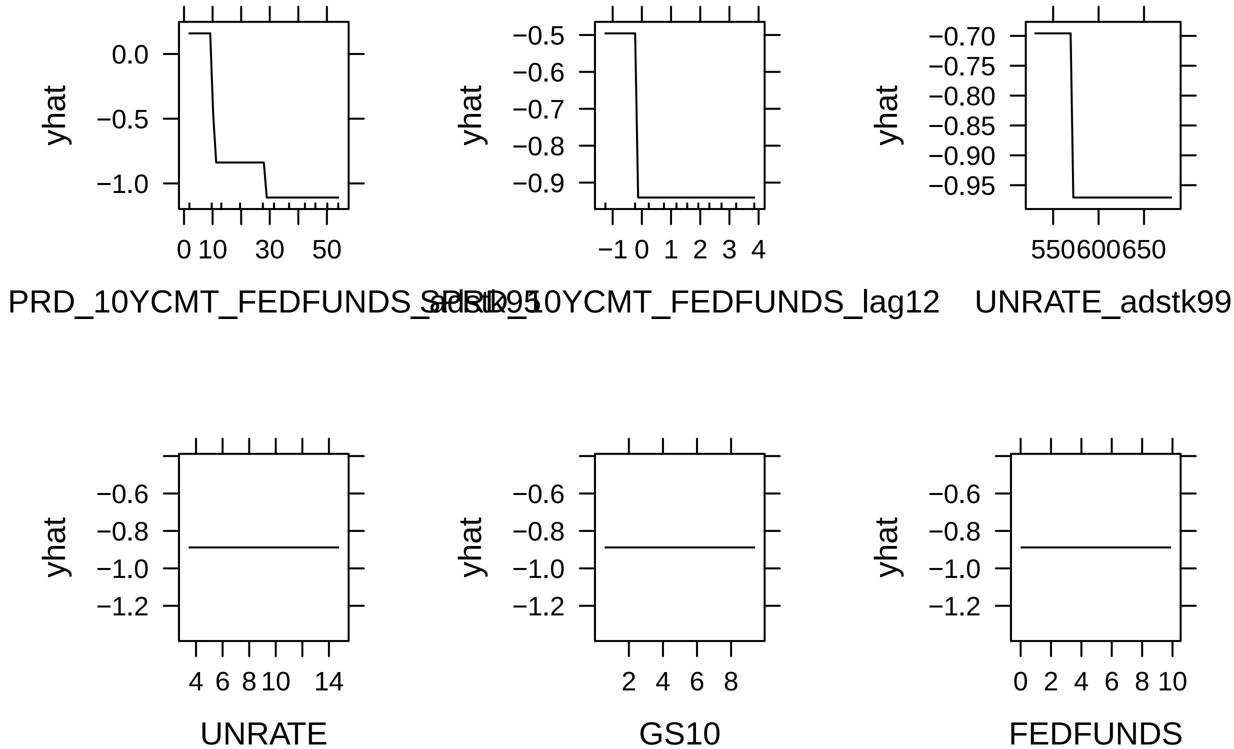
variable	Overall
SPRD_10YCMT_FEDFUNDS_adstk95	100.00000

variable	Overall
SPRD_10YCMT_FEDFUNDS_lag12	32.24702
UNRATE_adstk99	16.56033
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_adstk85	0.00000
UNRATE_adstk91	0.00000
UNRATE_adstk92	0.00000
UNRATE_adstk93	0.00000
UNRATE_adstk94	0.00000
UNRATE_adstk95	0.00000
GS10_adstk85	0.00000
GS10_adstk91	0.00000
GS10_adstk92	0.00000
GS10_adstk93	0.00000
GS10_adstk94	0.00000
GS10_adstk95	0.00000
GS10_adstk99	0.00000

variable	Overall
FEDFUNDS_adstk85	0.00000
FEDFUNDS_adstk91	0.00000
FEDFUNDS_adstk92	0.00000
FEDFUNDS_adstk93	0.00000
FEDFUNDS_adstk94	0.00000
FEDFUNDS_adstk95	0.00000
FEDFUNDS_adstk99	0.00000
SPRD_10YCMT_FEDFUNDS_adstk85	0.00000
SPRD_10YCMT_FEDFUNDS_adstk91	0.00000
SPRD_10YCMT_FEDFUNDS_adstk92	0.00000
SPRD_10YCMT_FEDFUNDS_adstk93	0.00000
SPRD_10YCMT_FEDFUNDS_adstk94	0.00000
SPRD_10YCMT_FEDFUNDS_adstk99	0.00000
D_UNRATE_adstk85	0.00000
D_UNRATE_adstk91	0.00000
D_UNRATE_adstk92	0.00000
D_UNRATE_adstk93	0.00000
D_UNRATE_adstk94	0.00000
D_UNRATE_adstk95	0.00000
D_UNRATE_adstk99	0.00000
G_CPIU_adstk85	0.00000
G_CPIU_adstk91	0.00000
G_CPIU_adstk92	0.00000
G_CPIU_adstk93	0.00000
G_CPIU_adstk94	0.00000
G_CPIU_adstk95	0.00000
G_CPIU_adstk99	0.00000
D_EFFR_adstk85	0.00000
D_EFFR_adstk91	0.00000
D_EFFR_adstk92	0.00000
D_EFFR_adstk93	0.00000
D_EFFR_adstk94	0.00000
D_EFFR_adstk95	0.00000
D_EFFR_adstk99	0.00000
D_GS10_adstk85	0.00000
D_GS10_adstk91	0.00000
D_GS10_adstk92	0.00000
D_GS10_adstk93	0.00000
D_GS10_adstk94	0.00000
D_GS10_adstk95	0.00000
D_GS10_adstk99	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)
```

```r
pdp.top2 <- partial(xgb_mod,
          pred.var = df_imp$variable[2],
          plot = TRUE,
          rug = TRUE)

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

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

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

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

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



Peeking

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

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

best_predictor <- head(top_predictors, 1)

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

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

Logistic Regression (with peeking)

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

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

summary(logit_mod)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.85885   0.01615   0.05089   0.21740   1.44457
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -2.35475    0.47256  -4.983 6.26e-07 ***
## SPRD_10YCMT_FEDFUNDS_adstk95  0.22758    0.03434   6.627 3.43e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 323.97  on 399  degrees of freedom
## Residual deviance: 169.43  on 398  degrees of freedom
## AIC: 173.43
##
## Number of Fisher Scoring iterations: 8
```

Compare Models

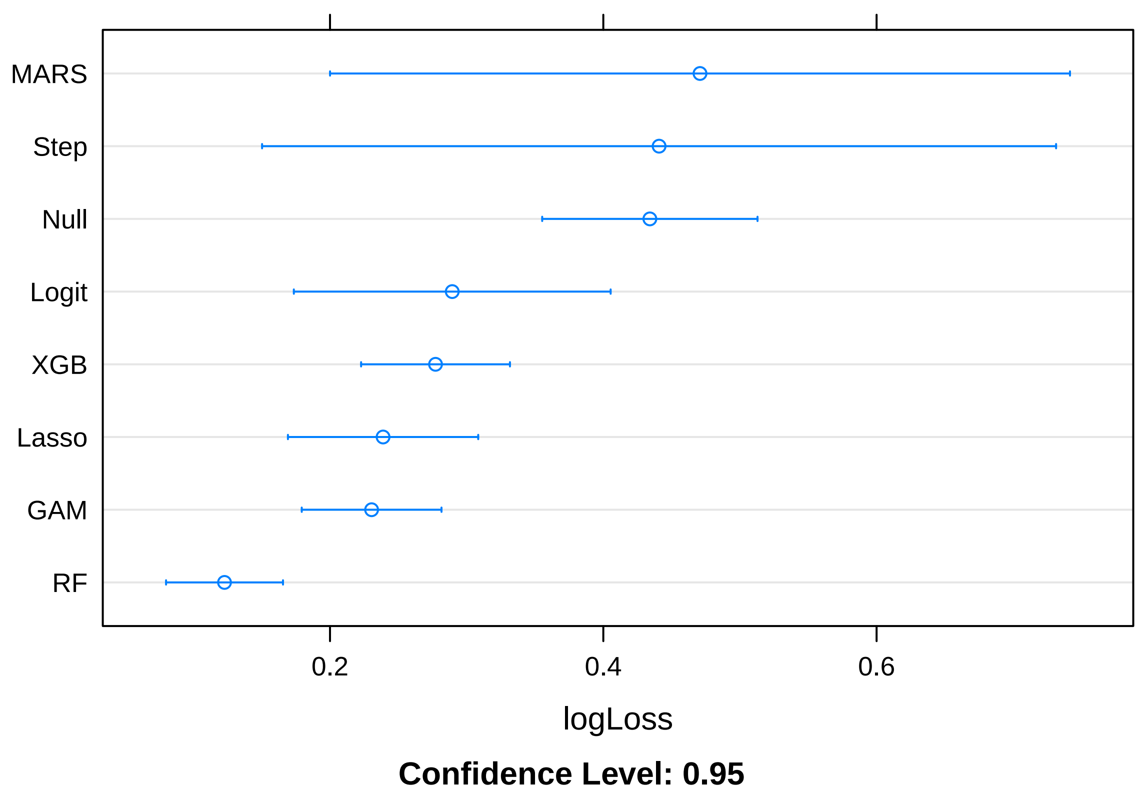
CV errors for models with peeking are misleadingly low. This will be addressed with a test set.

```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: 280
##
## logLoss
##               Min.      1st Qu.       Median      Mean       3rd Qu.      Max.
## XGB   5.602337e-02 7.602335e-02 9.952490e-02 0.2772170 0.1842396575  2.325495
## GAM   2.999232e-02 4.789796e-02 5.355077e-02 0.2304475 0.1707451150  3.358296
## RF    9.992007e-16 4.008021e-03 1.612938e-02 0.1228011 0.0882853626  3.729701
## Step  9.992007e-16 9.992007e-16 1.767475e-13 0.4408222 0.0004721395 34.538776
## Lasso 5.033888e-04 2.660753e-03 1.558069e-02 0.2388287 0.1083769170  4.758196
## MARS  9.992007e-16 4.364253e-03 3.867560e-02 0.4707717 0.0500827115 34.538776
## Null  8.940659e-02 1.395372e-01 1.562128e-01 0.4340732 0.1895729911  2.465489
## Logit 9.992007e-16 1.015517e-04 9.016187e-04 0.2894573 0.0238940266  6.026099
##       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	0.9627131
MARS	0.9595170
Lasso	0.9467330
XGB	0.9176136
RF	0.8417969
Null	0.5000000

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

model	LogLoss
XGB	0.3542969
RF	0.4518500
Lasso	0.5860258

model	LogLoss
Null	0.6349002
Step	0.6771614
Logit	0.6771614
GAM	0.7774056
MARS	2.7799214

Probability of Recession (Most Recent Month)

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

curr_data$date
```

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

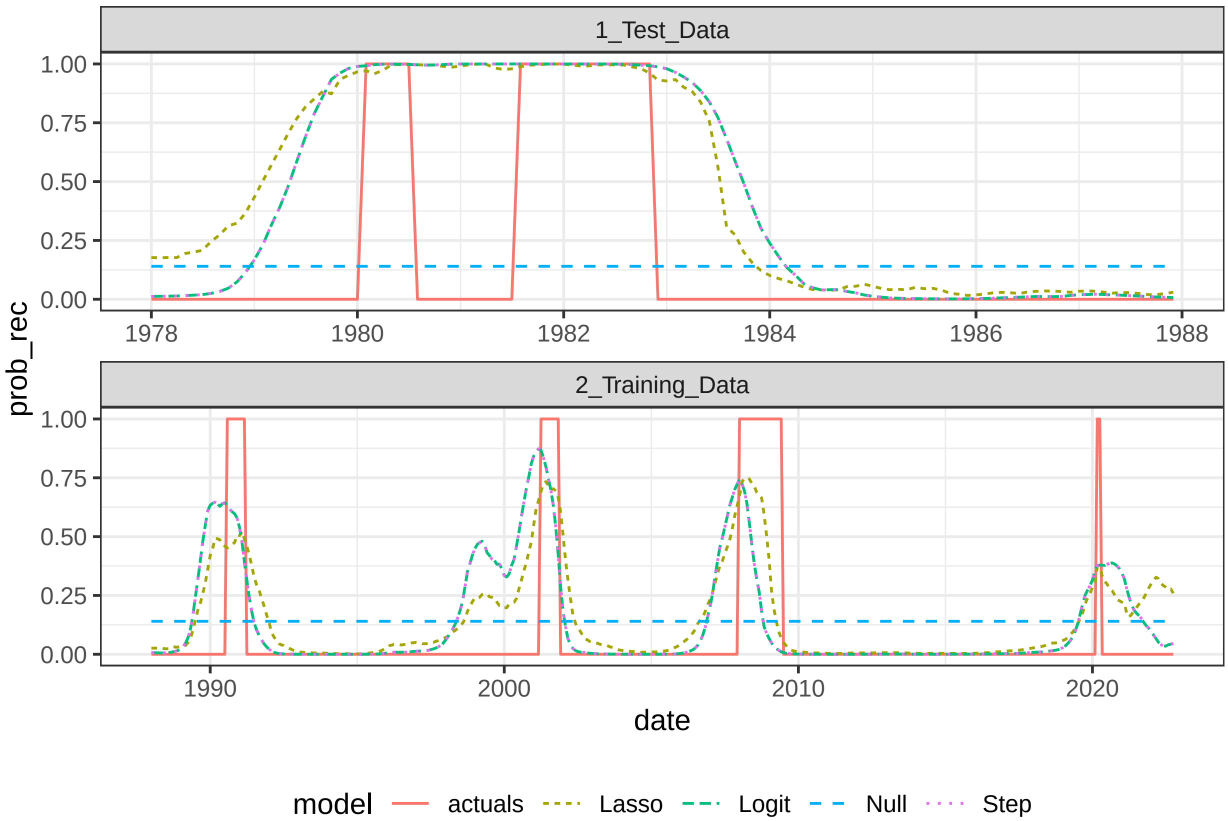
```
score_fun <- function(mods, dat) {
  output <- map_dfc(.x = mods, .f = function(x) {
    if(class(x)[1] != "train"){
      predict(x, newdata = dat, type = "response")
    } else(
       predict(x, newdata = dat, type = "prob")[,"yes"]

    )

  }) %>%
    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	19.85%
GAM	24.79%
RF	31.20%
Step	4.58%
Lasso	25.94%
MARS	5.15%
Null	14.00%
Logit	4.58%

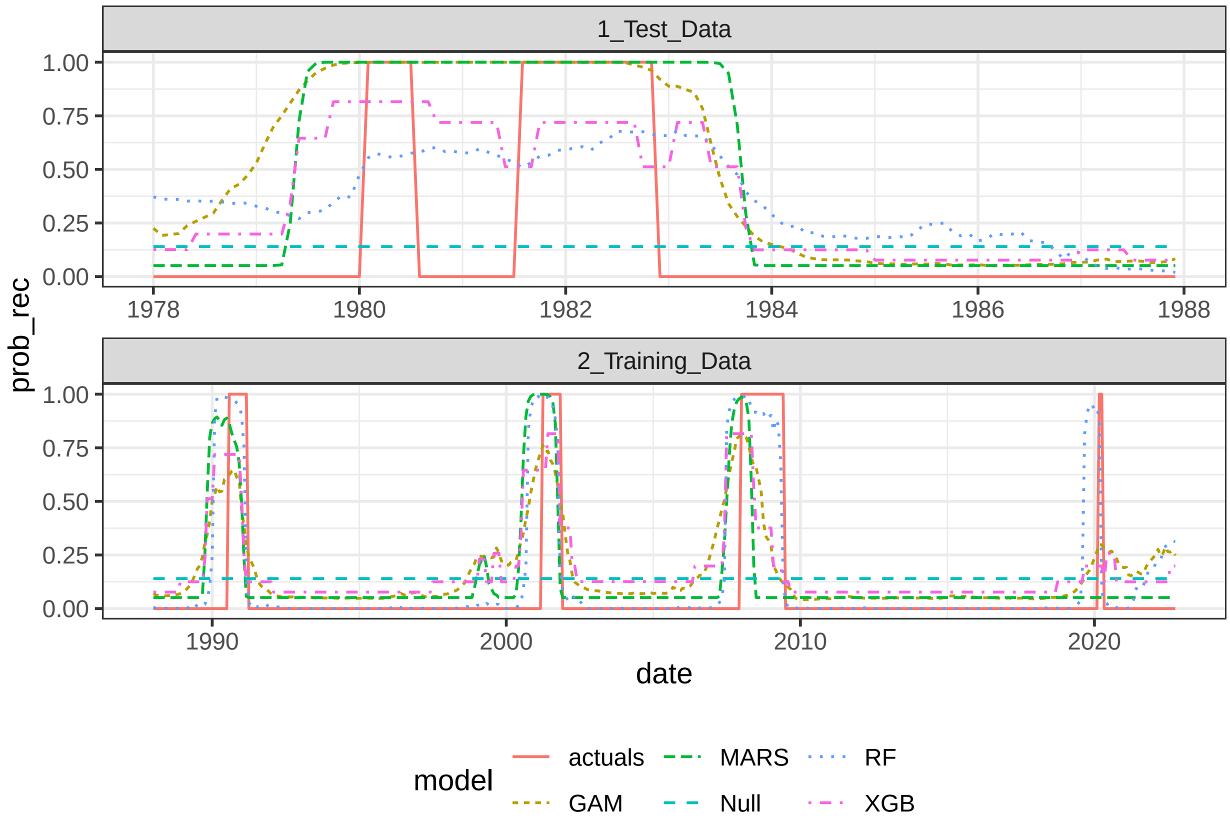
Backtesting

```
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'))
```

```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)
```