# SPY ETF ML Trading Strategy Analysis

Algorithmic Trading System

2025-07-16

## Contents

# Data Load

The numeric suffix for each data set represents 1. ATR multiplier 2. Trade timeout duration (minutes). View data_mining.ipynb for how this is populated.

```
data1_3 = read.csv("../csvs/train1_3.csv")
```

# Data exploration for 1_3

```
summary(data1_3)
```

```
##       X2                 Open           Close            High
##  Length:21492       Min.   :416.3   Min.   :416.3   Min.   :416.4
##  Class :character   1st Qu.:428.8   1st Qu.:428.8   1st Qu.:428.8
##  Mode  :character   Median :435.8   Median :435.8   Median :435.9
##                     Mean   :434.4   Mean   :434.4   Mean   :434.5
##                     3rd Qu.:439.8   3rd Qu.:439.8   3rd Qu.:439.8
##                     Max.   :450.4   Max.   :450.4   Max.   :450.4
##       Low              Volume             ATR                PL
##  Min.   :416.2   Min.   :      1   Min.   :0.006399   Min.   :0.0000
##  1st Qu.:428.7   1st Qu.:   2016   1st Qu.:0.067450   1st Qu.:0.0000
##  Median :435.7   Median :  54740   Median :0.106248   Median :0.0000
##  Mean   :434.3   Mean   :  83707   Mean   :0.130528   Mean   :0.4414
##  3rd Qu.:439.7   3rd Qu.: 110482   3rd Qu.:0.166756   3rd Qu.:1.0000
##  Max.   :450.2   Max.   :4816391   Max.   :1.285800   Max.   :1.0000
##     pl_value             SMA_k7          SMA_k20          SMA_k50
##  Min.   :-845.0000   Min.   :0.9966   Min.   :0.9938   Min.   :0.9930
##  1st Qu.: -90.2023   1st Qu.:0.9998   1st Qu.:0.9997   1st Qu.:0.9996
##  Median : -20.0000   Median :1.0000   Median :1.0000   Median :1.0001
##  Mean   :   0.5364   Mean   :1.0000   Mean   :1.0000   Mean   :1.0001
##  3rd Qu.:  90.0000   3rd Qu.:1.0002   3rd Qu.:1.0003   3rd Qu.:1.0006
##  Max.   :1610.0000   Max.   :1.0047   Max.   :1.0048   Max.   :1.0057
```

## View Summary and shape

### ATR Multiplier (Lambda)

**Mean Price Calculation:**

$$\mu_T = \frac{\sum_{i=T-k}^{T}(OHLC_i)}{k}$$

**Modified ATR Formula:**

$$ATR_T = \sqrt{\frac{\sum_{i=T-k}^{T}(OHLC_i - \mu_T)^2}{k}}$$

Controls the sensitivity to market volatility:

- **Purpose**: Scales profit/loss targets based on current market conditions
- **Example**: If SPY ATR = 0.50 and $\lambda = 10$:
  - Base volatility adjustment = $10 \times 0.50 = \$5.00$
  - Sell if SPY is up 5$ from entry price

### pl_value

It is determined by iterating through the dataframe and seeing if the price reaches take profit or stop loss first. It assumes a 1000 share position size.

**Profit-to-Loss Ratio (Chi)** Asymmetric risk-reward ratio: - **Default**: 1.5 (profit targets 50% wider than loss targets)

- **Profit Target**: $Price_T + \lambda Ö\chi ÖATR$
- **Loss Target**: $Price_T - \lambda ÖATR$

**Timeout Period (t)** Maximum holding period before forced exit:

- **Purpose**: Prevents indefinite position holding
- **Logic**: If neither target hit within t periods, compare exit price to entry
- **Classification**: Profit (1) if $Price_{T+t} > Price_T$, Loss (0) otherwise

### SMA

$$SMA_k = \frac{Open_T}{\frac{1}{k}\sum_{i=T-k}^{T-1} Close_i}$$

- $T$ = current time period
- $k$ = lookback period
- Hypothetically, values > 1.0 indicate price above historical average (bullish)
- Values < 1.0 indicate price below historical average (bearish)

**Multi-Timeframe Analysis** The system calculates SMA for three periods:

- **SMA_7**: Short-term momentum (1 week of 5-min bars)
- **SMA_20**: Medium-term trend (1 month of daily closes)
- **SMA_50**: Long-term trend (quarterly trend)

**Signal Interpretation**

- **Trend Confirmation**: Multiple SMAs above 1.0 = strong uptrend
- **Momentum Divergence**: SMA_7 > SMA_20 > SMA_50 = accelerating uptrend
- **Mean Reversion**: Extreme SMA values (>1.05 or <0.95) suggest potential reversal

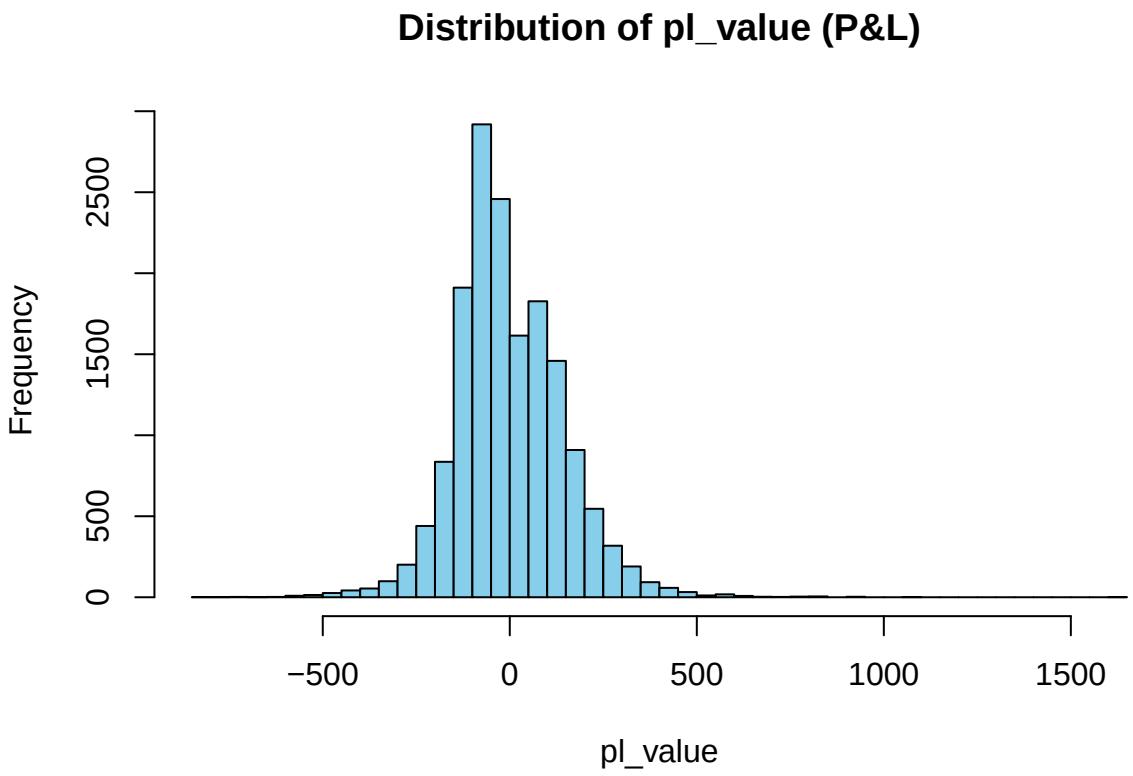
See data_mining.ipynb for information

**PL: Encodes 0 if `pl_value` < 0**

Split train-test data

```
seed_num = 213
set.seed(seed_num) # reproducibility
data1_3$norm_volume = (data1_3$Volume - mean(data1_3$Volume))/sd(data1_3$Volume)
ind = sample(1:nrow(data1_3),0.75*nrow(data1_3))
train = data1_3[ind,]
test = data1_3[-ind,]
```

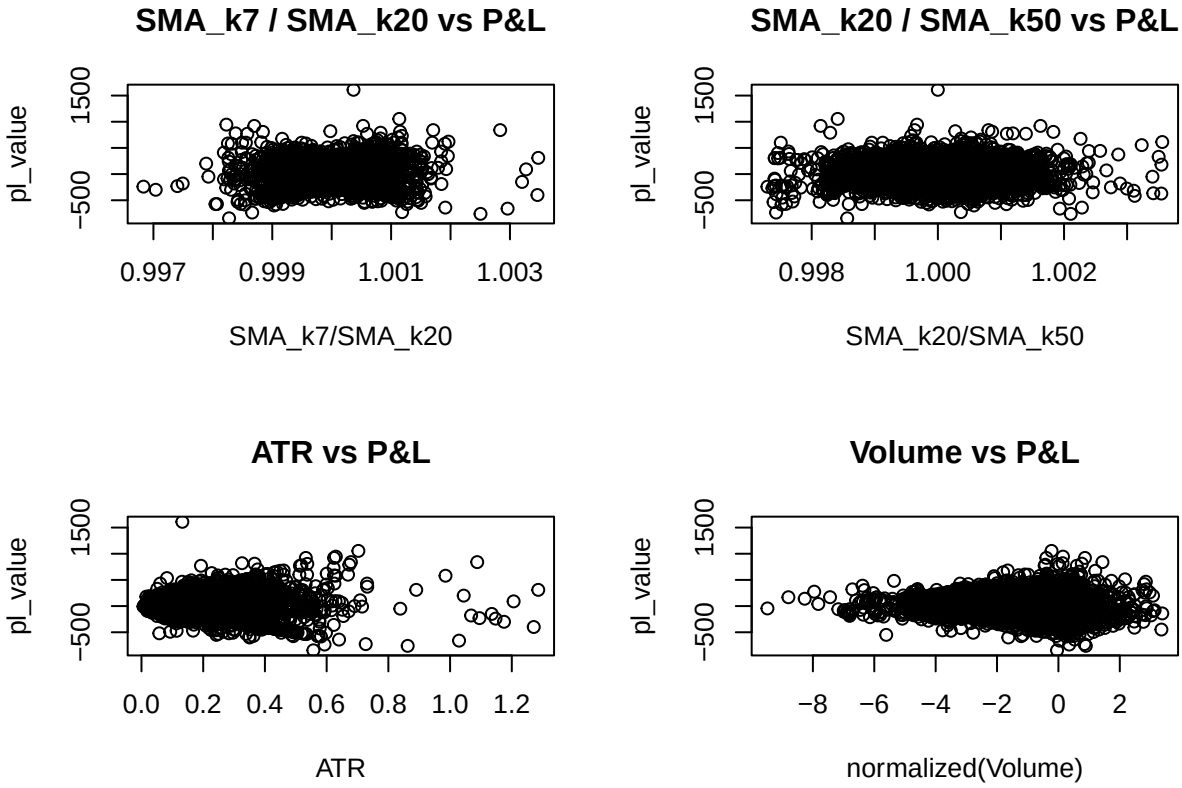
```
hist(train$pl_value, breaks = 50, col = "skyblue", main = "Distribution of pl_value (P&L)", xlab = "pl_
```




### Assess correlation

```
par(mfrow=c(2,2))
plot(train$SMA_k7/train$SMA_k20, train$pl_value, main="SMA_k7 / SMA_k20 vs P&L", xlab="SMA_k7/SMA_k20",
plot(train$SMA_k20/train$SMA_k50, train$pl_value, main="SMA_k20 / SMA_k50 vs P&L", xlab="SMA_k20/SMA_k5
plot(train$ATR, train$pl_value, main="ATR vs P&L", xlab="ATR", ylab="pl_value")
plot(log(train$norm_volume), train$pl_value, main="Volume vs P&L", xlab="normalized(Volume)", ylab="pl_
```

```
## Warning in log(train$norm_volume): NaNs produced
```




```
par(mfrow=c(1,1))
```

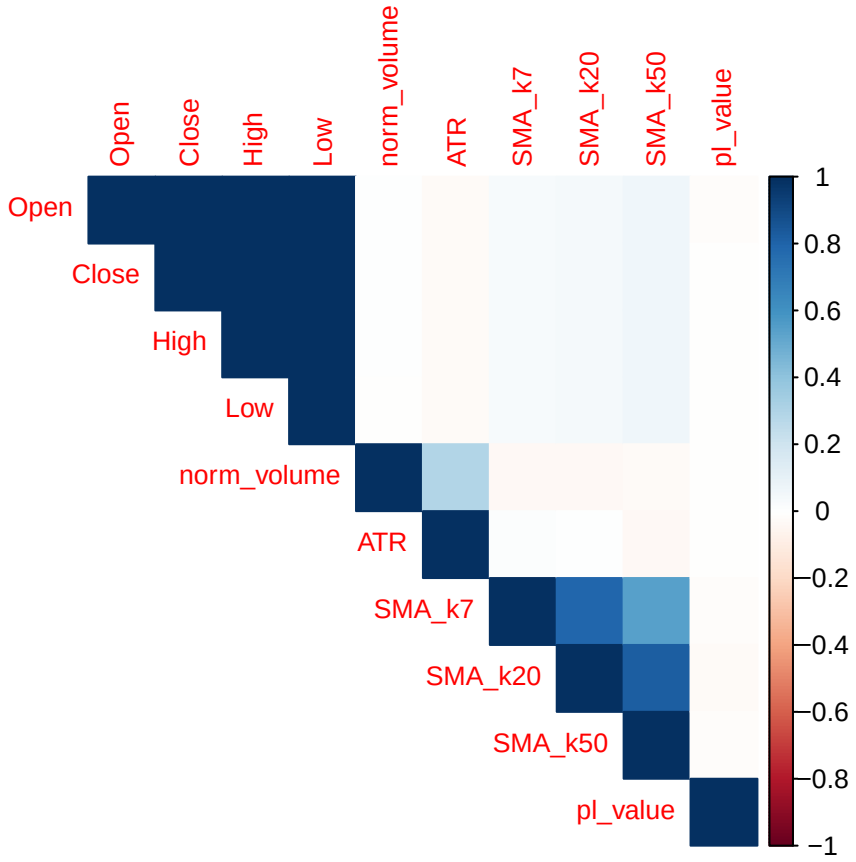
ATR shows to have a weak correlation. I think the interaction of predictors with each other will be more important.

### Visualize predictor correlation matrix

```
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
library(ggplot2)
numeric_vars <- train[, c("Open", "Close", "High", "Low", "norm_volume", "ATR", "SMA_k7", "SMA_k20", "SM
cor_matrix <- cor(numeric_vars, use="complete.obs")

corrplot(cor_matrix, method = "color", type = "upper", tl.cex = 0.8)
```




## Log regression

### Model Diagnostics and Theoretical Foundations

Logistic regression models the log-odds of a binary outcome as a linear function of predictors:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k$$

Where $p$ is the probability of a profitable trade. The coefficients represent the change in log-odds associated with a one-unit increase in the predictor, holding all other variables constant. We use all parameters and all interactions initially

```
glm1 = glm(
  data = train,
  PL ~ ATR + SMA_k7 + SMA_k20 + SMA_k50 + norm_volume +
       ATR:SMA_k7 +
       ATR:SMA_k20 +
       ATR:SMA_k50 +
       ATR:SMA_k7:SMA_k20 +
       ATR:SMA_k7:SMA_k50 +
       ATR:SMA_k20:SMA_k50 +
       ATR:SMA_k7:SMA_k20:SMA_k50 +
       norm_volume:ATR +
       norm_volume:SMA_k7 +
```

```
      norm_volume:SMA_k20 +
      norm_volume:SMA_k50 +
      norm_volume:ATR:SMA_k7 +
      norm_volume:ATR:SMA_k20 +
      norm_volume:ATR:SMA_k50 +
      norm_volume:SMA_k7:SMA_k20 +
      norm_volume:SMA_k7:SMA_k50 +
      norm_volume:SMA_k20:SMA_k50 +
      norm_volume:ATR:SMA_k7:SMA_k20 +
      norm_volume:ATR:SMA_k7:SMA_k50 +
      norm_volume:ATR:SMA_k20:SMA_k50 +
      norm_volume:ATR:SMA_k7:SMA_k20:SMA_k50,
  family = "binomial"
)
```

**Assess**

```
summary(glm1)
```

```
##
## Call:
## glm(formula = PL ~ ATR + SMA_k7 + SMA_k20 + SMA_k50 + norm_volume +
##     ATR:SMA_k7 + ATR:SMA_k20 + ATR:SMA_k50 + ATR:SMA_k7:SMA_k20 +
##     ATR:SMA_k7:SMA_k50 + ATR:SMA_k20:SMA_k50 + ATR:SMA_k7:SMA_k20:SMA_k50 +
##     norm_volume:ATR + norm_volume:SMA_k7 + norm_volume:SMA_k20 +
##     norm_volume:SMA_k50 + norm_volume:ATR:SMA_k7 + norm_volume:ATR:SMA_k20 +
##     norm_volume:ATR:SMA_k50 + norm_volume:SMA_k7:SMA_k20 + norm_volume:SMA_k7:SMA_k50 +
##     norm_volume:SMA_k20:SMA_k50 + norm_volume:ATR:SMA_k7:SMA_k20 +
##     norm_volume:ATR:SMA_k7:SMA_k50 + norm_volume:ATR:SMA_k20:SMA_k50 +
##     norm_volume:ATR:SMA_k7:SMA_k20:SMA_k50, family = "binomial",
##     data = train)
##
## Coefficients:
##                                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)                          7.141e+01  9.228e+01   0.774  0.43902
## ATR                                 -7.997e+07  5.275e+07  -1.516  0.12950
## SMA_k7                              -3.994e+02  1.457e+02  -2.742  0.00611
## SMA_k20                              3.469e+02  1.414e+02   2.453  0.01416
## SMA_k50                             -1.938e+01  6.485e+01  -0.299  0.76504
## norm_volume                          2.937e+03  6.251e+04   0.047  0.96252
## ATR:SMA_k7                           7.980e+07  5.273e+07   1.513  0.13023
## ATR:SMA_k20                          7.991e+07  5.279e+07   1.514  0.13007
## ATR:SMA_k50                          8.030e+07  5.267e+07   1.525  0.12733
## ATR:norm_volume                      4.502e+07  2.624e+07   1.716  0.08618
## SMA_k7:norm_volume                  -1.013e+05  8.826e+04  -1.148  0.25094
## SMA_k20:norm_volume                  3.479e+05  1.587e+05   2.192  0.02835
## SMA_k50:norm_volume                 -2.527e+05  1.218e+05  -2.075  0.03801
## ATR:SMA_k7:SMA_k20                  -7.974e+07  5.277e+07  -1.511  0.13079
## ATR:SMA_k7:SMA_k50                  -8.013e+07  5.265e+07  -1.522  0.12805
## ATR:SMA_k20:SMA_k50                 -8.024e+07  5.270e+07  -1.523  0.12788
## ATR:SMA_k7:norm_volume              -4.491e+07  2.622e+07  -1.713  0.08679
```

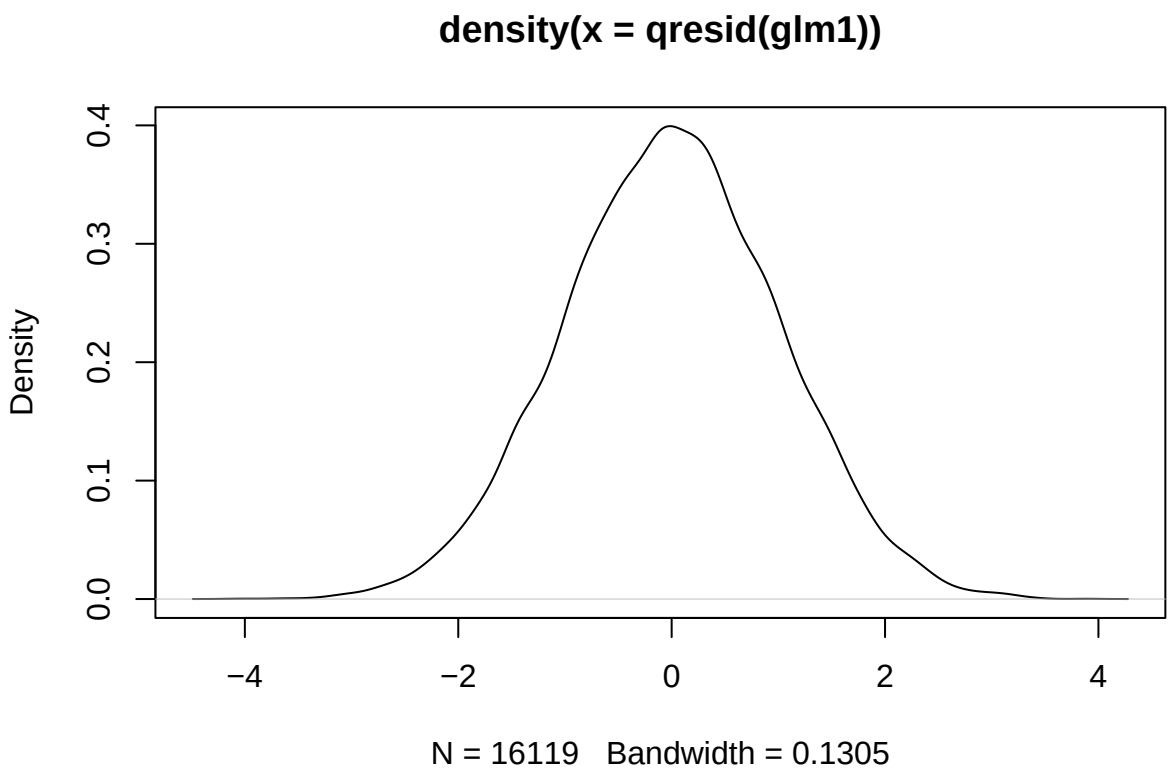
```
## ATR:SMA_k20:norm_volume                -4.541e+07  2.624e+07  -1.731  0.08351
## ATR:SMA_k50:norm_volume                -4.487e+07  2.632e+07  -1.705  0.08821
## SMA_k7:SMA_k20:norm_volume             -2.495e+05  1.494e+05  -1.671  0.09478
## SMA_k7:SMA_k50:norm_volume              3.514e+05  1.371e+05   2.564  0.01034
## SMA_k20:SMA_k50:norm_volume            -9.870e+04  5.308e+04  -1.859  0.06296
## ATR:SMA_k7:SMA_k20:SMA_k50              8.007e+07  5.269e+07   1.520  0.12859
## ATR:SMA_k7:SMA_k20:norm_volume          4.530e+07  2.622e+07   1.727  0.08409
## ATR:SMA_k7:SMA_k50:norm_volume          4.476e+07  2.630e+07   1.702  0.08884
## ATR:SMA_k20:SMA_k50:norm_volume         4.527e+07  2.632e+07   1.720  0.08542
## ATR:SMA_k7:SMA_k20:SMA_k50:norm_volume -4.515e+07  2.630e+07  -1.717  0.08602
##
## (Intercept)
## ATR
## SMA_k7                                 **
## SMA_k20                                *
## SMA_k50
## norm_volume
## ATR:SMA_k7
## ATR:SMA_k20
## ATR:SMA_k50
## ATR:norm_volume                        .
## SMA_k7:norm_volume
## SMA_k20:norm_volume                    *
## SMA_k50:norm_volume                    *
## ATR:SMA_k7:SMA_k20
## ATR:SMA_k7:SMA_k50
## ATR:SMA_k20:SMA_k50
## ATR:SMA_k7:norm_volume                 .
## ATR:SMA_k20:norm_volume                .
## ATR:SMA_k50:norm_volume                .
## SMA_k7:SMA_k20:norm_volume             .
## SMA_k7:SMA_k50:norm_volume             *
## SMA_k20:SMA_k50:norm_volume            .
## ATR:SMA_k7:SMA_k20:SMA_k50
## ATR:SMA_k7:SMA_k20:norm_volume         .
## ATR:SMA_k7:SMA_k50:norm_volume         .
## ATR:SMA_k20:SMA_k50:norm_volume        .
## ATR:SMA_k7:SMA_k20:SMA_k50:norm_volume .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22118  on 16118  degrees of freedom
## Residual deviance: 22013  on 16092  degrees of freedom
## AIC: 22067
##
## Number of Fisher Scoring iterations: 6
```

**Residuals analysis**

Regular residual plots don't make sense in glm. Dunn and Gordon (2018) introduce quantile residuals for discrete response variables. Their primary benefits are they do not show weird patterns (due to variable's

discreteness). They are available in R via statmod::qresid().

```
library(statmod)
plot(density(qresid(glm1)))
```




This residuals do appear normally distributed. Which means there is no unexplained variance (non linearity or omitted variables) in the model.

### Variable Selection

The Akaike Information Criterion (AIC) balances model fit against complexity:

$$AIC = -2\ln(L) + 2k$$

Where $L$ is the likelihood and $k$ is the number of parameters. Lower AIC values indicate better models. The stepwise procedure systematically adds or removes variables to minimize AIC, providing a theoretically sound approach to model selection. Let's use forward AIC to trim some of the predictors. AIC is usually computationally expensive, however I am not dealing with a lot of predictors here.

```
glm_step <- step(glm1, direction = "both")
```

```
## Start:  AIC=22067.2
## PL ~ ATR + SMA_k7 + SMA_k20 + SMA_k50 + norm_volume + ATR:SMA_k7 +
##     ATR:SMA_k20 + ATR:SMA_k50 + ATR:SMA_k7:SMA_k20 + ATR:SMA_k7:SMA_k50 +
```

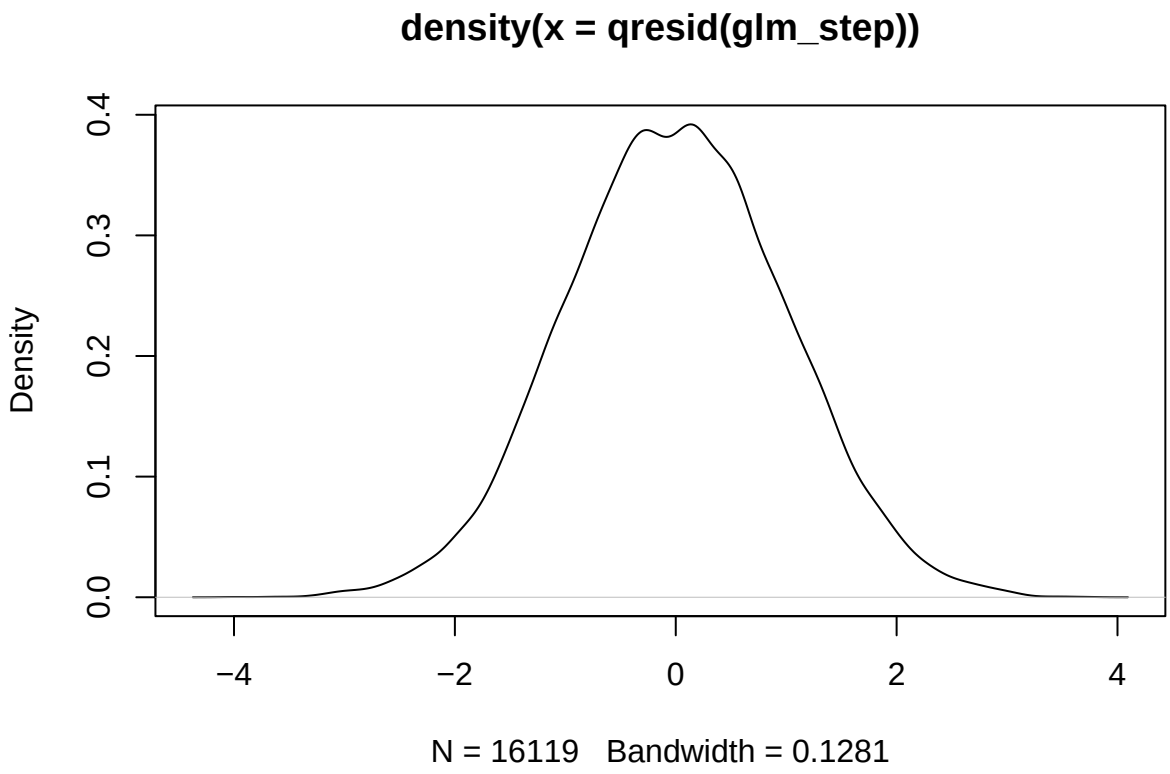
```
##     ATR:SMA_k20:SMA_k50 + ATR:SMA_k7:SMA_k20:SMA_k50 + norm_volume:ATR +
##     norm_volume:SMA_k7 + norm_volume:SMA_k20 + norm_volume:SMA_k50 +
##     norm_volume:ATR:SMA_k7 + norm_volume:ATR:SMA_k20 + norm_volume:ATR:SMA_k50 +
##     norm_volume:SMA_k7:SMA_k20 + norm_volume:SMA_k7:SMA_k50 +
##     norm_volume:SMA_k20:SMA_k50 + norm_volume:ATR:SMA_k7:SMA_k20 +
##     norm_volume:ATR:SMA_k7:SMA_k50 + norm_volume:ATR:SMA_k20:SMA_k50 +
##     norm_volume:ATR:SMA_k7:SMA_k20:SMA_k50
##
##                                         Df Deviance   AIC
## <none>                                       22013 22067
## - ATR:SMA_k7:SMA_k20:SMA_k50:norm_volume  1    22017 22069
```

```
summary(glm_step)
```

```
##
## Call:
## glm(formula = PL ~ ATR + SMA_k7 + SMA_k20 + SMA_k50 + norm_volume +
##     ATR:SMA_k7 + ATR:SMA_k20 + ATR:SMA_k50 + ATR:SMA_k7:SMA_k20 +
##     ATR:SMA_k7:SMA_k50 + ATR:SMA_k20:SMA_k50 + ATR:SMA_k7:SMA_k20:SMA_k50 +
##     norm_volume:ATR + norm_volume:SMA_k7 + norm_volume:SMA_k20 +
##     norm_volume:SMA_k50 + norm_volume:ATR:SMA_k7 + norm_volume:ATR:SMA_k20 +
##     norm_volume:ATR:SMA_k50 + norm_volume:SMA_k7:SMA_k20 + norm_volume:SMA_k7:SMA_k50 +
##     norm_volume:SMA_k20:SMA_k50 + norm_volume:ATR:SMA_k7:SMA_k20 +
##     norm_volume:ATR:SMA_k7:SMA_k50 + norm_volume:ATR:SMA_k20:SMA_k50 +
##     norm_volume:ATR:SMA_k7:SMA_k20:SMA_k50, family = "binomial",
##     data = train)
##
## Coefficients:
##                                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)                             7.141e+01  9.228e+01   0.774  0.43902
## ATR                                    -7.997e+07  5.275e+07  -1.516  0.12950
## SMA_k7                                 -3.994e+02  1.457e+02  -2.742  0.00611
## SMA_k20                                 3.469e+02  1.414e+02   2.453  0.01416
## SMA_k50                                -1.938e+01  6.485e+01  -0.299  0.76504
## norm_volume                             2.937e+03  6.251e+04   0.047  0.96252
## ATR:SMA_k7                              7.980e+07  5.273e+07   1.513  0.13023
## ATR:SMA_k20                             7.991e+07  5.279e+07   1.514  0.13007
## ATR:SMA_k50                             8.030e+07  5.267e+07   1.525  0.12733
## ATR:norm_volume                         4.502e+07  2.624e+07   1.716  0.08618
## SMA_k7:norm_volume                     -1.013e+05  8.826e+04  -1.148  0.25094
## SMA_k20:norm_volume                     3.479e+05  1.587e+05   2.192  0.02835
## SMA_k50:norm_volume                    -2.527e+05  1.218e+05  -2.075  0.03801
## ATR:SMA_k7:SMA_k20                     -7.974e+07  5.277e+07  -1.511  0.13079
## ATR:SMA_k7:SMA_k50                     -8.013e+07  5.265e+07  -1.522  0.12805
## ATR:SMA_k20:SMA_k50                    -8.024e+07  5.270e+07  -1.523  0.12788
## ATR:SMA_k7:norm_volume                 -4.491e+07  2.622e+07  -1.713  0.08679
## ATR:SMA_k20:norm_volume                -4.541e+07  2.624e+07  -1.731  0.08351
## ATR:SMA_k50:norm_volume                -4.487e+07  2.632e+07  -1.705  0.08821
## SMA_k7:SMA_k20:norm_volume             -2.495e+05  1.494e+05  -1.671  0.09478
## SMA_k7:SMA_k50:norm_volume              3.514e+05  1.371e+05   2.564  0.01034
## SMA_k20:SMA_k50:norm_volume            -9.870e+04  5.308e+04  -1.859  0.06296
## ATR:SMA_k7:SMA_k20:SMA_k50              8.007e+07  5.269e+07   1.520  0.12859
## ATR:SMA_k7:SMA_k20:norm_volume          4.530e+07  2.622e+07   1.727  0.08409
## ATR:SMA_k7:SMA_k50:norm_volume          4.476e+07  2.630e+07   1.702  0.08884
```

```
## ATR:SMA_k20:SMA_k50:norm_volume         4.527e+07  2.632e+07   1.720  0.08542
## ATR:SMA_k7:SMA_k20:SMA_k50:norm_volume -4.515e+07  2.630e+07  -1.717  0.08602
##
## (Intercept)
## ATR
## SMA_k7                                 **
## SMA_k20                                *
## SMA_k50
## norm_volume
## ATR:SMA_k7
## ATR:SMA_k20
## ATR:SMA_k50
## ATR:norm_volume                        .
## SMA_k7:norm_volume
## SMA_k20:norm_volume                    *
## SMA_k50:norm_volume                    *
## ATR:SMA_k7:SMA_k20
## ATR:SMA_k7:SMA_k50
## ATR:SMA_k20:SMA_k50
## ATR:SMA_k7:norm_volume                 .
## ATR:SMA_k20:norm_volume                .
## ATR:SMA_k50:norm_volume                .
## SMA_k7:SMA_k20:norm_volume             .
## SMA_k7:SMA_k50:norm_volume             *
## SMA_k20:SMA_k50:norm_volume            .
## ATR:SMA_k7:SMA_k20:SMA_k50
## ATR:SMA_k7:SMA_k20:norm_volume         .
## ATR:SMA_k7:SMA_k50:norm_volume         .
## ATR:SMA_k20:SMA_k50:norm_volume        .
## ATR:SMA_k7:SMA_k20:SMA_k50:norm_volume .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22118  on 16118  degrees of freedom
## Residual deviance: 22013  on 16092  degrees of freedom
## AIC: 22067
##
## Number of Fisher Scoring iterations: 6
```

```
plot(density(qresid(glm_step)))
```




This is still a lot of predictors, let's try and do some subset selection. This method uses cross validation and fits models with the least mean squared error. By splitting the data up we are able to understand what is actually needed by minimizing MSE.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-9
```

```
# Get interaction matrix
X <- model.matrix(PL ~ (ATR + SMA_k7 + SMA_k20 + SMA_k50 + norm_volume)^3, data = train)[, -1]
y <- train$PL
cvfit <- cv.glmnet(X, y, family = "binomial", alpha = 1)

# coefs with least mse
lasso_coef <- coef(cvfit, s = "lambda.min")
selected_vars <- rownames(lasso_coef)[lasso_coef[,1] != 0][-1]  # exclude intercept
selected_vars
```

```
## [1] "ATR"                 "norm_volume"         "ATR:SMA_k7"
## [4] "SMA_k50:norm_volume"
```

These are a lot fewer than earlier, let's fit this conservative model as well.

```r
glm_formula <- as.formula(paste("PL ~", paste(selected_vars, collapse = " + ")))
glm_cons <- glm(glm_formula, data = train, family = "binomial")
summary(glm_cons)
```

```
##
## Call:
## glm(formula = glm_formula, family = "binomial", data = train)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -0.37419    0.02796 -13.382   <2e-16 ***
## ATR                 -41.54800  153.21990  -0.271    0.786
## norm_volume           6.64180   12.80194   0.519    0.604
## ATR:SMA_k7           42.58200  153.21015   0.278    0.781
## norm_volume:SMA_k50  -6.67177   12.80731  -0.521    0.602
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 22118  on 16118  degrees of freedom
## Residual deviance: 22082  on 16114  degrees of freedom
## AIC: 22092
##
## Number of Fisher Scoring iterations: 4
```

**Assess performance**

```r
pred_glm = ifelse(predict(glm_step,test,type="response")>0.5,1,0)
winrate = mean(pred_glm == test$PL)
print(winrate)
```

```
## [1] 0.5585334
```

```r
pred_glm_cons = ifelse(predict(glm_cons,test,type="response")>0.5,1,0)
winrate = mean(pred_glm_cons == test$PL)
print(winrate)
```

```
## [1] 0.5592779
```

It appears that the conservative model performs worse. This accuracy is good, but it needs to be contextualized. The worst, bare bones model would trade on a coin flip. Let's simulate $nrow(test)$ coin flips and compare it to a 55% win rate.

Simulate 1000 trading simulations with random buy indicators and see if winrate if better than it. `p = 0.05`. This is to weed out strategies that don't perform better than random chance.

```r
n <- nrow(test)

# Simulate coin flips (baseline)
```

```r
baseline_accuracy = c()
for (i in 1:1000) {
  set.seed(i)
  coin_flips <- rbinom(n, size = 1, prob = 0.5)
  baseline_accuracy <- c(mean(coin_flips == test$PL), baseline_accuracy)
}
set.seed(seed_num) # reproducibility
cat("Coin toss simulations that performed better than strategy :",length(baseline_accuracy[baseline_accu
```

```
## Coin toss simulations that performed better than strategy : 0
```

```r
# Use normal approximation

cat("\nWinrate p value:",pnorm(winrate, mean(baseline_accuracy), sd(baseline_accuracy), lower.tail = F)
```

```
##
## Winrate p value: 2.369495e-18
```

This is definitely below the p value.

### Buy threshold

We need to determine when to buy based on the threshold. This improves the win rate at the cost of fewer trades. This will miss out on a lot of winning trades too (False negatives). But to succeed at trading we need to minimize false positives. Precision, or $\frac{TP}{FP+TP}$ is much more important. "For estimating a binomial proportion, at least 10 successes and 10 failures is recommended for normal approximation intervals." Reference: Agresti, A. (2013). Categorical Data Analysis (3rd ed.)

```r
# Define thresholds
probs <- 10^seq(-2, -10, length.out = 9)

fit_mean <- mean(glm_step$fitted.values)
fit_sd <- sd(glm_step$fitted.values)

prob_with_best_winrate = 0.05 # init
best_winrate = 0
for (prob in probs) {
  threshold <- qnorm(1 - prob, mean = fit_mean, sd = fit_sd)
  pred_glm <- ifelse(predict(glm_step, train, type = "response") > threshold, 1, 0)

  true_positive <- sum(pred_glm == 1 & train$PL == 1)
  false_positive <- sum(pred_glm == 1 & train$PL == 0)
  total_predicted_positive <- sum(pred_glm == 1)
  if ((true_positive + false_positive) < 30) {
    break # CLT
  }
  winrate <- if (total_predicted_positive > 0) {
    true_positive / total_predicted_positive
  } else {
    NA
  }
```

```
  if (winrate > best_winrate) {
    best_winrate = winrate
    prob_with_best_winrate = prob
  }

  cat(sprintf("Prob = %.5f | TP = %d | FP = %d | Winrate = %.3f\n",
              prob, true_positive, false_positive, winrate))
}
```

```
## Prob = 0.01000 | TP = 170 | FP = 140 | Winrate = 0.548
## Prob = 0.00100 | TP = 95 | FP = 78 | Winrate = 0.549
## Prob = 0.00010 | TP = 69 | FP = 48 | Winrate = 0.590
## Prob = 0.00001 | TP = 49 | FP = 29 | Winrate = 0.628
## Prob = 0.00000 | TP = 43 | FP = 19 | Winrate = 0.694
## Prob = 0.00000 | TP = 32 | FP = 12 | Winrate = 0.727
## Prob = 0.00000 | TP = 29 | FP = 8 | Winrate = 0.784
## Prob = 0.00000 | TP = 26 | FP = 6 | Winrate = 0.812
## Prob = 0.00000 | TP = 26 | FP = 4 | Winrate = 0.867
```

```
best_threshold = qnorm(1 - prob_with_best_winrate, mean = fit_mean, sd = fit_sd)
cat("Best winrate", best_winrate*100)
```

```
## Best winrate 86.66667
```

```
cat("\nProbability with highest precision", prob_with_best_winrate)
```

```
##
## Probability with highest precision 1e-10
```

```
cat("\nThreshold", best_threshold)
```

```
##
## Threshold 0.6893399
```

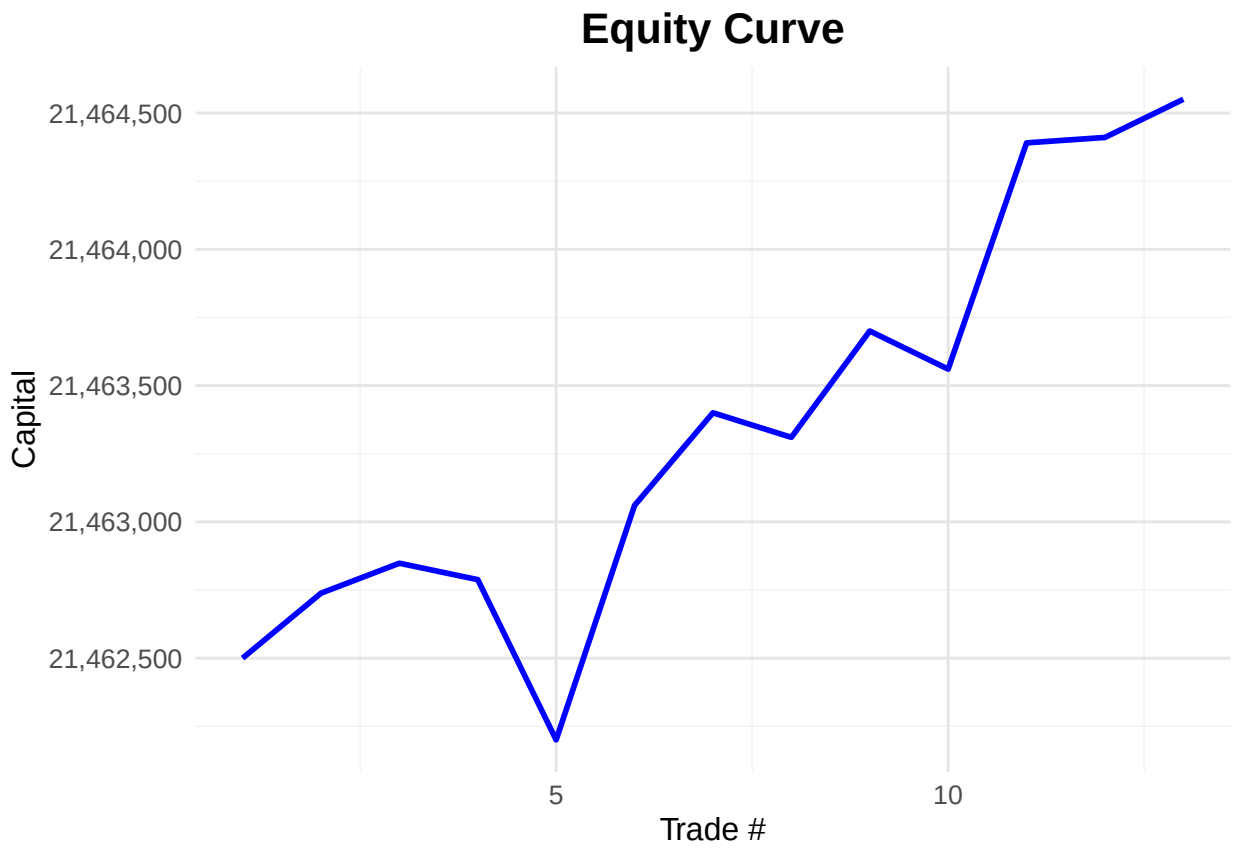
### Drawdown

`pl_value` is calculated with 1000 shares of SPY. In trading, using 2% of your portfolio in a trade is recommended, hence, The initial capital is set at $\frac{1}{0.02} * (1000 * Price)$.

```
initial_capital = (1/(0.02))*(1000 * sample(train$Open, 1)) # random price
capital <- initial_capital
drawdown <- c(capital)
wins <- c()

for (i in 1:nrow(test)) {
  pred_X <- predict(glm_step, test[i, ], type = "response")
  if (pred_X > best_threshold) {
    capital <- capital + test$pl_value[i]
    drawdown <- c(drawdown, capital)
    wins <- c(wins, test$pl_value[i])
```

```
  }
}

ggplot(data = data.frame(trade = 1:length(drawdown),
                         equity = drawdown),
       aes(x = trade, y = equity)) +
  geom_line(color = "blue", size = 1) +
  theme_minimal() +
  labs(title = "Equity Curve",
       x = "Trade #",
       y = "Capital") +
  theme(
    plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 12),
    axis.text = element_text(size = 10),
    panel.grid.major = element_line(color = "grey90"),
    panel.grid.minor = element_line(color = "grey95")
  ) +
  scale_y_continuous(labels = scales::comma)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



```r
# Sharpe ratio (simplified version: mean return over SD of returns)
sharpe <- (tail(drawdown, 1) - initial_capital) / sd(wins)


peak = max(drawdown)
trough = min(drawdown)
max_drawdown = (trough-peak)/peak


cat("Initial Capital", initial_capital)
```

```
## Initial Capital 21462500
```

```r
cat("\nFinal Capital:", tail(drawdown, 1), "\n")
```

```
##
## Final Capital: 21464550
```

```r
cat("Sharpe Ratio:", round(sharpe, 3), "\n")
```

```
## Sharpe Ratio: 5.053
```

```r
cat("Max Drawdown:", max_drawdown*100, "%\n")
```

```
## Max Drawdown: -0.01094934 %
```

```r
cat("Returns:", ((capital/initial_capital)-1)*100, "%\n")
```

```
## Returns: 0.009553513 %
```

## XGboost

In this section, we use **XGBoost**, a gradient-boosted decision tree ensemble, to classify trade direction. Our goal is not just accuracy but **precision**, the ratio of true positives to all predicted positives. This focus reduces false positives—critical for real-world trading where bad trades are costly.

### Load Data

We construct a design matrix using all two- and three-way interactions of trading indicators. The outcome variable `PL` is a binary indicator for whether a trade was profitable.

```r
library(xgboost)
library(Matrix)
library(caret)
```

```
## Loading required package: lattice
```

```
data = train
X = model.matrix(PL ~ (ATR + SMA_k7 + SMA_k20 + SMA_k50 + norm_volume)^3, data=data)[, -1]
y = data$PL
dtrain = xgb.DMatrix(data = X, label = y)
```

### XGBoost Theoretical Framework

XGBoost builds an ensemble of decision trees sequentially, with each new tree correcting errors made by previous trees. The algorithm optimizes:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

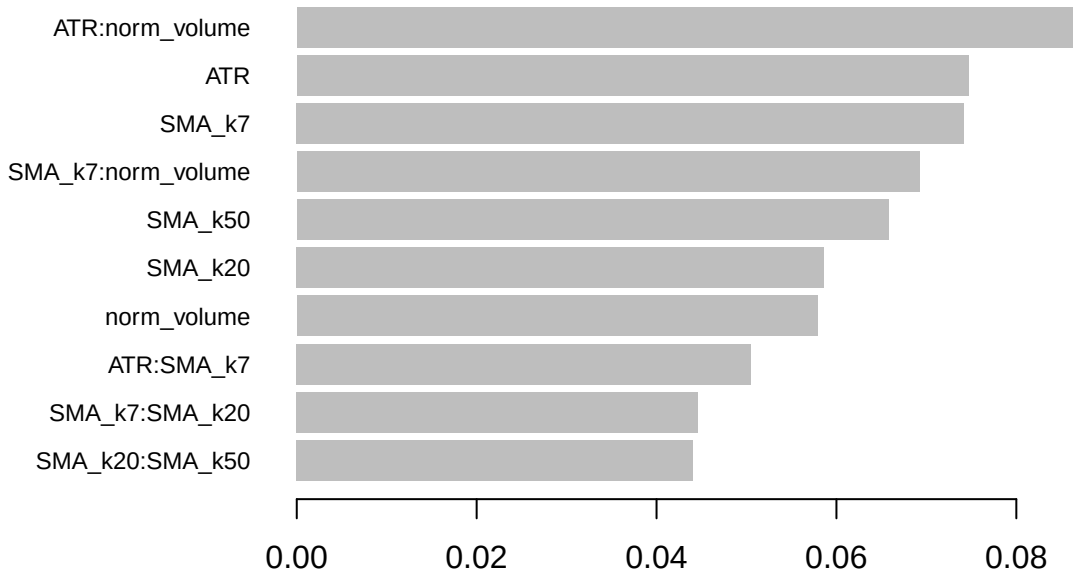
Where: - $l$ is a differentiable convex loss function - $\hat{y}_i$ is the prediction - $\Omega$ is a regularization term controlling model complexity - $f_k$ represents the $k$-th tree in the ensemble

This approach allows XGBoost to capture complex non-linear relationships and interactions that linear models might miss.

### Cross-Validation and Training

We optimize for **AUC-PR** (area under the precision-recall curve), which directly measures precision vs. recall trade-offs.

```
param_grid <- expand.grid(
  eta = 0.05,
  max_depth = 3,
  subsample = 0.8,
  colsample_bytree = 0.8,
  eval_metric = "aucpr",          # maximize precision-recall AUC
  objective = "binary:logistic"  # output probabilities
)

watchlist = list(train = dtrain)

model <- xgb.train(
  params = as.list(param_grid),
  data = dtrain,
  nrounds = 100,
  watchlist = watchlist,
  verbose = 0
)

importance_matrix <- xgb.importance(model = model)
xgb.plot.importance(importance_matrix[1:10,])
```




---

### Threshold Selection Based on Precision

Rather than defaulting to a 0.5 decision threshold, we **optimize threshold** $T$ to maximize:

$$\text{Precision} = \frac{TP}{TP + FP}$$

We assume model predictions $\hat{y} \sim N(\mu, \sigma^2)$ and scan high quantiles (e.g., 0.99999) to reduce false positives.

```
pred_probs <- predict(model, dtrain)
probs <- 10^seq(-2, -10, length.out = 9)

fit_mean = mean(pred_probs)
fit_sd = sd(pred_probs)

best_precision = 0
best_threshold = 0.5

for (prob in probs){
  threshold <- qnorm(1 - prob, mean = fit_mean, sd = fit_sd)
  preds = ifelse(pred_probs > threshold, 1, 0)
  TP = sum(preds == 1 & train$PL == 1)
  FP = sum(preds == 1 & train$PL == 0)
```

```
  if ((TP + FP) < 30) {
    break # CLT
  }
  precision = ifelse((TP + FP) == 0, NA, TP / (TP + FP))
  if (!is.na(precision) && precision > best_precision){
    best_precision = precision
    best_threshold = threshold
  }
}

cat("Best Precision:", round(best_precision * 100, 2), "% at Threshold", best_threshold, "\n")
```

```
## Best Precision: 83.67 % at Threshold 0.5804648
```

### Backtesting Equity Curve

Using the **test set**, we execute trades where the predicted probability $\hat{y} > T$, and track capital changes.

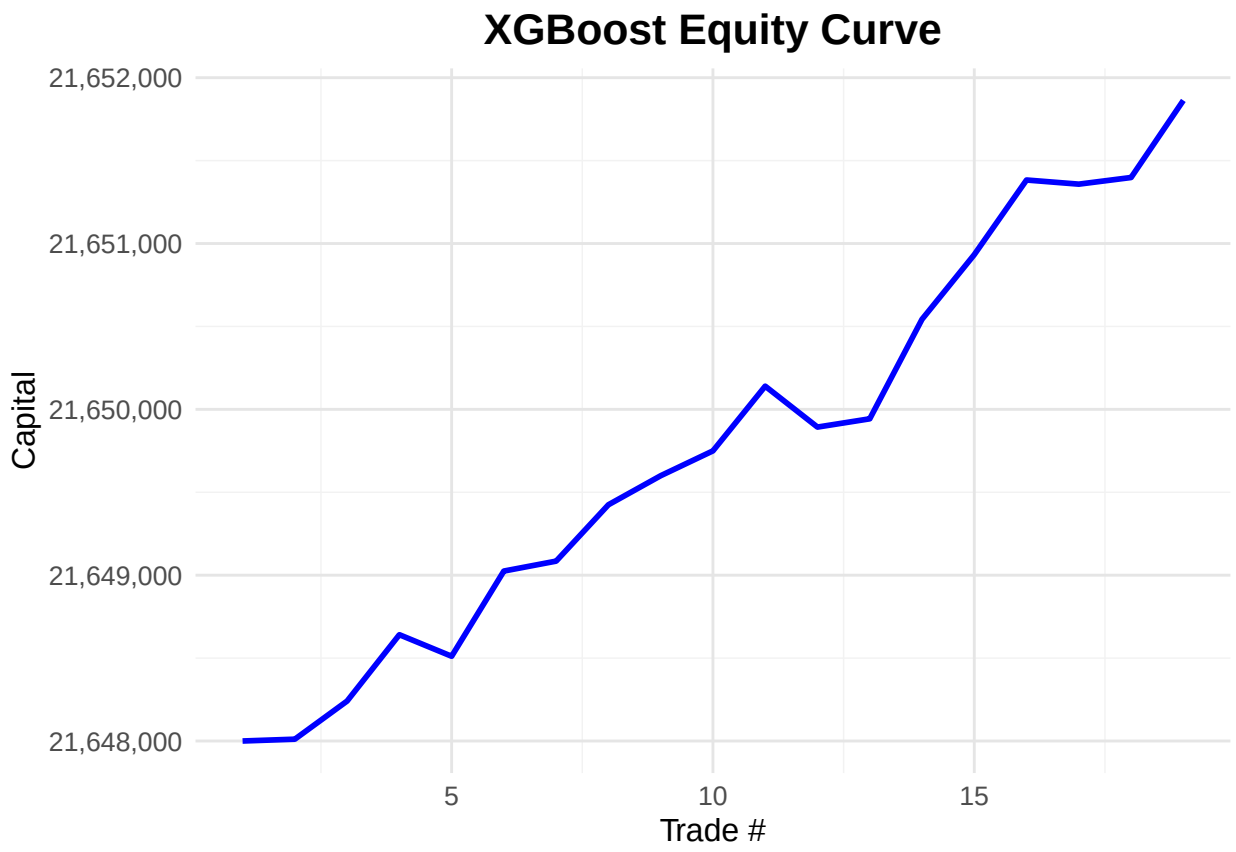
Let: - $C_0$: Initial capital - $r_i$: Profit/loss from trade $i$ - $T$: Threshold for high-precision entry

Then:

$$C_{i+1} = C_i + r_i \quad \text{only if } \hat{y}_i > T$$

```
test_X = model.matrix(PL ~ (ATR + SMA_k7 + SMA_k20 + SMA_k50 + norm_volume)^3, data=test)[, -1]
dtest = xgb.DMatrix(data = test_X)

pred_probs <- predict(model, dtest)

initial_capital = (1/(0.02)) * (1000 * sample(train$Open, 1))
capital = initial_capital
drawdown = c(capital)
wins = c()

for (i in 1:nrow(test)) {
  pred_X = pred_probs[i]
  if (pred_X > best_threshold) {
    capital = capital + test$pl_value[i]
    drawdown = c(drawdown, capital)
    wins = c(wins, test$pl_value[i])
  }
}

ggplot(data = data.frame(trade = 1:length(drawdown),
                         equity = drawdown),
       aes(x = trade, y = equity)) +
  geom_line(color = "blue", size = 1) +
  theme_minimal() +
  labs(title = "XGBoost Equity Curve",
       x = "Trade #",
       y = "Capital") +
  theme(
    plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 12),
```

```
    axis.text = element_text(size = 10),
    panel.grid.major = element_line(color = "grey90"),
    panel.grid.minor = element_line(color = "grey95")
  ) +
  scale_y_continuous(labels = scales::comma)
```




```
# Performance metrics
sharpe = (tail(drawdown, 1) - drawdown[1]) / sd(wins)
max_dd = min(drawdown - cummax(drawdown)) / max(drawdown)

cat("Final Capital:", capital, "\n")
```

```
## Final Capital: 21651863
```

```
cat("Sharpe Ratio:", round(sharpe, 3), "\n")
```

```
## Sharpe Ratio: 16.043
```

```
cat("Max Drawdown:", round(max_dd * 100, 2), "%\n")
```

```
## Max Drawdown: 0 %
```

### Commentary

XGBoost is particularly effective when: - Feature interactions matter - There are non-linear decision boundaries - **precision over recall**

Its ability to output probabilities makes it well-suited for **threshold tuning**, which we exploit using quantiles of the fitted distribution. By focusing on **extremely confident predictions**, we reduce bad trades and focus capital on high-reward setups.

## Support Vector Machines

Support Vector Machines (SVM) offer a powerful alternative approach for classification tasks in trading. Unlike linear models, SVMs can efficiently handle non-linear decision boundaries by projecting data into higher-dimensional spaces where separation becomes possible.

### Theoretical Foundation

The SVM algorithm finds the optimal hyperplane that maximizes the margin between classes. For linearly separable data, this involves solving:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ for all } i$$

For non-linear boundaries, SVMs employ the "kernel trick" to implicitly map data to higher dimensions without explicitly computing the transformation. The decision function becomes:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

Where $K(\mathbf{x}_i, \mathbf{x})$ is the kernel function measuring similarity between points.

### Model Implementation

RBF (Radial Basis Function) kernel in your SVM is a solid choice for optimizing precision

```
library(e1071)

X = model.matrix(PL ~ (ATR + SMA_k7 + SMA_k20 + SMA_k50 + norm_volume)^3, data=train)[, -1]
y = as.factor(train$PL)

svm_model <- svm(x = X, y = y,
                 kernel = "radial",
                 probability = TRUE,
                 cost = 10,
                 gamma = 0.1)

summary(svm_model)
```

```
##
## Call:
## svm.default(x = X, y = y, kernel = "radial", gamma = 0.1, cost = 10,
##     probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  13904
##
##  ( 7033 6871 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

### Kernel Selection Rationale

The radial basis function (RBF) kernel is chosen for its ability to model complex non-linear decision boundaries. The RBF kernel computes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2)$$

Where $\gamma$ controls the influence radius of each support vector. This allows the model to capture local patterns in the feature space that might be missed by linear models.

### Hyperparameter Tuning

Far too computationally expensive to run, `tune.svm` will take forever to compile. This is an area of improvement.

### Threshold Optimization for Precision

```r
svm_probs <- attr(predict(svm_model, X, probability = TRUE), "probabilities")[, "1"]

# Define threshold range
probs <- 10^seq(-2, -10, length.out = 9)
fit_mean <- mean(svm_probs)
fit_sd <- sd(svm_probs)

best_precision <- 0
best_threshold <- 0.5

for (prob in probs) {
  threshold <- qnorm(1 - prob, mean = fit_mean, sd = fit_sd)
```

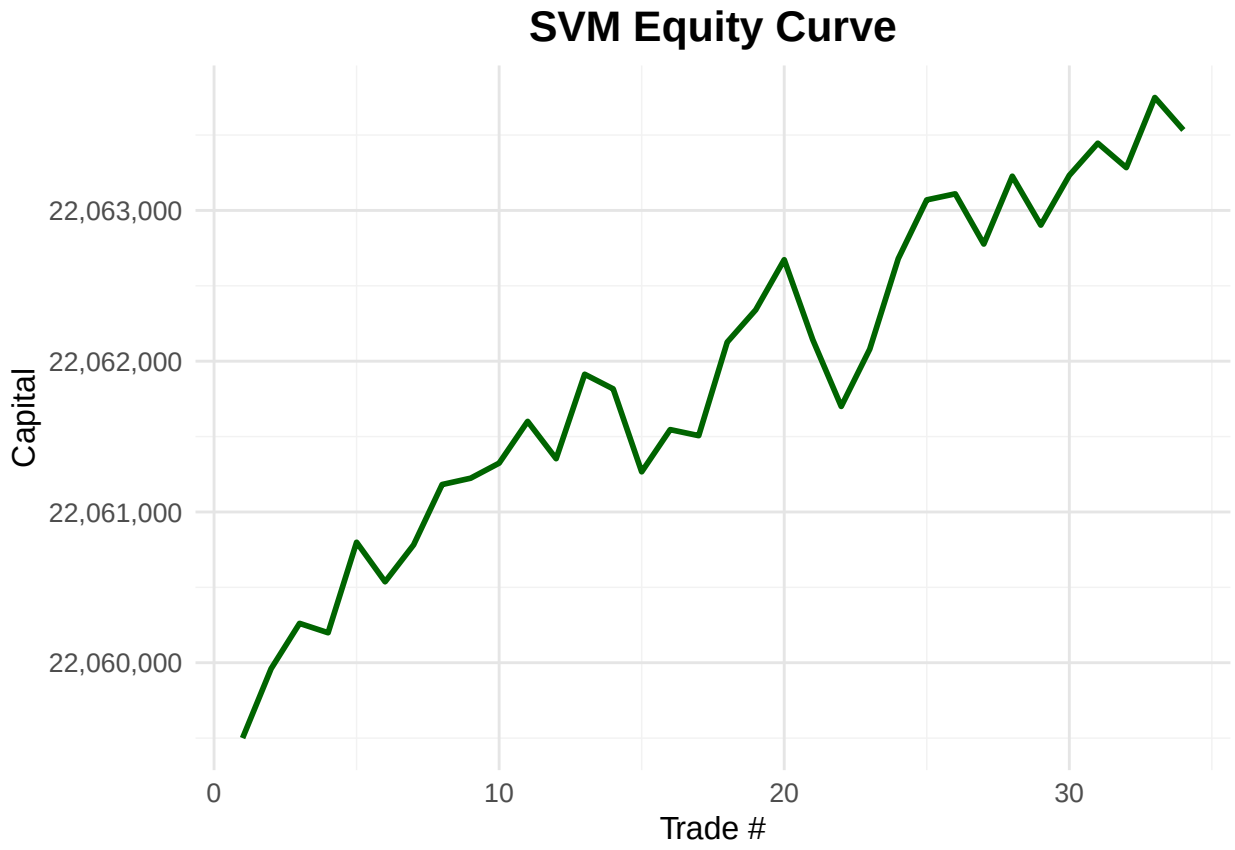
```
  preds <- ifelse(svm_probs > threshold, 1, 0)

  TP <- sum(preds == 1 & train$PL == 1)
  FP <- sum(preds == 1 & train$PL == 0)

  if ((TP + FP) < 30) {
    break # Ensure statistical significance
  }

  precision <- ifelse((TP + FP) == 0, NA, TP / (TP + FP))

  if (!is.na(precision) && precision > best_precision) {
    best_precision <- precision
    best_threshold <- threshold
  }
}

cat("SVM Best Precision:", round(best_precision * 100, 2), "% at Threshold", best_threshold, "\n")
```

```
## SVM Best Precision: 87.72 % at Threshold 0.6139943
```

### Backtesting SVM Strategy

```
#test data
test_X <- model.matrix(PL ~ (ATR + SMA_k7 + SMA_k20 + SMA_k50 + norm_volume)^3, data=test)[, -1]

# Get probability predictions for test set
test_probs <- attr(predict(svm_model, test_X, probability = TRUE), "probabilities")[, "1"]

initial_capital <- (1/(0.02)) * (1000 * sample(train$Open, 1))
capital <- initial_capital
drawdown <- c(capital)
wins <- c()

for (i in 1:nrow(test)) {
  pred_X <- test_probs[i]
  if (pred_X > best_threshold) {
    capital <- capital + test$pl_value[i]
    drawdown <- c(drawdown, capital)
    wins <- c(wins, test$pl_value[i])
  }
}

ggplot(data = data.frame(trade = 1:length(drawdown),
                         equity = drawdown),
       aes(x = trade, y = equity)) +
  geom_line(color = "darkgreen", size = 1) +
  theme_minimal() +
  labs(title = "SVM Equity Curve",
       x = "Trade #",
       y = "Capital") +
```

```
  theme(
    plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 12),
    axis.text = element_text(size = 10),
    panel.grid.major = element_line(color = "grey90"),
    panel.grid.minor = element_line(color = "grey95")
  ) +
  scale_y_continuous(labels = scales::comma)
```




```
# Calculate performance metrics
sharpe <- (tail(drawdown, 1) - drawdown[1]) / sd(wins)
max_dd <- min(drawdown - cummax(drawdown)) / max(drawdown)

cat("Initial Capital:", initial_capital, "\n")
```

```
## Initial Capital: 22059500
```

```
cat("Final Capital:", capital, "\n")
```

```
## Final Capital: 22063533
```

```
cat("Sharpe Ratio:", round(sharpe, 3), "\n")
```

```
## Sharpe Ratio: 11.621
```

```r
cat("Max Drawdown:", round(max_dd * 100, 2), "%\n")
```

```
## Max Drawdown: 0 %
```

```r
cat("Returns:", round(((capital/initial_capital)-1)*100, 2), "%\n")
```

```
## Returns: 0.02 %
```

### SVM Advantages for Trading

SVMs offer several advantages for financial prediction:

1. **Robustness to Outliers**: The support vector focus makes SVMs less sensitive to outliers compared to regression models.
2. **Effective in High-Dimensional Spaces**: SVMs perform well with many features relative to sample size, ideal for complex market data.
3. **Flexibility**: Through kernel selection, SVMs can model various decision boundaries without explicit feature engineering.
4. **Theoretical Guarantees**: SVMs are founded on statistical learning theory, providing formal bounds on generalization error.

The non-linear mapping capability is particularly valuable for capturing market regimes where relationships between indicators and outcomes change dynamically.

---

## Final Thoughts

We have now explored **three models** in depth:

| Model | Output Type | Thresholding | Precision Tuning | Purpose |
|---|---|---|---|---|
| Logistic Regression (GLM) | Probabilities (binary) | Yes | Yes via qnorm thresholds | Predict trade direction |
| XGBoost | Probabilities (binary) | Yes | Yes via precision-optimized threshold | Precision-focused execution |
| Support Vector Machines | Probabilities (binary) | Yes | Yes via precision-optimized threshold | Non-linear boundary detection |

Each model offers distinct advantages for trading strategy development:

**GLM** provides interpretable coefficients that directly quantify the impact of each feature on trade outcomes. This interpretability is valuable for understanding market dynamics and regulatory compliance.

**XGBoost** excels at capturing complex feature interactions and non-linear patterns without requiring explicit specification. Its ensemble nature provides robustness against overfitting, particularly important in noisy financial data.

**SVM** offers strong theoretical guarantees and effectively handles non-linear decision boundaries through kernel methods. Its focus on maximizing the margin between classes helps identify the most reliable trading signals.

In trading, minimizing **false positives** is often more critical than maximizing true positives due to the **asymmetric cost of bad trades**. All three models allow for threshold optimization to achieve precision-focused execution.

For production trading systems, an ensemble approach combining these models might offer the most robust performance across varying market conditions. Each model captures different aspects of market behavior, and their combined signals could provide more reliable trading decisions than any single model alone.