05 - Model

John Gilheany

9/13/2017

Model

Once the final data set was created and cleaned, with a number of response variables including trailing beta, trailing vol, and price to book value, and the associated outcome, which was measured by whether or not a stock was in the Min Vol index or not (1 if in, 0 if not in).

A snippet of the final data set is shown below

```r
head(trainingData)
```

```
## # A tibble: 6 x 8
## # Groups:   ticker [6]
##         date ticker      beta volatility price_to_book weight index_now
##       <date> <fctr>     <dbl>      <dbl>         <dbl>  <dbl>    <fctr>
## 1 2013-09-30     CL 1.0451272  0.6023338    14.8676397 0.7228         1
## 2 2013-05-31    MKC 4.1521567  1.7358281     5.0811086 0.9495         1
## 3 2014-12-31    SJM 0.7885774  1.9425497     1.9548413 0.0497         1
## 4 2014-07-31    SPG 0.4961165  0.9387595     4.5435197 0.3643         1
## 5 2014-08-29     RE 0.7246354  0.6597441     0.7347133 0.5964         1
## 6 2016-03-31    EXR 0.6859407  1.7823979     2.7733161 0.0890         1
## # ... with 1 more variables: index_before <fctr>
```

```r
tail(trainingData)
```

```
## # A tibble: 6 x 8
## # Groups:   ticker [6]
##         date ticker      beta volatility price_to_book weight index_now
##       <date> <fctr>     <dbl>      <dbl>         <dbl>  <dbl>    <fctr>
## 1 2012-08-31    MOS 1.3932857  1.0350886      1.692923      0         0
## 2 2013-01-31    HON 1.2007977  0.7540130      3.902511      0         0
## 3 2014-04-30   GILD 1.5797310  3.4051551     10.079184      0         0
## 4 2014-10-31     UA 1.4545868  0.9678980      3.496854      0         0
## 5 2016-10-31   XLNX 1.0518379  0.3432995      4.645901      0         0
## 6 2015-12-31     XL 0.7457147  0.8823093      1.719251      0         0
## # ... with 1 more variables: index_before <fctr>
```

```r
summary(trainingData)
```

```
##       date                ticker           beta         
##  Min.   :2012-01-31   MKC    :   86   Min.   :-23.3438  
##  1st Qu.:2013-07-31   CB     :   52   1st Qu.:  0.7327  
##  Median :2014-10-31   ADP    :   45   Median :  0.9262  
##  Mean   :2014-10-01   K      :   45   Mean   :  0.9541  
##  3rd Qu.:2015-11-30   SBAC   :   45   3rd Qu.:  1.1567  
##  Max.   :2016-12-30   XEL    :   45   Max.   : 17.5440  
##                       (Other):10880                     
##    volatility        price_to_book         weight       index_now
##  Min.   : 0.01073   Min.   :-524.357   Min.   :0.0000   0:5490   
##  1st Qu.: 0.53315   1st Qu.:   1.726   1st Qu.:0.0000   1:5708   
##  Median :  0.90939   Median :   2.954   Median :0.0526
##  Mean   :  1.81575   Mean   :   4.742   Mean   :0.3455
##  3rd Qu.:  1.57163   3rd Qu.:   4.957   3rd Qu.:0.6123
##  Max.   :152.96132   Max.   :1542.215   Max.   :2.7535
##
##  index_before
##  0:5988
##  1:5210
##
##
##
##
##
```

Given the nature of the data, a logit regression will be ran. Looking at all of the historical data and stock various characteristics, this would model the log odds of a stock being in the minimum volatility index as a combination of the linear predictors mentioned. Several models will be run in a panel, including one by certain months, and one by the entire pool of data.

Model 1: Entire Data Set (Monthly)

The first logit model that will be run is for the entire pool of monthly data.

Data Cleaning - Checking for Class Bias

Ideally, the proportion of stocks in and out of the USMV index should approximately be the same. Checking this, we can see that this is not the case. However, just around 24% of the data is from stocks that ereurrently in the index, so there is a class bias. As a result, we must sample the observations in approximately equal proportions to get better models.

```
table(monthly_final$index_now)
```

```
##
##     0     1
## 28639  8994
```

Create Training and Test Samples

One way to address the problem of class bias is to draw the 0's and 1's for the trainingData (development sample) in equal proportions. In doing so, we will put rest of the inputData not included for training into testData (validation sample). As a result, the size of development sample will be smaller that validation, which is okay, because, there are large number of observations.

```
# Create Training Data
input_ones <- monthly_final[which(monthly_final$index_now == 1), ]  # all 1's
input_zeros <- monthly_final[which(monthly_final$index_now == 0), ]  # all 0's
set.seed(100)  # for repeatability of samples
input_ones_training_rows <- sample(1:nrow(input_ones), 0.7*nrow(input_ones))  # 1's for training
input_zeros_training_rows <- sample(1:nrow(input_zeros), 0.7*nrow(input_ones))  # 0's for training. Pic
training_ones <- input_ones[input_ones_training_rows, ]
training_zeros <- input_zeros[input_zeros_training_rows, ]
trainingData <- rbind(training_ones, training_zeros)  # row bind the 1's and 0's
# Create Test Data
```

```
test_ones <- input_ones[-input_ones_training_rows, ]
test_zeros <- input_zeros[-input_zeros_training_rows, ]
testData <- rbind(test_ones, test_zeros)  # row bind the 1's and 0's
# Remove NA values in index_before
testData <- subset(testData, !is.na(index_before))
trainingData <- subset(trainingData, !is.na(index_before))
```

Now we can check class bias to see if it is more balanced. It is very close to being evenly weighted now.

```
table(trainingData$index_now)
```

```
##
##    0    1
## 5490 5708
```

Logistic Regression Model

Now the model can be run:

```
# Model 1
logit1 <- glm(index_now ~  volatility + beta + price_to_book + index_before, data=trainingData, family=

# Summary of Model 1
summary(logit1)
```

```
##
## Call:
## glm(formula = index_now ~ volatility + beta + price_to_book +
##     index_before, family = binomial(link = "logit"), data = trainingData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0173  -0.4514   0.1744   0.1891   2.6170
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.8759922  0.0592045 -31.687   <2e-16 ***
## volatility    -0.0043563  0.0054966  -0.793    0.428
## beta          -0.3127464  0.0371146  -8.426   <2e-16 ***
## price_to_book  0.0003945  0.0006032   0.654    0.513
## index_before1  6.1554851  0.1126370  54.649   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 15519  on 11197  degrees of freedom
## Residual deviance:  4772  on 11193  degrees of freedom
## AIC: 4782
##
## Number of Fisher Scoring iterations: 6
```

```
# Coefficient Interpretation
## Log Odds
exp(coef(logit1))
```

```
##   (Intercept)    volatility          beta price_to_book index_before1
##     0.1532029     0.9956532     0.7314354     1.0003946   471.2953929
```

```
## Probability
(exp(coef(logit1))) / (1+(exp(coef(logit1))))
```

```
##   (Intercept)    volatility          beta price_to_book index_before1
##     0.1328499     0.4989109     0.4224445     0.5000986     0.9978827
```

Looking at the monthly data is not a true representation of the results, because the index is rebalanced once every six months - not once a month.

Interpretation of Model

The model can be interpreted as:
$\ln[\frac{p}{1-p}]$ = -1.86 - 0.0044 x vol - 0.31 x beta + 0.00039 x price_to_book + 6.16 x index_before
$\frac{p}{1-p}$ = exp(-1.86 - 0.0044 x vol - 0.31 x beta + 0.00039 x price_to_book + 6.16 x index_before)
The coefficients can be interpreted as:
- Volatility: The odds ratio of being added to the index is 0.996 times smaller, given a one unit increase in volatility. This response variable is not statistically significant.
- Beta: The odds ratio of being added to the index is 0.731 times smaller, given a one unit increase in beta. This response variable is statistically significant.
- Price to Book: The odds ratio of being added to the index is 1.0051 times greater, given a one unit increase in price to book ratio. This response variable is not statistically significant.
- Index before: The odds ratio of being added to the index is 410.261 times greater if the stock was in the index 6 months ago. This response variable is statistically significant.

Sanity Check

To take a sample stock to understand the model, we can look at a stock that was not in the USMV index on 12-30-2016, as see how accurate our model would be in predicting the probability of this stock being in the index. We can take AAL (American Airlines), which had a beta of 1.6312867, volatility of 0.8067945, price to book ratio of 4.6943413, and was not in the USMV index 6 months ago. This stock ended up not being in the minimum volatility index on 12-30-2016, so we would expect a probability to be relatively low.
- Odds Ratio:
$\frac{p}{1-p}$ = exp(-3.094 - 0.0032 x 0.8067945 - 0.25 x 1.6312867 + 0.00051 x 4.6943413 + 6.017 x 0)
$\frac{p}{1-p}$ = 0.03013677

- Probability:
p = (exp(-3.094 - 0.0032 x 0.8067945 - 0.25 x 1.6312867 + 0.00051 x 4.6943413 + 6.017 x 0) / (1+exp(-3.094 - 0.0032 x 0.8067945 - 0.25 x 1.6312867 + 0.00051 x 4.6943413 + 6.017 x 0)))
p = 0.02925511

The odds of AAL being in the index on 12-30-2016 is 0.03013677, and this translates to a probability of 2.93%. As expected, already knowing that the stock was not in the index, this low probability seems reasonable.
To further understand the model, we can look at a stock that was was in the USMV index on 12-30-2016, as see how accurate our model would be in predicting the probability of this stock being in the index. We can take AAPL (Apple), which had a beta of 1.0099644, volatility of 0.6118842, price to book ratio of 4.7037726, and it was in the USMV index 6 months ago. This stock ended up being in the minimum volatility index on 12-30-2016, so we would expect a probability to be relatively high
- Odds Ratio:

$\frac{p}{1-p}$ = exp(-3.094 - 0.0032 x 0.6118842 - 0.25 x 1.0099644 + 0.00051 x 4.7037726 + 6.017 x 1)
$\frac{p}{1-p}$ = 14.45369
- Probability:
p = (exp(-3.094 - 0.0032 x 0.6118842 - 0.25 x 1.0099644 + 0.00051 x 4.7037726 + 6.017 x 1) / (1+exp(-3.094 - 0.0032 x 0.6118842 - 0.25 x 1.0099644 + 0.00051 x 4.7037726 + 6.017 x 1)))
p = 0.9352905

The odds of AAL being in the index on 12-30-2016 is 14.45369, and this translates to a probability of 93.53%. As expected, already knowing that the stock was in the index, this high probability seems reasonable.

Model Quality

To test the quality of the model, several tests were done:
Predictive Power
The default cutoff prediction probability score is 0.5 or the ratio of 1's and 0's in the training data. But sometimes, tuning the probability cutoff can improve the accuracy in both the development and validation samples. The InformationValue::optimalCutoff function provides ways to find the optimal cutoff to improve the prediction of 1's, 0's, both 1's and 0's and to reduce the misclassification error. Here, the optimal cut off is 0.74.

```
library(InformationValue)
optCutOff <- optimalCutoff(testData$index_now, predicted)[1]
```

*VIF***_
Like in case of linear regression, we should check for multicollinearity in the model. As seen below, all X variables in the model have VIF well below 4.

```
library(car)
vif(logit1)
```

```
##    volatility          beta price_to_book  index_before
##      1.016126      1.018653      1.000327      1.005611
```

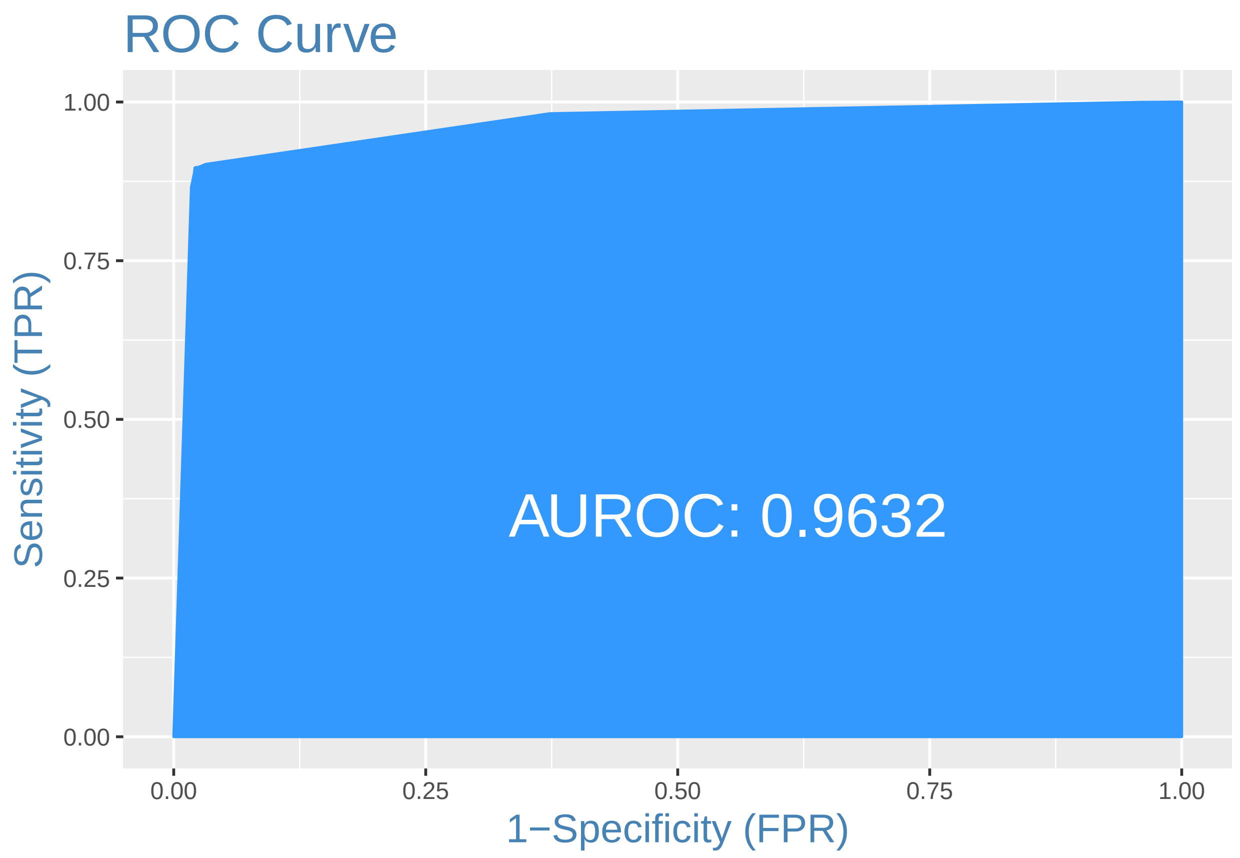
Misclassification Error
Misclassification error is the percentage mismatch of predicted vs actuals, irrespective of 1's or 0's. The lower the misclassification error, the better the model. Here it is 3.1%, which is quite low, and thus good.

```
predicted <- plogis(predict(logit1, testData))
misClassError(testData$index_now, predicted)
```

```
## [1] 0.0309
```

ROC
Receiver Operating Characteristics Curve traces the percentage of true positives accurately predicted by a given logit model as the prediction probability cutoff is lowered from 1 to 0. For a good model, as the cutoff is lowered, it should mark more of actual 1's as positives and lesser of actual 0's as 1's. So for a good model, the curve should rise steeply, indicating that the TPR (Y-Axis) increases faster than the FPR (X-Axis) as the cutoff score decreases. Greater the area under the ROC curve, better the predictive ability of the model. Here, it is 96.3%.

```
plotROC(testData$index_now, predicted)
```



Concordance
Ideally, the model-calculated-probability-scores of all actual Positive's, (aka Ones) should be greater than the model-calculated-probability-scores of ALL the Negatives (aka Zeroes). Such a model is said to be perfectly concordant and a highly reliable one. This phenomenon can be measured by Concordance and Discordance.

In simpler words, of all combinations of 1-0 pairs (actuals), Concordance is the percentage of pairs, whose scores of actual positive's are greater than the scores of actual negative's. For a perfect model, this will be 100%. So, the higher the concordance, the better is the quality of model. This model with a concordance of 97.2% is a good quality model.

```
Concordance(testData$index_now, predicted)
```

```
## $Concordance
## [1] 0.9724405
##
## $Discordance
## [1] 0.02755952
##
## $Tied
## [1] -4.510281e-17
##
## $Pairs
## [1] 47632140
```

Specificity and Sensitivity
- Sensitivity (or True Positive Rate) is the percentage of 1's (actuals) correctly predicted by the model, while, specificity is the percentage of 0's (actuals) correctly predicted. In this model, it was found to be 89.6%.
- Specificity can also be calculated as 1 - False Positive Rate. In this model, it was found to be 97.9%.

```
sensitivity(testData$index_now, predicted, threshold = optCutOff)
```

```
## [1] 0.8956805
```

```
specificity(testData$index_now, predicted, threshold = optCutOff)
```

```
## [1] 0.9789284
```

Confusion Matrix
In the confusion matrix, the columns are actuals, while rows are predicteds

```
confusionMatrix(testData$index_now, predicted, threshold = optCutOff)
```

```
##       0    1
## 0 19001  256
## 1   409 2198
```

Model 2: November Model

Since the index is rebalanced twice a year (once in November and once in May), it makes sense to look at a model for each of these individual months. Thus, a subset of the data was taken for November, and the same procedures done at with Model 1.

```
# Subset data for dates from November only
november_final <- filter(monthly_final, date == "2011-11-30" | date == "2012-11-30"| date == "2013-11-2
# Remove NA values from set
november_final <- subset(november_final, !is.na(index_before))
```

Data Cleaning - Checking for Class Bias

Ideally, the proportion of stocks in and out of the USMV index should approximately be the same. Checking this, we can see that this is not the case. However, just around 26% of the data is from stocks that ereurrently in the index, so there is a class bias. As a result, we must sample the observations in approximately equal proportions to get a better model.

```
table(november_final$index_now)
```

```
##
##    0    1
## 2161  750
```

Create Training and Test Samples

One way to address the problem of class bias is to draw the 0's and 1's for the trainingData (development sample) in equal proportions. In doing so, we will put rest of the inputData not included for training into testData (validation sample). As a result, the size of development sample will be smaller that validation, which is okay, because, there are large number of observations.

```
# Create Training Data
input_ones2 <- november_final[which(november_final$index_now == 1), ]  # all 1's
input_zeros2 <- november_final[which(november_final$index_now == 0), ]  # all 0's
set.seed(100)  # for repeatability of samples
input_ones_training_rows2 <- sample(1:nrow(input_ones2), 0.7*nrow(input_ones2))  # 1's for training
input_zeros_training_rows2 <- sample(1:nrow(input_zeros2), 0.7*nrow(input_ones2))  # 0's for training.
```

```r
training_ones2 <- input_ones2[input_ones_training_rows2, ]
training_zeros2 <- input_zeros2[input_zeros_training_rows2, ]
trainingData2 <- rbind(training_ones2, training_zeros2)  # row bind the 1's and 0's
# Create Test Data
test_ones2 <- input_ones2[-input_ones_training_rows2, ]
test_zeros2 <- input_zeros2[-input_zeros_training_rows2, ]
testData2 <- rbind(test_ones2, test_zeros2)  # row bind the 1's and 0's
```

Now we can check class bias to see if it is more balanced. It is evenly weighted now, with each being represented by 525 observations.

```r
table(trainingData2$index_now)
```

```
##
##   0   1
## 525 525
```

Logistic Regression Model

Now the model can be run:

```r
# Model 2
logit2 <- glm(index_now ~  volatility + beta + price_to_book + index_before, data=trainingData2, family=

# Summary of Model 2
summary(logit2)
```

```
##
## Call:
## glm(formula = index_now ~ volatility + beta + price_to_book +
##     index_before, family = binomial(link = "logit"), data = trainingData2)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.98863  -0.51069  -0.04623   0.27163   2.05545
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.4577528  0.1834842  -7.945 1.94e-15 ***
## volatility     0.0604954  0.0330935   1.828 0.067548 .
## beta          -0.4945911  0.1331544  -3.714 0.000204 ***
## price_to_book -0.0001288  0.0017214  -0.075 0.940368
## index_before1  5.0806517  0.2776866  18.296  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1455.61  on 1049  degrees of freedom
## Residual deviance:  585.48  on 1045  degrees of freedom
## AIC: 595.48
##
## Number of Fisher Scoring iterations: 6
```

```
# Coefficient Interpretation
## Log Odds
exp(coef(logit2))
```

```
##   (Intercept)    volatility          beta price_to_book index_before1
##     0.2327588     1.0623627     0.6098202     0.9998712   160.8788646
```

```
## Probability
(exp(coef(logit2))) / (1+(exp(coef(logit2))))
```

```
##   (Intercept)    volatility          beta price_to_book index_before1
##     0.1888113     0.5151192     0.3788126     0.4999678     0.9938225
```

Looking at the November model will be helpful for someone looking to predict index rebalancing between June and October.

Interpretation of Model

The model can be interpreted as:
$\ln[\frac{p}{1-p}]$ = -1.46 + 0.061 x vol - 0.49 x beta - 0.00013 x price_to_book + 5.08 x index_before
$\frac{p}{1-p}$ = exp(-1.46 + 0.061 x vol - 0.49 x beta - 0.00013 x price_to_book + 5.08 x index_before)
The coefficients can be interpreted as:
- Volatility: The odds ratio of being added to the index is 1.063 times greater, given a one unit increase in volatility. This response variable is statistically significant, at an alpha level of 0.1.
- Beta: The odds ratio of being added to the index is 0.61 times smaller, given a one unit increase in beta. This response variable is statistically significant.
- Price to Book: The odds ratio of being added to the index is 0.99 times smaller, given a one unit increase in price to book ratio. This response variable is not statistically significant.
- Index before: The odds ratio of being added to the index is 160.88 times greater if the stock was in the index 6 months ago. This response variable is statistically significant.

Sanity Check

Will do later, if useful.

Model Quality

To test the quality of the model, several tests were done:
Predictive Power
The default cutoff prediction probability score is 0.5 or the ratio of 1's and 0's in the training data. But sometimes, tuning the probability cutoff can improve the accuracy in both the development and validation samples. The InformationValue::optimalCutoff function provides ways to find the optimal cutoff to improve the prediction of 1's, 0's, both 1's and 0's and to reduce the misclassification error. Here, the optimal cut off is 0.95.

```
library(InformationValue)
optCutOff2 <- optimalCutoff(testData2$index_now, predicted2)[1]
```

*VIF***_
Like in case of linear regression, we should check for multicollinearity in the model. As seen below, all X variables in the model have VIF well below 4.

```r
library(car)
vif(logit2)
```

```
##    volatility          beta price_to_book  index_before
##      1.107794      1.106221      1.000799      1.003120
```

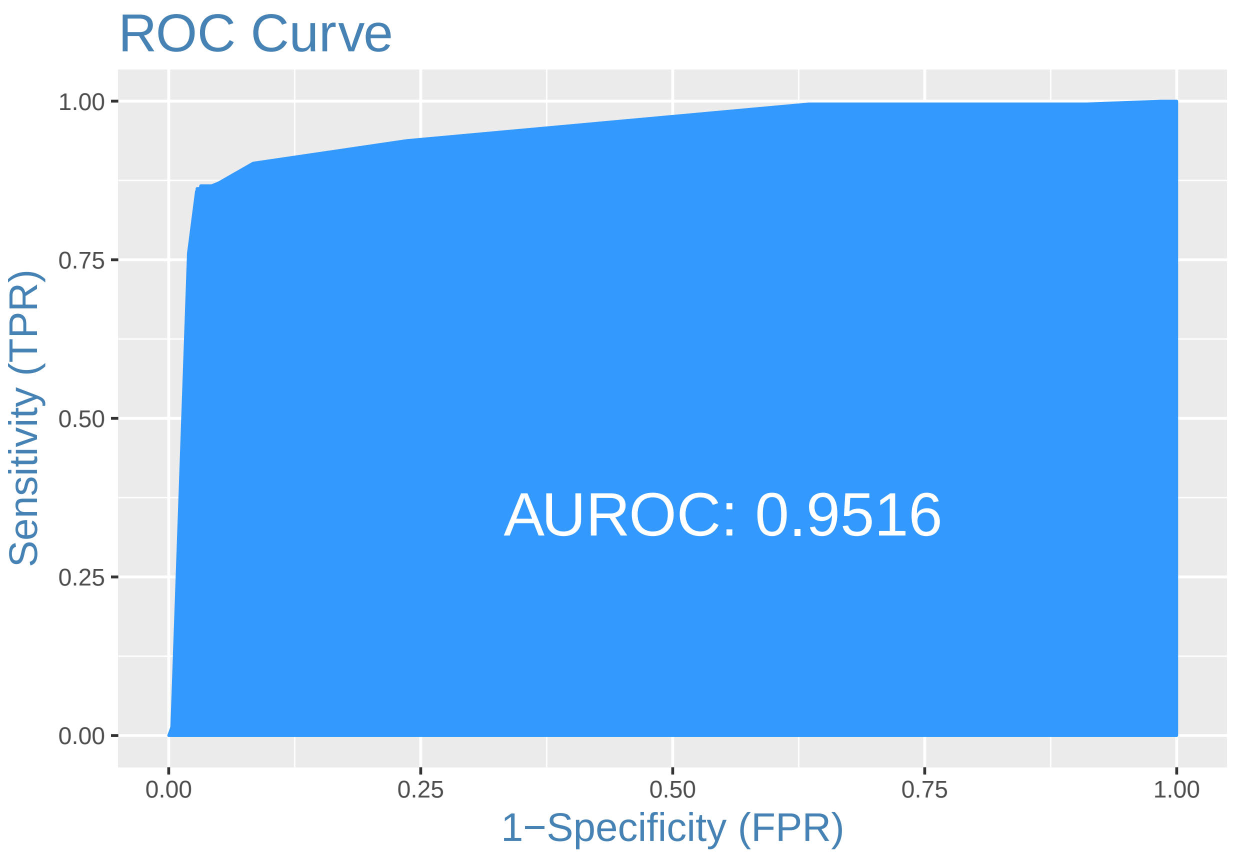
Misclassification Error
Misclassification error is the percentage mismatch of predicted vs actuals, irrespective of 1's or 0's. The lower the misclassification error, the better the model. Here it is 4.4%, which is quite low, and good.

```r
predicted2 <- plogis(predict(logit2, testData2))
misClassError(testData2$index_now, predicted2)
```

```
## [1] 0.0435
```

ROC
Receiver Operating Characteristics Curve traces the percentage of true positives accurately predicted by a given logit model as the prediction probability cutoff is lowered from 1 to 0. For a good model, as the cutoff is lowered, it should mark more of actual 1's as positives and lesser of actual 0's as 1's. So for a good model, the curve should rise steeply, indicating that the TPR (Y-Axis) increases faster than the FPR (X-Axis) as the cutoff score decreases. Greater the area under the ROC curve, better the predictive ability of the model. Here, it is 95.2%.

```r
plotROC(testData2$index_now, predicted2)
```



Concordance
Ideally, the model-calculated-probability-scores of all actual Positive's, (aka Ones) should be greater than the model-calculated-probability-scores of ALL the Negatives (aka Zeroes). Such a model is said to be perfectly

concordant and a highly reliable one. This phenomenon can be measured by Concordance and Discordance.

In simpler words, of all combinations of 1-0 pairs (actuals), Concordance is the percentage of pairs, whose scores of actual positive's are greater than the scores of actual negative's. For a perfect model, this will be 100%. So, the higher the concordance, the better is the quality of model. This model with a concordance of 95.5% is a good quality model.

```
Concordance(testData2$index_now, predicted2)
```

```
## $Concordance
## [1] 0.9558843
##
## $Discordance
## [1] 0.04411573
##
## $Tied
## [1] -6.938894e-18
##
## $Pairs
## [1] 368100
```

Specificity and Sensitivity
- Sensitivity (or True Positive Rate) is the percentage of 1's (actuals) correctly predicted by the model, while, specificity is the percentage of 0's (actuals) correctly predicted. In this model, it was found to be 85.8%.
- Specificity can also be calculated as 1 - False Positive Rate. In this model, it was found to be 97.3%.

```
sensitivity(testData2$index_now, predicted2, threshold = optCutOff2)
```

```
## [1] 0.8577778
```

```
specificity(testData2$index_now, predicted2, threshold = optCutOff2)
```

```
## [1] 0.9731051
```

Confusion Matrix
In the confusion matrix, the columns are actuals, while rows are predicteds

```
confusionMatrix(testData2$index_now, predicted2, threshold = optCutOff2)
```

```
##      0   1
## 0 1592  32
## 1   44 193
```

Model 3: May Model

Since the index is rebalanced twice a year (once in November and once in May), it makes sense to look at a model for each of these individual months. Thus, a subset of the data was taken for May, and the same procedures done at with Model 1.

```
# Subset data for dates from May only
may_final <- filter(monthly_final, date == "2012-05-31" |  date == "2013-05-31"| date == "2014-05-30" |
# Remove NA values from set
may_final <- subset(may_final, !is.na(index_before))
```

Data Cleaning - Checking for Class Bias

Ideally, the proportion of stocks in and out of the USMV index should approximately be the same. Checking this, we can see that this is not the case. However, just around 24% of the data is from stocks that ereurrently in the index, so there is a class bias. As a result, we must sample the observations in approximately equal proportions to get a better model.

```
table(may_final$index_now)
```

```
##
##    0    1
## 2188  705
```

Create Training and Test Samples

One way to address the problem of class bias is to draw the 0's and 1's for the trainingData (development sample) in equal proportions. In doing so, we will put rest of the inputData not included for training into testData (validation sample). As a result, the size of development sample will be smaller that validation, which is okay, because, there are large number of observations.

```
# Create Training Data
input_ones3 <- may_final[which(may_final$index_now == 1), ]  # all 1's
input_zeros3 <- may_final[which(may_final$index_now == 0), ]  # all 0's
set.seed(100)  # for repeatability of samples
input_ones_training_rows3 <- sample(1:nrow(input_ones3), 0.7*nrow(input_ones3))  # 1's for training
input_zeros_training_rows3 <- sample(1:nrow(input_zeros3), 0.7*nrow(input_ones3))  # 0's for training.
training_ones3 <- input_ones3[input_ones_training_rows3, ]
training_zeros3 <- input_zeros3[input_zeros_training_rows3, ]
trainingData3 <- rbind(training_ones3, training_zeros3)  # row bind the 1's and 0's
# Create Test Data
test_ones3 <- input_ones3[-input_ones_training_rows3, ]
test_zeros3 <- input_zeros3[-input_zeros_training_rows3, ]
testData3 <- rbind(test_ones3, test_zeros3)  # row bind the 1's and 0's
```

Now we can check class bias to see if it is more balanced. It is evenly weighted now, with each being represented by 493 observations.

```
table(trainingData3$index_now)
```

```
##
##   0   1
## 493 493
```

Logistic Regression Model

Now the model can be run:

```
# Model 3
logit3 <- glm(index_now ~  volatility + beta + price_to_book + index_before, data=trainingData3, family

# Summary of Model 3
summary(logit3)
```

```
##
## Call:
## glm(formula = index_now ~ volatility + beta + price_to_book +
##     index_before, family = binomial(link = "logit"), data = trainingData3)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -3.04816  -0.38611   0.00159   0.13885   2.34011
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.744382   0.249805  -6.983 2.89e-12 ***
## volatility    -0.039892   0.028585  -1.396 0.162842
## beta          -0.642378   0.165440  -3.883 0.000103 ***
## price_to_book -0.012360   0.007939  -1.557 0.119498
## index_before1  7.013927   0.497441  14.100  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1366.89  on 985  degrees of freedom
## Residual deviance:  327.87  on 981  degrees of freedom
## AIC: 337.87
##
## Number of Fisher Scoring iterations: 7
```

```r
# Coefficient Interpretation
## Log Odds
exp(coef(logit3))
```

```
##   (Intercept)    volatility          beta price_to_book index_before1
##     0.1747529     0.9608932     0.5260398     0.9877159  1112.0132792
```

```r
## Probability
(exp(coef(logit3))) / (1+(exp(coef(logit3))))
```

```
##   (Intercept)    volatility          beta price_to_book index_before1
##     0.1487572     0.4900283     0.3447091     0.4969100     0.9991015
```

Looking at the May model will be helpful for someone looking to predict index rebalancing between December and April.

Interpretation of Model

The model can be interpreted as:
$\ln[\frac{p}{1-p}]$ = -1.74 - 0.04 x vol - 0.64 x beta - 0.012 x price_to_book + 7.014 x index_before
$\frac{p}{1-p}$ = exp(-1.74 - 0.04 x vol - 0.64 x beta - 0.012 x price_to_book + 7.014 x index_before)
The coefficients can be interpreted as:
- Volatility: The odds ratio of being added to the index is 0.96 times smaller, given a one unit increase in volatility. This response variable is not statistically significant.
- Beta: The odds ratio of being added to the index is 0.52 times smaller, given a one unit increase in beta. This response variable is statistically significant.
- Price to Book: The odds ratio of being added to the index is 0.99 times smaller, given a one unit increase in price to book ratio. This response variable is not statistically significant.
- Index before: The odds ratio of being added to the index is 1112.01 times greater if the stock was in the index 6 months ago. This response variable is statistically significant.

Sanity Check

Will do later if useful.

Model Quality

To test the quality of the model, several tests were done:
Predictive Power
The default cutoff prediction probability score is 0.5 or the ratio of 1's and 0's in the training data. But sometimes, tuning the probability cutoff can improve the accuracy in both the development and validation samples. The InformationValue::optimalCutoff function provides ways to find the optimal cutoff to improve the prediction of 1's, 0's, both 1's and 0's and to reduce the misclassification error. Here, the optimal cut off is 0.98.

```
library(InformationValue)
optCutOff3 <- optimalCutoff(testData3$index_now, predicted3)[1]
```

*VIF***_
Like in case of linear regression, we should check for multicollinearity in the model. As seen below, all X variables in the model have VIF well below 4.

```
library(car)
vif(logit3)
```

```
##    volatility          beta price_to_book  index_before
##      1.124595      1.161319      1.006899      1.074031
```

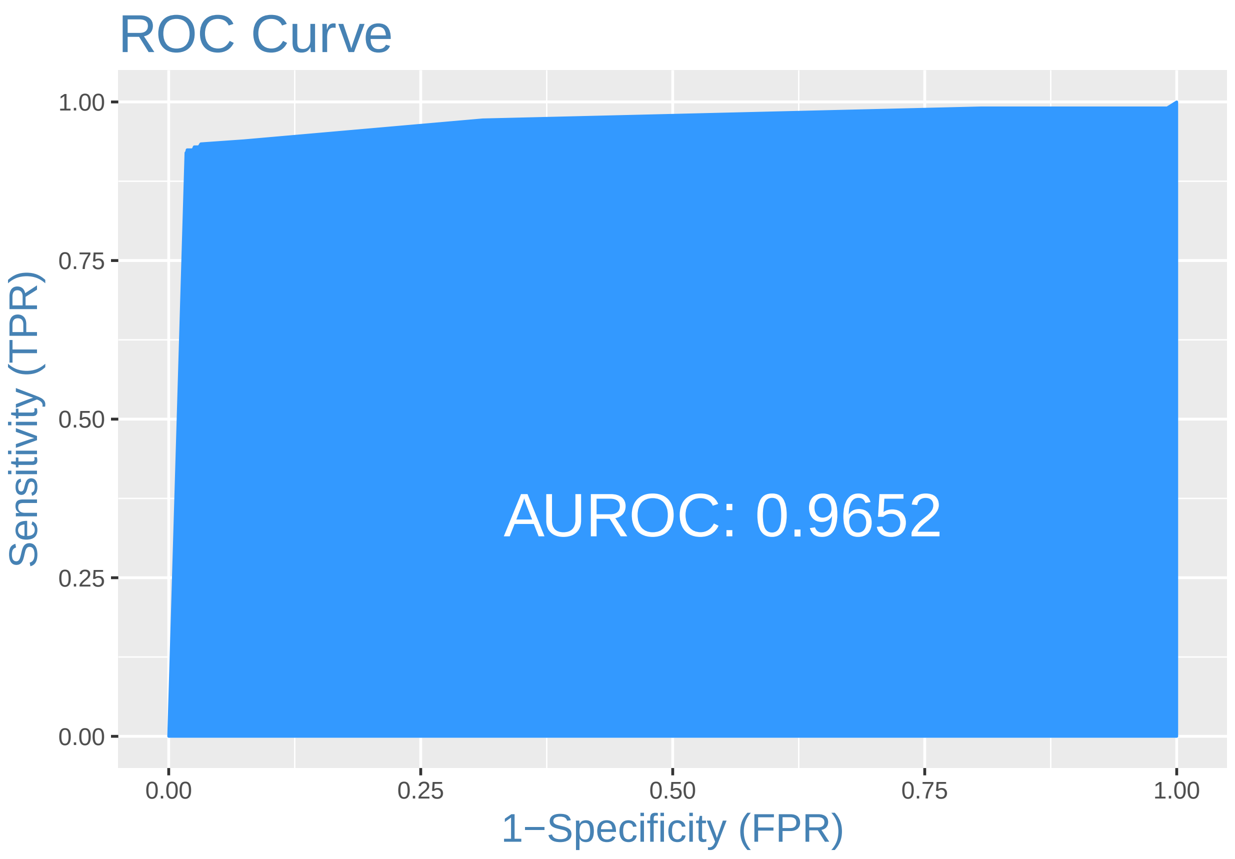
Misclassification Error
Misclassification error is the percentage mismatch of predicted vs actuals, irrespective of 1's or 0's. The lower the misclassification error, the better the model. Here it is 2.6%, which is quite low, and good.

```
predicted3 <- plogis(predict(logit3, testData3))
misClassError(testData3$index_now, predicted3)
```

```
## [1] 0.0262
```

ROC
Receiver Operating Characteristics Curve traces the percentage of true positives accurately predicted by a given logit model as the prediction probability cutoff is lowered from 1 to 0. For a good model, as the cutoff is lowered, it should mark more of actual 1's as positives and lesser of actual 0's as 1's. So for a good model, the curve should rise steeply, indicating that the TPR (Y-Axis) increases faster than the FPR (X-Axis) as the cutoff score decreases. Greater the area under the ROC curve, better the predictive ability of the model. Here, it is 96.5%.

```
plotROC(testData3$index_now, predicted3)
```



Concordance
Ideally, the model-calculated-probability-scores of all actual Positive's, (aka Ones) should be greater than the model-calculated-probability-scores of ALL the Negatives (aka Zeroes). Such a model is said to be perfectly concordant and a highly reliable one. This phenomenon can be measured by Concordance and Discordance.

In simpler words, of all combinations of 1-0 pairs (actuals), Concordance is the percentage of pairs, whose scores of actual positive's are greater than the scores of actual negative's. For a perfect model, this will be 100%. So, the higher the concordance, the better is the quality of model. This model with a concordance of 97.3% is a good quality model.

```
Concordance(testData3$index_now, predicted3)
```

```
## $Concordance
## [1] 0.9732621
##
## $Discordance
## [1] 0.02673791
##
## $Tied
## [1] -3.469447e-18
##
## $Pairs
## [1] 359340
```

Specificity and Sensitivity
- Sensitivity (or True Positive Rate) is the percentage of 1's (actuals) correctly predicted by the model, while, specificity is the percentage of 0's (actuals) correctly predicted. In this model, it was found to be 92.0%.
- Specificity can also be calculated as 1 - False Positive Rate. In this model, it was found to be 98.3%.

```
sensitivity(testData3$index_now, predicted3, threshold = optCutOff3)
```

```
## [1] 0.9198113
```

```
specificity(testData3$index_now, predicted3, threshold = optCutOff3)
```

```
## [1] 0.9828909
```

Confusion Matrix
In the confusion matrix, the columns are actuals, while rows are predicteds

```
confusionMatrix(testData3$index_now, predicted3, threshold = optCutOff3)
```

```
##      0   1
## 0 1666  17
## 1   29 195
```

Model 4: Total Rebalancing (November & May) Model

Since the index is rebalanced twice a year (once in November and once in May), it makes sense to look at a model for both of these months. Thus, a subset of the data was taken for May and November, by combining the data sets from Model 2 and Model 3.

```
# Subset data for dates from May only
both_final <- rbind(may_final, november_final)
```

Data Cleaning - Checking for Class Bias

Ideally, the proportion of stocks in and out of the USMV index should approximately be the same. Checking this, we can see that this is not the case. However, just around 25% of the data is from stocks that ereurrently in the index, so there is a class bias. As a result, we must sample the observations in approximately equal proportions to get a better model.

```
table(both_final$index_now)
```

```
##
##    0    1
## 4349 1455
```

Create Training and Test Samples

One way to address the problem of class bias is to draw the 0's and 1's for the trainingData (development sample) in equal proportions. In doing so, we will put rest of the inputData not included for training into testData (validation sample). As a result, the size of development sample will be smaller that validation, which is okay, because, there are large number of observations.

```
# Create Training Data
input_ones4 <- both_final[which(both_final$index_now == 1), ]  # all 1's
input_zeros4 <- both_final[which(both_final$index_now == 0), ]  # all 0's
set.seed(100)  # for repeatability of samples
input_ones_training_rows4 <- sample(1:nrow(input_ones4), 0.7*nrow(input_ones4))  # 1's for training
input_zeros_training_rows4 <- sample(1:nrow(input_zeros4), 0.7*nrow(input_ones4))  # 0's for training.
training_ones4 <- input_ones4[input_ones_training_rows4, ]
training_zeros4 <- input_zeros4[input_zeros_training_rows4, ]
```

```r
trainingData4 <- rbind(training_ones4, training_zeros4)  # row bind the 1's and 0's
# Create Test Data
test_ones4 <- input_ones4[-input_ones_training_rows4, ]
test_zeros4 <- input_zeros4[-input_zeros_training_rows4, ]
testData4 <- rbind(test_ones4, test_zeros4)  # row bind the 1's and 0's
```

Now we can check class bias to see if it is more balanced. It is evenly weighted now, with each being represented by 1018 observations.

```r
table(trainingData4$index_now)
```

```
##
##    0    1
## 1018 1018
```

Logistic Regression Model

Now the model can be run:

```r
# Model 4
logit4 <- glm(index_now ~  volatility + beta + price_to_book + index_before, data=trainingData4, family=

# Summary of Model 3
summary(logit4)
```

```
##
## Call:
## glm(formula = index_now ~ volatility + beta + price_to_book +
##     index_before, family = binomial(link = "logit"), data = trainingData4)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.76973  -0.46138   0.00366   0.21208   2.16144
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.840874   0.124092 -14.835  < 2e-16 ***
## volatility     0.002945   0.009558   0.308    0.758
## beta          -0.309455   0.071228  -4.345  1.4e-05 ***
## price_to_book -0.001894   0.001547  -1.224    0.221
## index_before1  5.886884   0.241988  24.327  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2822.50  on 2035  degrees of freedom
## Residual deviance:  931.66  on 2031  degrees of freedom
## AIC: 941.66
##
## Number of Fisher Scoring iterations: 6
```

```r
# Coefficient Interpretation
## Log Odds
exp(coef(logit4))
```

```
##   (Intercept)    volatility          beta price_to_book index_before1
##     0.1586787     1.0029492     0.7338469     0.9981080   360.2808848
```

```
## Probability
(exp(coef(logit4))) / (1+(exp(coef(logit4))))
```

```
##   (Intercept)    volatility          beta price_to_book index_before1
##     0.1369480     0.5007362     0.4232478     0.4995265     0.9972321
```

Looking at this model will be helpful for someone looking to predict index rebalancing, generally, for both months.

Interpretation of Model

The model can be interpreted as:
$\ln[\frac{p}{1-p}]$ = -1.84 + 0.003 x vol - 0.31 x beta - 0.0019 x price_to_book + 5.89 x index_before
$\frac{p}{1-p}$ = exp(-1.84 + 0.003 x vol - 0.31 x beta - 0.0019 x price_to_book + 5.89 x index_before)
The coefficients can be interpreted as:
- Volatility: The odds ratio of being added to the index is 1.0029 times greater, given a one unit increase in volatility. This response variable is not statistically significant.
- Beta: The odds ratio of being added to the index is 0.73 times smaller, given a one unit increase in beta. This response variable is statistically significant.
- Price to Book: The odds ratio of being added to the index is 0.99 times smaller, given a one unit increase in price to book ratio. This response variable is not statistically significant.
- Index before: The odds ratio of being added to the index is 360.28 times greater if the stock was in the index 6 months ago. This response variable is statistically significant.

Sanity Check

Will do later if useful.

Model Quality

To test the quality of the model, several tests were done:
Predictive Power
The default cutoff prediction probability score is 0.5 or the ratio of 1's and 0's in the training data. But sometimes, tuning the probability cutoff can improve the accuracy in both the development and validation samples. The InformationValue::optimalCutoff function provides ways to find the optimal cutoff to improve the prediction of 1's, 0's, both 1's and 0's and to reduce the misclassification error. Here, the optimal cut off is 0.77.

```
library(InformationValue)
optCutOff4 <- optimalCutoff(testData4$index_now, predicted4)[1]
```

*VIF**_*
Like in case of linear regression, we should check for multicollinearity in the model. As seen below, all X variables in the model have VIF well below 4.

```
library(car)
vif(logit4)
```

```
##    volatility          beta price_to_book  index_before
##      1.022471      1.033581      1.007330      1.022806
```

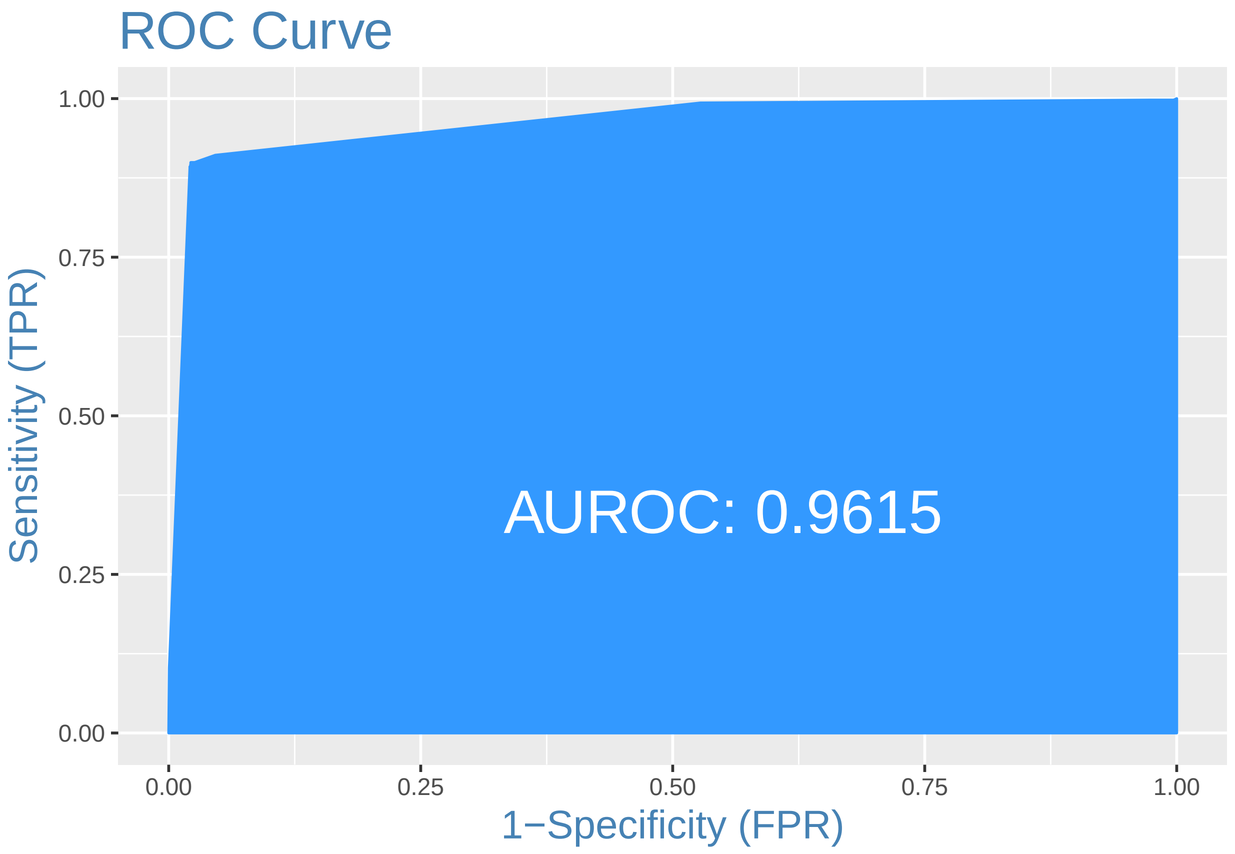
Misclassification Error
Misclassification error is the percentage mismatch of predicted vs actuals, irrespective of 1's or 0's. The lower the misclassification error, the better the model. Here it is 3.2%, which is quite low, and good.

```
predicted4 <- plogis(predict(logit4, testData4))
misClassError(testData4$index_now, predicted4)
```

```
## [1] 0.0321
```

ROC
Receiver Operating Characteristics Curve traces the percentage of true positives accurately predicted by a given logit model as the prediction probability cutoff is lowered from 1 to 0. For a good model, as the cutoff is lowered, it should mark more of actual 1's as positives and lesser of actual 0's as 1's. So for a good model, the curve should rise steeply, indicating that the TPR (Y-Axis) increases faster than the FPR (X-Axis) as the cutoff score decreases. Greater the area under the ROC curve, better the predictive ability of the model. Here, it is 96.2%.

```
plotROC(testData4$index_now, predicted4)
```



Concordance
Ideally, the model-calculated-probability-scores of all actual Positive's, (aka Ones) should be greater than the model-calculated-probability-scores of ALL the Negatives (aka Zeroes). Such a model is said to be perfectly concordant and a highly reliable one. This phenomenon can be measured by Concordance and Discordance.

In simpler words, of all combinations of 1-0 pairs (actuals), Concordance is the percentage of pairs, whose scores of actual positive's are greater than the scores of actual negative's. For a perfect model, this will be 100%. So, the higher the concordance, the better is the quality of model. This model with a concordance of 97.1% is a good quality model.

```
Concordance(testData4$index_now, predicted4)
```

```
## $Concordance
## [1] 0.9714409
##
## $Discordance
## [1] 0.02855912
##
## $Tied
## [1] -4.857226e-17
##
## $Pairs
## [1] 1455647
```

Specificity and Sensitivity
- Sensitivity (or True Positive Rate) is the percentage of 1's (actuals) correctly predicted by the model, while, specificity is the percentage of 0's (actuals) correctly predicted. In this model, it was found to be 89.9%.
- Specificity can also be calculated as 1 - False Positive Rate. In this model, it was found to be 97.8%.

```
sensitivity(testData4$index_now, predicted4, threshold = optCutOff4)
```

```
## [1] 0.8993135
```

```
specificity(testData4$index_now, predicted4, threshold = optCutOff4)
```

```
## [1] 0.9780847
```

Confusion Matrix
In the confusion matrix, the columns are actuals, while rows are predicteds

```
confusionMatrix(testData4$index_now, predicted4, threshold = optCutOff4)
```

```
##      0   1
## 0 3258  44
## 1   73 393
```

Side by Side Model Comparison

```
library(broom)
all_models <- rbind_list(
    tidy(logit1) %>% mutate(model = 1),
    tidy(logit2) %>% mutate(model = 2),
    tidy(logit3) %>% mutate(model = 3),
    tidy(logit4) %>% mutate(model = 4))
```

```
## Warning: 'rbind_list' is deprecated.
## Use 'bind_rows()' instead.
## See help("Deprecated")
```

```
ols_table <- all_models %>%
    select(-statistic, -p.value) %>%
    mutate_each(funs(round(., 2)), -term) %>%
    gather(key, value, estimate:std.error) %>%
    spread(model, value)
```

```
## `mutate_each()` is deprecated.
## Use `mutate_all()`, `mutate_at()` or `mutate_if()` instead.
## To map `funs` over a selection of variables, use `mutate_at()`
```

```
ols_table
```

```
## # A tibble: 10 x 6
##             term       key   `1`   `2`   `3`   `4`
##  *         <chr>     <chr> <dbl> <dbl> <dbl> <dbl>
##  1   (Intercept)  estimate -1.88 -1.46 -1.74 -1.84
##  2   (Intercept) std.error  0.06  0.18  0.25  0.12
##  3          beta  estimate -0.31 -0.49 -0.64 -0.31
##  4          beta std.error  0.04  0.13  0.17  0.07
##  5 index_before1  estimate  6.16  5.08  7.01  5.89
##  6 index_before1 std.error  0.11  0.28  0.50  0.24
##  7 price_to_book  estimate  0.00  0.00 -0.01  0.00
##  8 price_to_book std.error  0.00  0.00  0.01  0.00
##  9    volatility  estimate  0.00  0.06 -0.04  0.00
## 10    volatility std.error  0.01  0.03  0.03  0.01
```