Classification

Hrishabh

Contents

4.6.1 The Stock Market Data

The name of the data is *Smarket* and it is available inside the ISLR package. The data set contains the stock market percentage return for the S&P 500 stock index from 2001 to 2005. For each date, we have recorded the percentage returns for each of the five previous trading days.

Loading the data

```
smarket <- Smarket
kableExtra::kable(head(smarket,5))
```

Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
2001	0.381	-0.192	-2.624	-1.055	5.010	1.1913	0.959	Up
2001	0.959	0.381	-0.192	-2.624	-1.055	1.2965	1.032	Up
2001	1.032	0.959	0.381	-0.192	-2.624	1.4112	-0.623	Down
2001	-0.623	1.032	0.959	0.381	-0.192	1.2760	0.614	Up
2001	0.614	-0.623	1.032	0.959	0.381	1.2057	0.213	Up

Description of the Variables
Year: year
Lag1 ..... *Lag5* : The percentage returns for each of the five previous trading days.
Volume: no. of shares traded the previous day.
Today: The percentage return on the date in question.
Direction: whether the market was up or down.

Summary of the data

```
summary(smarket)
```

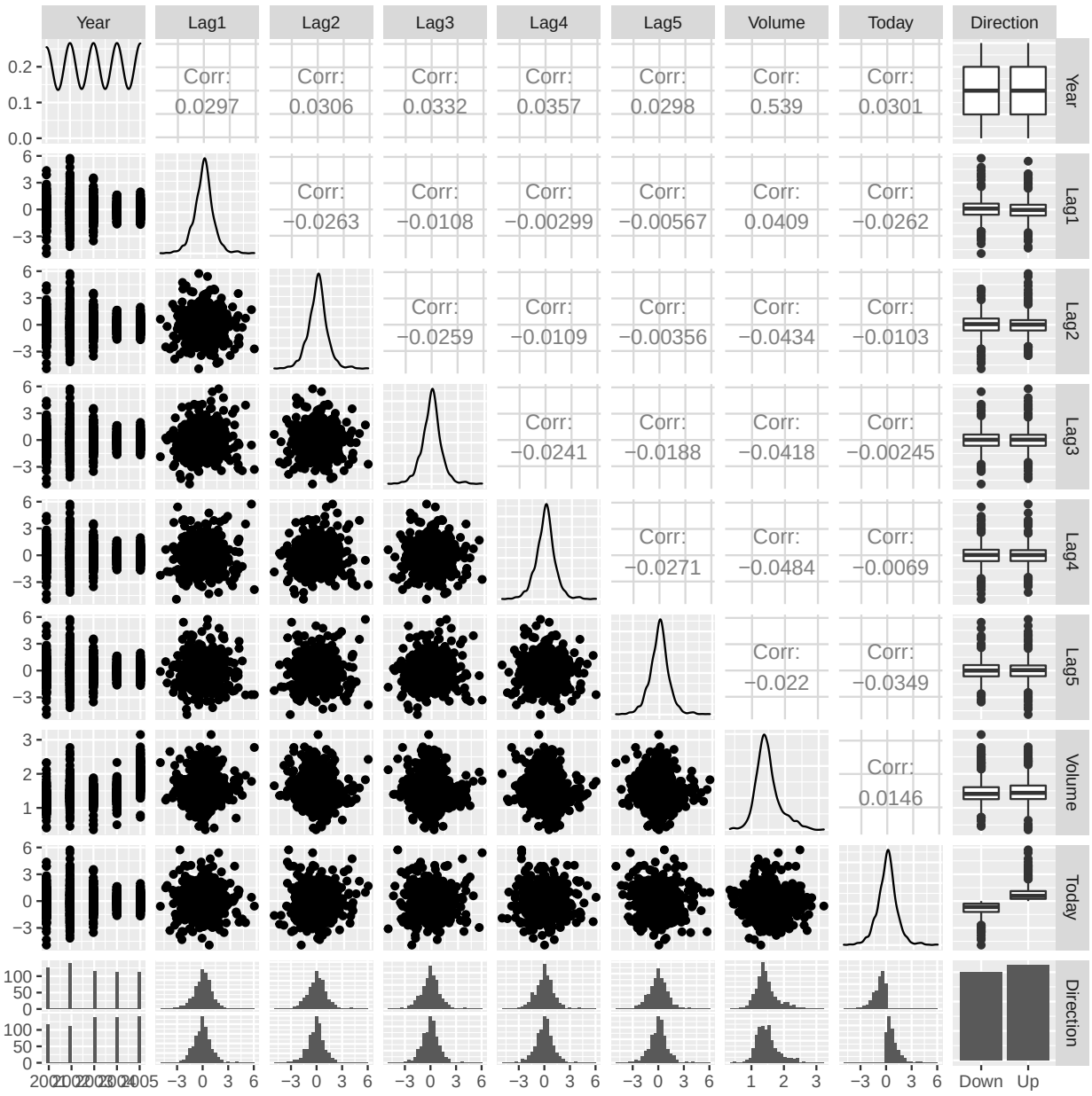
```
##       Year           Lag1                Lag2                Lag3          
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.922000  
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500   1st Qu.:-0.640000  
##  Median :2003   Median : 0.039000   Median : 0.039000   Median : 0.038500  
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919   Mean   : 0.001716  
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.596750  
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.733000  
##       Lag4                Lag5              Volume           Today          
##  Min.   :-4.922000   Min.   :-4.92200   Min.   :0.3561   Min.   :-4.922000  
##  1st Qu.:-0.640000   1st Qu.:-0.64000   1st Qu.:1.2574   1st Qu.:-0.639500  
##  Median : 0.038500   Median : 0.03850   Median :1.4229   Median : 0.038500  
##  Mean   : 0.001636   Mean   : 0.00561   Mean   :1.4783   Mean   : 0.003138  
##  3rd Qu.: 0.596750   3rd Qu.: 0.59700   3rd Qu.:1.6417   3rd Qu.: 0.596750  
##  Max.   : 5.733000   Max.   : 5.73300   Max.   :3.1525   Max.   : 5.733000  
##  Direction
##  Down:602
##  Up  :648
##
##
##
##
```

Pairs Plot

```
GGally::ggpairs(smarket,progress =FALSE)
```

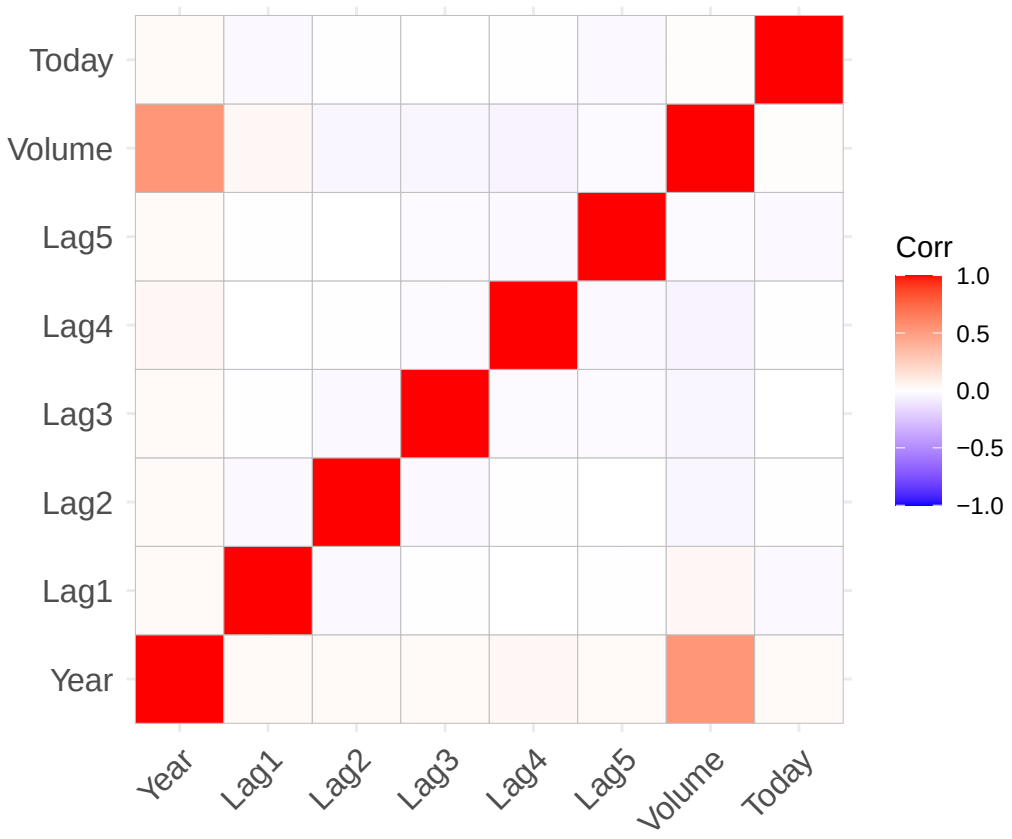
```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Correlation in the data

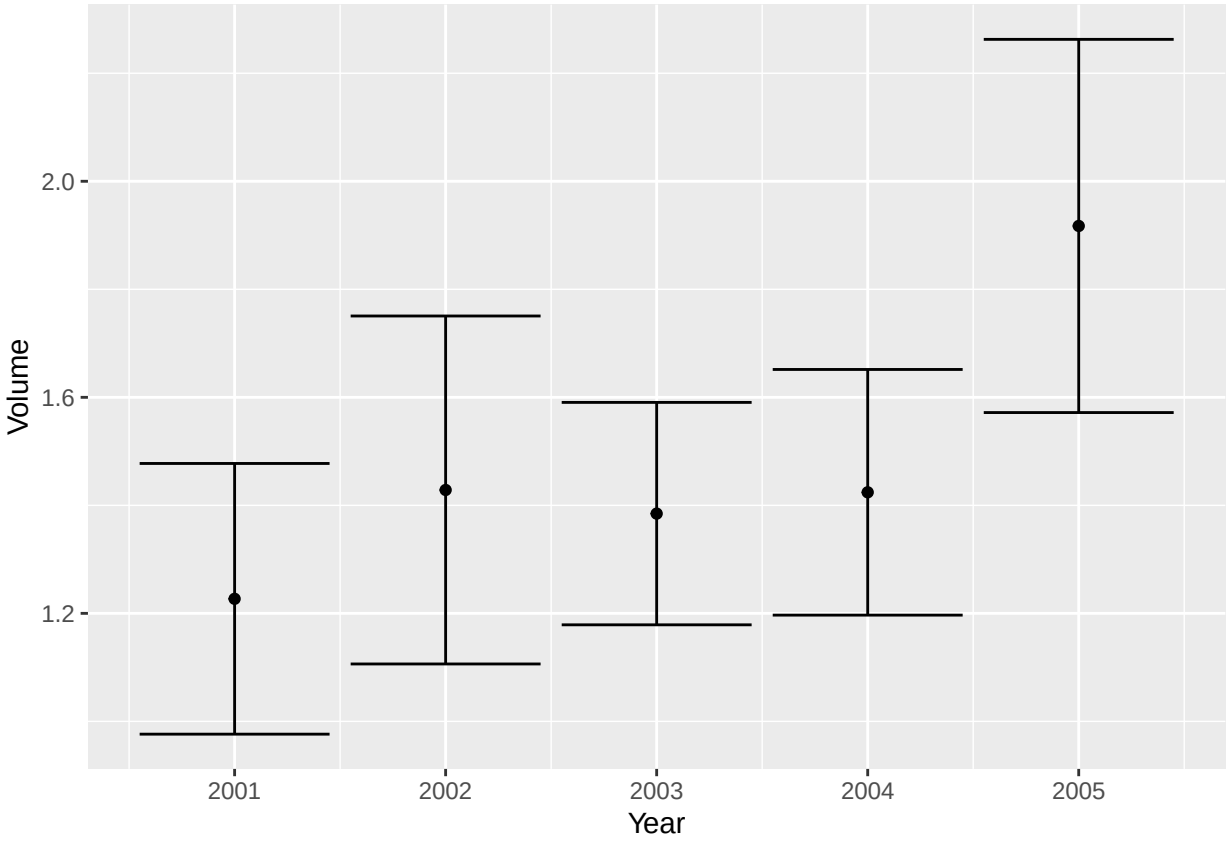
```
corr <- cor(smarket[,-9])
ggcorrplot::ggcorrplot(corr)
```



It is apparent that there is no correlation between the Lags. The only significant correlation is between the Volume and Year.

Let us plot Volume vs. Year to see there relation.

```
ggplot(data = smarket, aes(y = Volume, x = Year)) +
  stat_summary(fun.y = mean, geom = 'point') +
  stat_summary(fun.data = mean_sdl, fun.args = list(mult = 1),  geom = 'errorbar')
```



We can see the trend that the Volume traded is increasing as the year increases.

4.6.2 Logistic Regression

Now, we will be using the *glm()* function with *family = binomial* to run the logistic regression.

```
l1 <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = smarket, family = binomial)
summary(l1)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = smarket)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.446  -1.203   1.065   1.145   1.326
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.126000   0.240736  -0.523    0.601
## Lag1        -0.073074   0.050167  -1.457    0.145
## Lag2        -0.042301   0.050086  -0.845    0.398
## Lag3         0.011085   0.049939   0.222    0.824
## Lag4         0.009359   0.049974   0.187    0.851
## Lag5         0.010313   0.049511   0.208    0.835
## Volume       0.135441   0.158360   0.855    0.392
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1731.2  on 1249  degrees of freedom
## Residual deviance: 1727.6  on 1243  degrees of freedom
## AIC: 1741.6
##
## Number of Fisher Scoring iterations: 3
```

As we can see from the P-values, the relation between the variables and the Direction is not statistically significant.

Predicting the Direction

```
pred <- predict(l1,type = 'response')
pred_dir <- rep("Up",nrow(smarket))
pred_dir[pred < 0.5] = 'Down'

# Confusion Matrix
table(pred_dir,smarket$Direction)
```

```
##
## pred_dir Down  Up
##     Down  145 141
##     Up    457 507
```

Metrics

Now we will calculate metrics like Sensitivity, Specificity, Precision and Accuracy.

```
Values <- c(round(507/(141 + 507),4)*100, round(145/(145 + 457),4)*100, round(507/(457 + 507),4)*100,
            round((507 + 145)/nrow(smarket),4)*100)
Metrics <- c('Sensitivity','Specificity','Precision','Accuracy')
kableExtra::kable(data.frame(Metrics,Values))
```

Metrics	Values
Sensitivity	78.24
Specificity	24.09
Precision	52.59
Accuracy	52.16

Since, the Training Set Accuracy is only slightly better than 50%, this means this model is not such a good model. This fact was apparant from the P-values too. To check how this model performs on the test data, we will split the data and use the models again.

Train-Test Split and Model Validation

We will hold out the observations from the Year 2005 as a test set.

```
train <- (smarket$Year < 2005)
smarket.2005 <- smarket[!train,]
Direction.2005 <- smarket$Direction[!train]
```

Now, we will fit the model again.

```
l2 <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
          data = smarket, family = binomial, subset = train)
pred2 <- predict(l2, smarket.2005, type = 'response')
pred2_dir <- rep("Up",nrow(smarket.2005))
pred2_dir[pred2 < 0.5] = "Down"
# constructing the Confusion Matrix
table(pred2_dir, Direction.2005)
```

```
##          Direction.2005
## pred2_dir Down Up
##      Down   77 97
##      Up     34 44
```

Now, evaluating the metrics.

```
Values2 <- c(round(44/(44 + 907),4)*100, round(77/(77 + 34),4)*100, round(44/(44 + 34),4)*100,
             round((77 + 44)/nrow(smarket.2005),4)*100)
kableExtra::kable(data.frame(Metrics = Metrics, Values = Values))
```

Metrics	Values
Sensitivity	78.24
Specificity	24.09
Precision	52.59
Accuracy	52.16

The accuracy is 48.02% which is worse than prediction by random guessing.

Since a lot of the variables were not significant, we will try dropping them in hopes of increasing the accuracy.

Model after dropping some variables

We will only use Lag1 and Lag2 in this model.

```
l3 <- glm(Direction ~ Lag1 + Lag2, data = smarket, family = binomial, subset = train)
pred3 <- predict(l3, smarket.2005, type = 'response')
pred3_dir <- rep("Up",nrow(smarket.2005))
pred3_dir[pred3 < 0.5] = "Down"
# constructing the Confusion Matrix
table(pred3_dir, Direction.2005)
```

```
##          Direction.2005
## pred3_dir Down  Up
##      Down   35  35
##      Up     76 106
```

evaluating the metrics.

```
Values3 <- c(round(106/(106 + 35),4)*100, round(35/(35 + 76),4)*100, round(106/(106 + 76),4)*100,
             round((106 + 35)/nrow(smarket.2005),4)*100)
kableExtra::kable(data.frame(Metrics = Metrics, Values = Values3))
```

Metrics	Values
Sensitivity	75.18
Specificity	31.53
Precision	58.24
Accuracy	55.95

The accuracy(55.95%) is higher than the previous case. The precision of 58.24% suggests that when it predicts positive, it is accurate 58% of the time.

Predicting results for specific lags

In this section, we will predict the direction when Lag1 and Lag2 is equal to (1.2,1.1) and (1.5,-0.8) using the reduced model.

```
predict(l3, newdata = data.frame(Lag1 = c(1.2,1.5), Lag2 = c(1.5,-0.8)), type = 'response')
```

```
##         1         2
## 0.4747071 0.4960939
```

So according to our predictions the direction for both set of lags is down.

4.6.3 Linear Discriminant Analysis

In this section we will be using the Linear Discriminant Analysis(LDA) to perform the classification. LDA in R can be performed by using the *lda()* function from the *MASS* package.

Creating the LDA classifier

```
lda1 <- lda(Direction ~ Lag1 + Lag2, data = smarket, subset = train)
lda1
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
##
## Coefficients of linear discriminants:
##             LD1
## Lag1 -0.6420190
## Lag2 -0.5135293
```

The prior probabilities are show that the 49.2% of the training observations corresponds to days where the market went down and 50.1% of the training corresponds to the days when the market went up.

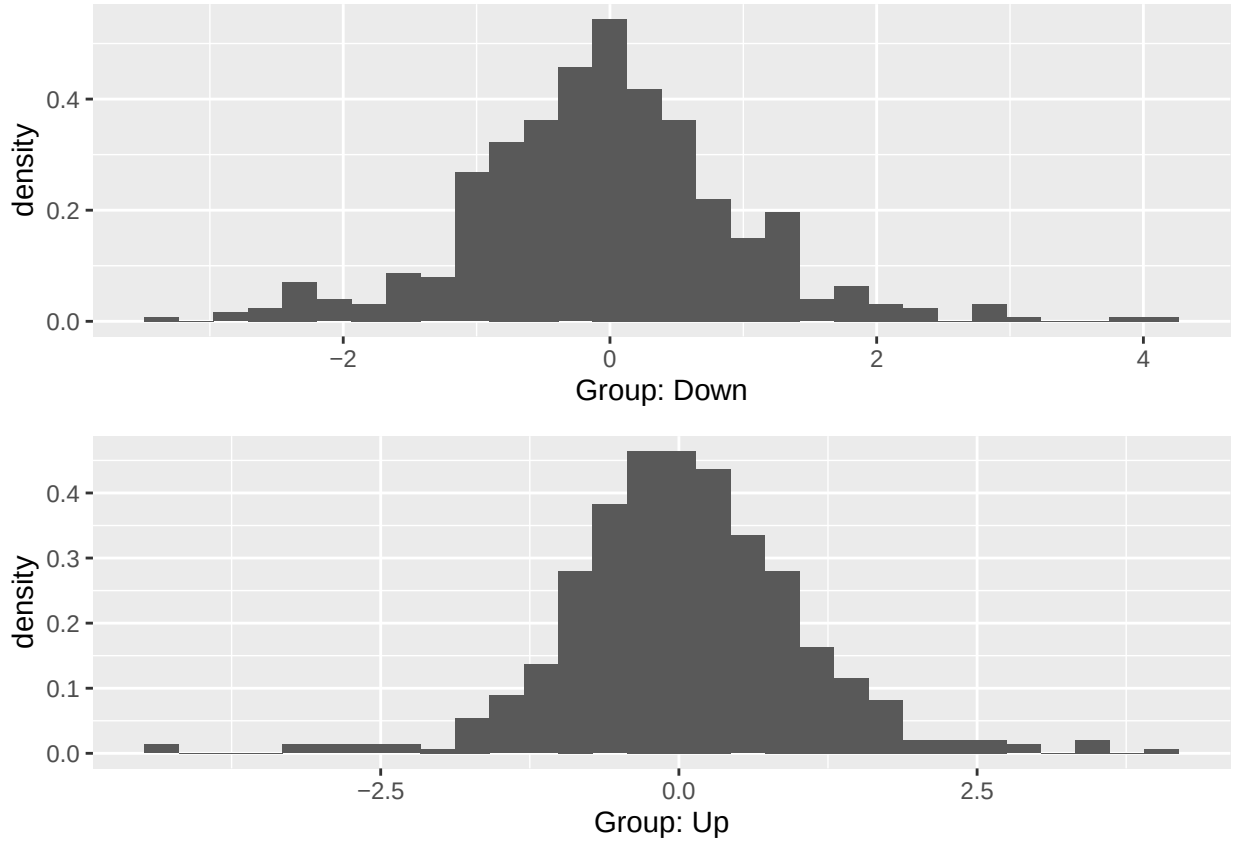
THe Group means suggests that when the market goes up, their is a tendency for the previous 2 days lags to be negative, wheras when the market goes down the previous two days lag has the tendency to be positive.

The coefficients of Linear discriminants is used to form the LDA decision rule. If (-0.642 * Lag1 - 0.513 * Lag2) is large than the LDA classifier predicts that the market goes up, and if it is small it predicts that the market goes down.

Visualizing the classifier

```
temp_data <- smarket[train,]
group1<- temp_data[temp_data$Direction == 'Up',]
group2 <- temp_data[temp_data$Direction == 'Down',]
prob_lda1 <- -0.642 * group1$Lag1 - 0.513 * group1$Lag2
prob_lda2 <- -0.642 * group2$Lag1 - 0.513 * group2$Lag2
plot1 <- ggplot(data.frame(prob_lda1),aes(prob_lda1))+
  geom_histogram(aes(y = ..density..)) +
  xlab("Group: Up")
plot2 <- ggplot(data.frame(prob_lda2), aes(prob_lda2)) +
  geom_histogram(aes(y = ..density..)) +
  xlab("Group: Down")
gridExtra::grid.arrange(plot2,plot1, ncol = 1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The plots are obtained by computing (-0.642 * Lag1 - 0.513 * Lag2) for each of the training observations. Instead of using ggplot one could use the base plot function.

Predicting results for test set

We will use the model that is trained in the previous step to classify the test observation.

```
pred_lda <- predict(lda1, smarket.2005)
class_lda <- pred_lda$class
table(class_lda, Direction.2005)
```

```
##          Direction.2005
## class_lda Down  Up
##      Down   35  35
##      Up     76 106
```

```
Values4 <- c(round(106/(106 + 35),4)*100, round(35/(35 + 76),4)*100, round(106/(106 + 76),4)*100,
             round((106 + 35)/nrow(smarket.2005),4)*100)
kableExtra::kable(data.frame(Metrics = Metrics, Values = Values4))
```

Metrics	Values
Sensitivity	75.18
Specificity	31.53
Precision	58.24
Accuracy	55.95

This result is obtained by using 50% as the threshold.

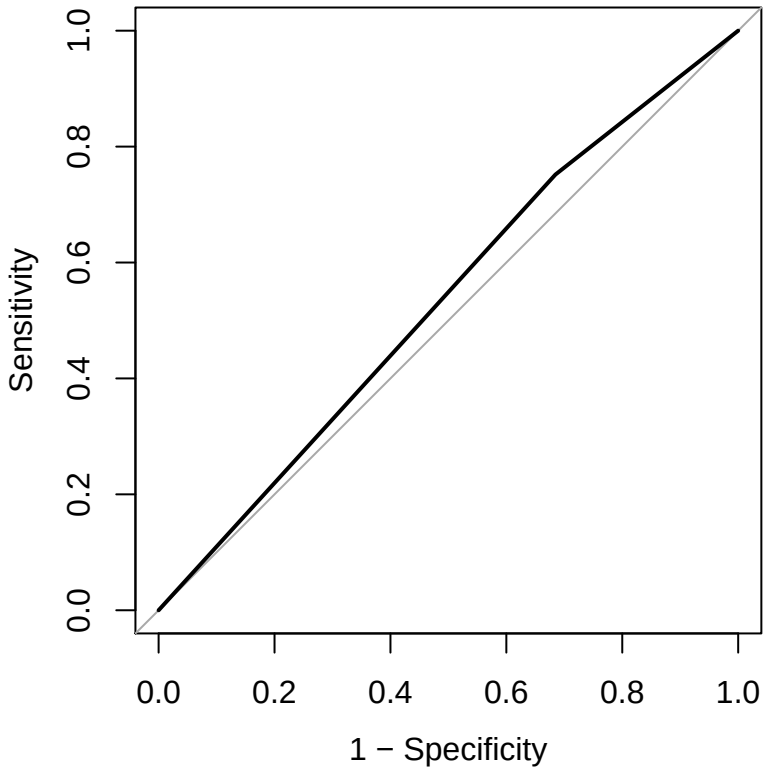
ROC

We can make the roc curve using the *roc* function of the *pROC* package.

```
true_dir <- rep(1,nrow(smarket.2005))
true_dir[smarket.2005$Direction == 'Down'] = 0
pred_dir_lda <- rep(1, nrow(smarket.2005))
pred_dir_lda[class_lda == 'Down'] = 0
par(pty = 's')
pROC::roc(true_dir, pred_dir_lda, plot = TRUE, legacy.axes = TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##
```

```
## Call:
## roc.default(response = true_dir, predictor = pred_dir_lda, plot = TRUE,     legacy.axes = TRUE)
##
## Data: pred_dir_lda in 111 controls (true_dir 0) < 141 cases (true_dir 1).
## Area under the curve: 0.5335
```

From the above ROC plot we can see that the classifier is not very good.

4.6.4 Quadratic Discriminant Analysis

We will use the *qda()* function from the *MASS* package to fit the Quadratic Discriminant Analysis(QDA) classifier.

```
qda_fit <- qda(Direction ~ Lag1 + Lag2, data = smarket, subset = train)
qda_fit
```

```
## Call:
## qda(Direction ~ Lag1 + Lag2, data = smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

The prior probabilities and the group means imply the same thing as in LDA.

Predicting the results

We will predict the result and make the confusion matrix.

```
pred_qda <- predict(qda_fit, smarket.2005)
class_qda <- pred_qda$class
table(class_qda, Direction.2005)
```

```
##          Direction.2005
## class_qda Down  Up
##      Down   30  20
##      Up     81 121
```

Metrics

```
Values5 <- c(round(121/(121 + 20),4)*100, round(30/(30 + 81),4)*100, round(121/(121 + 81),4)*100,
             round((121 + 30)/nrow(smarket.2005),4)*100)
kableExtra::kable(data.frame(Metrics = Metrics, Values = Values5))
```

Metrics	Values
Sensitivity	85.82
Specificity	27.03
Precision	59.90
Accuracy	59.92

This classifier attains a higher accuracy than previous classifiers.

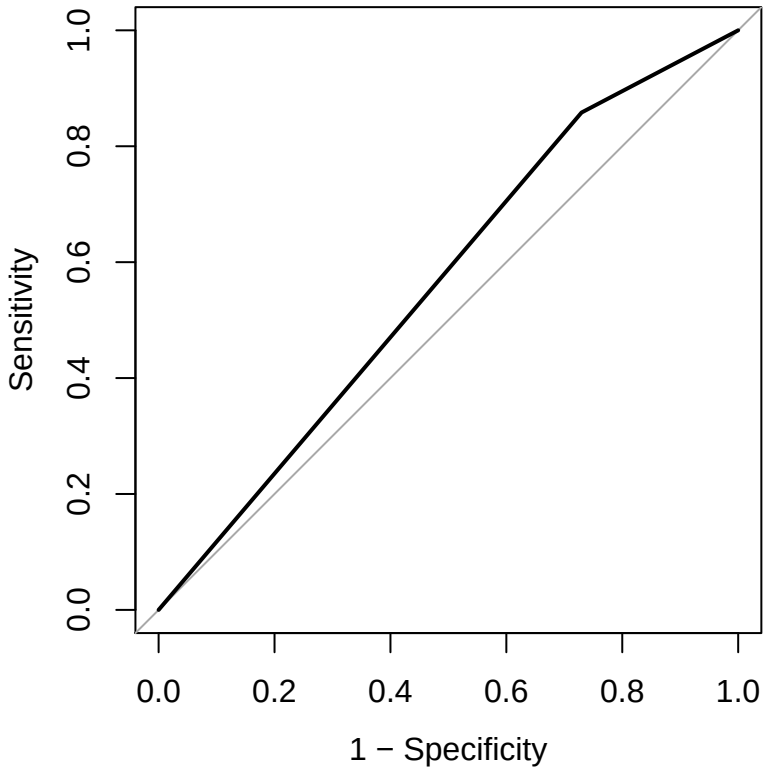
ROC

Similar as above LDA section.

```
pred_dir_qda <- rep(1, nrow(smarket.2005))
pred_dir_qda[class_qda == 'Down'] = 0
par(pty = 's')
pROC::roc(true_dir, pred_dir_qda, plot = TRUE, legacy.axes = TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##
## Call:
## roc.default(response = true_dir, predictor = pred_dir_qda, plot = TRUE,     legacy.axes = TRUE)
##
## Data: pred_dir_qda in 111 controls (true_dir 0) < 141 cases (true_dir 1).
## Area under the curve: 0.5642
```

This model is better than before but it is still not very accurate.

4.6.5 K-Nearest Neighbours

In this section we will be using the *knn()* function from the *class* library.

Required Input
1. Matrix containing predictors associated with training data.
2. Matrix contatining predictors associated with data for which we are trying to make predictions.
3. Vector contatining class labels for training observations.
4. Value of K.

We will now prepare the required inputs.

```
# Input 1
train_X <- cbind(smarket$Lag1,smarket$Lag2)[train,]
# Input 2
test_X <- cbind(smarket$Lag1,smarket$Lag2)[!train,]
train_Direction <- smarket$Direction[train]
```

Prediction using KNN

There exists a random component in the knn function. Hence, to get same results in each run we set seed.

```
set.seed(1)
library(class)
knn_pred <- knn(train_X,test_X,train_Direction, k = 1)
# Confusion Matrix
table(knn_pred, Direction.2005)
```

```
##         Direction.2005
## knn_pred Down Up
##     Down   43 58
##     Up     68 83
```

Metrics

```
Values6 <- c(round(83/(83 + 58),4)*100, round(43/(43 + 68),4)*100, round(83/(68 + 83),4)*100,
             round((83 + 43)/nrow(smarket.2005),4)*100)
kableExtra::kable(data.frame(Metrics = Metrics, Values = Values6))
```

Metrics	Values
Sensitivity	58.87
Specificity	38.74
Precision	54.97
Accuracy	50.00

The accuracy for this classifier is less than all the previous models. But we can improve the model by choosing a better value for K.

Prediction using KNN, K = 3

```
set.seed(1)
library(class)
knn_pred1 <- knn(train_X,test_X,train_Direction, k = 3)
# Confusion Matrix
table(knn_pred1, Direction.2005)
```

```
##          Direction.2005
## knn_pred1 Down Up
##      Down   48 55
##      Up     63 86
```

Metrics

```
Values7 <- c(round(86/(86 + 55),4)*100, round(48/(48 + 63),4)*100, round(86/(63 + 86),4)*100,
             round((86 + 48)/nrow(smarket.2005),4)*100)
kableExtra::kable(data.frame(Metrics = Metrics, Values = Values7))
```

Metrics	Values
Sensitivity	60.99
Specificity	43.24
Precision	57.72
Accuracy	53.17

We have improved the model to achieve an accuracy of 53.17%. We will discuss methods to select K for Knn in next chapter.

4.6.6 An Application to Caravan Insurance Data

We will now use the KNN approach to *Caravan* dataset of the *ISLR* package.

```
caravan <- Caravan
dim(caravan)
```

```
## [1] 5822   86
```

This is a big dataset with 86 variables and 5822 observations.

Remarks about KNN
In the knn algorithm, the scale of the variable has significant effect on the predictions. Variables on larger scale will have a higher effect than the variables on the smaller scale. Because of this, we need to standardized the data to get better model.

Standardization of the data and train-test split

In this section, we are going to standardize the data.

```
standardized_X <- scale(caravan[,-86])
# train-test split
test <- 1:1000
train_X2 <- standardized_X[-test,]
test_X2 <- standardized_X[test,]
train_Y2 <- caravan$Purchase[-test]
test_Y2 <- caravan$Purchase[test]
```

Prediction

We now use the above inputs to make prediction using the knn algorithm.

```
set.seed(1)
knn_pred2 <- knn(train_X2, test_X2, train_Y2, k=1)
#confusion matrix
table(knn_pred2, test_Y2)
```

```
##          test_Y2
## knn_pred2  No Yes
##       No  873  50
##       Yes  68   9
```

Metrics

```
Values8 <- c(round(9/(50 + 9),4)*100, round(873/(873 + 68),4)*100, round(9/(68 + 9),4)*100,
             round((873 + 9)/length(test_Y2),4)*100)
kableExtra::kable(data.frame(Metrics = Metrics, Values = Values8))
```

Metrics	Values
Sensitivity	15.25
Specificity	92.77
Precision	11.69
Accuracy	88.20

The accuracy is high for this model.

Prediction using KNN, K = 3

```
set.seed(1)
library(class)
knn_pred2 <- knn(train_X2,test_X2,train_Y2, k = 3)
# Confusion Matrix
table(knn_pred2, test_Y2)
```

```
##          test_Y2
## knn_pred2  No Yes
##       No  921  54
##       Yes  20   5
```

Metrics for K = 3

```
Values9 <- c(round(5/(5 + 54),4)*100, round(921/(921 + 20),4)*100, round(5/(20 + 5),4)*100,
             round((921 + 5)/length(test_Y2),4)*100)
kableExtra::kable(data.frame(Metrics = Metrics, Values = Values9))
```

Metrics	Values
Sensitivity	8.47
Specificity	97.87
Precision	20.00
Accuracy	92.60

The precision is higher in this case.

Prediction using KNN, K = 5

```
set.seed(1)
library(class)
knn_pred3 <- knn(train_X2,test_X2,train_Y2, k = 5)
# Confusion Matrix
table(knn_pred3, test_Y2)
```

```
##          test_Y2
## knn_pred3  No Yes
##       No  930  55
##       Yes  11   4
```

Metrics for K = 5

```
Values10 <- c(round(4/(4 + 55),4)*100, round(930/(930 + 11),4)*100, round(4/(4 + 11),4)*100,
              round((930 + 4)/length(test_Y2),4)*100)
kableExtra::kable(data.frame(Metrics = Metrics, Values = Values10))
```

Metrics	Values
Sensitivity	6.78
Specificity	98.83
Precision	26.67
Accuracy	93.40

We see that as we increase the value of K, our model accuracy and precision improves.

Comparison with Logistic Regression

Laslty, we will fit a logistic regression model to compare with KNN.

```
glm_fit <- glm(Purchase ~ ., data = caravan, family = binomial, subset = -test)
glm_prob <- predict(glm_fit, caravan[test,], type = 'response')
glm_pred <- rep("No", 1000)
glm_pred[glm_prob > 0.5] = "Yes"
table(glm_pred, test_Y2)
```

```
##         test_Y2
## glm_pred  No Yes
##      No  934  59
##      Yes   7   0
```

It seems that this model will not work since non are predicted to be Yes. But we can change the threshold to see what result we get.

```
glm_pred[glm_prob > 0.25] = "Yes"
table(glm_pred, test_Y2)
```

```
##         test_Y2
## glm_pred  No Yes
##      No  919  48
##      Yes  22  11
```

Metrics

```
Values11 <- c(round(11/(11+ 919),4)*100, round(919/(919 + 22),4)*100, round(11/(22 + 11),4)*100,
              round((919 + 11)/length(test_Y2),4)*100)
kableExtra::kable(data.frame(Metrics = Metrics, Values = Values11))
```

Metrics	Values
Sensitivity	1.18
Specificity	97.66
Precision	33.33
Accuracy	93.00

This model achieves similar efficiency as the KNN model.