Assessing Market Risk of S&P 500 Companies Using Support Vector Machines

Caleb Boateng and Lillian Caldwell

May 20, 2024

1 Introduction

1.1 Research Question

In this project, we aim to develop a predictive model to assess the market risk of publicly traded companies in the S&P 500 using beta and other financial indicators. Specifically, we will categorize companies into high risk and low risk based on their beta values and use a Support Vector Machine (SVM) to classify them.

1.2 Data Description

The data set used in this analysis consists of financial indicators for companies in the S&P 500. Key variables include beta, price-to-earnings ratio, earnings per share, dividend yield, and market capitalization. The data was produced by combining 'S&P 500 Stocks' from kaggle (https://www.kaggle.com/datasets/andrewmvd/sp-500-stocks) with the financial metrics taken from the Yahoo finance API for python.

2 Methodology

2.1 Introduction to Support Vector Machines

Support Vector Machines are a type of machine learning algorithm used for classification tasks. Classification involves predicting which category, or class, a data point belongs to. In our case, we aim to classify companies as either high risk or low risk based on their financial data.

2.2 SVM in the Context of Market Risk Classification

Concept of Hyperplane:

Think of a hyperplane as a line that separates different groups of data points in a space. In a simple two-dimensional plot, this would just be a straight line. In our project, the hyperplane separates companies into high-risk and low-risk categories based on their financial indicators.

Maximizing the Margin:

The SVM algorithm finds the hyperplane that leaves the largest possible space, or margin, between the high-risk and low-risk companies. This helps in making better predictions.

Support Vectors:

The support vectors are the data points that are closest to the hyperplane. These points are crucial because they determine the position and orientation of the hyperplane.For our project, these would be the financial indicators of companies that are closest to the decision boundary between high risk and low risk.

Linear Kernel:

We use a linear kernel, which is like drawing a straight line in a plot to separate the two classes. This approach is simple and effective for our data.

2.3 Data Preprocessing

```
library(readr)
data <- read_csv("Desktop/Stat Learning/tech_company_data3.csv")
```

```
## Rows: 505 Columns: 10
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): ticker
## dbl (9): beta, marketCap, forwardPE, debtToEquity, dividendYield, returnOnEq...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

2.3.1 Cleaning and Scaling Data

First, we will clean the data by removing unnecessary columns and handling missing values. We will also scale the features to ensure they are on the same scale. Also, SVM works best with classification, so in order to get the best results we turned beta into a categorical variable named RiskCategory. According to bankrate.com, any beta that exceeds 1 implies higher risk, vice-versa.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caTools)
library(tidyr)

cleaned_data =drop_na(data)

no_tick <- cleaned_data |> dplyr::select(-ticker)
no_tick$RiskCategory <- ifelse(no_tick$beta < 1, "Low Risk", "High Risk")
no_tick$beta <- NULL


data_scaled <- scale(no_tick[, -ncol(no_tick)])
data_scaled <- as.data.frame(data_scaled)
data_scaled$RiskCategory <- no_tick$RiskCategory
data_scaled$RiskCategory <- factor(data_scaled$RiskCategory, levels = c("Low Risk", "High Risk"))
data_scaled <- na.omit(data_scaled)
```

2.3.2 Exploratory Data Analysis

```
summary(data_scaled)
```

```
##    marketCap          forwardPE        debtToEquity      dividendYield
##  Min.   :-0.33416   Min.   :-3.4999   Min.   :-0.34136   Min.   :-1.5525
##  1st Qu.:-0.28306   1st Qu.:-0.5059   1st Qu.:-0.26112   1st Qu.:-0.8146
##  Median :-0.21911   Median :-0.2239   Median :-0.19236   Median :-0.1650
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.:-0.09342   3rd Qu.: 0.3719   3rd Qu.:-0.06424   3rd Qu.: 0.6360
##  Max.   : 9.48008   Max.   : 9.5566   Max.   :13.26179   Max.   : 3.7011
##  returnOnEquity     trailingEps        currentRatio      operatingMargins
##  Min.   :-0.4094   Min.   :-2.8143   Min.   :-0.90022   Min.   :-2.4205
##  1st Qu.:-0.1337   1st Qu.:-0.5784   1st Qu.:-0.41449   1st Qu.:-0.7237
##  Median :-0.1097   Median :-0.1858   Median :-0.22232   Median :-0.1292
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.:-0.0635   3rd Qu.: 0.3284   3rd Qu.: 0.03735   3rd Qu.: 0.5768
##  Max.   :17.2203   Max.   : 4.9068   Max.   :10.47256   Max.   : 3.8437
##     RiskCategory
##  Low Risk :158
##  High Risk:173
##
##
##
##
```

3 Model Implementations

3.1 Splitting the Data

```
set.seed(123)
split <- sample.split(data_scaled$RiskCategory, SplitRatio = 0.8)
train_data <- data_scaled[split, ]
test_data <- data_scaled[!split, ]

train_data$RiskCategory <- as.factor(train_data$RiskCategory)
test_data$RiskCategory <- as.factor(test_data$RiskCategory)
```

3.2 Training the SVM Model

```
library(e1071)


svm_model <- svm(RiskCategory ~ ., data = train_data, type = 'C-classification', kernel = 'linear')
```

3.3 Training LDA Model

LDA looks for a linear combination of the features (financial indicators) that can best separate the high-risk companies from the low-risk ones.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
lda_model <- lda(RiskCategory ~ ., data = train_data)
```

3.4 Model Evaluation

```
svm_predictions <- predict(svm_model, test_data)


library(caret)
```

```
## Loading required package: ggplot2
```

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

```
svm_confusion_matrix <- confusionMatrix(svm_predictions, test_data$RiskCategory)

lda_predictions <- predict(lda_model, test_data)
lda_predicted_classes <- lda_predictions$class

lda_confusion_matrix <- confusionMatrix(lda_predicted_classes, test_data$RiskCategory)
```

4 Results

```
svm_confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Low Risk High Risk
##   Low Risk        14         6
##   High Risk       18        29
##
##                Accuracy : 0.6418
##                  95% CI : (0.5153, 0.7553)
##     No Information Rate : 0.5224
##     P-Value [Acc > NIR] : 0.03257
##
##                   Kappa : 0.2704
##
##  Mcnemar's Test P-Value : 0.02474
##
##             Sensitivity : 0.4375
##             Specificity : 0.8286
##          Pos Pred Value : 0.7000
##          Neg Pred Value : 0.6170
##              Prevalence : 0.4776
##          Detection Rate : 0.2090
##    Detection Prevalence : 0.2985
##       Balanced Accuracy : 0.6330
##
##        'Positive' Class : Low Risk
##
```

```
lda_confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Low Risk High Risk
##   Low Risk        15         4
##   High Risk       17        31
##
##                Accuracy : 0.6866
##                  95% CI : (0.5616, 0.7944)
##     No Information Rate : 0.5224
##     P-Value [Acc > NIR] : 0.004694
##
##                   Kappa : 0.3607
##
##  Mcnemar's Test P-Value : 0.008829
##
##             Sensitivity : 0.4688
##             Specificity : 0.8857
##          Pos Pred Value : 0.7895
##          Neg Pred Value : 0.6458
##              Prevalence : 0.4776
##          Detection Rate : 0.2239
##    Detection Prevalence : 0.2836
##       Balanced Accuracy : 0.6772
##
##        'Positive' Class : Low Risk
##
```

5 Comparison of SVM and LDA

```
library(e1071)
library(MASS)
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

5.1 Perform PCA

```
pca <- prcomp(data_scaled[, -ncol(data_scaled)], scale. = TRUE)
data_pca <- as.data.frame(pca$x[, 1:2])
data_pca$RiskCategory <- data_scaled$RiskCategory
```

5.2 Train SVM on PCA-reduced data

```
svm_model_pca <- svm(RiskCategory ~ ., data = data_pca, type = 'C-classification', kernel = 'linear')
```

5.3 Train LDA on PCA-reduced data

```
lda_model_pca <- lda(RiskCategory ~ ., data = data_pca)
```

5.4 Create a grid of points to plot decision boundaries

```
xrange <- seq(min(data_pca$PC1), max(data_pca$PC1), length.out = 200)
yrange <- seq(min(data_pca$PC2), max(data_pca$PC2), length.out = 200)
grid <- expand.grid(PC1 = xrange, PC2 = yrange)
```

5.5 Predict classes for grid points

```
svm_grid_pred <- predict(svm_model_pca, grid)
lda_grid_pred <- predict(lda_model_pca, grid)$class
```

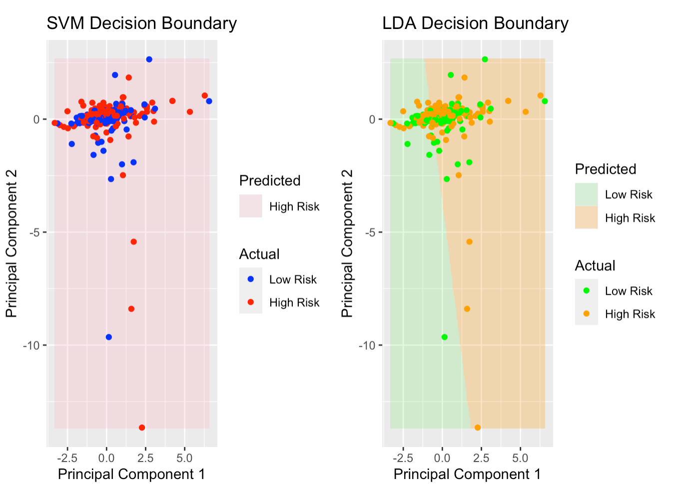
5.6 Plot decision boundaries

```
plot_svm <- ggplot() +
  geom_tile(data = grid, aes(x = PC1, y = PC2, fill = as.factor(svm_grid_pred)), alpha = 0.3) +
  geom_point(data = data_pca, aes(x = PC1, y = PC2, color = RiskCategory)) +
  labs(title = "SVM Decision Boundary", x = "Principal Component 1", y = "Principal Component 2") +
  scale_fill_manual(values = c("Low Risk" = "lightblue", "High Risk" = "pink"), name = "Predicted") +
  scale_color_manual(values = c("Low Risk" = "blue", "High Risk" = "red"), name = "Actual")

plot_lda <- ggplot() +
  geom_tile(data = grid, aes(x = PC1, y = PC2, fill = as.factor(lda_grid_pred)), alpha = 0.3) +
  geom_point(data = data_pca, aes(x = PC1, y = PC2, color = RiskCategory)) +
  labs(title = "LDA Decision Boundary", x = "Principal Component 1", y = "Principal Component 2") +
  scale_fill_manual(values = c("Low Risk" = "lightgreen", "High Risk" = "orange"), name = "Predicted") +
  scale_color_manual(values = c("Low Risk" = "green", "High Risk" = "orange"), name = "Actual")
```

5.7 Combine plots

```
grid.arrange(plot_svm, plot_lda, ncol = 2)
```



Overall, LDA slightly outperforms SVM in this task, showing better accuracy, precision, specificity, and Kappa values. Both models perform equally in terms of sensitivity, but LDA provides more reliable classification performance according to multiple metrics.

To better capture and visualize the differences and similarities between the high-risk and low-risk classes after creating the two models, we created 2D decision boundary plots. This type of plot helps visualize how the SVM and LDA models separate the two classes based on their decision boundaries. It is especially effective when combined with a Principal Component Analysis to reduce the dimensionality of the data to two dimensions, making it easier to plot and interpret.

The SVM model predicts most of the companies as high risk, as indicated by the red shading across almost the entire plot. This results in many low-risk companies being misclassified as high risk. The model does not effectively capture the separation between high-risk and low-risk companies in this PCA-reduced space.The LDA model provides a clearer and more balanced separation between high-risk and low-risk companies. The decision boundary effectively differentiates the two classes, with fewer misclassifications. The green and orange regions show a more logical separation based on the PCA components.

6 Discussion

6.1 Strengths and Limitations

6.1.1 Strengths

SVM is effective for high-dimensional data and provides a clear margin of separation.

LDA is a robust method when the class distributions are Gaussian with identical covariances.

6.1.2 Limitations

SVM can be sensitive to the choice of kernel and regularization parameters.

LDA assumes normally distributed classes with identical covariances, which might not always hold.

7 Future Work

Future research could explore more sophisticated models, such as non-linear kernels or ensemble methods, and include additional financial metrics.

8 Conclusion

This project demonstrates the use of SVM for classifying market risk in S&P 500 companies. The SVM model…