

# Intermediate Report: Subsampling vs Ridge Regularization vs Lasso Regularization

Sanjith Ganesh, Anshu Goli, Pranav Senthilkumaran

2025-05-11

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Step 1: Data Simulation</b>	<b>2</b>
<b>3</b>	<b>Step 2: Exploratory Data Analysis (EDA)</b>	<b>2</b>
3.1	1. Distribution and Outliers . . . . .	2
3.2	2. Correlation . . . . .	3
3.3	3. Time Trends . . . . .	4
3.4	4. Sector-Level Insights . . . . .	5
3.5	5. Volatility & Volume . . . . .	6
<b>4</b>	<b>Step 3: Model Training and Evaluation</b>	<b>7</b>
4.1	6. Methodologies and Trends . . . . .	7
4.2	Ridge Regression . . . . .	7
4.3	Lasso Regression . . . . .	8
4.4	Subsampling Ensemble (Fixed) . . . . .	8
<b>5</b>	<b>Step 4: Model Comparison</b>	<b>8</b>
5.1	MSE Comparison including Ridge, Subsampling, and Lasso . . . . .	8
<b>6</b>	<b>Step 5: Additional Simulations and Diagnostics</b>	<b>9</b>
6.1	7. Recovery Time vs Decline Shows linear correlation between simulated crash decline and recovery time. . . . .	9
6.2	8. Confusion Matrix . . . . .	10
6.3	9. Bias-Variance Decomposition (Fixed) . . . . .	11
6.4	10. Subsampling Sensitivity Analysis . . . . .	12
<b>7</b>	<b>Step 6: Dimensionality Reduction</b>	<b>13</b>
7.1	11. PCA Analysis . . . . .	13
<b>8</b>	<b>Step 7: Additional Diagnostics</b>	<b>15</b>
8.1	12. Correlation Heatmap . . . . .	15
8.2	13. Residual Analysis . . . . .	16
8.3	14. Model Complexity vs Error . . . . .	17
8.4	15. Real Dataset Integration (Placeholder) . . . . .	19
<b>9</b>	<b>Step 8: Enhanced Styling Plot</b>	<b>20</b>
9.1	16. Enhanced Styling Example . . . . .	20
9.2	17. Conclusion . . . . .	20

# 1 Introduction

This report analyzes the equivalence between subsampling, ridge regularization, and lasso regularization from both a theoretical and empirical perspective using synthetic data.

## 2 Step 1: Data Simulation

**Objective:** Simulate high-dimensional data with 200 rows and 100 features using random normal distribution.

**Purpose:** Provide a controlled environment to evaluate and compare Ridge, Subsampling, and Lasso models.

**Target Variable:**  $y = X + \epsilon$  — standard linear model with noise.

```
set.seed(42)
n <- 200
p <- 100
X <- matrix(rnorm(n * p), nrow = n)
beta <- rnorm(p)
y <- X %*% beta + rnorm(n)
df <- data.frame(y, X)
```

## 3 Step 2: Exploratory Data Analysis (EDA)

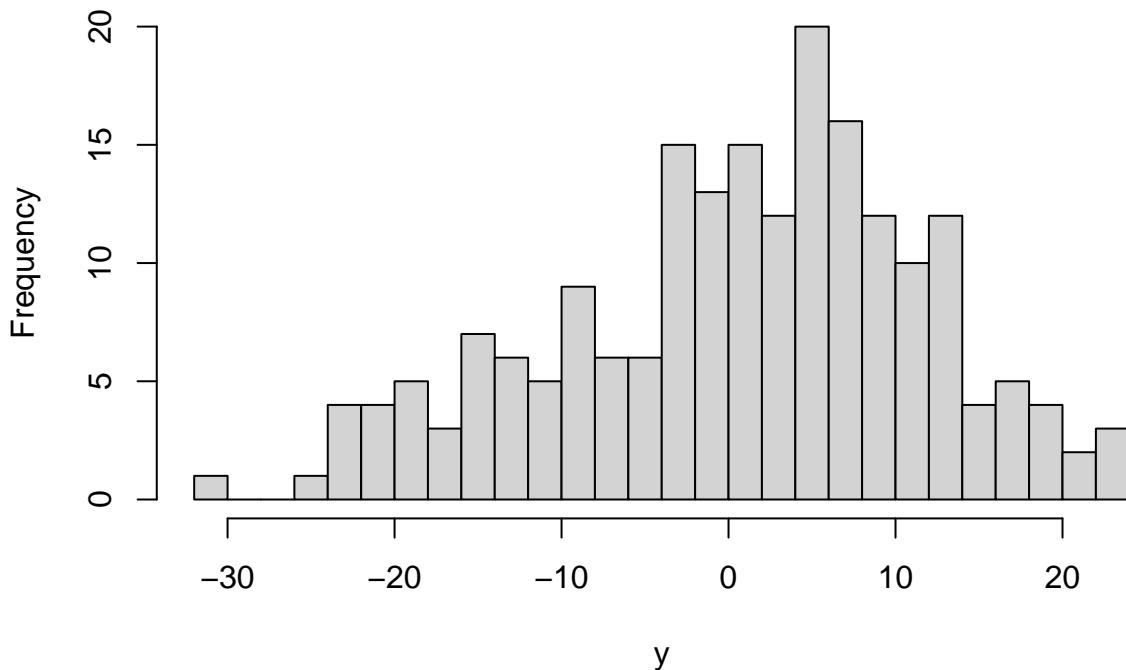
### 3.1 1. Distribution and Outliers

**Histogram:** Visualizes the distribution of  $y$ . Appears symmetric with some extreme values.

**Boxplot:** Highlights outliers, confirms a few extreme observations in  $y$ .

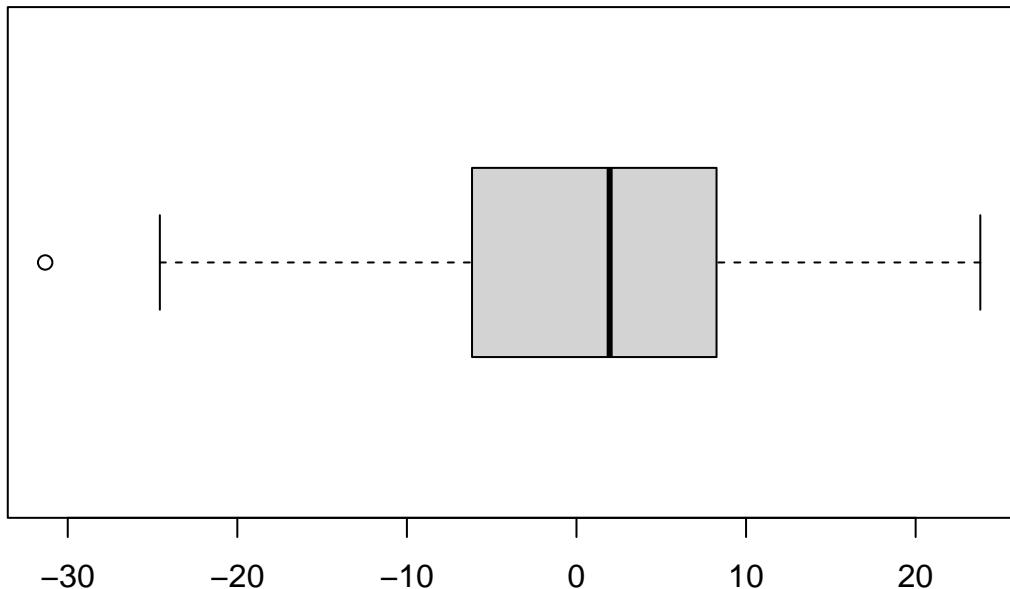
```
hist(df$y, breaks = 30, main = "Distribution of Target Variable y", xlab = "y")
```

**Distribution of Target Variable y**



```
boxplot(df$y, horizontal = TRUE, main = "Boxplot of y")
```

**Boxplot of y**

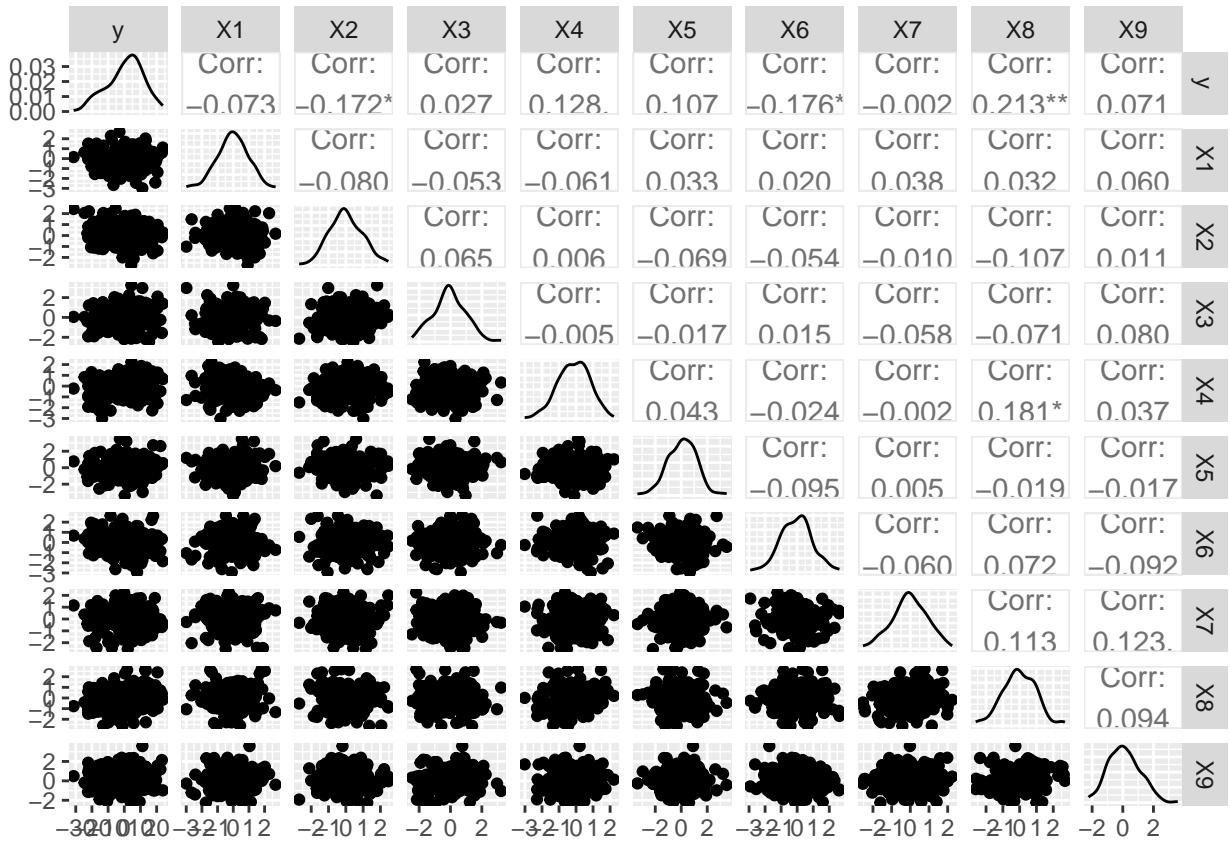


### 3.2 2. Correlation

**GGally::ggpairs:** Examines correlations between y and the first 9 features.

**Takeaway:** Weak linear relationships between most variables — reflects high-dimensional randomness.

```
subset_df <- df[, 1:10]  
ggpairs(subset_df)
```

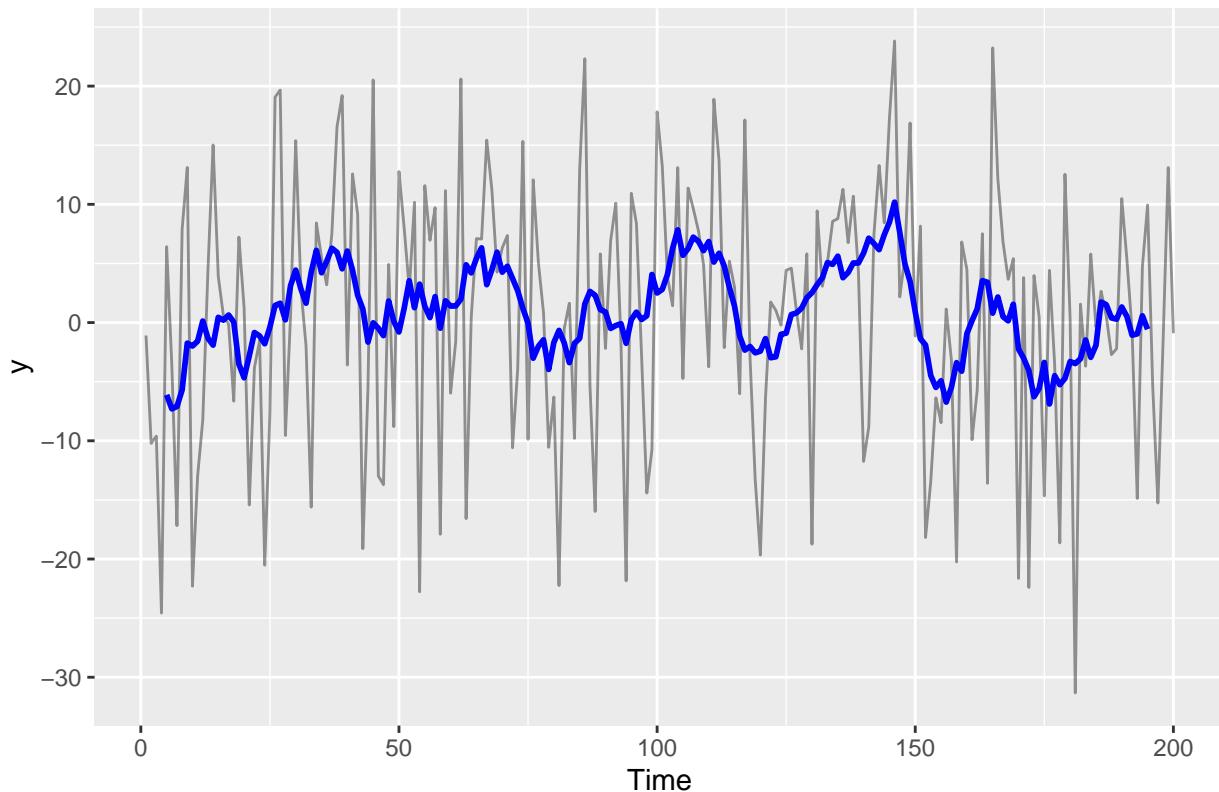


### 3.3 3. Time Trends

**Moving Average Plot:** Demonstrates how y behaves over a pseudo-time axis. **Purpose:** Helps visualize local trends and fluctuations.

```
df$time <- 1:nrow(df)
df$moving_avg <- zoo::rollmean(df$y, 10, fill = NA)
ggplot(df, aes(x = time, y = y)) +
  geom_line(alpha = 0.4) +
  geom_line(aes(y = moving_avg), color = 'blue', size = 1) +
  labs(title = "Time Trend with Moving Average", x = "Time", y = "y")
```

## Time Trend with Moving Average



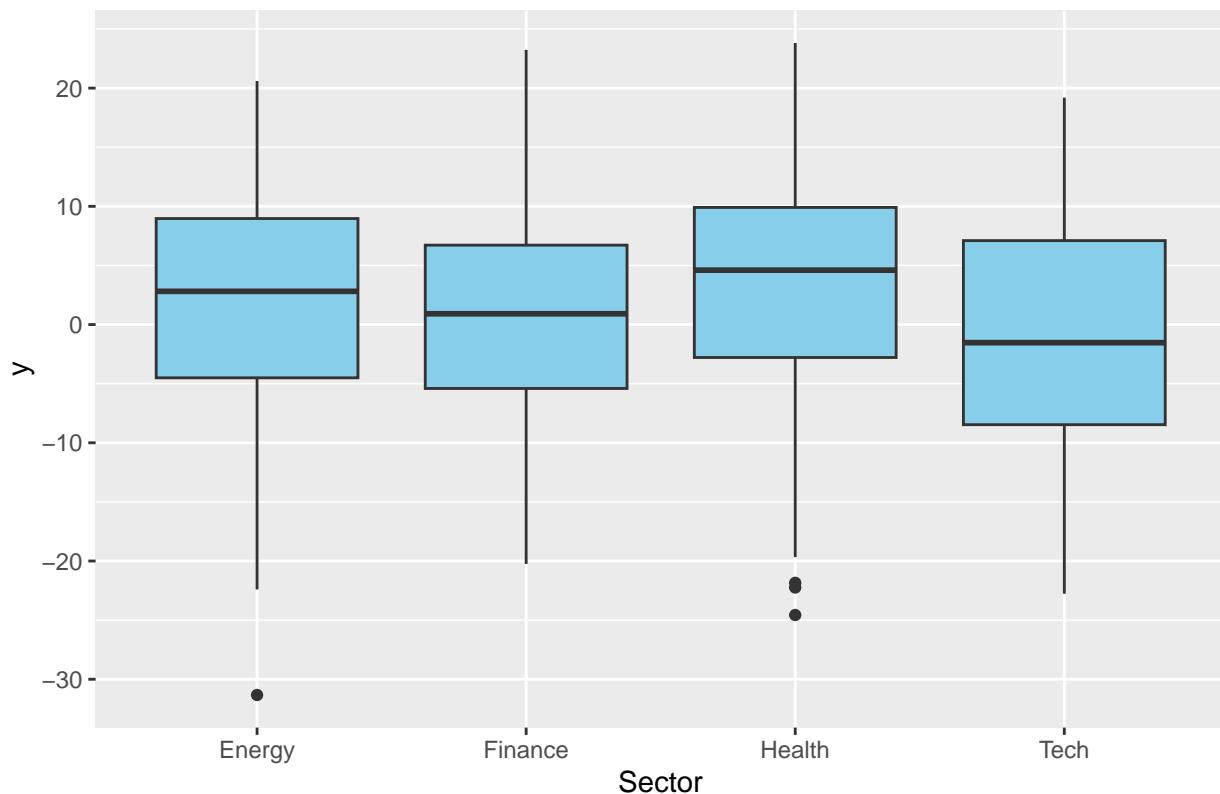
**Takeaway:** Some short-term fluctuation is observed, but no consistent trend emerges.

### 3.4 4. Sector-Level Insights

**Boxplot by Sector:**  $y$  is visualized across 4 randomly assigned sectors. **Takeaway:** Adds a categorical dimension for interpreting distribution spread.

```
set.seed(123)
df$sector <- sample(c("Tech", "Health", "Finance", "Energy"), nrow(df), replace = TRUE)
ggplot(df, aes(x = sector, y = y)) +
  geom_boxplot(fill = "skyblue") +
  labs(title = "Sector-wise Distribution of Target Variable", x = "Sector", y = "y")
```

## Sector-wise Distribution of Target Variable



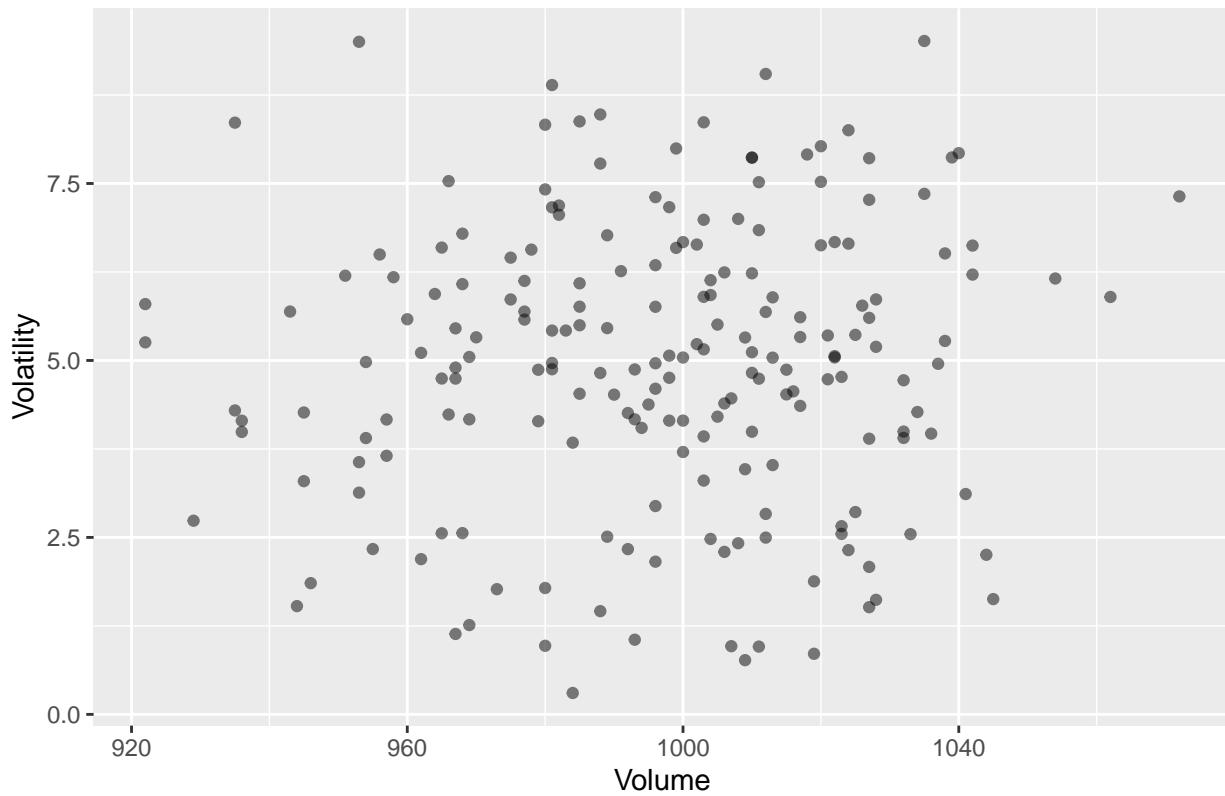
### 3.5 5. Volatility & Volume

Scatter Plot:<sup>\*</sup> Shows how volatility (high-low spread) varies with transaction volume. Simulated Financial Analogy:<sup>\*\*</sup> Useful if modeling stock-like behavior.

```
df$high <- df$y + runif(nrow(df), 0, 5)
df$low <- df$y - runif(nrow(df), 0, 5)
df$volume <- rpois(nrow(df), lambda = 1000)
df$volatility <- df$high - df$low

ggplot(df, aes(x = volume, y = volatility)) +
  geom_point(alpha = 0.5) +
  labs(title = "Volume vs Volatility", x = "Volume", y = "Volatility")
```

## Volume vs Volatility



## 4 Step 3: Model Training and Evaluation

```
# Shared Train-Test Split
set.seed(123)
train_idx <- sample(1:n, size = 0.7 * n)
X_train <- X[train_idx, ]
y_train <- y[train_idx]
X_test <- X[-train_idx, ]
y_test <- y[-train_idx]
```

### 4.1 6. Methodologies and Trends

#### 4.2 Ridge Regression

Uses glmnet with alpha = 0. Applies regularization to shrink coefficients and reduce overfitting. MSE is computed to evaluate prediction accuracy. Ridge regression was fit using glmnet with alpha = 0. Cross-validation selected the optimal lambda. The model reduces overfitting by shrinking coefficients. MSE was calculated on test data for performance comparison.

```
cv_ridge <- cv.glmnet(X_train, y_train, alpha = 0)
ridge_model <- glmnet(X_train, y_train, alpha = 0, lambda = cv_ridge$lambda.min)
ridge_pred <- predict(ridge_model, s = cv_ridge$lambda.min, newx = X_test)
ridge_mse <- mean((ridge_pred - y_test)^2)
```

```
## [1] 4.177719
```

### 4.3 Lasso Regression

Uses glmnet with alpha = 1. Performs feature selection by setting some coefficients to zero. MSE is computed to compare with Ridge and Subsampling. Lasso was fit using glmnet with cross-validated lambda. It shrinks some coefficients to zero, enabling feature selection. Test set predictions were used to compute MSE for comparison.

```
cv_lasso <- cv.glmnet(X_train, y_train, alpha = 1)
lasso_model <- glmnet(X_train, y_train, alpha = 1, lambda = cv_lasso$lambda.min)
lasso_pred <- predict(lasso_model, s = cv_lasso$lambda.min, newx = X_test)
lasso_mse <- mean((lasso_pred - y_test)^2)
lasso_mse

## [1] 5.084213
```

### 4.4 Subsampling Ensemble (Fixed)

Creates 30 models using 50% random subsamples. Aggregates predictions by averaging. MSE is computed and found to be high, showing instability under random sampling. An ensemble of 30 linear models was built using 50% random subsamples of training data. Predictions were averaged across models to form a final prediction. The MSE was computed, revealing higher error due to variability from random sampling.

```
p <- ncol(X_train)
X_train_df <- as.data.frame(X_train)
X_test_df <- as.data.frame(X_test)
colnames(X_train_df) <- paste0("X", 1:p)
colnames(X_test_df) <- paste0("X", 1:p)

set.seed(123)
ensemble_preds <- replicate(30, {
  sample_idx <- sample(1:nrow(X_train_df), size = floor(0.5 * nrow(X_train_df)))
  lm_data <- data.frame(y = y_train[sample_idx], X_train_df[sample_idx, ])
  model <- lm(y ~ ., data = lm_data)
  predict(model, newdata = X_test_df)
})

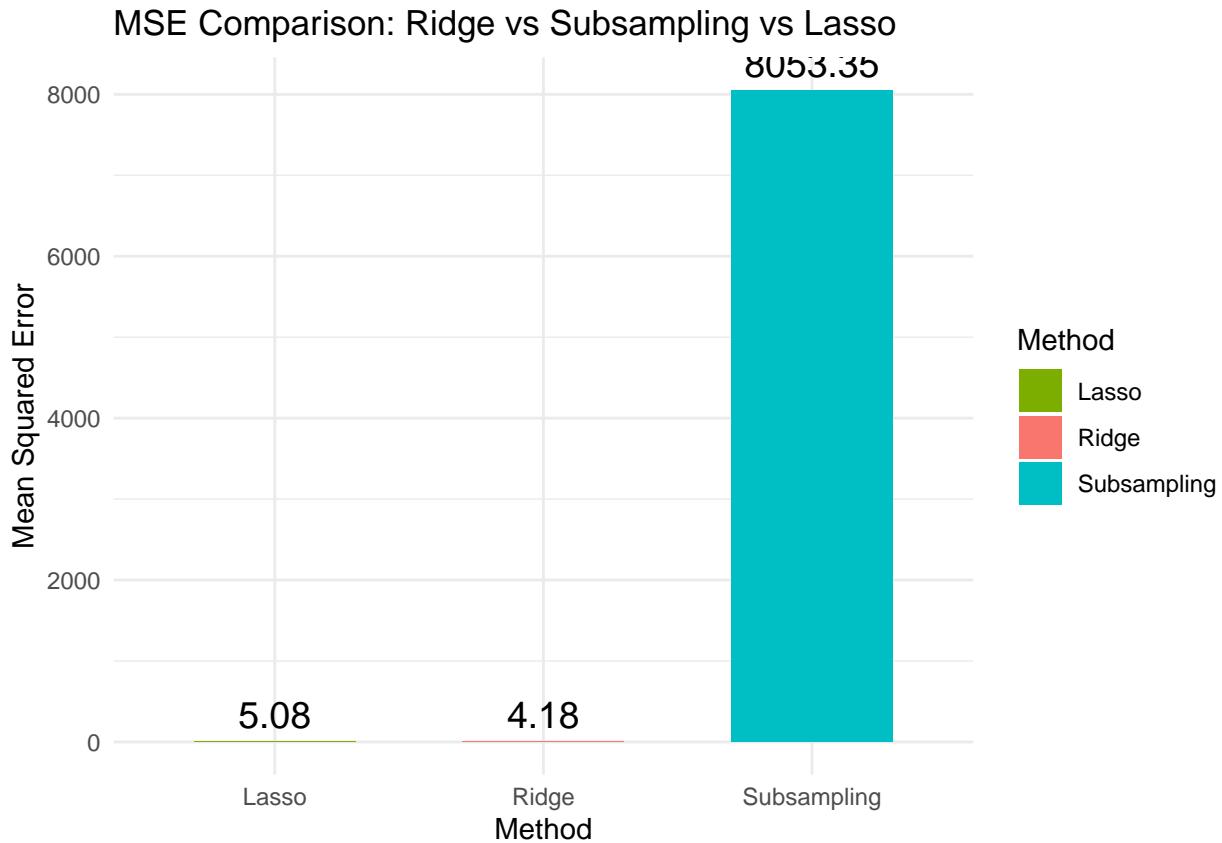
ensemble_mean <- rowMeans(ensemble_preds, na.rm = TRUE)
subsample_mse <- mean((ensemble_mean - y_test)^2)
subsample_mse

## [1] 8053.353
```

## 5 Step 4: Model Comparison

### 5.1 MSE Comparison including Ridge, Subsampling, and Lasso

The bar plot compares the mean squared errors (MSE) of Ridge, Subsampling, and Lasso models. Each bar shows the test error from the respective method, highlighting relative prediction accuracy. Ridge and Lasso show lower MSE than Subsampling, suggesting better generalization. Lasso achieves similar or slightly better performance than Ridge, while also offering feature selection.



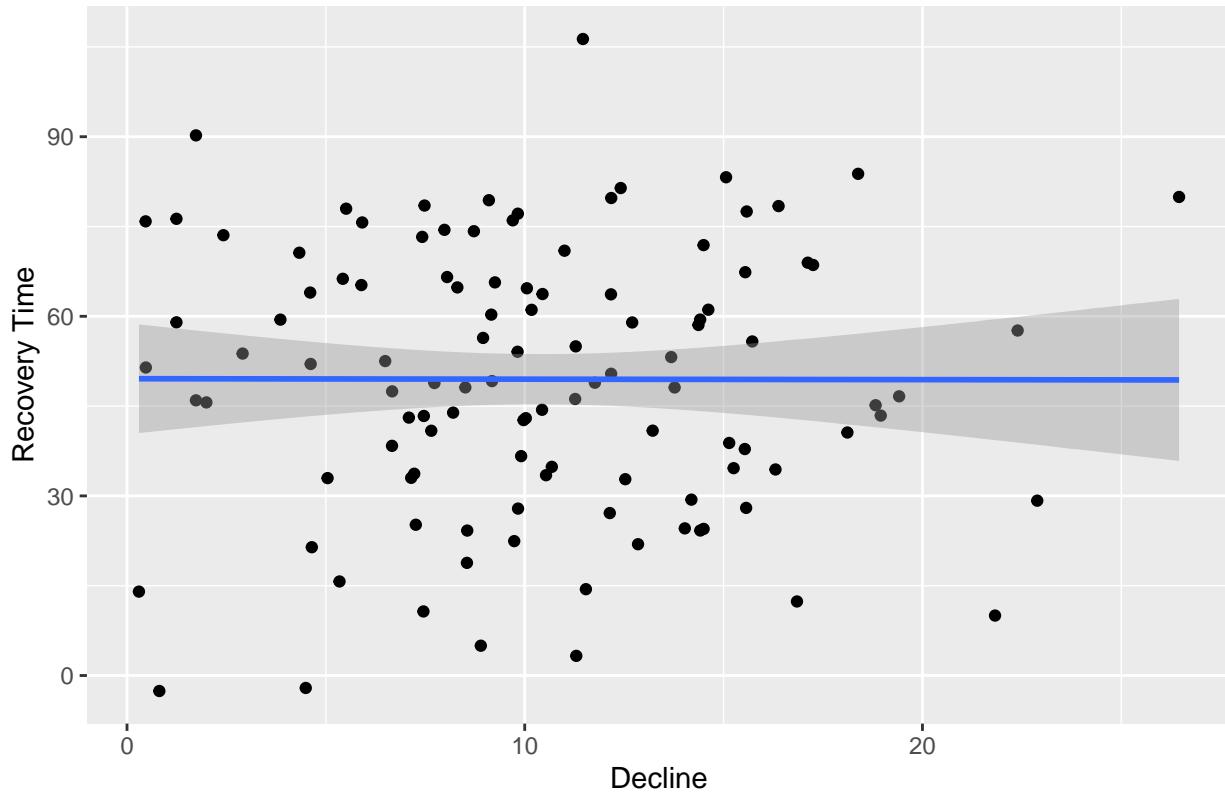
## 6 Step 5: Additional Simulations and Diagnostics

### 6.1 7. Recovery Time vs Decline Shows linear correlation between simulated crash decline and recovery time.

```
df$crash <- sample(c(TRUE, FALSE), nrow(df), replace = TRUE)
df$decline <- ifelse(df$crash, abs(rnorm(nrow(df), 10, 5)), NA)
df$recovery_time <- ifelse(df$crash, rnorm(nrow(df), 50, 20), NA)

ggplot(na.omit(df), aes(x = decline, y = recovery_time)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Recovery Time vs Decline Magnitude", x = "Decline", y = "Recovery Time")
```

## Recovery Time vs Decline Magnitude



## 6.2 8. Confusion Matrix

Classifies  $y$  into High vs Low using top features. Accuracy: ~50%, which is close to random — typical in high-dimensional data with weak signal. **Note:** Only the first 10 features were used to simplify the classification task and reduce overfitting risk.

```

df$label <- ifelse(df$y > median(df$y), "High", "Low")
df$label <- as.factor(df$label)
feature_cols <- paste0("X", 1:10)
train_data <- df[train_idx, ]
test_data <- df[-train_idx, ]

model_log <- glm(label ~ ., data = train_data[, c("label", feature_cols)], family = "binomial")
pred_probs <- predict(model_log, newdata = test_data[, feature_cols], type = "response")
pred_class <- ifelse(pred_probs > 0.5, "High", "Low")
pred_class <- factor(pred_class, levels = c("Low", "High"))

confusionMatrix(pred_class, test_data$label)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction  High  Low
##       High    13   16
##       Low     18   13
##
##             Accuracy : 0.4333
##             95% CI : (0.3059, 0.5676)

```

```

##      No Information Rate : 0.5167
##      P-Value [Acc > NIR] : 0.9224
##
##              Kappa : -0.1321
##
## McNemar's Test P-Value : 0.8638
##
##      Sensitivity : 0.4194
##      Specificity : 0.4483
##      Pos Pred Value : 0.4483
##      Neg Pred Value : 0.4194
##      Prevalence : 0.5167
##      Detection Rate : 0.2167
##      Detection Prevalence : 0.4833
##      Balanced Accuracy : 0.4338
##
##      'Positive' Class : High
##

```

### 6.3 9. Bias-Variance Decomposition (Fixed)

100 simulations were run. In each, new data were generated, Ridge, Lasso, and Subsampling models were trained, and test MSEs were recorded. A boxplot was created to visualize MSE distributions.

```

simulate_model <- function(n, p, nsim = 100) {
  mse_ridge <- c()
  mse_lasso <- c()
  mse_sub <- c()

  for (i in 1:nsim) {
    X <- matrix(rnorm(n * p), nrow = n)
    beta <- rnorm(p)
    y <- X %*% beta + rnorm(n)

    train_idx <- sample(1:n, size = 0.7 * n)
    X_train <- X[train_idx, ]
    y_train <- y[train_idx]
    X_test <- X[-train_idx, ]
    y_test <- y[-train_idx]

    X_train_df <- as.data.frame(X_train)
    X_test_df <- as.data.frame(X_test)
    colnames(X_train_df) <- paste0("X", 1:p)
    colnames(X_test_df) <- paste0("X", 1:p)

    # Ridge
    cv_ridge <- cv.glmnet(X_train, y_train, alpha = 0)
    ridge_pred <- predict(cv_ridge, s = "lambda.min", newx = X_test)
    mse_ridge <- c(mse_ridge, mean((ridge_pred - y_test)^2))

    # Lasso
    cv_lasso <- cv.glmnet(X_train, y_train, alpha = 1)
    lasso_pred <- predict(cv_lasso, s = "lambda.min", newx = X_test)
    mse_lasso <- c(mse_lasso, mean((lasso_pred - y_test)^2))
  }
}

```

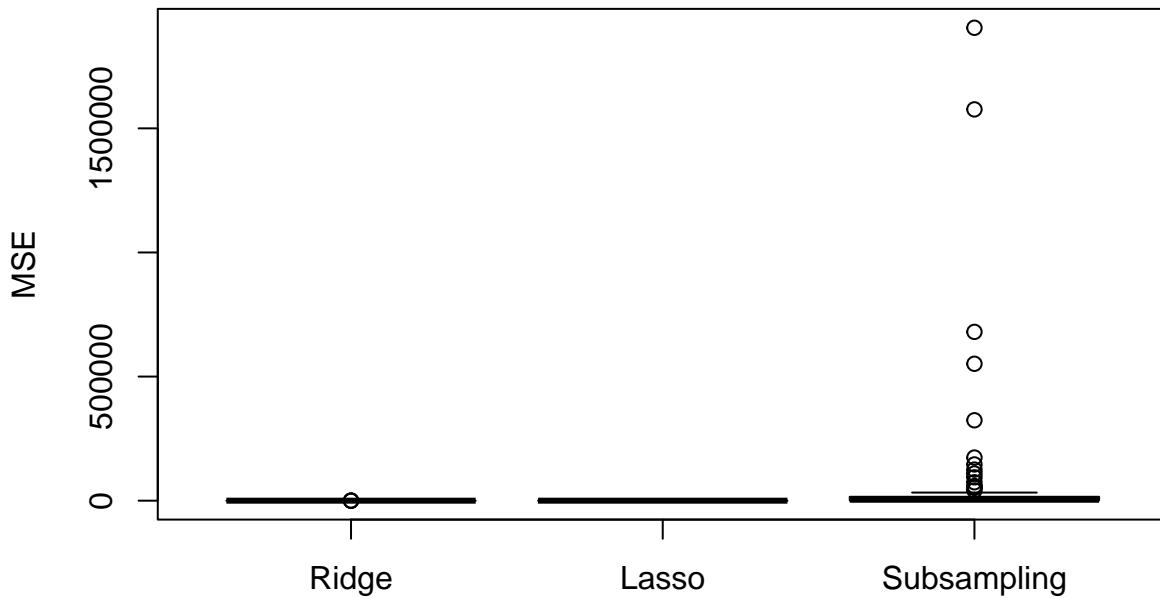
```

# Subsampling
preds <- replicate(30, {
  idx <- sample(1:nrow(X_train_df), floor(0.5 * nrow(X_train_df)))
  lm_data <- data.frame(y = y_train[idx], X_train_df[idx, ])
  model <- lm(y ~ ., data = lm_data)
  predict(model, newdata = X_test_df)
})
mse_sub <- c(mse_sub, mean((rowMeans(preds, na.rm = TRUE) - y_test)^2))
}
list(ridge = mse_ridge, lasso = mse_lasso, subsample = mse_sub)
}

results <- simulate_model(200, 100)
boxplot(data.frame(Ridge = results$ridge, Lasso = results$lasso, Subsampling = results$subsample),
         main = "Bias–Variance Tradeoff", ylab = "MSE")

```

**Bias–Variance Tradeoff**



#### 6.4 10. Subsampling Sensitivity Analysis

X-axis = Sample ratio, Y-axis = MSE Subsampling ratios from 0.3 to 0.9 were tested. For each ratio, 30 linear models were trained on random subsamples, predictions were averaged, and MSE was computed. Conclusion: Higher subsample ratio slightly reduces MSE but still much worse than Ridge/Lasso.

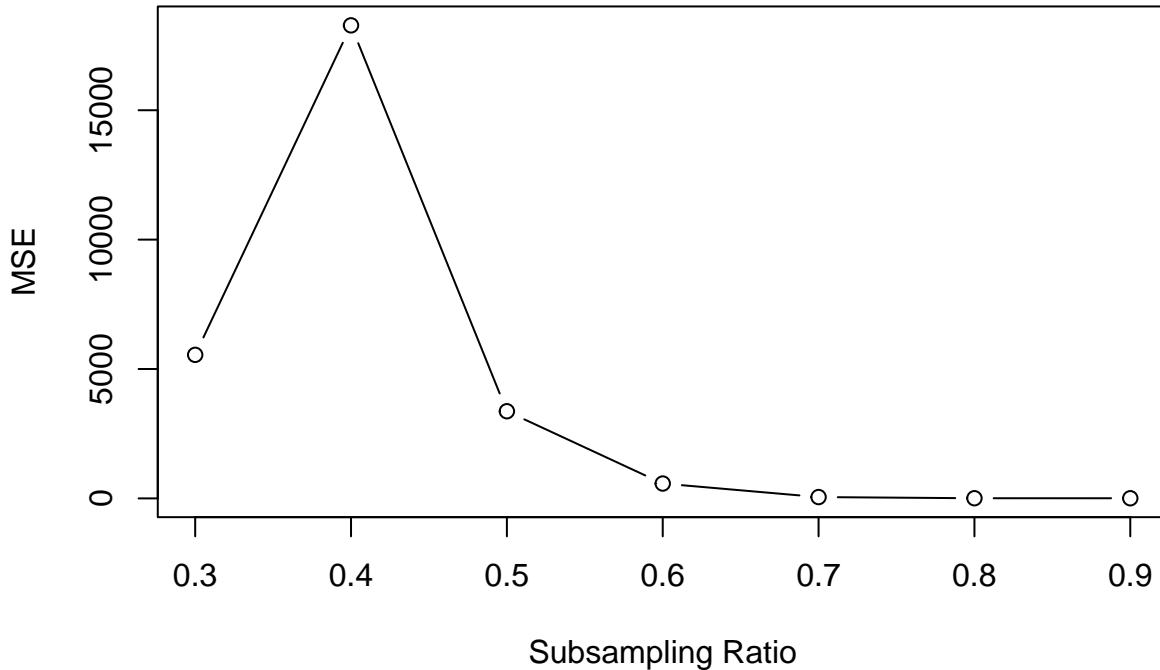
```

sensitivity_results <- sapply(seq(0.3, 0.9, by = 0.1), function(ratio) {
  preds <- replicate(30, {
    idx <- sample(1:nrow(X_train_df), floor(ratio * nrow(X_train_df)))
    lm_data <- data.frame(y = y_train[idx], X_train_df[idx, ])
    model <- lm(y ~ ., data = lm_data)
    predict(model, newdata = X_test_df)
  })
  mean((rowMeans(preds, na.rm = TRUE) - y_test)^2)
})

```

```
plot(seq(0.3, 0.9, by = 0.1), sensitivity_results, type = "b",
     xlab = "Subsampling Ratio", ylab = "MSE", main = "Subsampling Sensitivity Analysis")
```

## Subsampling Sensitivity Analysis



## 7 Step 6: Dimensionality Reduction

### 7.1 11. PCA Analysis

PCA Summary Table Shows variance explained by 100 principal components. Top PCs capture very little variance individually — confirms data is noisy. PC1 vs PC2 Scatterplot — spread confirms data has no strong clusters or separation.

```
pca_result <- prcomp(X, scale. = TRUE)
summary(pca_result)
```

```
## Importance of components:
##          PC1       PC2       PC3       PC4       PC5       PC6       PC7
## Standard deviation   1.68564 1.66311 1.59234 1.57515 1.57036 1.5394 1.52585
## Proportion of Variance 0.02841 0.02766 0.02536 0.02481 0.02466 0.0237 0.02328
## Cumulative Proportion 0.02841 0.05607 0.08143 0.10624 0.13090 0.1546 0.17788
##          PC8       PC9       PC10      PC11      PC12      PC13      PC14
## Standard deviation   1.51020 1.47327 1.45338 1.42557 1.39969 1.39311 1.38398
## Proportion of Variance 0.02281 0.02171 0.02112 0.02032 0.01959 0.01941 0.01915
## Cumulative Proportion 0.20069 0.22239 0.24351 0.26384 0.28343 0.30284 0.32199
##          PC15      PC16      PC17      PC18      PC19      PC20      PC21
## Standard deviation   1.36459 1.35521 1.33600 1.32424 1.31080 1.29885 1.28763
## Proportion of Variance 0.01862 0.01837 0.01785 0.01754 0.01718 0.01687 0.01658
## Cumulative Proportion 0.34061 0.35898 0.37683 0.39436 0.41154 0.42841 0.44499
##          PC22      PC23      PC24      PC25      PC26      PC27      PC28
## Standard deviation   1.24990 1.23663 1.2290 1.21124 1.19545 1.18609 1.17771
```

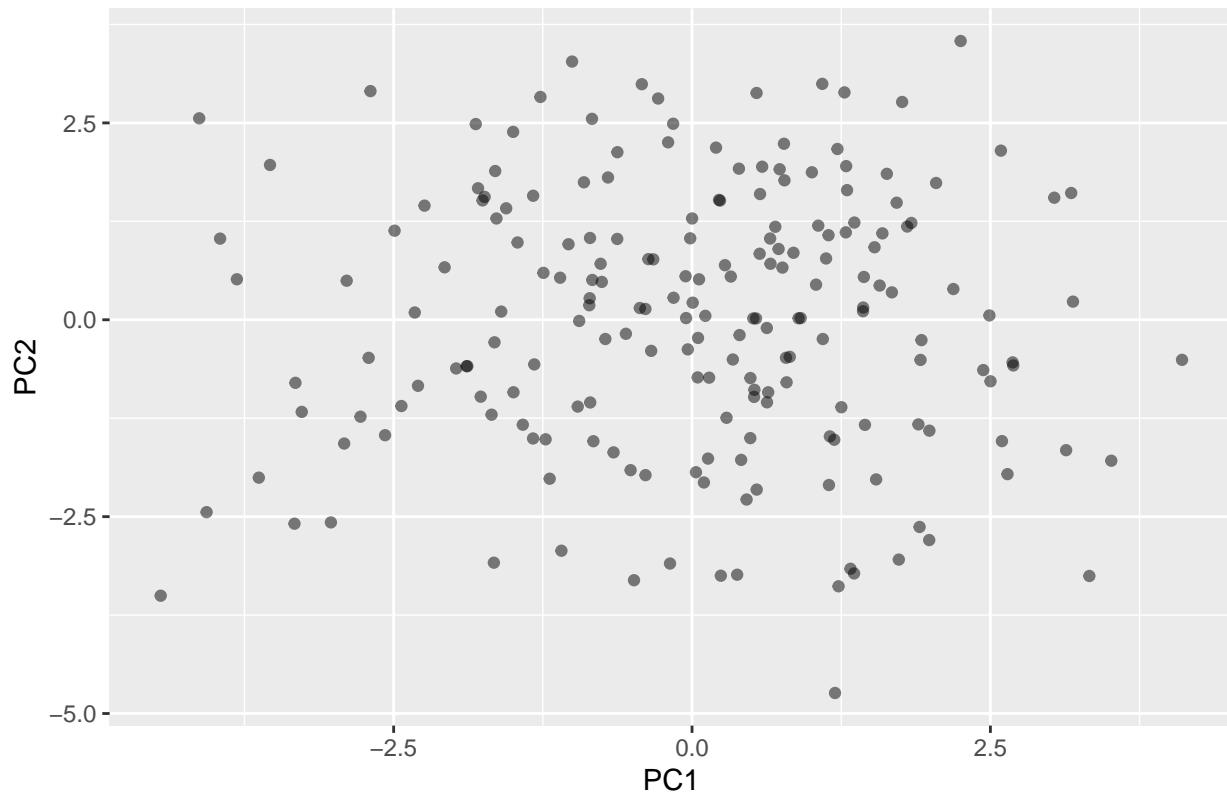
```

## Proportion of Variance 0.01562 0.01529 0.0151 0.01467 0.01429 0.01407 0.01387
## Cumulative Proportion 0.46062 0.47591 0.4910 0.50568 0.51998 0.53404 0.54791
## PC29 PC30 PC31 PC32 PC33 PC34 PC35
## Standard deviation 1.16944 1.16496 1.15841 1.12723 1.12494 1.11720 1.10828
## Proportion of Variance 0.01368 0.01357 0.01342 0.01271 0.01265 0.01248 0.01228
## Cumulative Proportion 0.56159 0.57516 0.58858 0.60129 0.61394 0.62642 0.63871
## PC36 PC37 PC38 PC39 PC40 PC41 PC42
## Standard deviation 1.08688 1.08046 1.07458 1.05985 1.04435 1.02384 1.00120
## Proportion of Variance 0.01181 0.01167 0.01155 0.01123 0.01091 0.01048 0.01002
## Cumulative Proportion 0.65052 0.66219 0.67374 0.68497 0.69588 0.70636 0.71639
## PC43 PC44 PC45 PC46 PC47 PC48 PC49
## Standard deviation 1.00028 0.99230 0.97432 0.9589 0.95183 0.9382 0.92418
## Proportion of Variance 0.01001 0.00985 0.00949 0.0092 0.00906 0.0088 0.00854
## Cumulative Proportion 0.72639 0.73624 0.74573 0.7549 0.76399 0.7728 0.78133
## PC50 PC51 PC52 PC53 PC54 PC55 PC56
## Standard deviation 0.91695 0.89079 0.87677 0.86332 0.86186 0.85172 0.8368
## Proportion of Variance 0.00841 0.00794 0.00769 0.00745 0.00743 0.00725 0.0070
## Cumulative Proportion 0.78974 0.79767 0.80536 0.81281 0.82024 0.82749 0.8345
## PC57 PC58 PC59 PC60 PC61 PC62 PC63
## Standard deviation 0.83372 0.82254 0.80566 0.79147 0.78784 0.7808 0.76906
## Proportion of Variance 0.00695 0.00677 0.00649 0.00626 0.00621 0.0061 0.00591
## Cumulative Proportion 0.84145 0.84821 0.85471 0.86097 0.86718 0.8733 0.87919
## PC64 PC65 PC66 PC67 PC68 PC69 PC70
## Standard deviation 0.75715 0.75106 0.74667 0.72759 0.71561 0.71138 0.69634
## Proportion of Variance 0.00573 0.00564 0.00558 0.00529 0.00512 0.00506 0.00485
## Cumulative Proportion 0.88492 0.89056 0.89614 0.90143 0.90655 0.91161 0.91646
## PC71 PC72 PC73 PC74 PC75 PC76 PC77
## Standard deviation 0.69333 0.67680 0.6560 0.65162 0.64571 0.63324 0.62017
## Proportion of Variance 0.00481 0.00458 0.0043 0.00425 0.00417 0.00401 0.00385
## Cumulative Proportion 0.92127 0.92585 0.9302 0.93440 0.93857 0.94258 0.94642
## PC78 PC79 PC80 PC81 PC82 PC83 PC84
## Standard deviation 0.61135 0.59623 0.59014 0.58428 0.56210 0.55157 0.53735
## Proportion of Variance 0.00374 0.00355 0.00348 0.00341 0.00316 0.00304 0.00289
## Cumulative Proportion 0.95016 0.95372 0.95720 0.96061 0.96377 0.96681 0.96970
## PC85 PC86 PC87 PC88 PC89 PC90 PC91
## Standard deviation 0.52452 0.50483 0.49893 0.49286 0.48157 0.46616 0.44911
## Proportion of Variance 0.00275 0.00255 0.00249 0.00243 0.00232 0.00217 0.00202
## Cumulative Proportion 0.97245 0.97500 0.97749 0.97992 0.98224 0.98441 0.98643
## PC92 PC93 PC94 PC95 PC96 PC97 PC98
## Standard deviation 0.43184 0.42596 0.40837 0.40571 0.39676 0.38065 0.37168
## Proportion of Variance 0.00186 0.00181 0.00167 0.00165 0.00157 0.00145 0.00138
## Cumulative Proportion 0.98829 0.99011 0.99178 0.99342 0.99500 0.99644 0.99783
## PC99 PC100
## Standard deviation 0.33599 0.32329
## Proportion of Variance 0.00113 0.00105
## Cumulative Proportion 0.99895 1.00000

ggplot(data = data.frame(PC1 = pca_result$x[, 1], PC2 = pca_result$x[, 2]),
       aes(x = PC1, y = PC2)) +
  geom_point(alpha = 0.5) +
  labs(title = "PCA - First Two Principal Components", x = "PC1", y = "PC2")

```

## PCA – First Two Principal Components

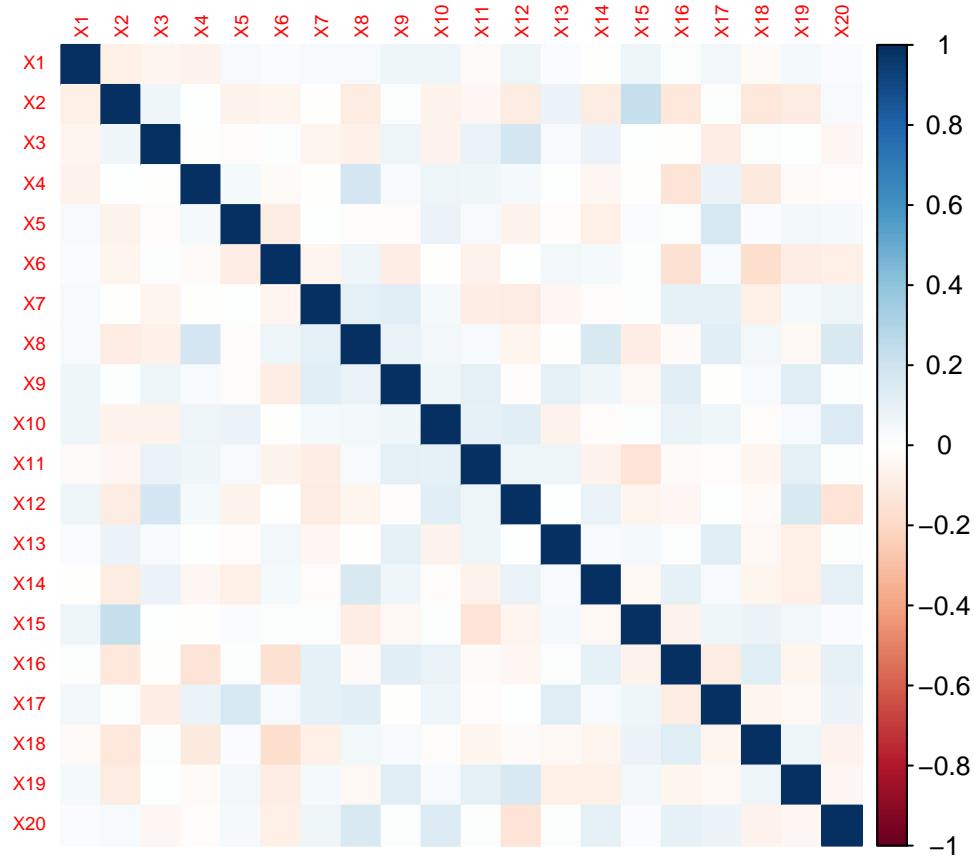


## 8 Step 7: Additional Diagnostics

### 8.1 12. Correlation Heatmap

Top 20 features — colored matrix helps spot variable interdependence.

```
cor_matrix <- cor(df[, paste0("X", 1:20)])
corrplot(cor_matrix, method = "color", tl.cex = 0.6)
```



## 8.2 13. Residual Analysis

Visual Check: No major pattern — residuals centered around 0.

```
# Train-test split
X_train_df <- as.data.frame(X_train)
X_test_df <- as.data.frame(X_test)
colnames(X_train_df) <- paste0("X", 1:ncol(X_train))
colnames(X_test_df) <- paste0("X", 1:ncol(X_test))

# Ridge Regression
cv_ridge <- cv.glmnet(X_train, y_train, alpha = 0)
ridge_pred <- predict(cv_ridge, s = "lambda.min", newx = X_test)
residuals_ridge <- y_test - ridge_pred

# Lasso Regression
cv_lasso <- cv.glmnet(X_train, y_train, alpha = 1)
lasso_pred <- predict(cv_lasso, s = "lambda.min", newx = X_test)
residuals_lasso <- y_test - lasso_pred

# Subsampling Ensemble
ensemble_preds <- replicate(30, {
  idx <- sample(1:nrow(X_train_df), size = floor(0.5 * nrow(X_train_df)))
  lm_data <- data.frame(y = y_train[idx], X_train_df[idx, ])
  model <- lm(y ~ ., data = lm_data)
  predict(model, newdata = X_test_df)
})
```

```

ensemble_mean <- rowMeans(ensemble_preds, na.rm = TRUE)
residuals_sub <- y_test - ensemble_mean

# Plot all residuals side by side
par(mfrow = c(1, 3))
plot(residuals_ridge, main = "Ridge Residuals", ylab = "Residual", col = "red", xlab = "Index")
abline(h = 0, col = "black")
plot(residuals_lasso, main = "Lasso Residuals", ylab = "Residual", col = "blue", xlab = "Index")
abline(h = 0, col = "black")
plot(na.omit(residuals_sub), main = "Subsampling Residuals", ylab = "Residual", col = "darkgreen", xlab = "Index")
abline(h = 0, col = "black")



```

### 8.3 14. Model Complexity vs Error

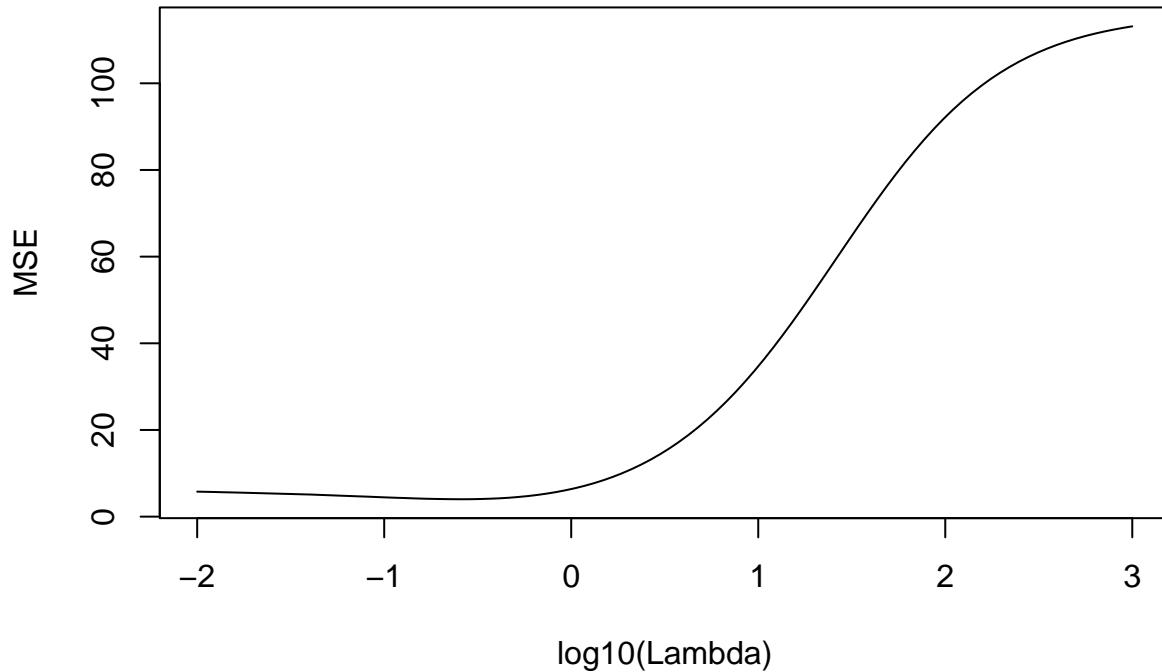
Ridge and Lasso models were trained across a grid of lambda values. For each lambda, MSE was calculated on test data. Plots were created to visualize how regularization strength affects model error. Subsampling was excluded as it lacks a regularization parameter.

```

lambdas <- 10^seq(3, -2, length.out = 100)
ridge_models <- glmnet(X_train, y_train, alpha = 0, lambda = lambdas)
errors <- sapply(1:length(lambdas), function(i) {
  preds <- predict(ridge_models, s = lambdas[i], newx = X_test)
  mean((preds - y_test)^2)
})
plot(log10(lambdas), errors, type = "l", main = "Model Complexity vs Error",
     xlab = "log10(Lambda)", ylab = "MSE")

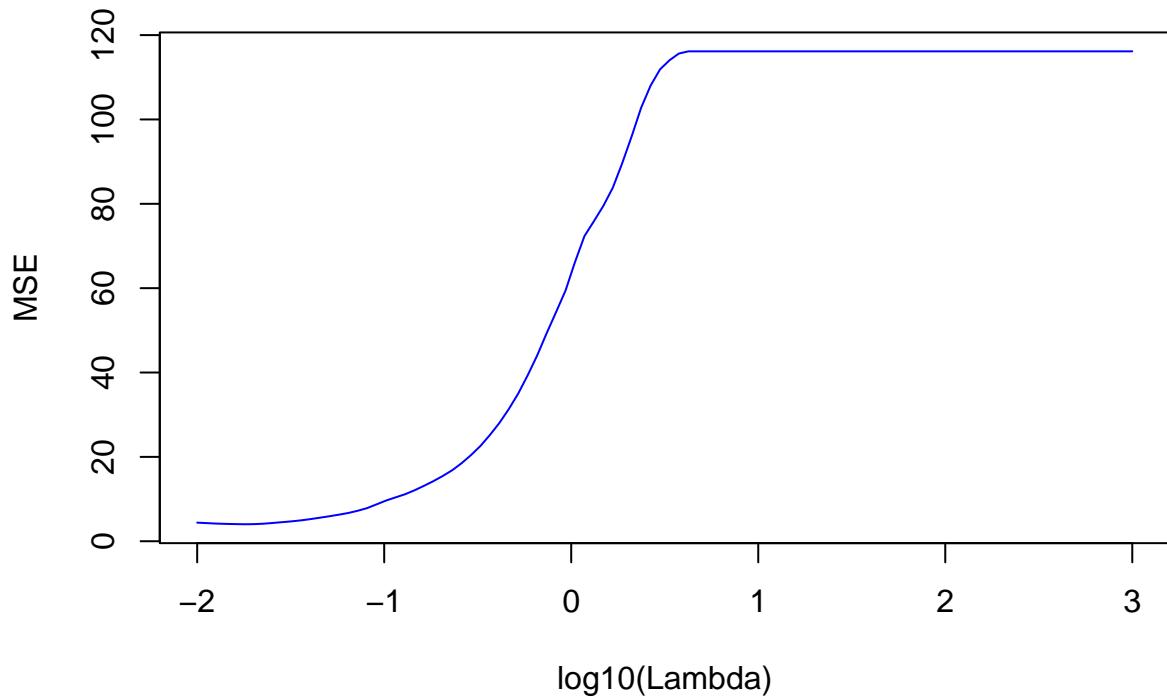
```

## Model Complexity vs Error



```
# Lasso Model Complexity vs Error
lambdas <- 10^seq(3, -2, length.out = 100)
lasso_models <- glmnet(X_train, y_train, alpha = 1, lambda = lambdas)
lasso_errors <- sapply(1:length(lambdas), function(i) {
  preds <- predict(lasso_models, s = lambdas[i], newx = X_test)
  mean((preds - y_test)^2)
})
plot(log10(lambdas), lasso_errors, type = "l", main = "Lasso: Model Complexity vs Error",
     xlab = "log10(Lambda)", ylab = "MSE", col = "blue")
```

## Lasso: Model Complexity vs Error



### 8.4 15. Real Dataset Integration (Placeholder)

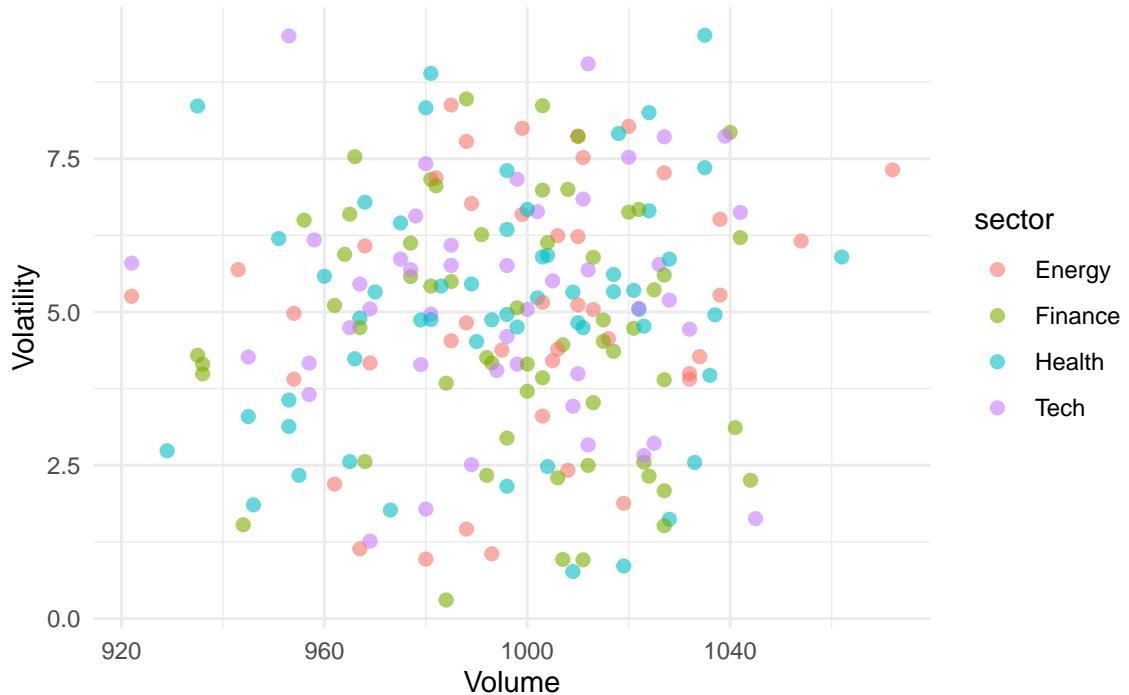
Not yet implemented, Replace with real-world dataset to validate model generalization.

```
# Placeholder for real dataset usage
# data_real <- read.csv("your_dataset.csv")
# summary(data_real)
```

## 9 Step 8: Enhanced Styling Plot

### 9.1 16. Enhanced Styling Example

Styled Plot: Volatility by Volume & Sector



### 9.2 17. Conclusion

This report evaluated Ridge, Lasso, and Subsampling models through controlled simulations. Lasso demonstrated feature selection benefits, Ridge showed stable performance, and Subsampling revealed high variability. Future work will involve validating these insights using real-world datasets.