Predicting Glass Types Using Chemical Properties
By: Shane Gao, Tristan Kao, Ethan Dang, Amir Yaacoobi

I. Introduction

In this exploration, we aim to identify the type of glass based on its chemical properties using multivariate data analysis techniques. Our focus is on developing a reliable predictive model that can assist in forensic investigations by accurately classifying glass fragments found at crime scenes. This analysis is crucial in forensic science, as determining the source of glass fragments can provide pivotal leads in criminal investigations. Glass fragments are often found at crime scenes, and determining their origin can provide significant leads. By leveraging chemical composition data, forensic scientists can compare glass samples found at crime scenes with those from known sources, thereby establishing connections and potentially identifying suspects.

We utilize the Glass Identification Database, derived from the R package 'mlbench', which offers a collection of chemical properties for different glass samples. This dataset includes measurements of refractive index (RI), the weight percentage of several oxides (Na, Mg, Al, Si, K, Ca, Ba, Fe), and the type of glass classified into seven categories.

Our objective is to develop a predictive model that can identify the type of glass based on its chemical composition. To achieve this goal, we integrate the multivariate data analysis techniques we learned in class, like Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Random Forest.

II. Data Description

The Glass Identification Database used in this project comes from the mlbench R package. This dataset contains chemical composition data for various glass types. Our analysis involves examining both predictor and target variables to develop a model capable of classifying glass types based on their chemical properties. Below, we detail the predictor variables, which are chemical components measured as weight percentages, and the target variable, which categorizes the glass type.

II.I Predictor Variables

1. RI (Refractive Index): Measures the bending of light as it passes through the glass. This index is crucial as it varies with the composition and thickness of the glass.
2. Na (Sodium): Indicates the presence of sodium oxide. Sodium is common in glass and affects its melting temperature and durability.
3. Mg (Magnesium): The amount of magnesium oxide, which influences the glass's optical and mechanical properties.
4. Al (Aluminum): Aluminum oxide content affects the viscosity and stiffness of the glass.
5. Si (Silicon): Silicon dioxide is the primary constituent in most glasses, providing structural integrity.
6. K (Potassium): Potassium oxide is used to reduce the melting temperature and to increase the strength of the glass
7. Ca (Calcium): Calcium oxide is added to improve the chemical durability and weather resistance.

8. Ba (Barium): Barium oxide is used to increase the refractive index and to improve stability against water and acids.
9. Fe (Iron): Iron oxide can influence the color and thermal properties of the glass.

II.II Target Variable

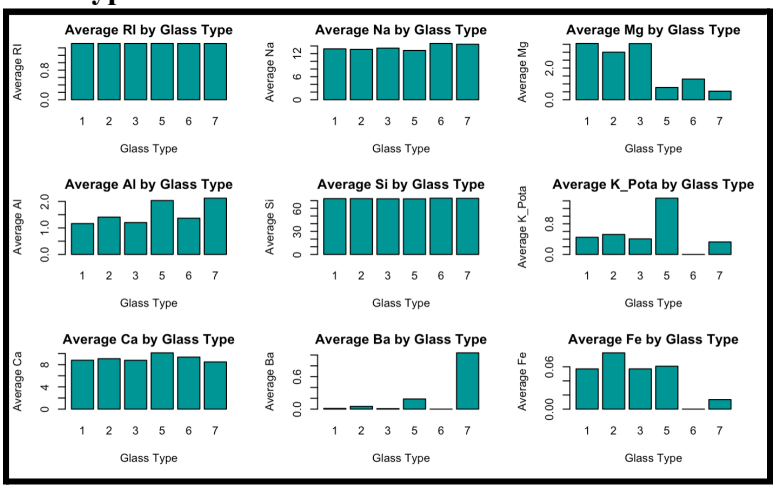
Type: The type of glass, which is an integral part of forensic classification. It is categorized into seven distinct classes:

- Type 1: Building windows (float processed)
- Type 2: Building windows (non-float processed)
- Type 3: Vehicle windows (float processed)
- Type 5: Containers
- Type 6: Tableware
- Type 7: Headlamps

III. Exploratory Data Analysis (EDA)

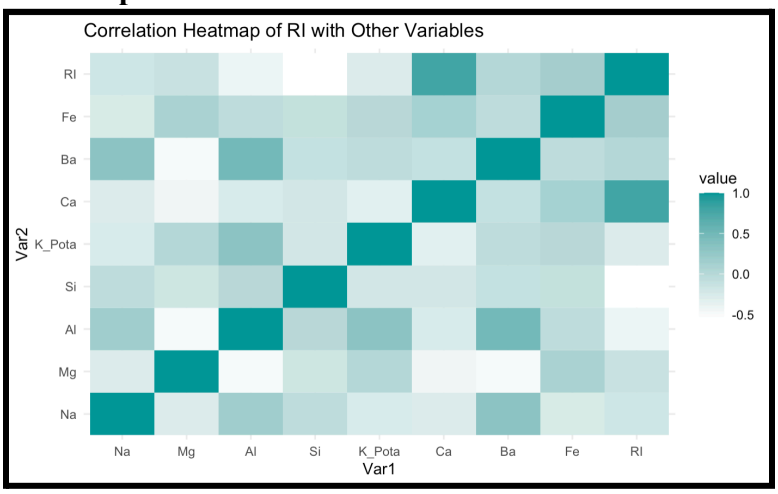
In this section, we will delve into a comprehensive EDA to uncover underlying patterns and insights from the chemical composition of glass types. Our EDA aims to highlight significant trends and anomalies in the dataset that could inform the development of a predictive model. This analysis is crucial for understanding the relationship between the various chemical elements within the glass and their impact on glass classification. Our findings from this analysis will lay the groundwork for more advanced statistical modeling and machine learning applications in subsequent sections of this study.

III.I Average by Glass Type



In the diagram above, we analyze the average concentrations of various chemical elements across different glass types. This analysis helps us understand the contribution of each chemical to the composition of the glass and identifies potential key markers for differentiating among glass types. Notably, chemical elements such as Na, Ca, and Si display relatively consistent levels across all types, indicating their common usage in glass production and suggesting that they might not serve as effective discriminators in classifying glass types. In contrast, elements like Mg and K exhibit considerable variations. Specifically, both elements are significantly lower in types 6 and 7. Furthermore, Ba shows a notable spike in type 7, and Fe is particularly absent in type 6, presenting unique patterns that could be pivotal for our predictive modeling.

III.II Correlation Heatmap



The correlation heatmap gives a clear visual representation of the relationships between the variables in the dataset. Each cell in the heatmap shows the correlation coefficient between pairs of variables. The color's intensity shows the correlation's strength, with darker colors representing stronger correlations. We can see that there is a strong negative correlation between Na and Mg, suggesting that these elements are inversely related to the glass composition. Similarly, the weak correlations between most variables and the RI indicate that RI alone may not be a strong predictor of glass type, meaning that there is the need to consider multiple variables in the predictive modeling process. This emphasizes the importance of a multifaceted approach in our predictive modeling, where numerous variables must be considered to improve accuracy and effectiveness in classifying glass types for forensic analysis.

IV. Multivariate Analysis

With our analysis, our main goal is to create the best predictive model and see the patterns that appear in each model. We will first determine which principal components we should use. Then, we will create an LDA, a random forest model, and a random forest model based on the newly transformed PCA dataset.

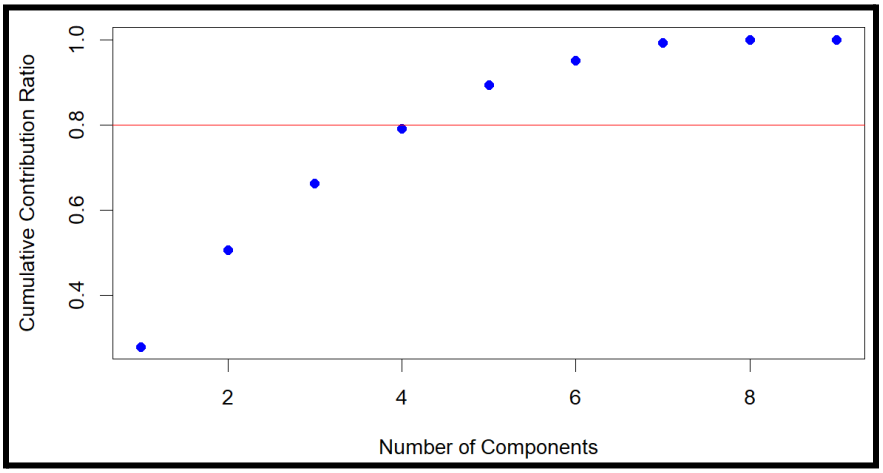
The purpose of PCA is to remove the less significant details to only focus on the important features of the whole dataset. This is done by transforming the variables into a new principal component set. The principal components are based on the eigenvectors of the covariance matrix of the data. We then determine the number of PCs needed to cover most data. Thus, we are looking for the right dimension reduction. We determine the number of PCs by looking at the models such as the Scree and Cumulative Contribution Ratio plots.

IV.I LDA - Linear Discriminants Table

Coefficients of Linear Discriminants					
Variable	**LD1**	**LD2**	**LD3**	**LD4**	**LD5**
RI	-382.2985258	124.623908	-344.957602	230.519284	-736.181805
Na	-1.8403020	4.246797	-1.286137	6.622420	2.453514
Mg	-0.2127494	4.065250	-2.257212	6.380583	2.903273
Al	-2.8682125	2.927329	-2.756048	6.431031	1.065987
Si	-2.0387398	4.209085	-2.634968	6.923383	1.384865
K	-1.1714389	3.220615	-2.372597	7.581718	3.067277
Ca	-0.3722468	3.365933	-1.470566	6.396811	3.877336
Ba	-1.4573146	4.450135	-3.237427	5.422651	4.337572
Fe	0.6882057	1.117227	-1.363287	1.921015	-4.444905

The coefficients of the linear discriminants from our Linear Discriminant Analysis (LDA) reveal the importance of various elements in distinguishing between glass types. For both the first and third linear discriminant (LD1 and LD3), the RI has the most significant negative coefficient of -382.2985 and -344.9576, respectively. Na, Mg, Al, Si, K, Ca, and Ba also have negative coefficients, while Fe has a positive coefficient in LD1 but a negative coefficient in LD3. LD2 and LD4 show different patterns, with most elements such as Na, Mg, Si, and Ba having positive coefficients, indicating that higher values of these elements contribute significantly to class separation along these dimensions. RI also has positive coefficients in LD2 (124.6239) and LD4 (230.5193), highlighting its consistent influence in distinguishing between the classes. Notably, LD5 has a very high negative coefficient for RI (-736.18) and a high positive coefficient for Ba (4.34). Overall, the analysis indicates that RI and Ba are consistently influential across multiple discriminants, with other elements like Na, Mg, Si, and Fe also playing significant roles in classifying different glass types. These coefficients highlight the complex interplay of these variables in distinguishing between glass types.

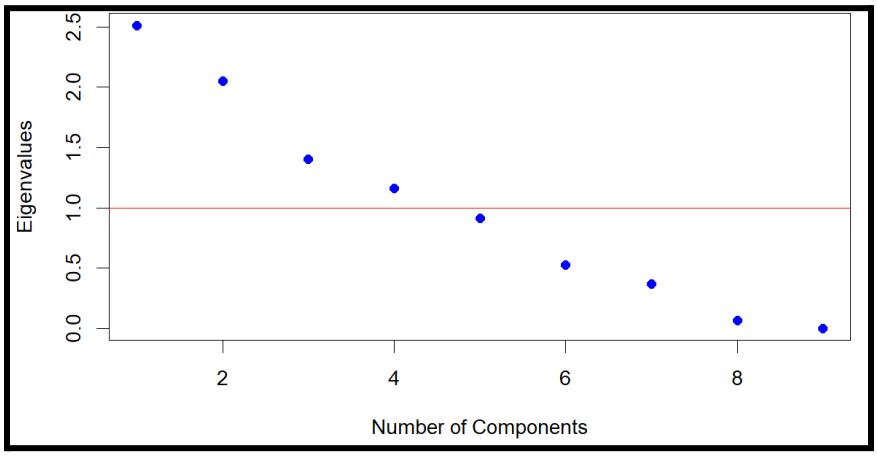
IV.II Cumulative Contribution Ratio Plot



The Cumulative Contribution Ratio plot is designed to determine how many principal components are needed to effectively capture the variance in our dataset. The plot suggests that

using all nine components accounts for nearly 100% of the variance. However, an important observation from the graph is that the first four principal components already capture approximately 80% of the variance. This indicates that additional components beyond the fourth contribute minimally to increasing our understanding of the data, as evidenced by the curve leveling off after the fourth component. Therefore, focusing on the first four components may be sufficient for our analysis, allowing us to reduce dimensionality without losing significant information.

IV.III Scree Plot



The Scree Plot displayed above illustrates the eigenvalues for each principal component derived from the covariance matrix of our dataset. These eigenvalues are indicative of the amount of variance each component captures. As we add more components, there is a noticeable decline in the magnitude of the eigenvalues, suggesting that most of the significant variance is captured early on. Specifically, the first four components are particularly valuable as their eigenvalues are above the widely-used Kaiser criterion threshold of one, indicating substantial variance coverage. After the fourth component, the drop in eigenvalues becomes more pronounced, signaling diminishing returns on additional components. Therefore, we will focus on the first four components for our analysis, as they provide the most meaningful information without unnecessary complexity.

IV.IV Linear Combinations of the Elements

Variable	PC1	PC2	PC3	PC4
RI	-0.5451766	0.2856832	0.0869108	0.1473810
Na	0.2581256	0.2703501	-0.3849196	0.4912420
Mg	-0.1108810	-0.5935583	0.0084180	0.3787858
Al	0.4287086	0.2952115	0.3292371	-0.1375059
Si	0.2288364	-0.1550989	-0.4587088	-0.6525377
K	0.2193440	-0.1539701	0.6625741	-0.0385354
Ca	-0.4923061	0.3453798	-0.0009847	-0.2764432
Ba	0.2503751	0.4847022	0.0740547	0.1331755
Fe	-0.1858415	-0.0620388	0.2844506	-0.2304920

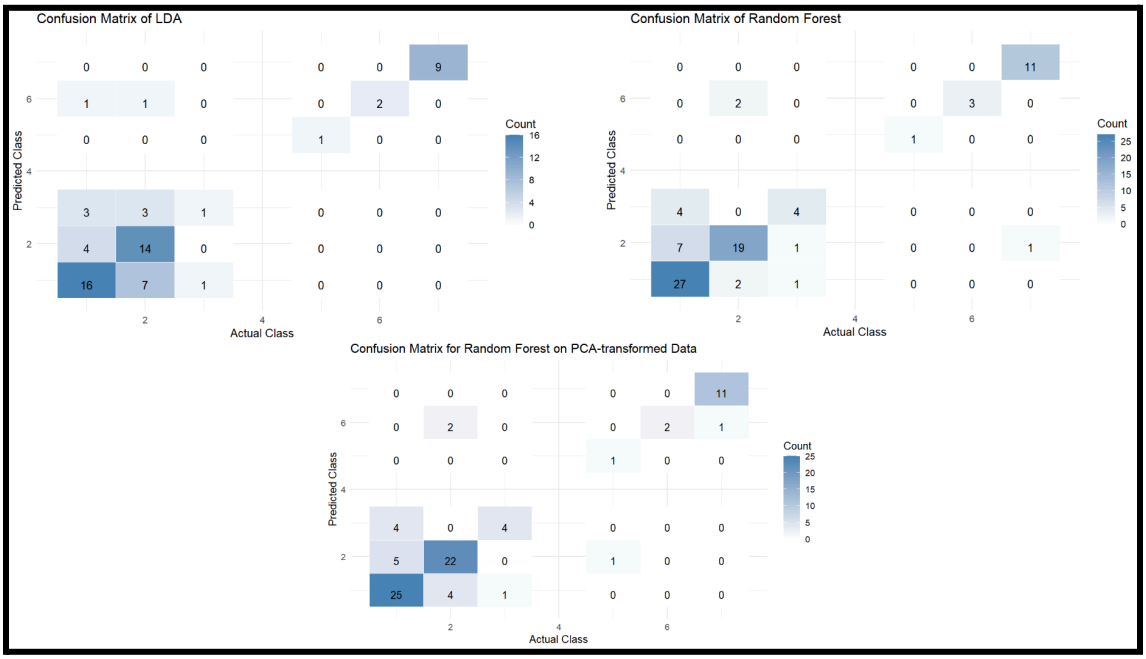
The table presented above details the linear combinations of chemical elements as they contribute to the first four principal components (PCs) identified in our analysis. Each entry in the table, referred to as a coefficient, quantifies the influence of a specific element on a particular principal component. Positive coefficients indicate an increase in the component value with an increase in the element's proportion, while negative coefficients indicate a decrease. For example, the negative coefficient for RI in PC1 (-0.5451766) indicates that higher RI values tend to decrease this component's value. Conversely, Na shows a substantial positive influence in PC4 (0.4912420), suggesting its significant role in this component. This breakdown helps us understand the relative importance of each chemical element within the principal components, providing insights into their interactions and impact on glass type differentiation.

IV.V Prediction Models with Different Splits

Model	Split	Accuracy
LDA	0.6	0.6626506
LDA	0.7	0.6825397
LDA	0.8	0.6521739
Random Forest	0.6	0.7831325
Random Forest	0.7	0.8095238
Random Forest	0.8	0.8478261
Random Forest PCA	0.6	0.7831325
Random Forest PCA	0.7	0.7142857
Random Forest PCA	0.8	0.6956522

The table displayed above summarizes the performance of various predictive models at different training-test splits. These splits represent the proportion of data allocated to training versus testing, with varying impacts on model accuracy. For the regular LDA splits, the model with the 0.8 splits had the lowest accuracy. However, for the regular Random Forest model the 0.6 split had the lowest accuracy. Interestingly, the Random Forest model enhanced with PCA transformation shows a drop in accuracy at the 0.8 split, possibly indicating overfitting as the training set size increases while working with reduced dimensionality due to PCA. Despite these variations, the highest accuracy was observed at different splits within each model category, suggesting that split ratio alone is not a definitive predictor of performance. Notably, the regular Random Forest model at an 80/20 split achieved the highest accuracy, emphasizing its effectiveness in handling the dataset with a larger training proportion.

IV.VI Confusion Matrix



The confusion matrix depicted above compares the predicted classifications from our models against the actual types of glass, with each model using a data split of 0.6. These matrices are instrumental in revealing which types of glass are more commonly misclassified and which ones are distinct enough to be consistently identified correctly. A notable observation across all models is the frequent misclassification between types 1 and 2, suggesting these glass types share similar characteristics that confuse the models. Conversely, type 7 glass consistently shows zero misclassifications across all models, indicating its unique composition makes it distinctly different from other types. This pattern highlights the strengths and weaknesses of our predictive models in identifying specific glass types, which is crucial for refining our approaches and improving accuracy in practical forensic applications.

V. Conclusion

In conclusion, our investigation into dimension reduction and machine learning revealed some unexpected results. While PCA was intended to enhance the performance of our machine learning models by simplifying the data structure, we found that reducing the dimension did not necessarily improve the accuracy since the regular random forest model generally performed better than the random forest model with the PCA transformation. This suggests that there is a limit to how much the dataset's dimensionality can be reduced before it begins to detrimentally affect the prediction accuracy. These findings underscore the complexity of balancing data simplification with the retention of essential information for effective model performance.

Contributions

Shane Gao - LDA, Random Forest, analysis
Tristan Kao - PCA, Random Forest with PCA, analysis
Ethan Dang - Introduction writing, data, LATEX tables, analysis
Amir Yaacoobi - EDA and EDA analysis, conclusion, analysis

Appendix

```
# Heatmap
cor_matrix <- cor(Glass[, -10])

cor_matrix_melt <- melt(cor_matrix)

ggplot(cor_matrix_melt, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1, 1), space = "Lab",
                       name = "Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1)) +
  coord_fixed() +
  labs(title = "Correlation Heatmap of Glass Elements",
       x = "Element",
       y = "Element")

# Split the data into training and testing sets
set.seed(999)
ind <- sample(2, nrow(Glass), replace = TRUE, prob = c(0.6, 0.3))
training <- Glass[ind == 1, ]
testing <- Glass[ind == 2, ]

# LDA
lda_model <- lda(Type ~ ., data = training)
print(lda_model)

# Perform LDA
lda_model <- lda(Type ~ ., data = training)
print(lda_model)

par(mar = c(5, 4, 4, 2) + 0.1)
par(mfrow = c(1, 1))

# Open a new plotting device with specified dimensions
dev.new(width = 12, height = 8)

# Predictions on training
p1 <- predict(lda_model, training)$class
tab <- table(Predicted = p1, Actual = training$Type)
print(tab)

# Predictions on testing
p2 <- predict(lda_model, testing)$class
tab1 <- table(Predicted = p2, Actual = testing$Type)
print(tab1)

# Accuracy
accuracy <- sum(diag(tab1)) / sum(tab1)
print(accuracy)

# PCA
glass_scaled <- scale(Glass[, -10])  # Exclude the target variable
```

```
pca <- prcomp(glass_scaled, center = TRUE, scale. = TRUE)
summary(pca)

# Plot PCA
pca_df <- data.frame(pca$x, Type = Glass$Type)
ggplot(pca_df, aes(x = PC1, y = PC2, color = as.factor(Type))) +
  geom_point() +
  labs(title = "PCA of Glass Identification Data",
       x = "Principal Component 1",
       y = "Principal Component 2")

# Eigenvalues and explained variance
ev <- eigen(cov(glass_scaled))$values
eve <- eigen(cov(glass_scaled))$vectors

# Plot cumulative contribution ratio
Cr <- cumsum(ev) / sum(ev)
plot(Cr, xlab = "Number of Components", ylab = "Cumulative Contribution Ratio", pch =
19, col = "blue")
abline(h = 0.8, col = "red")

# Plot eigenvalues
plot(ev, xlab = "Number of Components", ylab = "Eigenvalues", pch = 19, col = "blue")
abline(h = mean(ev), col = "red")

# Scree plot
plot(ev, xlab = "Number of Components", ylab = "Eigenvalues", pch = 19, col = "blue")

# Plot differences between successive eigenvalues
plot(ev[1:8] - ev[2:9], xlab = "Number", ylab = "Difference in Eigenvalues", pch = 19,
col = "blue")

# Hypothesis testing for number of components to retain
u <- rep(0, 8)
for (k in 1:8) {
  intermediate <- 1 / (9 - k + 1) * sum(ev[k:9])
  u[k] <- (nrow(glass_scaled) - (2 * 9 + 11) / 6) * ((9 - k + 1) * log(intermediate) -
sum(log(ev[k:9])))
}

conf_matrix <- confusionMatrix(tab1)
conf_matrix_long <- melt(conf_matrix$table)
colnames(conf_matrix_long) <- c("Actual", "Predicted", "Count")

# Confusion matrix
ggplot(data = conf_matrix_long, aes(x = Actual, y = Predicted, fill = Count)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Count), vjust = 1) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(title = "Confusion Matrix",
       x = "Actual Class",
       y = "Predicted Class") +
  theme_minimal()

# Prior probabilities of groups
prior_prob <- data.frame(lda_model$prior)
colnames(prior_prob) <- "Prior Probability"
prior_prob$Group <- rownames(prior_prob)
prior_prob <- prior_prob[, c("Group", "Prior Probability")]
kable(prior_prob, caption = "Prior Probabilities of Groups")

# Group means
group_means <- data.frame(lda_model$means)
```

```
group_means$Variable <- rownames(group_means)
group_means <- group_means[, c(ncol(group_means), 1:(ncol(group_means)-1))]
kable(group_means, caption = "Group Means")

# Coefficients of linear discriminants
coefficients_ld <- data.frame(lda_model$scaling)
coefficients_ld$Variable <- rownames(coefficients_ld)
coefficients_ld <- coefficients_ld[, c(ncol(coefficients_ld),
1:(ncol(coefficients_ld)-1))]
kable(coefficients_ld, caption = "Coefficients of Linear Discriminants")

# Proportion of trace
proportion_trace <- data.frame(Proportion = lda_model$svd^2 / sum(lda_model$svd^2))
proportion_trace$LD <- paste0("LD", 1:nrow(proportion_trace))
kable(proportion_trace, caption = "Proportion of Trace")

loadings <- pca$rotation[, 1:4]

# PCA data
loadings_df <- as.data.frame(loadings)
colnames(loadings_df) <- c("PC1", "PC2", "PC3", "PC4")

# Add the variable names as a column
loadings_df$Variable <- rownames(loadings_df)

# Reorder the columns
loadings_df <- loadings_df[, c("Variable", "PC1", "PC2", "PC3", "PC4")]

# Table
kable(loadings_df, format = "html", table.attr = "class='table table-striped'") %>%
  kable_styling(full_width = FALSE, position = "center", bootstrap_options =
c("striped", "hover", "condensed", "responsive")) %>%
  column_spec(1, bold = TRUE) %>%
  row_spec(0, bold = TRUE)

# Split the data into training and testing sets
set.seed(999)
ind <- sample(2, nrow(Glass), replace = TRUE, prob = c(0.6, 0.4))
training <- Glass[ind == 1, ]
testing <- Glass[ind == 2, ]

# Random Forest
rf_model <- randomForest(Type ~ ., data = training, ntree = 10000)

# Predictions on testing
rf_pred <- predict(rf_model, testing)

# RF result
rf_table <- table(Predicted = rf_pred, Actual = testing$Type)
print(rf_table)

rf_accuracy <- sum(diag(rf_table)) / sum(rf_table)
print(paste("Random Forest Accuracy:", rf_accuracy))

# Confusion matrix metrics
conf_matrix <- as.data.frame.matrix(rf_table)
print(conf_matrix)

# Visualization of the confusion matrix
# Convert confusion matrix to a long format for ggplot
conf_matrix_long <- melt(rf_table)
colnames(conf_matrix_long) <- c("Actual", "Predicted", "Count")
```

```
# Plot confusion matrix
ggplot(data = conf_matrix_long, aes(x = Actual, y = Predicted, fill = Count)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Count), vjust = 1) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(title = "Confusion Matrix of Random Forest",
       x = "Actual Class",
       y = "Predicted Class") +
  theme_minimal()

data("Glass", package = "mlbench")

# PCA
glass_scaled <- scale(Glass[, -10])  # Exclude the target variable
pca <- prcomp(glass_scaled, center = TRUE, scale. = TRUE)

# Create a dataframe with the first four principal components and the target variable
pca_scores <- data.frame(pca$x[, 1:4], Type = Glass$Type)
colnames(pca_scores) <- c("PC1", "PC2", "PC3", "PC4", "Type")

# Split the PCA-transformed data into training and testing sets
set.seed(999)
ind <- sample(2, nrow(pca_scores), replace = TRUE, prob = c(0.6, 0.4))
training_pca <- pca_scores[ind == 1, ]
testing_pca <- pca_scores[ind == 2, ]

# Random Forest classification on PCA-transformed data
rf_model_pca <- randomForest(Type ~ PC1 + PC2 + PC3 + PC4, data = training_pca, ntree
= 10000)

# Predictions on testing
rf_pred_pca <- predict(rf_model_pca, testing_pca)

# Evaluate the performance
rf_table_pca <- table(Predicted = rf_pred_pca, Actual = testing_pca$Type)
print(rf_table_pca)

rf_accuracy_pca <- sum(diag(rf_table_pca)) / sum(rf_table_pca)
print(paste("Random Forest Accuracy on PCA-transformed data:", rf_accuracy_pca))

# Confusion matrix metrics
conf_matrix_pca <- as.data.frame.matrix(rf_table_pca)
print(conf_matrix_pca)

# Optionally, create a visualization of the confusion matrix
# Convert confusion matrix to a long format for ggplot
conf_matrix_pca_long <- melt(rf_table_pca)
colnames(conf_matrix_pca_long) <- c("Actual", "Predicted", "Count")

# Confusion matrix
ggplot(data = conf_matrix_pca_long, aes(x = Actual, y = Predicted, fill = Count)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Count), vjust = 1) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(title = "Confusion Matrix for Random Forest on PCA-transformed Data",
       x = "Actual Class",
       y = "Predicted Class") +
  theme_minimal()

# Calculate averages for each glass type
averages <- Glass %>%
  group_by(Type) %>%
  summarise(across(everything(), mean, na.rm = TRUE))
```

```
# Print the averages
print(averages)

# Individual plots
create_plot <- function(data, col_name) {
  ggplot(data, aes_string(x = "factor(Type)", y = col_name)) +
    geom_bar(stat = "identity", fill = "steelblue") +
    labs(title = paste("Average", col_name, "by Glass Type"),
         x = "Glass Type",
         y = paste("Average", col_name)) +
    theme_minimal()
}

# Individual plots for each attribute
plots <- list()
attributes <- names(averages)[-1] # Exclude 'Type' column
for (col in attributes) {
  p <- create_plot(averages, col)
  plots <- append(plots, list(p))
}

# Arrange plots in a 3x3 grid
do.call(grid.arrange, c(plots, ncol = 3))
```