PROYECTO FINAL MACHINE LEARNING

Ariana Silva y Valeria M. Cuevas

2024-11-27

Part 1: Dataset Selection and Initial Analysis

1.1 Dataset Documentation

This RNA-seq dataset is derived from a project by the Cancer Genetics & Bioinformatics Laboratory at the International Laboratory for Human Genome Research. The data from this project have not been published yet, so we will not share very specific details.

The data from this study come from an experiment involving four groups of rats, each consisting of 20 rats, for a total of 80 rats, subjected to different feeding protocols. Two of the groups were administered a hepatotoxic to induce the development of liver carcinoma, while the other two groups were kept as healthy controls. The main objective was to assess changes in gene expression related to the diets and the administration of the hepatotoxic.

Final distribution of the groups:

- Group 1: healthy (normal diet)
- Group 2: hepatotoxic (normal diet)
- Group 3: healthy (treatment diet)
- Group 4: hepatotoxic (treatment diet)

As initial data, we have the raw counts for each of the 80 samples, covering a total of 15,309 genes (features). In this project, our goal is to classify the experimental groups based on the GSVA scores for each pathway across the samples, capturing the variation in pathway activity. To achieve this, we will implement multinomial logistic regression models, Decision Trees, and Random Forests, aiming to identify the model with the best performance.

1.2 Exploratory Data Analysis

```
#---- 01 Load data ----
# We load the raw counts stored within the dds object, which also contains the experimental design and
other relevant data for the analysis.
dds <- readRDS("/home/user/Documents/Proyectod_ML/dds_object.rds")

#---- 02 normalization ----
dds <- estimateSizeFactors(dds) # Normalization of sample counts
vds<- vst(dds,blind = FALSE) # A variance stabilizing transformation (VST) is applied

# head(assay(vds), 3)
vst_counts <- assay(vds)
colnames(vst_counts) <- substr(colnames(vst_counts), 1, 2)

#---- 03 Exploratory Data Analysis ----
# create_report(vst_counts)
```

Data Quality and Missing Values

We generated a report using the DataExplorer package to gain a general overview of the data. Our dataset contains 80 samples and 15,309 genes, consisting solely of continuous variables. Furthermore, there are no missing values in any column or row of the dataset.

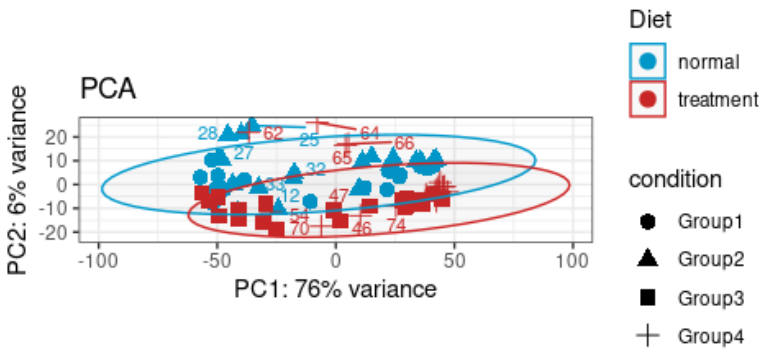
Histograms

The report shows the distribution of gene expression for each sample. Most samples exhibit similar distributions, slightly skewed to the left, which is expected since the data has already been normalized. However, samples 61 and 62, belonging to group 4, display noticeably different distributions. According to the MultiQC analysis, these samples are not of poor quality but have a significantly lower number of reads compared to the other samples. This discrepancy could explain the globally lower gene expression observed in the histograms for these samples.

Correlation Analysis

Our dataset has been normalized using VST (Variance Stabilizing Transformation). This method stabilizes the variance of gene counts, reducing variability among samples. In our case, the correlation plot shows that the samples are highly positively correlated, likely as a result of the VST normalization process.

```r
#---- 04 PCA ----
plotPCA(vds, intgroup= c("condition"))

pca_data <- plotPCA(vds, intgroup = "condition", returnData = TRUE)

#Plot
percentVar <- round(100 * attr(pca_data, "percentVar"))
ggplot(pca_data, aes(x = PC1, y = PC2, col = Diet)) +
  geom_point(aes(shape = condition), size = 3) +
  geom_text_repel(aes(label = name), size = 3, max.overlaps = 10) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +
  coord_fixed() +
  theme_bw() +
  stat_ellipse(geom = "polygon", level = 0.95, alpha = 0.03) +
  ggtitle("PCA") +
  scale_colour_manual(values = c("#009ACD", "#CD2626"))
```



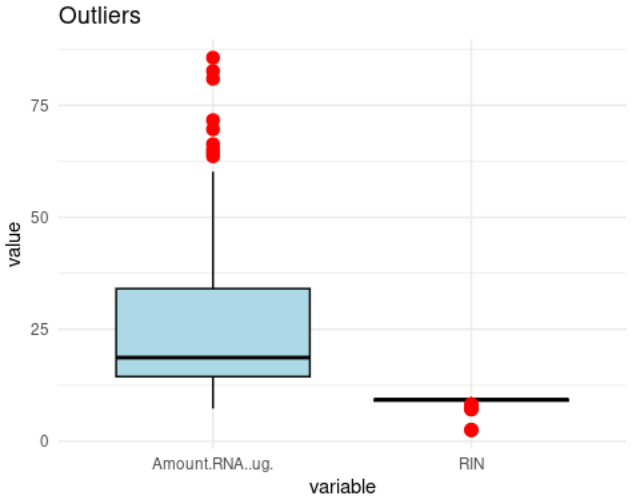
PCA of gene expression

We can see that the first two principal components explain 76% and 8% of the variance, respectively, totaling 84%. In the PCA, two well-defined groups emerge based on the type of

diet: the first group corresponds to the normal diet (Blue), and the second group corresponds to the treatment diet (Red). This suggests that diet plays an important role in explaining the variation in our data and is likely influencing gene expression.

Both groups show dispersion, indicating that each group has internal variability and is not completely homogeneous. PC1 captures most of the variance, likely reflecting differences due to the diet type, while PC2 seems to capture more subtle differences between the diets. Additionally, we observe outliers in the treatment diet group, specifically samples 62 and 64, which were also identified in the DataExplorer report.

```r
#---- 05 Outliers ----
explore <- read.csv("/home/user/Documents/Proyectod_ML/explore.csv") # load the data
explore <-  explore %>% # drop the sample column
  select( -X )

df_long <- explore %>% # Adjust the data set
  pivot_longer(cols = everything(), names_to = "variable", values_to = "value")

# Calculate the quartiles and the lower and upper limits for each variable in the data set.
df_outliers <- df_long %>%
  group_by(variable) %>%
  mutate(
    Q1 = quantile(value, 0.25),  # First quartile
    Q3 = quantile(value, 0.75),  # Third quartile
    IQR = Q3 - Q1,               # Interquartile range
    lower_limit = Q1 - 1.5 * IQR,  # Lower limit
    upper_limit = Q3 + 1.5 * IQR,  # Upper limit
    is_outlier = value < lower_limit | value > upper_limit  # Identify outliers
  )

#Outliers plot
ggplot(df_outliers, aes(x = variable, y = value)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  geom_point(data = filter(df_outliers, is_outlier),
             aes(x = variable, y = value),
             color = "red", size = 3) +
  labs(title = "Outliers") +
  theme_minimal()
```



Outliers

Additionally, we examined our metadata to identify any variables that could be affecting samples 62 and 64. While these samples are outliers in terms of RNA Integrity Number (RIN), they are not outliers in the amount of RNA present, and they do not show poor quality in the FastQC report. However, they do have a significantly lower number of sequences compared to other samples. This places them as outliers, but it does not compromise the overall quality of the dataset.

We decided to keep these samples in the analysis because the machine learning models showed good performance (77% accuracy across all models). This suggests that, although samples 62 and 64 are outliers, they do not negatively impact the models' ability to make correct classifications. Furthermore, retaining these samples helps increase the diversity and representativeness of the dataset, which may improve the robustness and generalizability of our models.

Gene Set Variation Analysis (GSVA)

Now we will perform a Gene Set Variation Analysis for our dataset.

```
library(recount3)
library(GSVA)
library(GSVAdata)
library(msigdbr)
library(SummarizedExperiment)
library(dplyr)
library(tidyr)
library(ggplot2)
library(pheatmap)
library(AnnotationDbi)
library(org.Rn.eg.db)

#--------- 01 Res object ----------
expr_matrix <- vst_counts

# Creating sample metadata
col_data <- data.frame(
  sample_id = colnames(expr_matrix),
  condition = rep(c("control", "treated"), each = ncol(expr_matrix) / 2),
  row.names = colnames(expr_matrix)
)

# Creating the SummarizedExperiment object
rse <- SummarizedExperiment(
  assays = list(counts = as.matrix(expr_matrix)),
  colData = col_data
)

# SummarizedExperiment object
# rse

#--------- 02 Pathway-data ---------
#| label: pathway-data
# Load C2 canonical pathways for Rattus norvegicus
c2_gene_sets <- msigdbr(species = "Rattus norvegicus",
                        category = "C2") %>%
  dplyr::select(gs_name, gene_symbol) %>%
  split(x = .$gene_symbol, f = .$gs_name)
print(paste("Number of pathways loaded:", length(c2_gene_sets)))

## [1] "Number of pathways loaded: 6366"
```

```
#--------- 03 Data-cleaning ---------
#| label: clean-data

# Create the expression matrix from the data in `rse`
expr_matrix <- assays(rse)$counts
gene_symbols <- rownames(rse)
rownames(expr_matrix) <- gene_symbols

# Remove genes with duplicate names or `NA`
expr_matrix <- expr_matrix[!is.na(rownames(expr_matrix)), ]
# Remove duplicates
expr_matrix <- expr_matrix[!duplicated(rownames(expr_matrix)), ]

# Get the rse metadata
metadata <- colData(rse) %>%
  as.data.frame()

print(paste("Número de genes después de limpiar:", nrow(expr_matrix)))

## [1] "Número de genes después de limpiar: 15309"

print(paste("Número de muestras:", ncol(expr_matrix)))

## [1] "Número de muestras: 80"

#--------- 04 Match gene names ---------
gene_ids <- rownames(expr_matrix)

# Mapping Ensembl id's to gene symbols
gene_names <- mapIds(org.Rn.eg.db, keys = gene_ids, column = "SYMBOL", keytype = "ENSEMBL", multiVals =
"first")

# Number of genes without a symbol
# sum(is.na(gene_names))

# Replace Ensembl identifiers with gene names
rownames(expr_matrix) <- gene_names

# Delete rows with names NA
expr_matrix <- expr_matrix[!is.na(gene_names), ]

# Verificar el resultado
# head(rownames(expr_matrix))

#--------- 05 Filter the expression matrix ---------
# Obtaining the unique genes of the pathways
genes_in_pathways <- unique(unlist(c2_gene_sets))

# Filter the expression matrix
expr_matrix <- expr_matrix[rownames(expr_matrix) %in% genes_in_pathways, ]

print(paste("Number of genes after filtering:", nrow(expr_matrix)))

## [1] "Number of genes after filtering: 12102"

#--------- 06 Filter duplicate genes ---------
# Delete or merge duplicate genes
expr_matrix <- expr_matrix[!duplicated(rownames(expr_matrix)), ]

# Merge duplicate genes (taking the average of the expression values)
expr_matrix <- aggregate(expr_matrix, by = list(rownames(expr_matrix)), FUN = mean)

# Reassign gene names
rownames(expr_matrix) <- expr_matrix$Group.1
# Remove the extra column from the groups
expr_matrix <- expr_matrix[, -1]
```

```r
# Check for duplicates
anyDuplicated(rownames(expr_matrix))
```

```
## [1] 0
```

```r
# This should return 0 if there are no duplicates

#--------- 07 Run gsva ---------
#| label: run-gsva

# Set up GSVA parameters
expr_matrix <- as.matrix(expr_matrix)
gsvaPar <- GSVA::gsvaParam(expr = expr_matrix,
                           geneSets = c2_gene_sets,
                           kcdf = "Gaussian")

# GSVA
gsva_results <- gsva(gsvaPar, verbose = TRUE)

# Results
print(paste("Dimensions of GSVA results:",
            nrow(gsva_results), "pathways by",
            ncol(gsva_results), "samples"))
```

```
## [1] "Dimensions of GSVA results: 6348 pathways by 80 samples"
```

```r
#--------- 08 Basic-analysis ---------
# Calculate the average enrichment scores for each pathway
mean_enrichment <- rowMeans(gsva_results)

# Sort pathways descendingly by enrichment score
sorted_pathways <- sort(mean_enrichment, decreasing = TRUE)

# Select the top 20 pathways
top_pathways <- head(sorted_pathways, 20)

# Clean up pathway names
pathway_names <- names(top_pathways)

# Eliminate common prefixes
shortened_names <- gsub("REACTOME_|KEGG_|BIOCARTA_|PID_|WP_", "", pathway_names)
shortened_names <- gsub("_", " ", shortened_names)

# Assign the new names to the pathways
names(top_pathways) <- shortened_names

# Print the 5 most enriched pathways
print("Top 5 enriched pathways:")
```

```
## [1] "Top 5 enriched pathways:"
```

```r
head(top_pathways, 5)
```

```
## FICOLINS BIND TO REPETITIVE CARBOHYDRATE STRUCTURES ON THE TARGET CELL SURFACE
##                                                                      0.13244514
##                                                  TRANSPORT AND SYNTHESIS OF PAPS
##                                                                      0.10008005
##                           PKA MEDIATED PHOSPHORYLATION OF KEY METABOLIC FACTORS
##                                                                      0.09979163
##                                                                CYP2E1 REACTIONS
##                                                                      0.09877404
##                                                          MITOCHONDRIAL UNCOUPLING
##                                                                      0.09583079
```

```r
#--------- 09 Subset pathways ---------
# Calculate average enrichment scores
mean_enrichment <- rowMeans(gsva_results)
```

```r
# Select the first 100 most enriched routes
top_pathways <- sort(mean_enrichment, decreasing = TRUE)[1:25]

# Subset of GSVA results with the first 100 routes
gsva_subset <- gsva_results[names(top_pathways), ]

# Remove NA, NaN, or Inf values
gsva_subset_clean <- gsva_subset
gsva_subset_clean <- gsva_subset_clean[complete.cases(gsva_subset_clean), ]
gsva_subset_clean <- gsva_subset_clean[, apply(gsva_subset_clean, 2, function(x) all(is.finite(x)))]

# Heatmap
pheatmap(gsva_subset_clean,
         main = "Top 100 Enriched C2 Pathways",
         scale = "row",
         show_colnames = FALSE,
         clustering_distance_rows = "correlation",
         clustering_distance_cols = "correlation",
         clustering_method = "complete",
         color = colorRampPalette(c("blue", "white", "red"))(100),
         fontsize_row = 4
)

#--------- 10 Clean final data ---------
# Remove NA, NaN, or Inf values from the entire dataset
gsva_clean <- gsva_results
gsva_clean <- gsva_clean[complete.cases(gsva_clean), ]
gsva_clean <- gsva_clean[, apply(gsva_clean, 2, function(x) all(is.finite(x)))]

# Dimensions
print(paste("Número de filas y columnas después de la limpieza:", dim(gsva_clean)))

## [1] "Número de filas y columnas después de la limpieza: 6347"
## [2] "Número de filas y columnas después de la limpieza: 80"

# write.csv
# write.csv(gsva_clean, "/home/user/Documents/Proyectod_ML/gsva_clean.csv", row.names = TRUE)
```

Final result

We obtained a data set with 6347 pathways and kept our 80 samples after GSVA.

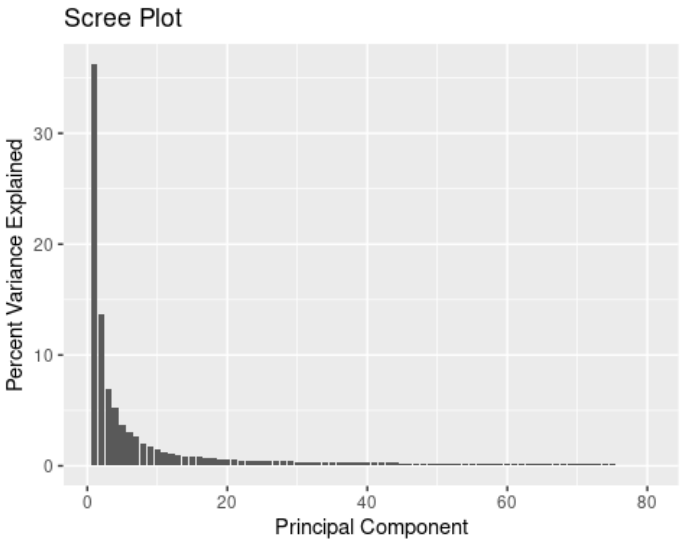
Required Dimensionality Reduction Analysis

```r
#----01 package loading-----
library(corrplot)
library(tidymodels)
library(corrr)
library(palmerpenguins)
library(janitor)
library(tidyverse)
library(DataExplorer)
library(dplyr)
library(ggplot2)

#----02 Load the data set -----
df <-  read.csv("/home/user/Documents/Proyectod_ML/gsva_clean.csv")
df <- df %>% rename(pathway = X)
df <- drop_na(df)

# Transpose the data frame
df <- t(df)
df <- as.data.frame(df)
colnames(df) <- as.character(df[1,])
df <- df[-1,]
df <- tibble::rownames_to_column(df, var = "sampleID")
df <- df %>%
```

```r
  mutate(across(-sampleID, as.numeric))
# str(df)
# View(df)

# Load metadata to add groups
df$group <- pca_data$condition
df$condition <- pca_data$Diet

#Add hepatotoxic
df <- df %>%
  mutate(hepatotoxic = case_when(
    group %in% c("Group1", "Group3") ~ "healthy",
    group %in% c("Group2", "Group4") ~ "cancer"
  ))

df <- df %>%
  select(sampleID, group, condition, hepatotoxic, everything())
# write.csv(df, file = "/home/user/Documents/Proyectod_ML/df.csv")

#---- 03 Recipe ----
data_recipe <-
  recipe(~., data = df) %>%
  update_role(sampleID, group, condition, hepatotoxic, new_role = "id") %>%
  step_naomit(all_predictors(), id = "na") %>%   # Delete NA
  step_normalize(all_predictors(), id= "normalize") %>% # Normalize predictors
  step_pca(all_predictors(), num_comp = 4,  id = "pca") %>%  # Add PCA
  prep()

# Extract the PCA
data_pca <- data_recipe %>%
  tidy(id = "pca")
# data_pca

# Extract NAs IDs
data_na <- data_recipe %>%
  tidy(id = "na")
# data_na

# Extract Normalize IDs
data_nor <- data_recipe %>%
  tidy(id = "normalize")
# data_nor
```

Variance ratio

```r
#---- 03 Scree Plot ----
#Extract the variance
variance <- data_recipe %>%
  tidy(id = "pca", type = "variance")

# Filter for percent variance
percent_variance <- variance %>%
  filter(terms == "percent variance")

# Plot
ggplot(percent_variance, aes(x = component, y = value)) +
  geom_col() +
  ylab("Percent Variance Explained") +
  xlab("Principal Component") +
  ggtitle("Scree Plot")
```



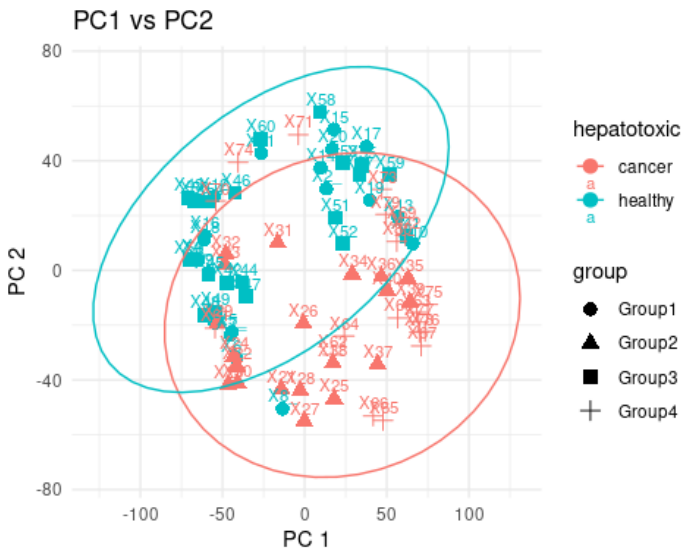

```
# We obtain the percentage of variance explained by each component
variance_table <- data_recipe %>%
  tidy(id = "pca", type = "variance") %>%
  filter(terms == "percent variance") %>%
  mutate(percent_variance = round(value, 2)) %>%
  select(component, percent_variance)

#  Table
head(variance_table,10)

## # A tibble: 10 × 2
##    component percent_variance
##        <int>            <dbl>
##  1         1             36.3
##  2         2             13.6
##  3         3              6.93
##  4         4              5.21
##  5         5              3.74
##  6         6              3.11
##  7         7              2.71
##  8         8              1.95
##  9         9              1.71
## 10        10              1.46
```

Number of components selected

We have 80 principal components, of which the first 4 explain 62.04% of the total variance. Starting from the fifth component, each accounts for less than 4% of the variance. Therefore, we decided to consider only the first 4 components and discard the remaining ones.

```
#---- 04 PCA ----
#Prep
pca_estimates <- prep(data_recipe)

#Bake
pca_results <- bake(pca_estimates, new_data = NULL)

#PCA
pca_results$sampleID <- substr(pca_results$sampleID , 1, 3)
ggplot(pca_results, aes(PC1, PC2, color = hepatotoxic, shape = group)) +
  geom_point(size = 3) +
  geom_text(aes(label = sampleID), vjust = -0.5, size = 3) +
  stat_ellipse(aes(group = hepatotoxic), level = 0.95) +
```

```
  labs(title = "PC1 vs PC2", x = "PC 1", y = "PC 2") +
  theme_minimal() +
  theme(legend.position = "right")
```



PCA 1 and 2

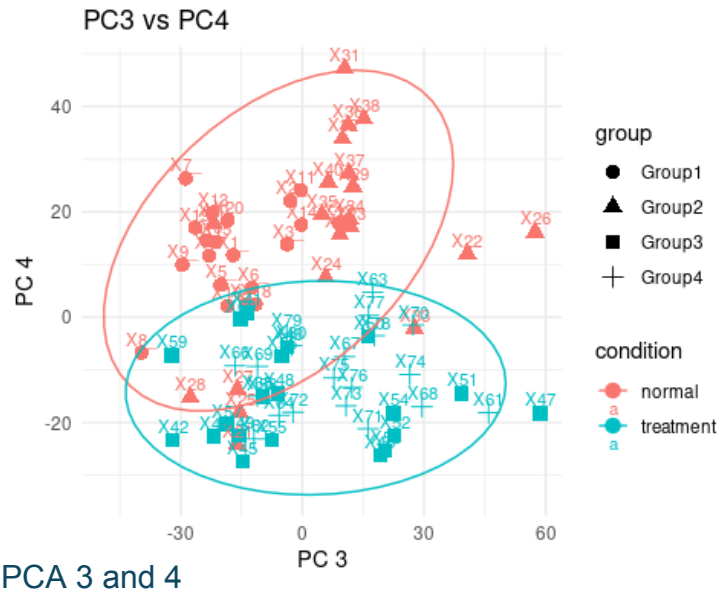
In this first PCA, we can observe that PC1 explains the largest portion of the data variability, accounting for 36.29%, while PC2 explains 13.61%, resulting in a total of 49.9% of the variability captured by these two components. Two clusters are moderately separated along PC1: the first cluster includes the groups that developed cancer due to the hepatotoxic agent (red), and the second cluster includes the healthy groups (blue).

Some samples from Group 4 (treatment diet + hepatotoxic agent) overlap with the healthy samples. This makes biological sense, as the treatment diet used (Time Caloric-Restriction Protocol) has been shown to have a protective effect in models of liver damage. Time-caloric restriction not only reduces hepatomegaly but also prevents the increase in blood leukocytes induced by hepatotoxic agents such as diethylnitrosamine. Furthermore, this protocol preserves the liver's functional and histological characteristics, favoring fibrotic areas over cirrhotic ones and limiting the progression to advanced stages of liver damage.

Although an increase in collagen deposits was observed, the protocol demonstrated positive effects, such as preventing systemic inflammation, reducing carcinoembryonic antigen levels, and limiting neoplastic transformation. Time Caloric-Restriction is an effective strategy to mitigate the adverse effects of liver damage and preserve liver function (Molina-Aguilar et al., 2017). This explains the behavior of the samples from Group 4, as we can observe expression patterns and pathways resembling healthy conditions in this group, despite the presence of the hepatotoxic agent.

```
# Prep
pca_estimates <- prep(data_recipe)

# Bake
pca_results <- bake(pca_estimates, new_data = NULL)

# Biplot para PC3 y PC4
pca_results$sampleID <- substr(pca_results$sampleID , 1, 3)
ggplot(pca_results, aes(PC3, PC4, color = condition, shape = group)) +
  geom_point(size = 3) +
```

```
  geom_text(aes(label = sampleID), vjust = -0.5, size = 3) +
 stat_ellipse(aes(group = condition), level = 0.95) +
 labs(title = "PC3 vs PC4", x = "PC 3", y = "PC 4") +
 theme_minimal() +
 theme(legend.position = "right")  # Coloca la leyenda a la derecha
```



PCA 3 and 4

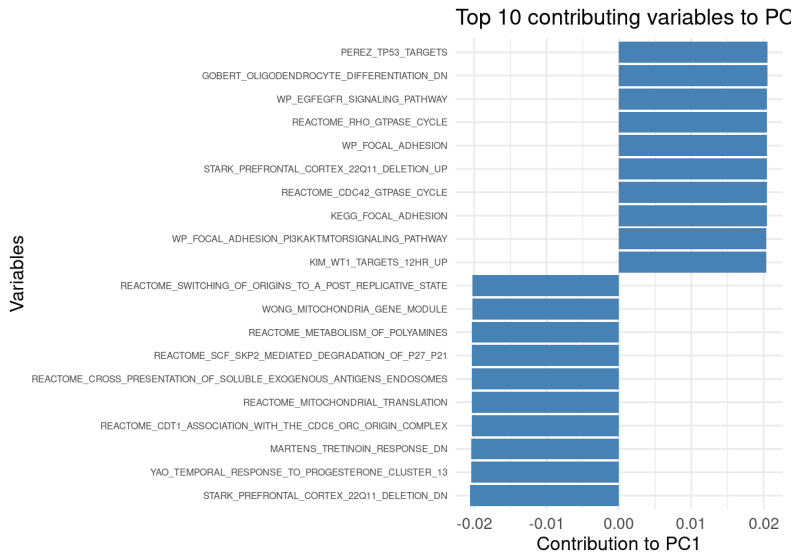
In this PCA, we are analyzing PC3, which captures 6.93% of the variability, and PC4, which captures 5.21%. Together, they account for 12.14% of the total variability. There is a clear differentiation between the samples based on diet—normal (red) and treatment (blue)—especially along PC4. This indicates that we are also observing differences in the pathways expressed as a result of the type of diet.

We have the following:

- PC1 and PC2 capture the global differences between healthy samples and cancer samples.
- PC3 and PC4 reflect diet-specific effects, distinguishing between treatment and normal diets.

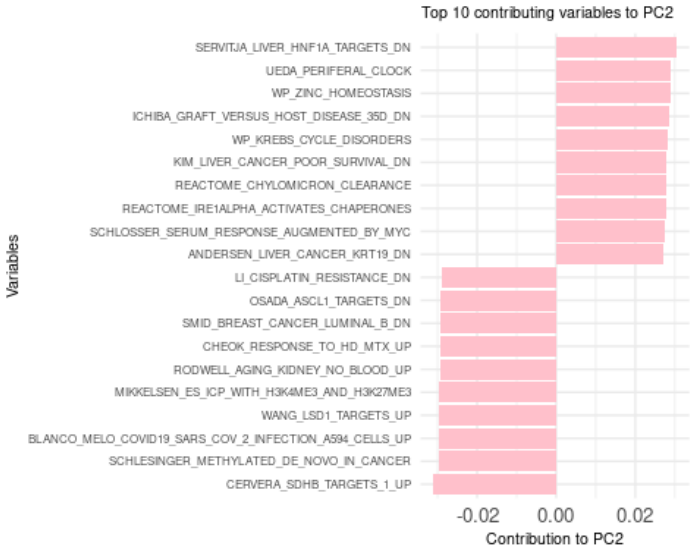
```
#---- 05 Importance plot ----

################################### PC1 ########################################
# Get PCA loadings from recipe
loadings <- data_recipe %>%
  tidy(id = "pca", type = "coef")

# Filter for only one componente
loadings_pc1 <- loadings %>%
  filter(component == "PC1") %>%
  arrange(desc(value))

# Select the 10 variables with the greatest positive and negative contribution
top_contrib_positive <- loadings_pc1 %>%
  top_n(10, value)

top_contrib_negative <- loadings_pc1 %>%
  top_n(-10, value)

top_contrib <- bind_rows(top_contrib_positive, top_contrib_negative)
```

```r
# Importance plot
ggplot(top_contrib, aes(x = reorder(terms, value), y = value)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  xlab("Variables") +
  ylab("Contribution to PC1") +
  ggtitle("Top 10 contributing variables to PC1") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 6),
    axis.title = element_text(size = 12),
    plot.title = element_text(size = 14)
  )
```

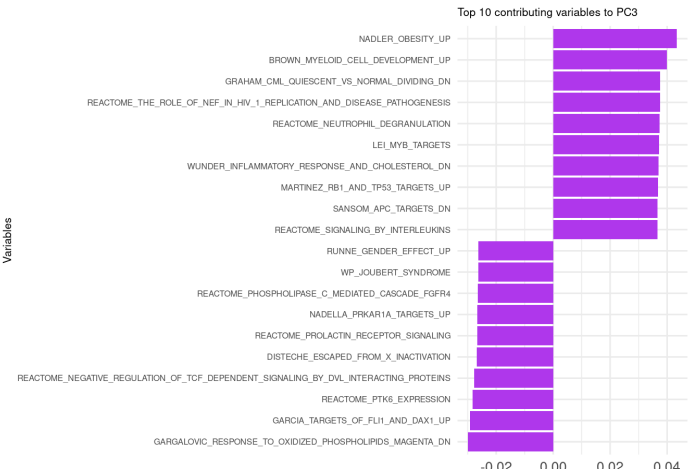



```r
################################## PC2 ######################################

loadings <- data_recipe %>%
  tidy(id = "pca", type = "coef")

loadings_pc2 <- loadings %>%
  filter(component == "PC2") %>%
  arrange(desc(value))

top_contrib_positive <- loadings_pc2 %>%
  top_n(10, value)

top_contrib_negative <- loadings_pc2 %>%
  top_n(-10, value)

top_contrib <- bind_rows(top_contrib_positive, top_contrib_negative)

ggplot(top_contrib, aes(x = reorder(terms, value), y = value)) +
  geom_col(fill = "pink") +
  coord_flip() +
  xlab("Variables") +
  ylab("Contribution to PC2") +
  ggtitle("Top 10 contributing variables to PC2") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 6),
    axis.title = element_text(size = 8),
```

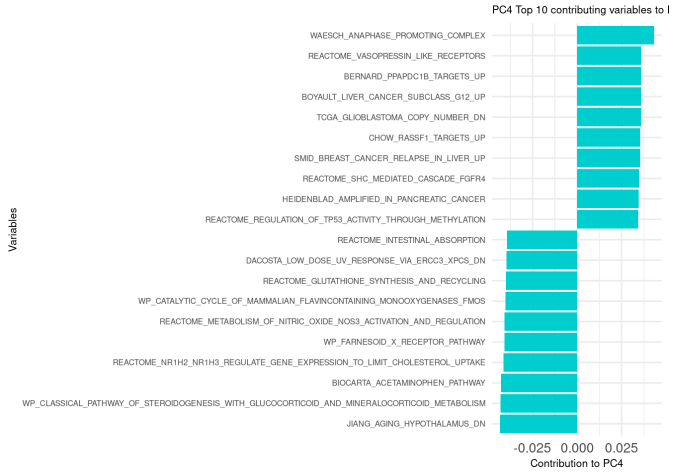
```
    plot.title = element_text(size = 8)
  )
```




```
################################## PC3 ########################################
loadings <- data_recipe %>%
  tidy(id = "pca", type = "coef")

loadings_pc3 <- loadings %>%
  filter(component == "PC3") %>%
  arrange(desc(value))

top_contrib_positive <- loadings_pc3 %>%
  top_n(10, value)

top_contrib_negative <- loadings_pc3 %>%
  top_n(-10, value)

top_contrib <- bind_rows(top_contrib_positive, top_contrib_negative)

ggplot(top_contrib, aes(x = reorder(terms, value), y = value)) +
  geom_col(fill = "#B23AEE") +
  coord_flip() +
  xlab("Variables") +
  ylab("Contribution to PC3") +
  ggtitle("Top 10 contributing variables to PC3") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 6),
    axis.title = element_text(size = 8),
    plot.title = element_text(size = 8)
  )
```



```r
################################## PC4 ########################################
loadings <- data_recipe %>%
  tidy(id = "pca", type = "coef")

loadings_pc4 <- loadings %>%
  filter(component == "PC4") %>%
  arrange(desc(value))

top_contrib_positive <- loadings_pc4 %>%
  top_n(10, value)

top_contrib_negative <- loadings_pc4 %>%
  top_n(-10, value)

top_contrib <- bind_rows(top_contrib_positive, top_contrib_negative)

ggplot(top_contrib, aes(x = reorder(terms, value), y = value)) +
  geom_col(fill = "#00CDCD") +
  coord_flip() +
  xlab("Variables") +
  ylab("Contribution to PC4") +
  ggtitle("PC4 Top 10 contributing variables to PC4") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(size = 10),
    axis.text.y = element_text(size = 6),
    axis.title = element_text(size = 8),
    plot.title = element_text(size = 8)
  )
```



PC1

This principal component captures:

- Positive Contributions: Processes related to cell differentiation, signaling pathways (e.g., EGFR signaling, focal adhesion), and cell cycle regulation (e.g., Rho GTPase cycle, CDC42 GTPase cycle).
- Negative Contributions: Processes related to mitochondrial function, polyamine metabolism, and degradation pathways, such as SCF-SKP2-mediated degradation.

PC1 separates the samples based on cancer status, distinguishing cancerous samples by processes related to cell proliferation, signaling, and differentiation, while healthy samples are characterized by mitochondrial and metabolic processes. This suggests that PC1 helps differentiate between healthy and cancer samples based on their gene expression profiles.

PC2

This principal component captures:

- Positive Contributions: Processes related to liver cancer, peripheral clock regulation, zinc homeostasis, and chylomicron clearance, linked to metabolism and liver function.
- Negative Contributions: Cancer resistance mechanisms, such as cisplatin resistance and various cancer subtypes, indicating downregulated processes in cancer.

PC2 separates samples based on cancer status, with cancer samples showing disrupted metabolism, DNA damage response, and treatment resistance, while healthy samples are associated with normal metabolic and cellular functions.

PC3

This principal component captures:

-Positive Contributions: Processes related to obesity, myeloid cell development, inflammatory response, and signaling by interleukins, which are linked to immune response and cell differentiation.

- Negative Contributions: Processes involving negative regulation of signaling pathways, such as phospholipase C-mediated cascade and TCF-dependent signaling, indicating potential downregulation in certain cancer-related pathways.

PC3 primarily separates samples based on diet, with distinct clustering based on dietary factors rather than cancer status, highlighting differences in immune and metabolic responses.

PC4

This principal component captures:

- Positive Contributions: Processes related to cell proliferation, cancer, and cell cycle regulation.
- Negative Contributions: Metabolic processes, antioxidant responses, and cellular stress.

PC4 separates the samples based on diet type, with a distinction between healthy and treatment diet groups. This component helps to differentiate between samples based on the influence of the diet.

Variable selection

We decided to select the 25 most relevant variables from each of the 4 principal components to build our final dataset, which we will use to develop our machine learning models. This will help us reduce the dimensionality of the data while retaining the most significant features that explain the variability within it.

```r
#---- 06 Variable selection ----
# store the most important variables of each component
important_vars <- list()

# Filter contributions for the 4 PCs
for (pc in paste0("PC", 1:4)) {
  loadings_pc <- loadings %>%
    filter(component == pc) %>%
    arrange(desc(abs(value)))  # # Sort by magnitude of contribution
  # Select the 25 variables with the greatest contribution
  top_vars_pc <- loadings_pc %>%
    slice_head(n = 25) %>%
    pull(terms)

  # list of variables
  important_vars[[pc]] <- top_vars_pc
}

# Merge all selected variables into a single vector
selected_vars <- unique(unlist(important_vars))
extra_vars <- c("sampleID",  "group", "condition", "hepatotoxic")

# Combine both vectors
selected_vars <- c(selected_vars, extra_vars)
selected_vars <- unique(selected_vars)

# Subset of the original dataset with the selected variables
reduced_data <- df[, selected_vars]
reduced_data <- reduced_data %>%
  select(sampleID, group, condition, hepatotoxic, everything())

# write.csv(reduced_data, file = "/home/user/Documents/Proyectod_ML/reduced_data.csv")
```

1.3 Biological Relevance

This dataset and analysis aim to advance hepatocellular carcinoma (HCC) research. By using GSVA to classify samples based on gene set activity, we can identify patterns of gene expression associated with specific diets and key pathways involved in liver disease progression. Notably, the TCR protocol's ability to slow fibrosis progression (Molina-Aguilar et al., 2017) provides a valuable opportunity to explore early stages of liver disease and improve clinical outcomes.

The findings from this analysis could have substantial clinical and research implications. By applying machine learning models to classify patient samples based on pathway signatures, we aim to develop predictive tools that could expand into broader clinical applications, such as early diagnosis and personalized treatment strategies. However, some limitations should be considered. The sample size, while informative, may limit generalizability, and the GSVA method captures only part of the complexity of liver disease biology.

Part 2: Machine Leaning Approach

2.1 Problem Formulation

Multi-class classification

We implemented a multinomial logistic regression model to classify the four experimental groups in our dataset, using the 25 most relevant variables from each of the 4 principal components. This approach allowed us to reduce the dataset's dimensionality and focus the

model on the most representative features, maximizing its predictive capacity while minimizing the noise from less informative variables.

```
#----01 package loading-----
library(tidyverse)
library(tidymodels)
```

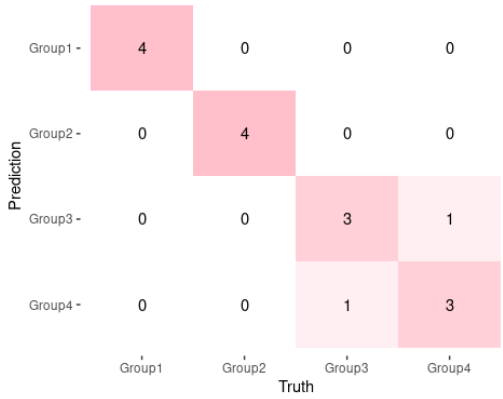
We loaded the dataset selected from the principal component analysis. Our dataset contains 80 samples and 105 pathways, with no missing data. Each group contains 20 samples. In terms of diet, two of these groups follow the treatment diet, while the other two follow a normal diet. Additionally, there are two groups with cancer and two healthy groups. Therefore, there is no bias in the number of samples across the different categories.

```
#----01 Load the data -----
df <- read.csv("/home/user/Documents/Proyectod_ML/reduced_data.csv")
# dim(df)

#----02 Clean the data ----
rattus_clean <- df %>%
  drop_na()

rattus_clean <- rattus_clean %>%
  select(-X )
rattus_clean$group <- as.factor(rattus_clean$group)
rattus_clean$condition <- as.factor(rattus_clean$condition)
rattus_clean$hepatotoxic <- as.factor(rattus_clean$hepatotoxic)
rattus_clean <- rattus_clean %>%
  select(-condition, -hepatotoxic, -sampleID )
# str(rattus_clean)
# View(rattus_clean)

#----03 Logistic Regression ----
# Split the data into training and testing
set.seed(123)
rattus_split <- initial_split(rattus_clean, prop = 0.8, strata = group)
rattus_train <- training(rattus_split)
rattus_test <- testing(rattus_split)

#----04 Specify the model ----
log_reg_model <- multinom_reg() %>%
  set_engine("nnet")  # We use the 'nnet' engine for multinomial logistic regression

#----05 Recipe -----
rattus_recipe <- recipe(group ~ ., data = rattus_train) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors())  # Creating dummy variables for categorical variables

#----06 workflow -----
rattus_workflow <- workflow() %>%
  add_model(log_reg_model) %>%
  add_recipe(rattus_recipe)

#----07 Fit the model -----
rattus_fit <- rattus_workflow %>%
  fit(data = rattus_train)

#----08 Predict on test set -----
predictions <- rattus_fit %>%
  predict(new_data = rattus_test) %>%
  bind_cols(rattus_test)

#----09 Performance -----
metrics <- predictions %>%
  metrics(truth = group, estimate = .pred_class)
```

```
# Metrics
metrics

## # A tibble: 2 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.875
## 2 kap      multiclass     0.833
```

These two metrics indicate very good model performance. The 87.5% accuracy means that the model correctly assigned the samples to their respective groups 87.5% of the time in the test set. On the other hand, the Kappa index of 0.833 shows that, in addition to accuracy, the model has high consistency in its predictions.

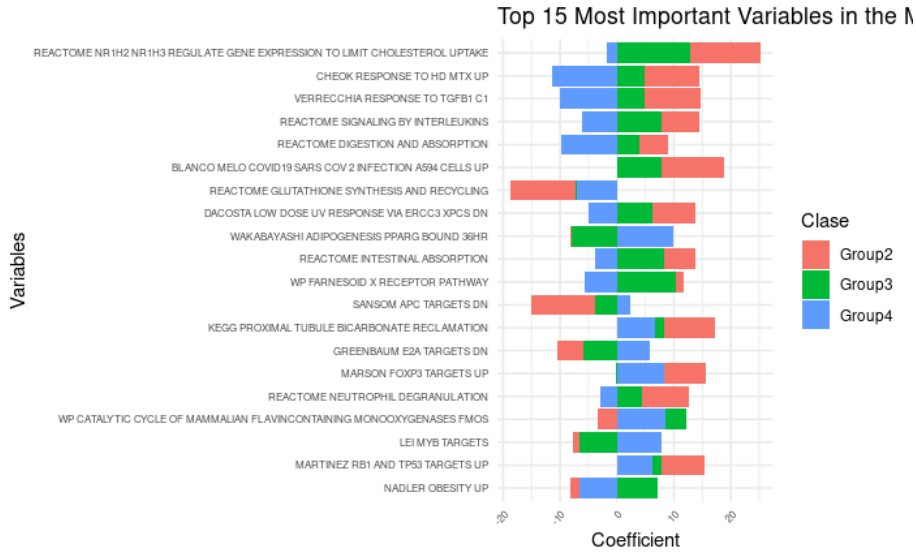
```
#----10 Confusion Matrix -----
conf_mat <- predictions %>%
  conf_mat(truth = group, estimate = .pred_class)

autoplot(conf_mat, type = "heatmap") +
  scale_fill_gradient(low = "white", high = "pink")
```



We can observe that we have a high number of true positives and very few false positives and false negatives, suggesting that the model is correctly classifying most of the samples and making few errors.

```
#----11 Extraction of parameters -----
# Coefficients for the multinomial model
model_fit <- extract_fit_parsnip(rattus_fit)

# Fitted values
model_fit$fit$fitted.values
coef(model_fit$fit)

# Extract the coefficients from the multinomial model
coefs <- coef(model_fit$fit)
coefs_df <- as.data.frame(coefs)
coefs_df$class <- rownames(coefs_df)
coefs_long <- coefs_df %>%
  pivot_longer(cols = -class, names_to = "variable", values_to = "coefficient")

#----12 Importance plot -----
# Sort the coefficients by absolute value and select the 15 most important variables
top_15_vars <- coefs_long %>%
  mutate(abs_coefficient = abs(coefficient)) %>%
  group_by(variable) %>%
  summarise(mean_abs_coefficient = mean(abs_coefficient)) %>%
  arrange(desc(mean_abs_coefficient)) %>%
```

```
  head(20)

# Filter the original dataframe to keep only the 15 most important variables
top_15_coefs <- coefs_long %>%
  filter(variable %in% top_15_vars$variable)

# Importance plot
ggplot(top_15_coefs, aes(x = reorder(variable, abs(coefficient)), y = coefficient, fill = class)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Top 15 Most Important Variables in the Multinomial Model",
       x = "Variables",
       y = "Coefficient",
       fill = "Clase") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size =6 ),
        axis.text.y = element_text(size = 6)) +
  scale_x_discrete(labels = function(x) gsub("_", " ", x))
```



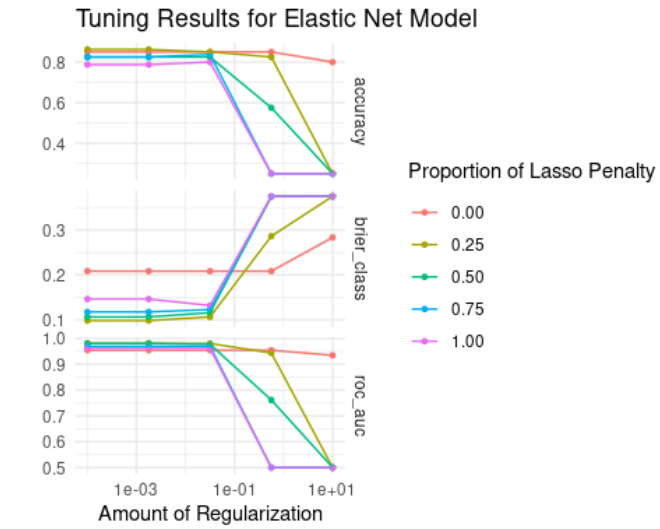
The pathways that contribute the most to the classifier are mainly related to lipid metabolism, treatment response, cell cycle, antioxidant defense, and immune signaling. These are key areas in cancer development and progression. This suggests that the model is capturing relevant information across the different groups.

Regularization

We used Elastic Net as a regularization method, as it combines Lasso (L1) and Ridge (L2). Given that we have 100 pathways and 80 samples, this method is well-suited for handling high-dimensional situations, as it helps select relevant features while controlling for overfitting. By combining the properties of Lasso and Ridge, Elastic Net can improve model accuracy and manage multicollinearity between variables, which is crucial when working with a large number of features relative to the number of available samples.

```
#----12 Regularization -----
elastic_net_spec <- multinom_reg(penalty = tune(), mixture = tune()) %>%
  set_engine("glmnet")

#----13 cross validation -----
rattus_folds <- vfold_cv(rattus_clean, v = 5, strata = group)
```

```r
#----14 hyperparameters -----

# Define the hyperparameter grid:
elastic_net_grid <- grid_regular(
  penalty(range = c(-4, 1)),
  mixture(range = c(0, 1)),   # Mixture between Lasso (L1) and Ridge (L2)
  levels = 5
)

#  workflow:
rattus_recipe <- recipe(group ~ ., data = rattus_train) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors())

rattus_workflow <- workflow() %>%
  add_model(elastic_net_spec) %>%
  add_recipe(rattus_recipe)

#Tune_grid
elastic_net_tune <- tune_grid(
  rattus_workflow,
  resamples = rattus_folds,
  grid = elastic_net_grid
)

#----15 metrics -----
elastic_net_tune %>%
  collect_metrics()

autoplot(elastic_net_tune) +
  ggtitle("Tuning Results for Elastic Net Model") +
  theme_minimal()
```



With low or moderate regularization, the performance metrics (accuracy, brier_class, roc_auc) are good, and the Lasso proportion has no significant impact. In contrast, with high regularization , all metrics deteriorate, especially with higher Lasso proportions, indicating an overly simplistic model. The best results are observed with intermediate Lasso proportions (between 0.25 and 0.75), achieving a good balance across metrics.

Best model

```
# Select the best model based on the roc_auc metric
best_elastic_net <- select_best(elastic_net_tune, metric = "roc_auc")
print(best_elastic_net)
```

```
## # A tibble: 1 × 3
##   penalty mixture .config
##     <dbl>   <dbl> <chr>
## 1  0.0001    0.25 Preprocessor1_Model06
```

The model uses weak regularization (penalty = 1e-04) with 25% Lasso (L1) and 75% Ridge (L2). This means it focuses more on smoothing coefficients like Ridge but still allows some coefficients to be zero thanks to the Lasso component.

Comparación de modelos

```
#----16 Model comparison -----
# Tune the model with the best hyperparameters
final_workflow <- finalize_workflow(rattus_workflow, best_elastic_net)

# Train the model with the full training set
elastic_net_fit <- fit(final_workflow, data = rattus_train)

# Function to evaluate the model and calculate only general metrics
evaluate_model <- function(model_fit) {
  predictions <- augment(model_fit, new_data = rattus_test)
  metrics <- predictions %>%
    metrics(truth = group, estimate = .pred_class)

  return(metrics)
}

# Evaluate the Elastic Net model
elastic_net_results <- evaluate_model(elastic_net_fit)

# metrics
elastic_net_results
```

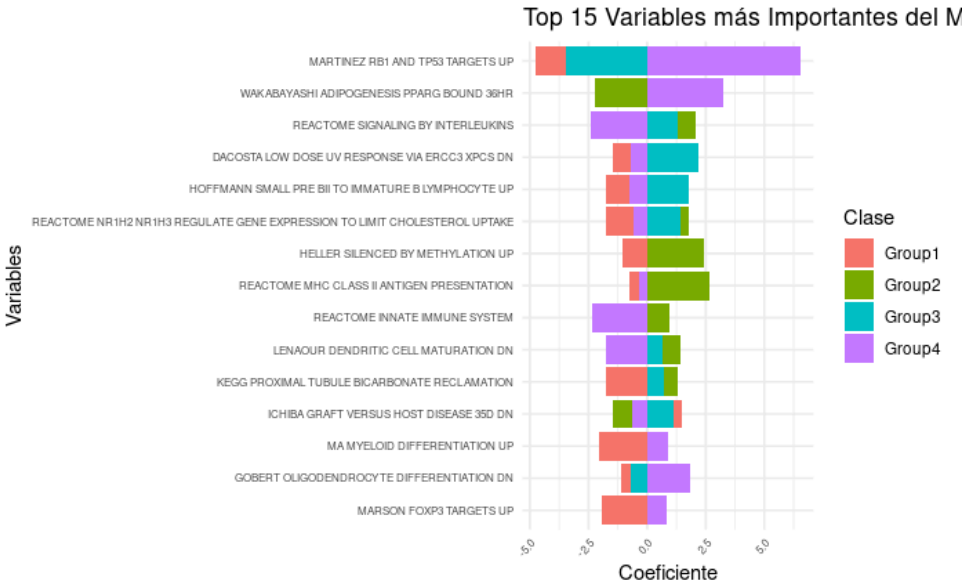
```
## # A tibble: 2 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.75
## 2 kap      multiclass     0.667
```

The model with all variables outperforms the Elastic Net in accuracy (87.5% vs. 75%) and Kappa index (83.33% vs. 66.67%), indicating that the full model correctly predicts more cases and has greater agreement between the predictions and the true classes.

The Elastic Net model performs worse in both metrics compared to the model with all variables. This is likely due to Elastic Net introducing regularization, penalizing the coefficients, and reducing the model's complexity.

```
#Importance plot
coefs <- tidy(elastic_net_fit)

coefs$class <- rep(c("Group1", "Group2", "Group3", "Group4"), each = nrow(coefs) / 4)

top_15_vars <- coefs %>%
  mutate(abs_coefficient = abs(estimate)) %>%
  arrange(desc(abs_coefficient)) %>%
  head(15)
```

```r
top_15_coefs <- coefs %>%
  filter(term %in% top_15_vars$term)


library(ggplot2)

ggplot(top_15_coefs, aes(x = reorder(term, abs(estimate)), y = estimate, fill = class)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Top 15 Variables ",
       x = "Variables",
       y = "Coeficiente",
       fill = "Clase") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 6),
        axis.text.y = element_text(size = 6),
        plot.title = element_text(size = 10),
        axis.title.x = element_text(size = 8),
        axis.title.y = element_text(size = 8)) +
  scale_x_discrete(labels = function(x) gsub("_", " ", x))
```



The pathways that have the greatest impact on the classifier are primarily associated with lipid metabolism, treatment response, the cell cycle, antioxidant defense, and immune signaling. These are crucial areas in cancer development and progression. This indicates that the model is capturing relevant information across the various groups.

Decision tree and random forest

This project aims to classify rats based on their feeding schedules and the impact of hepatotoxin administration, using pathway enrichment scores derived from RNA sequencing (RNAseq) data. These scores were computed using Gene Set Variation Analysis (GSVA), which summarizes the activity of biological pathways instead of focusing on individual gene expression.

By leveraging *machine learning* models, the study seeks to identify patterns in pathway activity associated with feeding schedules, while accounting for the effects of hepatotoxicity.

This project employs a *supervised* learning approach, as the dataset includes labeled samples that indicate the feeding schedule and treatment condition for each rat. The objective is to develop a model capable of accurately classifying unknown rats into their respective feeding schedule categories based on pathway activity profiles.

The performance of the model will be *evaluated* using standard classification metrics such as accuracy, ROC-AUC, and Kappa score. By focusing on pathway-level data, this approach aims to provide a more functional and biologically relevant interpretation of the relationships between feeding schedules, hepatotoxin exposure, and their effects on cellular pathways.

2.2 Required Model Implementation

For the project, we used decision tree and random forest models to predict the categories of rats evaluated in the study (group 1, group 2, group 3 and group 4), each with specific treatment and diet schedules. For further explanation, some explanations of each of the models will be included below.

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes (Ibm, 2024).

Meanwhile, random forest is a machine learning algorithm that combines the output of multiple decision trees to reach a single result. It handles both classification and regression problems (Ibm, 2024).

```
library(ggplot2)
library(dplyr)
library(purrr)
library(tidyr)
library(tibble)
library(stringr)
library(corrplot)
library(corrr)
library(tidymodels)
library(janitor)
library(DataExplorer)
library(tidyverse)
library(tidymodels)
library(vip)
library(viridis)

df <- read.csv("/Users/arianasilva/Documents/ML Renan/Proyecto/reduced_data.csv")
dim(df)

## [1]  80 105
```

We have included the dataset containing the pathways, groups, conditions, and treatments. Given that conditions and treatments are integral components of the experimental design for group formation, we hypothesize that they may be strongly correlated with our target variable. Nevertheless, we will conduct a Pearson's Chi-squared test to assess this relationship.

```
# Create contigency table between condition and hepatotoxic, condition and group, hepatotoxic and group
table_condition_hepatotoxic <- table(df$condition, df$hepatotoxic)

table_condition_group <- table(df$condition, df$group)

table_hepatotoxic_group <- table(df$hepatotoxic, df$group)

# Chi-squeared test
```

```r
chi_sq_condition_hepatotoxic <- chisq.test(table_condition_hepatotoxic)

chi_sq_condition_group <- chisq.test(table_condition_group)

chi_sq_hepatotoxic_group <- chisq.test(table_hepatotoxic_group)

# Print results
print(chi_sq_condition_hepatotoxic)

##
##  Pearson's Chi-squared test
##
## data:  table_condition_hepatotoxic
## X-squared = 0, df = 1, p-value = 1

print(chi_sq_condition_group)

##
##  Pearson's Chi-squared test
##
## data:  table_condition_group
## X-squared = 80, df = 3, p-value < 2.2e-16

print(chi_sq_hepatotoxic_group)

##
##  Pearson's Chi-squared test
##
## data:  table_hepatotoxic_group
## X-squared = 80, df = 3, p-value < 2.2e-16
```

Condition vs. Hepatotoxic: The result yields an X-squared value of 0, with a p-value of 1; this thereby indicates no significant association between the condition and hepatotoxic variable. This indicates that these two variables are independent and there is no meaningful relationship between these variables in the dataset.

Condition vs. Group: The test gives an X-squared value of 80 with a p-value smaller than 2.2e-16, which is highly significant. This therefore indicates very strong association between the condition and group variables; hence, these two variables are not independent and may be related.

Hepatotoxic vs. Group: In the test, X-squared value of 80 was given with a p-value smaller than 2.2e-16, also indicating strong and statistically significant association of hepatotoxic with group.

In other words, while there is no significant relationship between condition and hepatotoxic, both condition and hepatotoxic are strongly and significantly related to the group variable. Therefore, we will eliminate both the conditional variable and the hepatotoxic variable for the creation of the model.

Then, we performed data preprocessing on the `rattus_clean` dataset by cleaning the data (removing missing values and irrelevant columns) and normalizing predictors. The goal is to prepare the data for further analysis or modeling.

```r
rattus_clean <- df %>%
  drop_na()

# Check the number of remaining rows
nrow(rattus_clean)

## [1] 80
```

```r
rattus_clean <- rattus_clean %>%
  select(-c(condition, hepatotoxic, X, sampleID))
rattus_clean$group <- as.factor(rattus_clean$group)
View(rattus_clean)

# Recipe
data_recipe <- recipe(~., data = rattus_clean) %>%
  update_role(group, new_role = "id") %>%
  step_naomit(all_predictors(), id = "na") %>%   # Delete NA
  step_normalize(all_predictors(), id = "normalize") %>% # Normalize predictors
  prep()
```

Splitting data

The dataset is split into training and testing subsets using initial_split, ensuring the stratification is done by the *group* variable, representing the different groups of rats based on their feeding schedule.

```r
# Data split
set.seed(123)
subset_split <- initial_split(rattus_clean, prop = 0.7, strata = group)
subset_train <- training(subset_split)
subset_test <- testing(subset_split)
```

Recipe for classification

A preprocessing recipe was created, specifying how the data should be transformed before being used in the model. The target variable (group) is considered an identifier, with all other variables being predictors, involving dummy encoding for categorical variables and normalization for numerical variables.

```r
# Recipe for the mode
subset_recipe <- recipe(group ~ ., data = subset_train) %>%
  step_dummy(all_nominal_predictors()) %>%  # Convert categorical features into dummy variables (binary)
  step_zv(all_predictors()) # Remove all predictors with zero variance
```

Cross-validation

10-fold cross-validation is used to evaluate the model's performance, where the data is split into 10 subsets (folds), and the model is trained on 9 folds, with the remaining fold used as the test set.

```r
set.seed(123)
subset_folds <- vfold_cv(subset_train, v = 10, repeats = 5, strata = group)
```

Classification

This is the moment when the decision tree and random forest models begin to be built, with the subsequent creation of workflows where the recipe created previously is combined with the specific model.

```r
library(ranger)
# Specifying decision tree model
dt_spec <- decision_tree() %>%
  set_mode("classification") %>% # Set the mode to sorting
  set_engine("rpart") # "rpart" is tipically used in decision tree models

# Specifying random forest model
rf_spec <- rand_forest() %>%
  set_mode("classification") %>%  # Set the mode to sorting
  set_engine("ranger", importance = "impurity") # "ranger" is an eficient engine for random forest
```

```r
dt_workflow <- workflow() %>%
  add_model(dt_spec) %>%
  add_recipe(subset_recipe)

rf_workflow <- workflow() %>%
  add_model(rf_spec) %>%
  add_recipe(subset_recipe)
```

The Random Forest model is trained and evaluated using the cross-validation folds. The workflow combines the preprocessing recipe and the model, and the fit_resamples() function trains the model on each fold while collecting performance metrics.

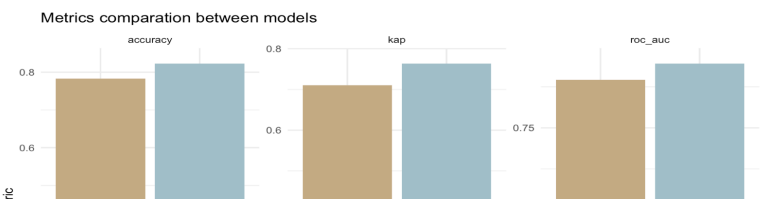
Model evaluation

Performance metrics from the cross-validation process are aggregated using collect_metrics.

```r
dt_res <- dt_workflow %>%
  fit_resamples(
    resamples = subset_folds,
    metrics = metric_set(accuracy, roc_auc, kap),
    control = control_resamples(save_pred = TRUE)
  )

rf_res <- rf_workflow %>%
  fit_resamples(
    resamples = subset_folds,
    metrics = metric_set(accuracy, roc_auc, kap),
    control = control_resamples(save_pred = TRUE)
  )

# Decision tree metrics
dt_metrics <- dt_res %>%
  collect_metrics()
# Random forest metrics
rf_metrics <- rf_res %>%
  collect_metrics()

# Combine both metrics
combined_metrics <- bind_rows(
  dt_metrics %>% mutate(model = "Decision Tree"),
  rf_metrics %>% mutate(model = "Random Forest")
)

print(combined_metrics)
```

```
## # A tibble: 6 × 7
##   .metric  .estimator  mean     n std_err .config              model
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>                <chr>
## 1 accuracy multiclass 0.782    50  0.0247 Preprocessor1_Model1 Decision Tree
## 2 kap      multiclass 0.71     50  0.0329 Preprocessor1_Model1 Decision Tree
## 3 roc_auc  hand_till  0.896    50  0.0134 Preprocessor1_Model1 Decision Tree
## 4 accuracy multiclass 0.805    50  0.0275 Preprocessor1_Model1 Random Forest
## 5 kap      multiclass 0.74     50  0.0366 Preprocessor1_Model1 Random Forest
## 6 roc_auc  hand_till  0.950    50  0.0122 Preprocessor1_Model1 Random Forest
```

```r
ggplot(combined_metrics, aes(x = model, y = mean, fill = model)) +
  geom_bar(stat = "identity", position = "dodge", show.legend = FALSE) +
  facet_wrap(~.metric, scales = "free_y") +
  labs(title = "Comparación de métricas entre modelos",
       x = "Modelo",
       y = "Valor de la Métrica Promedio") +
  theme_minimal() +
  scale_fill_manual(values = c("Decision Tree" = "#CDAA7D", "Random Forest" = "lightblue3"))
```



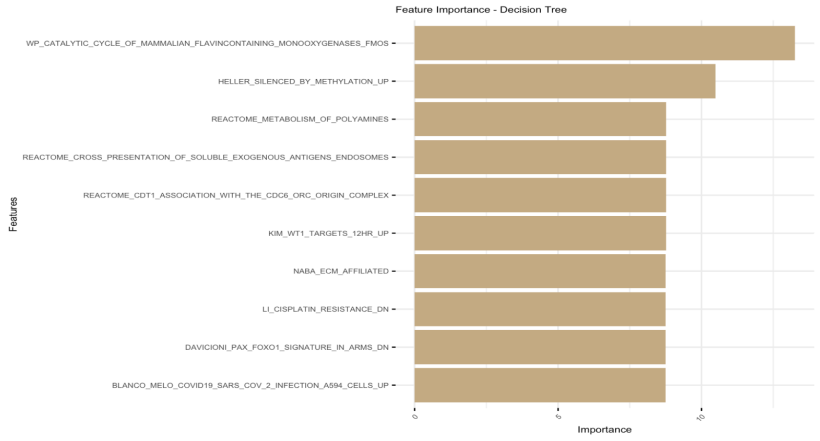
Accuracy: In decision tree model, on average, it correctly classified 70% of the cases, in contrast with random forest model, which correctly classified 80% of the cases, better than decision tree model.

Kappa score: Kappa score of 0.6111 for the decision tree model indicates moderate agreement between predictions and actual values, while the score of 0.7444 for the random forest model suggests better agreement.

ROC AUC: Since the ROC AUC of the decision tree model is 0.8842 and that of the random forest is 0.9481, we can conclude that the random forest performs significantly better in classifying this data set.

Subsequently, we fitted a decision tree model to the training data, visualized the feature importance, and extracted the most influential variables. In a biological context, these key features correspond to the pathways most impacted by the model, helping to identify which biological processes play a crucial role in the observed outcomes.

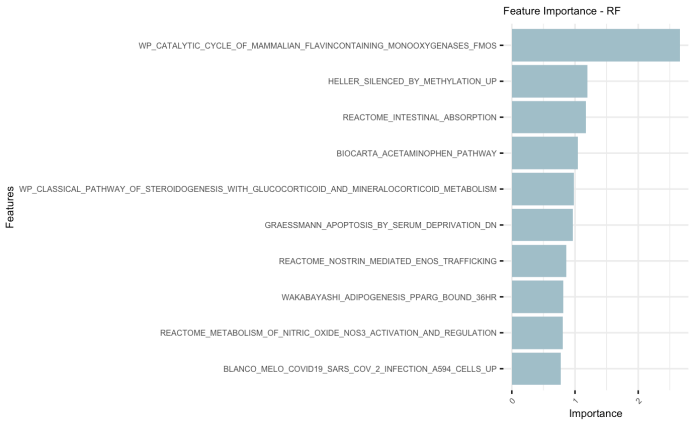
```r
# Visualizar la importancia de características para Decision Tree
modelo_dt <- dt_workflow %>%
  fit(subset_train) %>%
  extract_fit_parsnip()

vip(modelo_dt, geom = "col", aesthetics = list(fill = "#CDAA7D")) +
  labs(title = "Feature Importance - Decision Tree",
       x = "Features",
       y = "Importance") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(size = 6, angle = 45, hjust = 1), # Reduce size and rotate labels
    axis.title = element_text(size = 8), # Adjust axis title size
    plot.title = element_text(size = 8), # Adjust plot title size
    plot.margin = margin(8, 8, 8, 8), # Adjust margins to save space
    axis.text.y = element_text(size = 6), # Reduce size of y-axis labels (importance values)
    axis.ticks = element_line(size = 0.5), # Reduce size of axis ticks
    legend.position = "none" # Hide legend if not necessary
  ) +
  coord_flip() # Flip coordinates if the variable names are still too large
```



In the decision tree model, we observe that the most influential pathways are associated with biological processes such as drug metabolism, gene silencing through methylation, pathways regulated by the WT1 gene, which plays key roles in development and cell survival, as well as immune system responses and the metabolism of specific substances.

Similarly, the same approach was applied to the random forest model. By fitting the model to the training data and extracting the feature importance, we identified the pathways most influential in the random forest model.

```r
# Visualizar la importancia de características para Decision Tree
modelo_rf <- rf_workflow %>%
  fit(subset_train) %>%
  extract_fit_parsnip()

vip(modelo_rf, geom = "col", aesthetics = list(fill = "lightblue3")) +
  labs(title = "Feature Importance - Decision Tree",
       x = "Features",
       y = "Importance") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(size = 6, angle = 45, hjust = 1), # Reduce size and rotate labels
    axis.title = element_text(size = 8), # Adjust axis title size
    plot.title = element_text(size = 8), # Adjust plot title size
    plot.margin = margin(8, 8, 8, 8), # Adjust margins to save space
    axis.text.y = element_text(size = 6), # Reduce size of y-axis labels (importance values)
    axis.ticks = element_line(size = 0.5), # Reduce size of axis ticks
    legend.position = "none" # Hide legend if not necessary
  )
```



In the random forest model, we observe pathways that regulate hormones and metabolic functions, adipogenesis, gene silencing, apoptosis, and liver toxicity. Additionally, pathways related to nutrient absorption and immune system functions are also present.

Model prediction

After the model has been trained, it is utilized to predict outcomes on the test set. The accuracy of the model is assessed by comparing its predictions with the actual labels of the test set to determine its ability to work with new data.

```r
# Fit the best performing model to the entire training set
dt_best_model <- dt_workflow
dt_final_fit <- dt_best_model %>%
  last_fit(split = subset_split)

rf_best_model <- rf_workflow
rf_final_fit <- rf_best_model %>%
  last_fit(split = subset_split)

# Collect the results
dt_final_results <- collect_metrics(dt_final_fit)
rf_final_results <- collect_metrics(rf_final_fit)

print(dt_final_results)

## # A tibble: 3 × 4
##   .metric     .estimator .estimate .config
##   <chr>       <chr>          <dbl> <chr>
## 1 accuracy    multiclass     0.5   Preprocessor1_Model1
## 2 roc_auc     hand_till      0.711 Preprocessor1_Model1
## 3 brier_class multiclass     0.420 Preprocessor1_Model1

print(rf_final_results)

## # A tibble: 3 × 4
##   .metric     .estimator .estimate .config
##   <chr>       <chr>          <dbl> <chr>
## 1 accuracy    multiclass     0.792 Preprocessor1_Model1
## 2 roc_auc     hand_till      0.910 Preprocessor1_Model1
## 3 brier_class multiclass     0.202 Preprocessor1_Model1
```

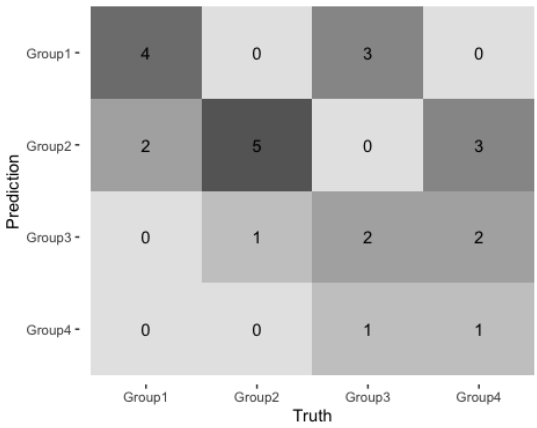
Accuracy: the decision tree model correctly classified, on average, 50% of the cases, while the random forest model demonstrated a higher accuracy, correctly classifying 79% of the cases.

ROC AUC: the decision tree model achieved a ROC AUC of 0.7106, compared to 0.9481 for the random forest model. This indicates that the random forest significantly outperforms the decision tree in terms of classification ability, as reflected by the higher ROC AUC value.

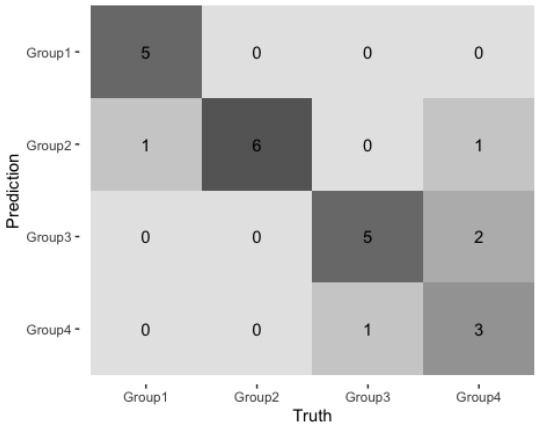
Brier class: the decision tree model has a score of 0.4197 indicates the model's predicted probabilities are fairly accurate, however, there is still potential for enhancement. Meanwhile, in random forest the brier class value is 0.1987, which shows good probability calibration for multiclass predictions.

Then, a confusion matrix was created to assess how well the classification model performed by comparing the predicted and actual class labels in the test set. This matrix shows how well the model categorizes each group, detecting true positives, false positives, true negatives, and false negatives.

```r
dt_predictions <- dt_final_fit %>%
  collect_predictions()
rf_predictions <- rf_final_fit %>%
  collect_predictions()

dt_conf_mat <- dt_predictions %>%
  conf_mat(truth = group, estimate = .pred_class)
rf_conf_mat <- rf_predictions %>%
  conf_mat(truth = group, estimate = .pred_class)

autoplot(dt_conf_mat, type = "heatmap")
```




```
autoplot(rf_conf_mat, type = "heatmap")
```



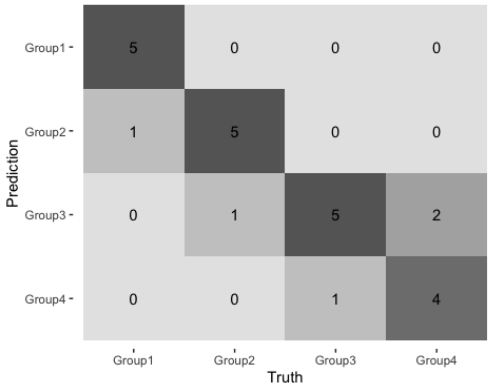
While *decision tree* showed decent performance, especially in group 2, but struggled more with misclassifying instances from group 1 and group 4, *random forest* demonstrated better overall performance, particularly in group 4, with fewer misclassifications across all groups. It showed a balanced prediction approach, correctly identifying instances in group 1, group 2, and group 3.

Hyperparameter adjustment

Also, we tuned the hyperparameters of a Random Forest model by fine-tuning parameters like mtry, number of trees (trees), and minimum data points for a split (min_n). It establishes a range of potential values for every parameter and employs cross-validation to assess the model's effectiveness. The optimal hyperparameter combination is chosen considering accuracy, and the test set is used to train and assess the final model. The model's performance is optimized by visualizing the results with a confusion matrix.

```
# Especificar modelo de Random Forest con hiperparametros ajustables
rf_spec_tuned <- rand_forest(
  mtry = tune(),       # Ajustar el número de predictores seleccionados en cada división
  trees = tune(),
  min_n = tune()       # Ajustar el número mínimo de datos para realizar una división
) %>%
```

```r
  set_mode("classification") %>%
  set_engine("ranger", importance = "impurity")

# Actualizar el workflow con el modelo que tiene tune()
rf_workflow_tuned <- workflow() %>%
  add_model(rf_spec_tuned) %>%
  add_recipe(subset_recipe)

# Definir la cuadrícula de hiperparámetros
grid_rf <- grid_regular(
  mtry(range = c(1, 10)), # Rango para 'mtry'
  min_n(range = c(2, 20)),
  trees(range = c(100, 800)),# Rango para 'min_n'
  levels = 5                   # Número de valores a probar en cada rango
)

# Ajuste del modelo usando tune_grid
set.seed(123) # Fijar semilla para reproducibilidad
rf_tuned_res <- rf_workflow_tuned %>%
  tune_grid(
    resamples = subset_folds,     # Validación cruzada definida previamente
    grid = grid_rf,               # Cuadrícula de hiperparámetros
    metrics = metric_set(accuracy, roc_auc, kap),  # Métricas a evaluar
    control = control_grid(save_pred = TRUE)  # Guardar predicciones
  )

# Seleccionar los mejores hiperparámetros según la métrica deseada (por ejemplo, accuracy)
best_rf_params <- select_best(rf_tuned_res, metric = "accuracy")

# Finalizar el workflow con los mejores hiperparámetros
final_rf_workflow <- finalize_workflow(
  rf_workflow_tuned,
  best_rf_params
)

# Entrenar el modelo final usando los mejores hiperparámetros
final_train_fit <- final_rf_workflow %>%
  fit(data = subset_train)

# Evaluar el modelo en el conjunto de prueba
final_test_fit <- final_rf_workflow %>%
  last_fit(split = subset_split)

# Recoger los resultados finales
final_results <- collect_metrics(final_test_fit)
print(rf_final_results)

## # A tibble: 3 × 4
##   .metric     .estimator .estimate .config
##   <chr>       <chr>          <dbl> <chr>
## 1 accuracy    multiclass     0.792 Preprocessor1_Model1
## 2 roc_auc     hand_till      0.910 Preprocessor1_Model1
## 3 brier_class multiclass     0.202 Preprocessor1_Model1

# Obtener las predicciones en el conjunto de prueba
final_predictions <- final_test_fit %>%
  collect_predictions()  # Recoge las predicciones de la última evaluación

# Crear la matriz de confusión
final_conf_mat <- final_predictions %>%
  conf_mat(truth = group, estimate = .pred_class)

# Visualizar la matriz de confusión
autoplot(final_conf_mat, type = "heatmap")
```



Talking about metrics, we have accuracy, ROC AUC and brier class.

Accuracy: The accuracy of the initial random forest model was 79%, while the hyperparameter-tuned model achieved 79.17%. Although the difference is small, the hyperparameter-tuned model slightly improved the classification performance, indicating that fine-tuning the model parameters has led to a marginal enhancement in correctly classifying instances.

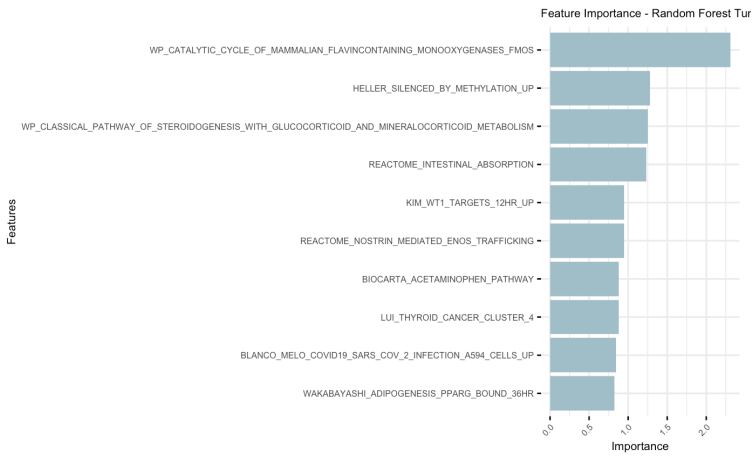
ROC AUC: The initial random forest model had a ROC AUC of 0.9481, which was slightly higher than the hyperparameter-tuned model's ROC AUC of 0.92. Despite this small difference, both values indicate strong classification performance, with the tuned model still demonstrating excellent discriminatory power. The ROC AUC values suggest that both models can effectively distinguish between classes, but the initial model performed slightly better in this regard.

Brier Class: The initial random forest model had a Brier Class score of 0.1987, which indicates moderate calibration of predicted probabilities. The hyperparameter-tuned model showed the same score of 0.1987.

In addition, as it is visualized in the confusion matrix, the Random Forest model shows better classification performance for group 4 compared to other models, with fewer misclassifications. While group 3 is sometimes confused with group 4 and group 2, the overall model performs well, especially for group 4, with more correct predictions than errors.

```
# Visualizar la importancia de características para Random forest
modelo_rf_final <- rf_workflow %>%
  finalize_workflow(best_rf_params) %>%
  fit(subset_train) %>%
  extract_fit_parsnip()

vip(modelo_rf_final, geom = "col", aesthetics = list(fill = "lightblue3")) +
  labs(title = "Feature Importance - Random Forest Tuned",
       x = "Features",
       y = "Importance") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(size = 6, angle = 45, hjust = 1), # Reduce size and rotate labels
    axis.title = element_text(size = 8), # Adjust axis title size
    plot.title = element_text(size = 8), # Adjust plot title size
    plot.margin = margin(8, 8, 8, 8), # Adjust margins to save space
    axis.text.y = element_text(size = 6), # Reduce size of y-axis labels (importance values)
    axis.ticks = element_line(size = 0.5), # Reduce size of axis ticks
    legend.position = "none" # Hide legend if not necessary
```

```
) +
coord_flip() # Flip coordinates if the variable names are still too large
```



The pathways involved in the random forest model with tuned hyperparameters is partially aligned with those of the initial random forest model. Here, pathways are associated with metabolism, intestinal absorption, gene expression regulation (silencing), transport regulation, apoptosis, liver metabolism and toxicity of acetaminophen, regulation of cell growth and immune system responses.

Part 3: Literature Review

3.1 Primary Literature Analysis

Multimetric feature selection for analyzing multicategory outcomes of colorectal cancer: random forest and multinomial logistic regression models (Feng, C. H., et al., 2022)

The study aimed to classify the four different survival outcomes of colorectal cancer (CRC) patients using RNA-seq data, focusing on identifying the most effective set of features for classification. The categories included various survival outcomes, such as “alive with no progression” and “dead with progression.”

The research utilized both Random Forest (RF) and Multinomial Logistic Regression (MLR) models, optimizing feature selection through multimetric approaches.

Due to validation, authors used cross-validation, including 5-fold and 10-fold approaches to assess accuracy and other performance metrics consistently.

The methodology presented by Feng, C. H., et al. (2022). demonstrates the effectiveness of combining Random Forest and Multinomial Logistic Regression for classifying multicategory outcomes using RNA-Seq data.

Using supervised learning methods for gene selection in RNA-Seq Case-Control studies (Wenric & Shemirani, 2018)

This study investigates the use of supervised learning methods to prioritize genes in RNA-Seq data for case-control studies in the context of disease-related gene expression. The authors introduced two key innovations: 1. The use of Random Forests to rank genes based on permutation importance 2. The Extreme Pseudo-Samples (EPS) method, which leverages Variational Autoencoders (VAE) to generate extreme pseudo-samples, enhancing gene selection.

To validate their approach, the authors applied survival analysis to cancer datasets, in addition, results were validated using separate training and testing datasets, with accuracy metrics evaluated for both.

Limitations include variability in gene rankings across iterations and computational trade-offs in Random Forests, as well as inconsistent performance of the EPS method across different datasets, and a small dataset.

The methodology presented by Wenric & Shemirani (2018) is directly relevant to this project, where machine learning is used to classify and prioritize genes based on RNA-Seq data from liver cancer in rats.

Machine learning model for predicting Major Depressive Disorder using RNA-Seq data: optimization of classification approach (Verma & Shakya, 2021)

The primary objective was the classification of Major Depressive Disorder (MDD), including distinguishing between suicidal and non-suicidal MDD patients.

The key innovation was the use of Random Forest (RF) and K-Nearest Neighbors (KNN) for gene classification and feature selection, coupled with PCA to reduce data dimensionality.

To validate their approach, the authors used the model's accuracy scores. As it is well-known, Random Forest has a significant limitation, the overfitting indicated by the high accuracy on the training data compared to the lower accuracy on the test data.

The methodology presented by Verma & Shakya (2021) illustrates the effectiveness of combining PCA and Random Forest to handle high-dimensional RNA-Seq data, which is directly applicable to the present project.

3.2 Methods comparison

In the literature, various machine learning techniques are utilized to analyze RNA-Seq data and classify biological outcomes. The studies discussed in this section employed different methods, each with its unique advantages and drawbacks.

1. Random Forest:

In all three studies examined, random forest played a key role in both feature selection and classification. This model excels at managing high-dimensional datasets, such as RNA-Seq, because it can accommodate a large number of features without overfitting.

Although random forest is generally resistant to overfitting, it can still experience this issue if not properly tuned, particularly with smaller datasets or when the number of trees is not optimized

(as noted in the Verma & Shakya, 2021 study). Furthermore, random forest's lack of interpretability can be a disadvantage in biological contexts where understanding the relationships between features is essential.

2. Multinomial Logistic Regression:

In Feng, C. H. et al. (2022), authors integrates Random Forest with Multinomial Logistic Regression to classify various survival outcomes in colorectal cancer patients. MLR is a straightforward and interpretable approach for multiclass classification, making it valuable when the aim is to uncover the relationships between features and multiple outcome categories. It offers clear insights into how each predictor influences the likelihood of class membership.

On the other hand, MLR may face challenges with high-dimensional data (like RNA-Seq data) and might necessitate dimensionality reduction to be effective.

In our study of liver cancer outcomes in rats using RNA-Seq data, we first utilized Multinomial Logistic Regression (MLR) to tackle the issue of multiclass classification. This approach was selected for its clarity and ability to differentiate between various results, in line with the objective of categorizing different cancer stages or treatment results. After that, we utilized Random Forest (RF) because of its ability to handle high-dimensional biological data effectively and its capability to detect non-linear correlations among features.

Part 4. Implementation and results

In this study, we aimed to classify four experimental groups of rats, two of which were administered a hepatotoxin to induce liver carcinoma and two were kept as healthy controls. The analysis was based on RNA-Seq data, focusing on pathway activity scores related to different feeding protocols and the administration of the hepatotoxin.

Decision Trees and Random Forest models

We compared the performance of Decision Trees (DT) and Random Forest (RF) for the classification task. The models were evaluated using several performance metrics: accuracy, ROC AUC, and Kappa.

Visual representations of the performance measurements show that Random Forest outperforms Decision Trees across all metrics, indicating it is more suitable for managing the intricacies of RNA-Seq data. This finding is consistent with prior research, which has demonstrated that Random Forest models are highly effective at managing high-dimensional biological data and identifying intricate non-linear connections among features.

As a component of our investigation, we utilized *hyperparameter tuning* on the Random Forest model in order to enhance its performance even more. However, the metrics indicated that the tuned model performed similarly to the original. This could be due to the initial model being well-configured, or it might suggest that further hyperparameter optimization was not necessary, despite testing models with tree counts ranging from 100 to 800 and various combinations of variable and observation numbers, as the performance remained very similar. Future analysis could be useful with more data or computing power.

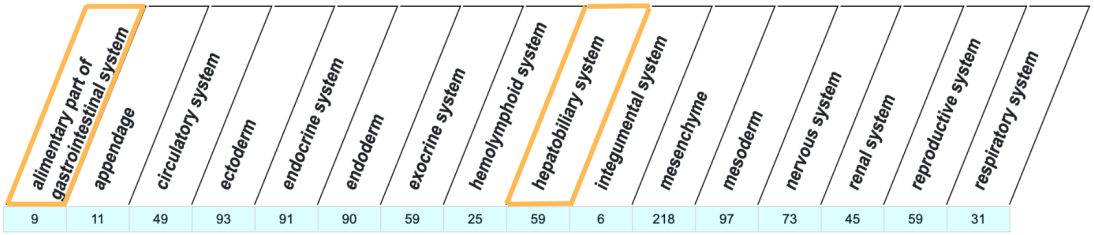
It is important to mention that Decision Trees were not tuned due to inferior performance compared to Random Forest's complexity and flexibility.

Biological interpretation

Both models, Logistic Multinomial Regression and Random Forest, emphasize comparable biological processes, especially in metabolism, immune system responses, gene expression regulation, and adipogenesis. The Random Forest (tuned) model emphasizes gene regulation and metabolic control mechanisms, whereas the LMR model offers a wider perspective, encompassing immune system differentiation and disease response. Both models exhibit common pathways associated with cholesterol metabolism, immune activation, and gene silencing, highlighting the significance of these processes within the dataset's context.

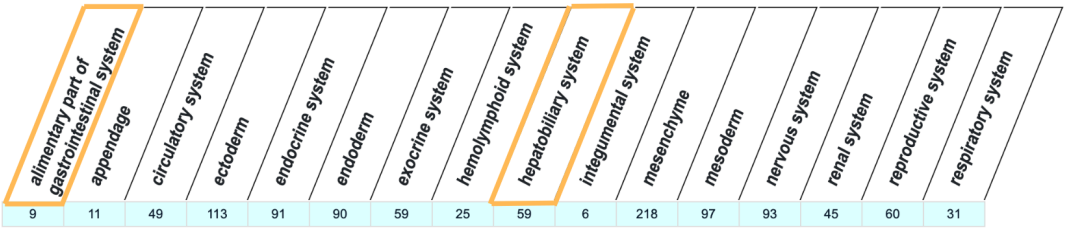
So far, we have extensively discussed the pathways influencing the experimental groups of rats exposed to different diets and a hepatotoxic agent. However, certain genes stand out in our models and are prominently featured in these pathways. In the **Logistic Multinomial Regression model**, two key genes, Rb1 and TP53, emerge as noteworthy.

According to UniProt, **Rb1** is a tumor suppressor that plays a crucial role in regulating the G1/S transition of the cell cycle. Additionally, the Rat Genome Database indicates that this gene is expressed across various systems, including the hepatobiliary system and gastrointestinal system, aligning with the central focus of our current project.



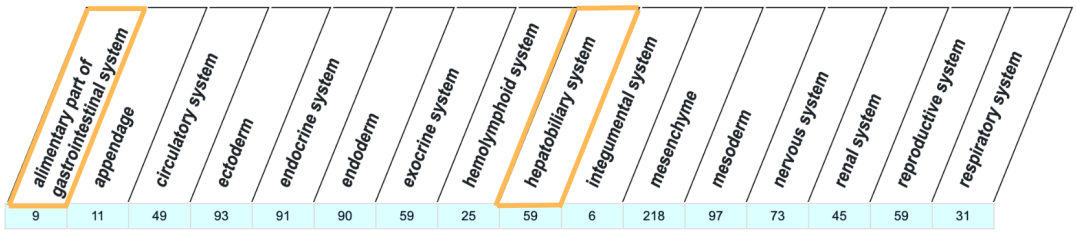
Rb1 expression in different systems

TP53, according to UniProt, encodes a multifunctional transcription factor that regulates cell cycle arrest, DNA repair, or apoptosis by binding to its target DNA sequences. It functions as a tumor suppressor in various tumor types, inducing growth arrest or apoptosis depending on the physiological context and cell type. Additionally, it negatively regulates cell division by controlling the expression of genes involved in this process. Similar to **Rb1**, Tp53 is represented in multiple tissues and systems in the Rat Genome Database, including the hepatobiliary and gastrointestinal systems.



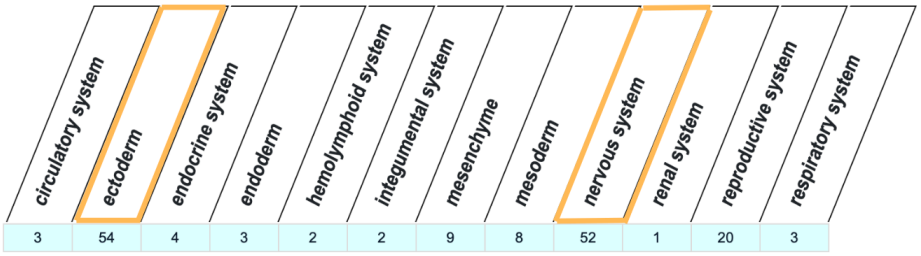
Tp53 expression in different systems

In the **tuned Random Forest model**, the **Pparg** gene is prominently featured within the pathways identified. According to UniProt, Pparg is described as a nuclear receptor that binds to peroxisome proliferators such as hypolipidemic drugs and fatty acids. Additionally, according to Rat Genome Database, it acts as a regulator of **BMAL1** transcription, a gene involved in cardiovascular circadian rhythms. Considering that dietary timing is a critical aspect of the experimental design and can influence molecular clock genes, the role of Pparg holds significant importance in the study.



Pparg expression in different systems

Another gene that was highlighted as important in this model is Wnt1, which according to UniProt, is a transcription factor that plays a crucial role in cellular development and cell survival. Its expression is predominantly reported during embryonic development, as stated by RGD.



Pparg expression in different systems

To conclude, the analysis highlights the interconnected biological processes identified across both Logistic Multinomial Regression and Random Forest models, with a particular focus on metabolism, immune response, and gene regulation. Key genes such as Rb1, TP53, Pparg and Wnt1 play central roles in these pathways.

Rb1 and TP53 underscore the importance of tumor suppression, cell cycle regulation, and tissue-specific expression in systems impacted by hepatotoxicity and dietary changes. Meanwhile, Pparg adds another layer of significance by linking gene regulation, metabolic control, and circadian rhythms, aligning closely with the experimental design's focus on dietary timing and its impact on molecular clock genes. The gene Wnt1, a crucial transcription factor involved in cellular development and survival, also emerges as significant, highlighting its role in tissue development and cellular functions.

These findings suggest a complex interaction of metabolic and regulatory mechanisms influenced by diet and hepatotoxic exposure, underscoring the pathways' relevance to understanding the experimental groups. Integrating these insights furthers our understanding of how specific genes and pathways contribute to the physiological responses observed in this study.

This study's clinical and research implications showcase the possibility of treating metabolic disorders, liver diseases, and cancers by focusing on genes such as Pparg, Rb1, Wnt1 and Tp53. Pparg's involvement in circadian rhythms presents possibilities for chrononutrition and chronotherapy, whereas Tp53 and Rb1, Wnt1 could offer insights for addressing hepatotoxicity or cellular stress. Future studies should prioritize investigations into gene interactions with circadian rhythms through time-series analysis, explore the effects of diet on Wnt1, Tp53 and Rb1 through mechanistic research, and explore into other genes and pathways associated with diet-induced physiological alterations. This will enhance our comprehension and could result in innovative treatment methods.

References

Ibm. (2024, August 15). Decision tree. IBM. https://www.ibm.com/topics/decision-trees

Ibm. (2024, October 25). Random Forest. IBM. https://www.ibm.com/topics/random-forest

Wenric, S., & Shemirani, R. (2018). Using supervised learning methods for gene selection in RNA-Seq Case-Control studies. Frontiers in Genetics, 9. https://doi.org/10.3389/fgene.2018.00297

Verma, P., & Shakya, M. (2021). Machine learning model for predicting Major Depressive Disorder using RNA-Seq data: optimization of classification approach. Cognitive Neurodynamics, 16(2), 443–453. https://doi.org/10.1007/s11571-021-09724-8

Feng, C. H., et al. (2022). Multimetric feature selection for analyzing multicategory outcomes of colorectal cancer: Random forest and multinomial logistic regression models. Laboratory Investigation, 102(3), 236-244. https://doi.org/10.1038/s41374-021-00662-x

UniProt. (s. f.). https://www.uniprot.org/uniprotkb/P33568/entry

Rgd. (s. f.). RAT Genome Database. https://rgd.mcw.edu/