Module 4 Technique Practice Report

Muhammad Umer Mirza

College of Professional Studies, Northeastern University Toronto

ALY6040 - Data Mining

Dr. Shahram Sattar

May 05, 2024

Introduction

In Module 4 Technique Practice Assignment, I worked with the Pumpkin Seed dataset, which encompasses a comprehensive list of variables reflecting various attributes of pumpkin seeds. The dataset categorizes these seeds into two distinct classes: 'Çerçevelik' and 'Ürgüp Sivrisi.' These classes represent different varieties of pumpkin seeds, each with unique characteristics that are pivotal in agricultural practices and seed classification. Understanding these differences is crucial for stakeholders in the agriculture sector, as it helps in optimizing seed selection for planting and processing.

In this report, I delve into the dataset by conducting an extensive Exploratory Data Analysis (EDA). This involves computing summary statistics for each variable to identify any correlations and imbalances within the dataset. Following the EDA, I employ the Support Vector Machines (SVM) technique to mine the data. This advanced analytical approach enables the classification and prediction of seed categories based on the measured attributes. The goal is to provide a model that not only accurately classifies pumpkin seeds but also offers insights into the factors that most significantly influence seed classification. This analysis and the resultant model could be pivotal in enhancing decision-making processes related to agricultural planning and seed quality assessment.

Analysis

I started my analysis by loading the necessary libraries in R, which are crucial for various data manipulation, cleaning, visualization, and modeling tasks.

```
# Load packages
library(pacman)
p_load(tidyverse, janitor, lubridate, ggthemes, ggplot2, ggeasy, knitr, kableExtra, psych,
       corrplot, gclus, RColorBrewer, dplyr, caret, pROC, e1071)
```

Figure 1: Loading libraries

The pacman package helps manage other packages, ensuring they are loaded and installed as needed. The tidyverse suite is crucial for data manipulation and visualization, while janitor assists in cleaning tasks like standardizing column names. Date and time operations are handled by lubridate. For visualizations, ggplot2 is used for creating graphics, supplemented by ggthemes and ggeasy for additional styling options. Reporting and tabulation are enhanced with knitr and kableExtra. The psych package supports psychological and psychometric analysis. Correlations and clustering visualizations are managed using corrplot, gclus, and RColorBrewer, with the latter providing attractive color palettes. Data manipulation functions are primarily performed by dplyr. For machine learning, caret offers tools for creating modeling workflows, and pROC is specialized for analyzing ROC curves. Finally, e1071 is used for implementing machine learning algorithms, including the support vector machines critical for this project. These packages collectively form a comprehensive toolkit for detailed data analysis from preparation through to predictive modeling.

As mentioned in Figure 2, I started by loading the Pumpkin Seeds Dataset into the R environment using the read.csv function. Following this, I conducted an initial inspection of the dataset to understand its structure and contents better.

```r
# Load data
psd <- read.csv("Pumpkin_Seeds_Dataset.csv")

# Initial data inspection
cat("Dimensions of data:", dim(psd), "\n")
colnames(psd)
head(psd, n = 10)
str(psd)

# Count unique values for each variable
sapply(psd, function(x) length(unique(x)))
# Count missing values for each variable
sapply(psd, function(x) sum(is.na(x)))

# Check for duplicate entries
cat("Number of duplicate rows:", sum(duplicated(psd)), "\n")
```

Figure 2: Load and inspect data

I began by checking the dimensions of the dataset to know the number of rows and columns it contains. This was followed by a quick review of the column names and the first ten entries of the dataset using the head function, allowing me to get a preliminary idea of the data types and values present. Per figure 3 it appears that the dataset contains 2500 rows and 13 columns.

```
> cat("Dimensions of data:", dim(psd), "\n")
Dimensions of data: 2500 13
> colnames(psd)
 [1] "Area"             "Perimeter"         "Major_Axis_Length" "Minor_Axis_Length" "Convex_Area"
 [6] "Equiv_Diameter"   "Eccentricity"      "Solidity"          "Extent"            "Roundness"
[11] "Aspect_Ration"    "Compactness"       "Class"
```

Figure 3: Dataset dimension and column names

To understand the dataset's integrity and detail further, I performed a structure check using the str function, which provides a concise summary of the data types and the number of

non-missing values in each column. According to figure 4 it appears that majority of the variables are numerical.

```
> str(psd)
'data.frame':   2500 obs. of  13 variables:
 $ Area             : int  56276 76631 71623 66458 66107 73191 73338 69692 95727 73465 ...
 $ Perimeter        : num  888 1068 1083 992 998 ...
 $ Major_Axis_Length: num  326 417 436 382 384 ...
 $ Minor_Axis_Length: num  220 234 211 223 220 ...
 $ Convex_Area      : int  56831 77280 72663 67118 67117 73969 73859 70442 96831 74089 ...
 $ Equiv_Diameter   : num  268 312 302 291 290 ...
 $ Eccentricity     : num  0.738 0.828 0.875 0.812 0.819 ...
 $ Solidity         : num  0.99 0.992 0.986 0.99 0.985 ...
 $ Extent           : num  0.745 0.715 0.74 0.74 0.675 ...
 $ Roundness        : num  0.896 0.844 0.767 0.849 0.834 ...
 $ Aspect_Ration    : num  1.48 1.78 2.07 1.71 1.74 ...
 $ Compactness      : num  0.821 0.749 0.693 0.762 0.756 ...
 $ Class            : chr  "Çerçevelik" "Çerçevelik" "Çerçevelik" "Çerçevelik" ...
```

Figure 4: Structure of the dataset

Ensuring data quality, I then assessed the uniqueness of the entries in each column with the sapply function, which applied the length and unique functions to count unique values in each column. This helps in identifying columns that may have a high degree of redundancy or very few unique data points, which could influence the analysis.

```
> # Count unique values for each variable
> sapply(psd, function(x) length(unique(x)))
            Area        Perimeter Major_Axis_Length Minor_Axis_Length       Convex_Area
            2424             2490              2499              2497              2445
  Equiv_Diameter      Eccentricity         Solidity            Extent         Roundness
            2424             1295               166              1392              1480
   Aspect_Ration       Compactness             Class
            2237             1405                 2
```

Figure 5: Number of unique values in each variable

Additionally, I checked for missing values across the dataset by applying another sapply function that sums up all NA values per column, ensuring that the subsequent analysis would not be skewed by missing data. Per figure 6 it appears that there are no missing values in the dataset.

```
> # Count missing values for each variable
> sapply(psd, function(x) sum(is.na(x)))
             Area         Perimeter Major_Axis_Length Minor_Axis_Length       Convex_Area
                0                 0                 0                 0                 0
   Equiv_Diameter      Eccentricity          Solidity            Extent         Roundness
                0                 0                 0                 0                 0
    Aspect_Ration       Compactness             Class
                0                 0                 0
```

Figure 6: Number of missing/NA values

Finally, to ensure the dataset did not contain duplicate records that could distort analytical results, I calculated the number of duplicate rows. The presence of duplicates could potentially lead to biased statistical analyses and was therefore crucial to identify early in the preprocessing stage. No duplicate rows were found in the dataset.

```
> # Check for duplicate entries
> cat("Number of duplicate rows:", sum(duplicated(psd)), "\n")
Number of duplicate rows: 0
```

Figure 7: Number of duplicate rows

These steps formed the foundation of my data quality assessment and initial exploration, setting the stage for more detailed analysis and modeling.

```
# Data Cleaning
# Clean column names with janitor
psd <- psd %>% janitor::clean_names()

# Convert the class column as factor
psd$class <- factor(psd$class)

# Check structure to confirm data cleaning changes
str(psd)
```

Figure 8: Data cleaning steps

In Figure 8, the data cleaning process for the Pumpkin Seeds Dataset is outlined, emphasizing the steps taken to ensure the data is properly formatted and ready for analysis.

The first step involved using the janitor::clean_names() function. This function standardizes the column names by converting them to lower case, replacing spaces with underscores, and removing any special characters. This step is crucial for preventing issues related to column name formatting that could potentially disrupt subsequent data manipulations and analyses.

Next, the class column, which contains the categorical labels for each seed type, was converted to a factor data type using the factor() function. Converting categorical columns to factor is essential in R for statistical modeling as it correctly defines the data's structure for functions and models that differentiate between numerical and categorical inputs.

Finally, I used the str() function to check the structure of the dataset again. This helps verify that the cleaning steps were correctly applied and allows for a quick review of the entire dataset's structure post-cleaning, confirming that the data types and column names are now appropriate for further analysis.

```
> # Check structure to confirm data cleaning changes
> str(psd)
'data.frame':   2500 obs. of  13 variables:
 $ area             : int  56276 76631 71623 66458 66107 73191 73338 69692 95727 73465 ...
 $ perimeter        : num  888 1068 1083 992 998 ...
 $ major_axis_length: num  326 417 436 382 384 ...
 $ minor_axis_length: num  220 234 211 223 220 ...
 $ convex_area      : int  56831 77280 72663 67118 67117 73969 73859 70442 96831 74089 ...
 $ equiv_diameter   : num  268 312 302 291 290 ...
 $ eccentricity     : num  0.738 0.828 0.875 0.812 0.819 ...
 $ solidity         : num  0.99 0.992 0.986 0.99 0.985 ...
 $ extent           : num  0.745 0.715 0.74 0.74 0.675 ...
 $ roundness        : num  0.896 0.844 0.767 0.849 0.834 ...
 $ aspect_ration    : num  1.48 1.78 2.07 1.71 1.74 ...
 $ compactness      : num  0.821 0.749 0.693 0.762 0.756 ...
 $ class            : Factor w/ 2 levels "Çerçevelik","Ürgüp Sivrisi": 1 1 1 1 1 1 1 1 1 1 ...
```

Figure 9: Cleaned dataset structure

These cleaning steps are critical for ensuring that the dataset does not contain syntactical errors and is organized in a way that aligns with good data analysis practices. Such preparations help minimize errors in the modeling phase and improve the clarity and efficiency of the code.

In Figure 10, as part of my Exploratory Data Analysis (EDA), I focused on summarizing the numerical variables in the Pumpkin Seeds Dataset.

```
# EDA

# Summary statistics for numerical variables
# List of numerical variables in the psd dataset
numerical_vars <- c("area", "perimeter", "major_axis_length", "minor_axis_length",
                    "convex_area", "equiv_diameter", "eccentricity", "solidity",
                    "extent", "roundness", "aspect_ration", "compactness")

# Function to calculate and round summary statistics
calculate_stats <- function(data, var) {
  stats <- c(
    Min = round(min(data[[var]], na.rm = TRUE), 2),
    Max = round(max(data[[var]], na.rm = TRUE), 2),
    Mean = round(mean(data[[var]], na.rm = TRUE), 2),
    SD = round(sd(data[[var]], na.rm = TRUE), 2),
    Median = round(median(data[[var]], na.rm = TRUE), 2),
    IQR = round(IQR(data[[var]], na.rm = TRUE), 2)
  )
  return(stats)
}

# Apply the function to each numerical variable and store results
stats_list <- lapply(numerical_vars, calculate_stats, data = psd)

summary_stats <- tibble(
  Variable = numerical_vars,
  Min = sapply(stats_list, "[", "Min"),
  Max = sapply(stats_list, "[", "Max"),
  Mean = sapply(stats_list, "[", "Mean"),
  SD = sapply(stats_list, "[", "SD"),
  Median = sapply(stats_list, "[", "Median"),
  IQR = sapply(stats_list, "[", "IQR")
)

# Print and create a table using kable
print(summary_stats)
kable(summary_stats, caption = "Summary Statistics for Numerical Variables in Pumpkin Seed Dataset",
  kable_classic(full_width = F, html_font = "Cambria")  %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

Figure 10: Summary statistics table for numerical variables

I started by creating a custom function named calculate_stats that computes and rounds off various summary statistics for each numerical variable: minimum, maximum, mean, standard deviation, median, and interquartile range (IQR). These statistics are essential for understanding the distribution and scale of each feature in the dataset.

After defining the list of numerical variables, I applied the calculate_stats function to each of these using the lapply function. This ensured that I had a comprehensive set of statistics for each variable, which I then organized into a tidy data frame using the tibble function for easier readability and manipulation.

Finally, I used the kable function from the kableExtra package to create a well-formatted table of these summary statistics. This table, titled "Summary Statistics for Numerical Variables in Pumpkin Seed Dataset," provides a clear and professional presentation of the data, highlighting important statistical details in a format suitable for inclusion in reports or presentations. This approach not only aids in the visualization of data characteristics but also prepares the groundwork for further analysis and model building.

Figure 11 shows a detailed summary of the distribution for each numerical variable in the dataset. This table provides insights into the central tendency and dispersion of data across different seed characteristics, crucial for understanding the diversity and patterns in pumpkin seeds.

Summary Statistics for Numerical Variables in Pumpkin Seed Dataset

Variable	Min	Max	Mean	SD	Median	IQR
area	47939.00	136574.00	80658.22	13664.51	79076.00	18992.50
perimeter	868.48	1559.45	1130.28	109.26	1123.67	154.51
major_axis_length	320.84	661.91	456.60	56.24	449.50	77.78
minor_axis_length	152.17	305.82	225.79	23.30	224.70	29.43
convex_area	48366.00	138384.00	81508.08	13764.09	79872.00	19285.75
equiv_diameter	247.06	417.00	319.33	26.89	317.31	37.89
eccentricity	0.49	0.95	0.86	0.05	0.86	0.07
solidity	0.92	0.99	0.99	0.00	0.99	0.00
extent	0.47	0.83	0.69	0.06	0.71	0.08
roundness	0.55	0.94	0.79	0.06	0.80	0.08
aspect_ration	1.15	3.14	2.04	0.32	1.98	0.46
compactness	0.56	0.90	0.70	0.05	0.71	0.08

Figure 11: Summary statistics

Both 'area' and 'convex area' show large ranges with values spanning from around 47,939 to 1,365,740 and 48,366 to 1,383,840 respectively. The median and mean are closely aligned, indicating a relatively symmetric distribution despite the broad spread, as shown by the high standard deviations and IQRs.

Perimeter, Major Axis Length, and Minor Axis Length variables also exhibit large variances in their values but maintain a near-symmetry between the mean and median. The

minor axis length has the least variability among them, suggesting more consistency in this particular dimension of the seeds.

Equiv Diameter and Eccentricity both show smaller ranges and lower variability compared to size measurements, with means and medians nearly identical, indicating symmetric distributions. Eccentricity's low standard deviation and IQR point to tight clustering around the mean.

Solidity, Extent, and Roundness characteristics display tight distributions with very low standard deviations and IQRs, and the median approximating the mean closely, indicating consistent measurements across the samples.

Aspect Ratio and Compactness variables show slight deviation between mean and median, suggesting minor skewness in their distributions. Their moderate IQRs relative to their means suggest a modest spread around the central value.

Overall, the similarity of the mean and median in most variables suggests that the data does not have significant skewness. The wide ranges in variables related to size (area, perimeter, axis lengths) reflect considerable variability in physical dimensions among the pumpkin seeds.

In Figure 12, I generated a bar chart to visually examine the distribution of pumpkin seed classes in the dataset.

```r
# Bar chart for the categorical variable class

# Reorder class variable based on count
psd$class <- factor(psd$class,
                            levels = names(sort(table(psd$class), decreasing = FALSE)))
ggplot(psd, aes(x = class)) +
  geom_bar(fill = "steelblue") +
  labs(x = "Class", y = "Frequency", title = "Frequency of Each Class in PSD Dataset",
       caption = "Data source: https://doi.org/10.1007/s10722-021-01226-0") +
  coord_flip() +  # Flips the axes to put class on the y-axis
  theme_bw() +  # Using black and white theme
  theme(axis.title = element_text(color = "black", face = "bold"),
        axis.text = element_text(color = "black"),
        axis.ticks = element_blank(),  # Removing axis ticks
        plot.title = element_text(face = "bold", hjust = 0.5, color = "black", size = 16),
        panel.grid = element_blank(),  # Removing grid lines
        panel.background = element_rect(fill = "white", colour = NA))  # White background
```

Figure 12: Bar chart for categorical variable

The bar chart was created using ggplot2, with the geom_bar function, and the bars were filled in a striking steel blue color to enhance visual appeal. I reordered the 'class' variable based on frequency, ensuring that the classes are displayed in ascending order of their counts. This makes it easier to compare the relative frequencies at a glance.

The chart employs a coordinate flip (coord_flip()) which places the class categories on the y-axis and their frequencies on the x-axis, providing a clear and easy-to-read horizontal bar chart. I applied the theme_bw() for a clean, black and white theme, which helps highlight the data without visual distractions.

Overall, the bar chart effectively displays the frequency of each class within the Pumpkin Seed Dataset, allowing stakeholders to quickly assess which classes are more or less prevalent. This visualization is an essential tool for understanding the dataset's composition before delving deeper into more complex analyses.

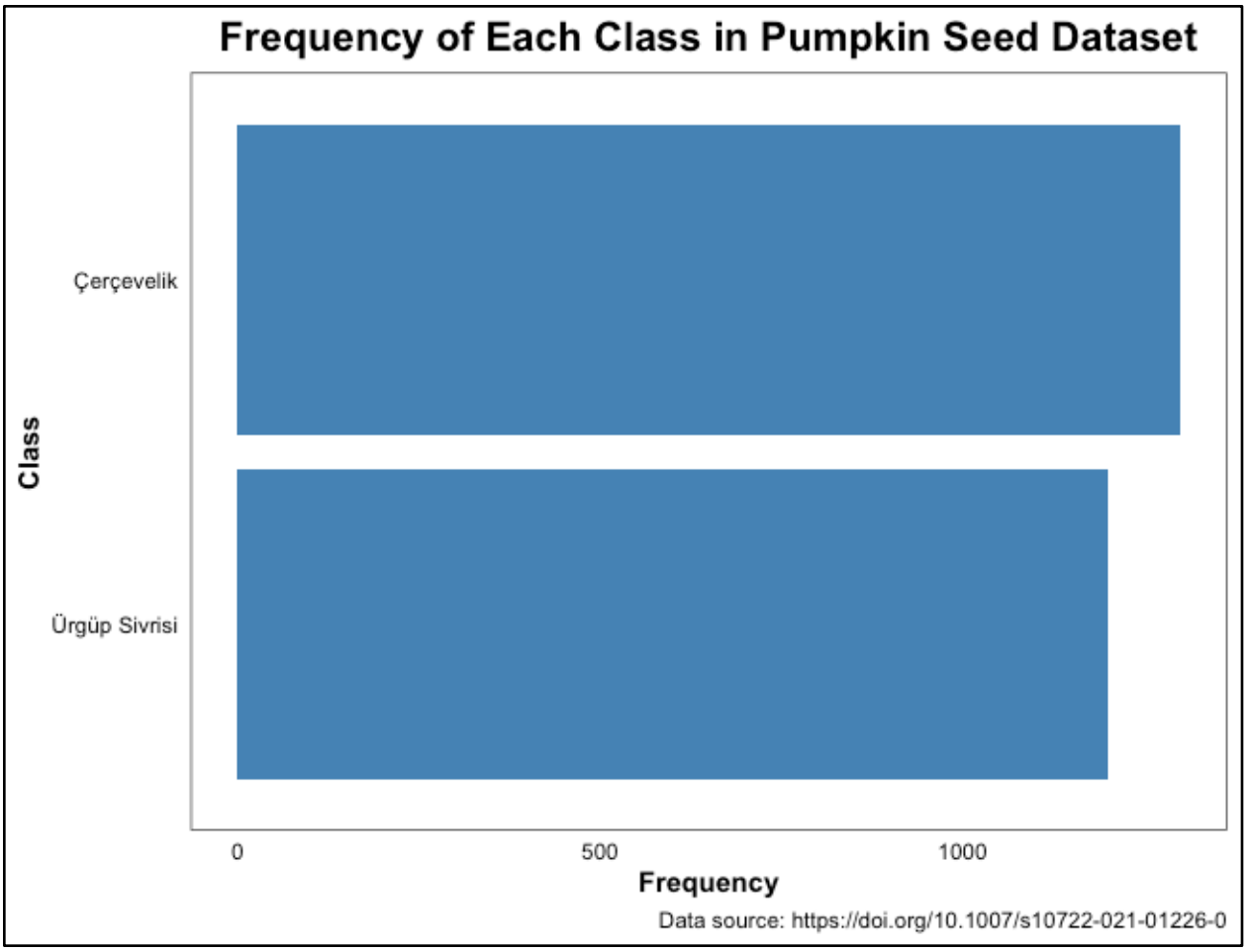

Figure 13: Class frequency

In Figure 13, the bar chart illustrates the frequency of each class in the Pumpkin Seed Dataset. The classes, 'Çerçevelik' and 'Ürgüp Sivrisi', are represented by horizontal bars, demonstrating their respective counts within the dataset.

Both classes appear to be nearly equally represented, with 'Çerçevelik' showing a slightly higher count compared to 'Ürgüp Sivrisi'. This near-equal representation is advantageous for predictive modeling as it reduces the likelihood of class imbalance that could potentially bias the model's performance. A balanced dataset ensures that the predictive model developed from this data is less likely to overfit to the more frequent class and can generalize better to new, unseen

data. This balanced distribution is crucial for maintaining the integrity and accuracy of classification models, making the insights derived from such analyses more reliable and robust.

In Figure 14, the correlation matrix visually presents the relationships between various numerical variables in the Pumpkin Seed Dataset. The matrix uses color intensity and numerical values to indicate the strength and direction of the correlations between variables.

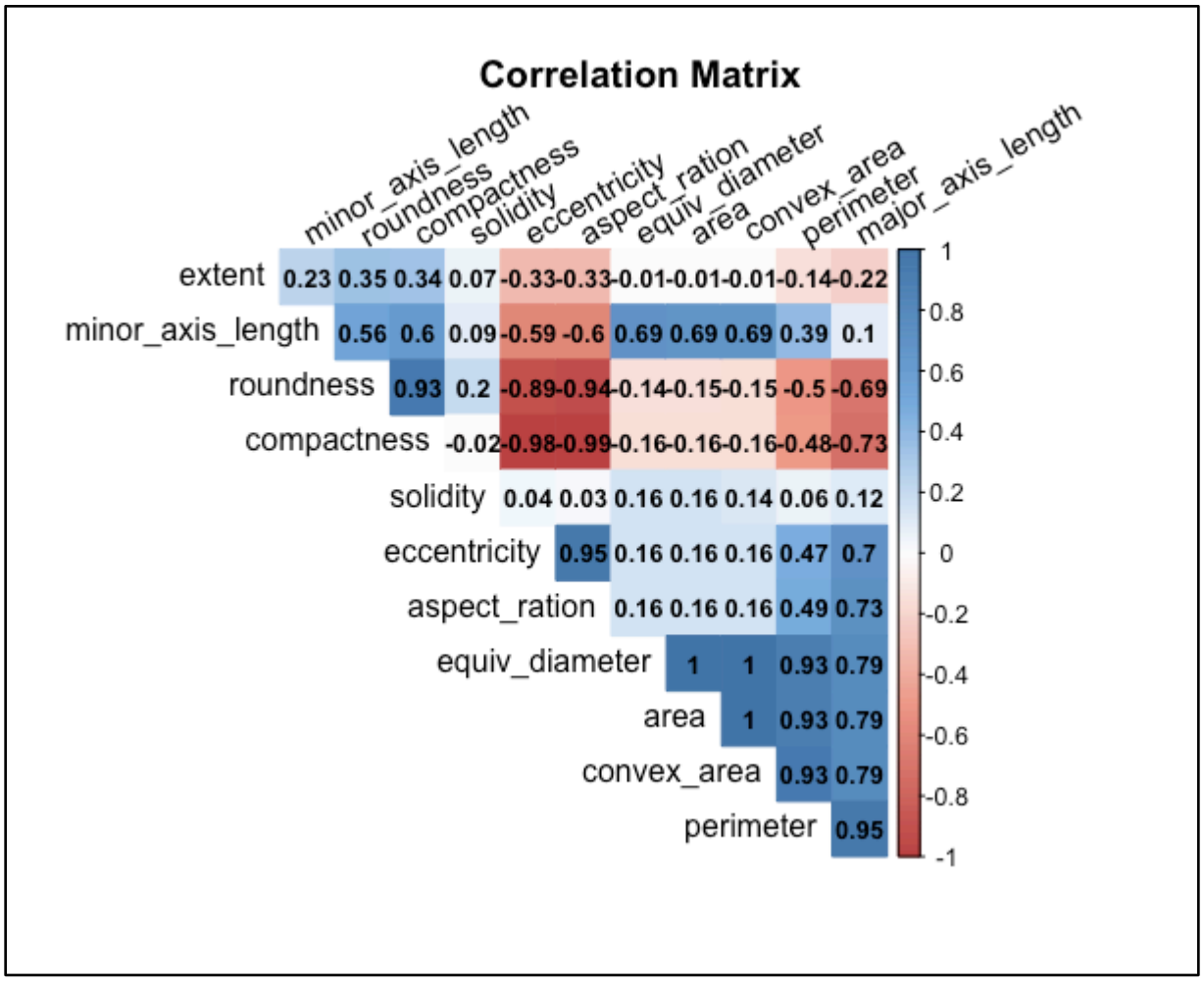

Figure 14: Correlation Matrix

Variables such as 'perimeter', 'area', 'equiv_diameter', and 'convex_area' show very high correlations with each other (values close to 1), suggesting that these measurements, which are all related to the size and shape of the seeds, are interdependent.

The 'roundness' shows a strong negative correlation with 'major_axis_length' (around -0.89), indicating that as seeds become longer in their major axis, they tend to be less round.

'Eccentricity' is strongly positively correlated with 'aspect_ratio' and 'major_axis_length', and shows varying degrees of negative correlation with 'solidity' and 'roundness', which aligns with the expected geometric properties of seeds where more elongated seeds tend to be less solid and round.

These correlations are critical for understanding the relationships between different seed characteristics and can guide further analyses, such as feature selection for predictive modeling, ensuring that redundant features are identified and managed appropriately.

In Figure 15, I proceed with the data mining phase of my analysis using the Support Vector Machine (SVM) technique, a powerful machine learning method suited for classification tasks.

```
# Data Mining
# Support Vector Machine (SVM)
# Load the necessary library
#library(e1071)

# Train SVM model using all features
m1 <- svm(class ~ ., data = psd)
summary(m1)  # Display the summary of the trained model

# Generate confusion matrix
table1 <- table(psd$class, predict(m1))
print(table1)  # Print the confusion matrix

# Calculate the classification rate
classification_rate <- sum(diag(table1)) / sum(table1)
print(classification_rate)  # Print the classification rate

# Graphical output to visualize the decision boundary for two features
plot(m1, psd, area ~ perimeter)

plot(m1, psd, area ~ roundness)

plot(m1, psd, area ~ compactness)
```

Figure 15: Data mining steps

I utilized the svm function from the e1071 package to train the SVM model using all available features in the dataset. The model is trained to classify the pumpkin seeds into their respective classes (Çerçevelik and Ürgüp Sivrisi).

After training, I displayed the summary of the SVM model using the summary(m1) command. This summary provides insights into the model's structure, including the type of SVM kernel used, the number of support vectors, and other relevant model parameters.

To evaluate the model's performance, I generated a confusion matrix using the table function, which compares the actual class labels with the predicted ones from the model. Printing

this matrix helps in visually assessing how well the model is performing in terms of correctly classifying the seeds.

I calculated the overall classification rate of the model by summing the diagonal elements of the confusion matrix (which represent correct classifications) and dividing by the total number of observations. This metric is crucial for understanding the effectiveness of the model in practical terms.

Lastly, I used plots to visualize the decision boundaries created by the SVM for different pairs of features (area vs. perimeter, area vs. roundness, and area vs. compactness). These visualizations are particularly useful for seeing how the SVM model separates the classes based on combinations of two features at a time.

Overall, these steps encapsulate the application of SVM to the Pumpkin Seeds Dataset, illustrating not only the classification process but also the evaluation of the model's performance in predicting seed classes.

Figure 16 displays the summary of the SVM (Support Vector Machine) model trained on the Pumpkin Seeds Dataset.

```
> summary(m1)  # Display the summary of the trained model

Call:
svm(formula = class ~ ., data = psd)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  743

 ( 374 369 )


Number of Classes:  2

Levels:
 Ürgüp Sivrisi Çerçevelik
```

Figure 16: Summary of SVM model

The model utilizes C-classification, a common choice for binary classification tasks. This method focuses on finding a hyperplane that separates the data with the maximum margin while penalizing misclassifications.

The kernel used is radial (also known as Radial Basis Function or RBF), which is effective for handling cases where the relationship between class labels and attributes is non-linear. The radial kernel can map the inputs into a higher-dimensional space, making it easier to find a linear separation.

The cost parameter is set to 1, balancing the trade-off between achieving a low error on the training data and minimizing model complexity for better generalization. This parameter can be tuned to optimize performance, particularly to adjust how much the model penalizes misclassification.

The model has 743 support vectors, indicating the instances from the training data that define the decision boundary. Among these, 374 are for one class and 369 for the other, suggesting a fairly balanced influence from both classes on the model's decision-making process.

There are two classes being predicted: 'Ürgüp Sivrisi' and 'Çerçevelik'. The representation of these classes in terms of support vectors shows that the model considers a substantial amount of information from both categories, which is crucial for maintaining accuracy across diverse seed types.

The summary provides valuable insights into how the SVM model was structured and parameterized, giving us a clear view of its complexity and the strategies used for classification. This information is crucial for understanding model behavior and for making any necessary adjustments to improve its predictive performance.

In Figure 17, shows the confusion matrix generated from the SVM model applied to the Pumpkin Seeds Dataset. This matrix is a powerful tool for evaluating the performance of the classification model by showing the actual versus predicted classifications.

```
> # Generate confusion matrix
> table1 <- table(psd$class, predict(m1))
> print(table1)  # Print the confusion matrix

                Ürgüp Sivrisi Çerçevelik
Ürgüp Sivrisi            1021        179
Çerçevelik                 92       1208
```

Figure 17: Confusion matrix

True Positives (TP) for 'Ürgüp Sivrisi': 1021 instances were correctly classified as 'Ürgüp Sivrisi'. False Negatives (FN) for 'Ürgüp Sivrisi': 92 instances were incorrectly classified as 'Çerçevelik'. True Positives (TP) for 'Çerçevelik': 1208 instances were correctly classified as 'Çerçevelik'. False Positives (FP) for 'Ürgüp Sivrisi': 179 instances were incorrectly classified as 'Ürgüp Sivrisi'.

The confusion matrix allows us to calculate key performance metrics such as precision, recall, and the overall accuracy of the model.

Precision for 'Ürgüp Sivrisi' can be calculated as TP / (TP + FP) = 1021 / (1021 + 179) which comes out to be 85.08%. Recall for 'Ürgüp Sivrisi' is TP / (TP + FN) = 1021 / (1021 + 92) which is about 91.73%.

Overall, the matrix shows that the SVM model performs robustly in classifying both classes with more correct predictions than incorrect. However, the presence of misclassifications, particularly the 179 instances where 'Çerçevelik' was incorrectly predicted as 'Ürgüp Sivrisi', suggests areas where the model might be improved, possibly by tuning hyperparameters or by further feature engineering.

In Figure 18, the classification rate of the SVM model applied to the Pumpkin Seeds Dataset is displayed.

```
> # Calculate the classification rate
> classification_rate <- sum(diag(table1)) / sum(table1)
> print(classification_rate)  # Print the classification rate
[1] 0.8916
```

Figure 18: Classification rate

The classification rate here is 0.8916, indicating that approximately 89.16% of the total predictions made by the model were correct. This rate is a direct measure of the model's accuracy, reflecting how effectively the model can classify the pumpkin seed types into 'Ürgüp Sivrisi' and 'Çerçevelik'. A classification rate of over 89% is generally considered good, showing that the SVM model has performed well in distinguishing between the two classes based on the given features.

However, while this rate shows high accuracy, there is still room for improvement, especially considering the potential impact of false positives and false negatives on the practical application of such a model in a real-world setting. Fine-tuning the model's parameters or incorporating additional or more discriminative features could potentially increase this accuracy further.

Figure 19, 19.1 and 19.2 consists of three SVM classification plots that visually represent the decision boundaries between the two classes of pumpkin seeds ('Çerçevelik' and 'Ürgüp Sivrisi') based on different pairs of features.

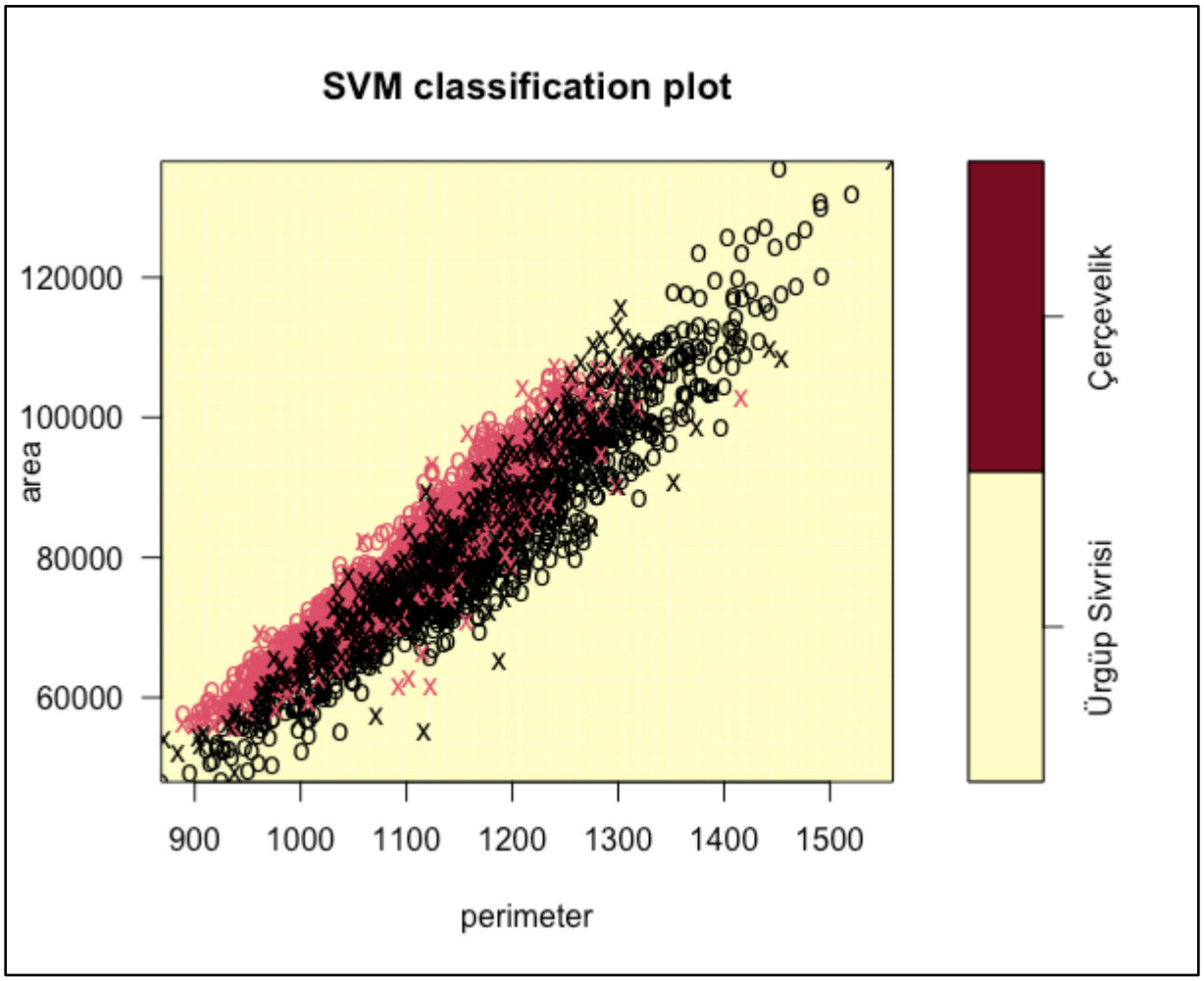

Figure 19: Classification plot for area and perimeter features

The first plot shows the distribution of seeds based on 'area' and 'perimeter'. A clear linear relationship is evident, with the SVM model creating a boundary that efficiently separates the two classes along these dimensions. The plot shows that as both area and perimeter increase, the likelihood of a seed belonging to a particular class changes.

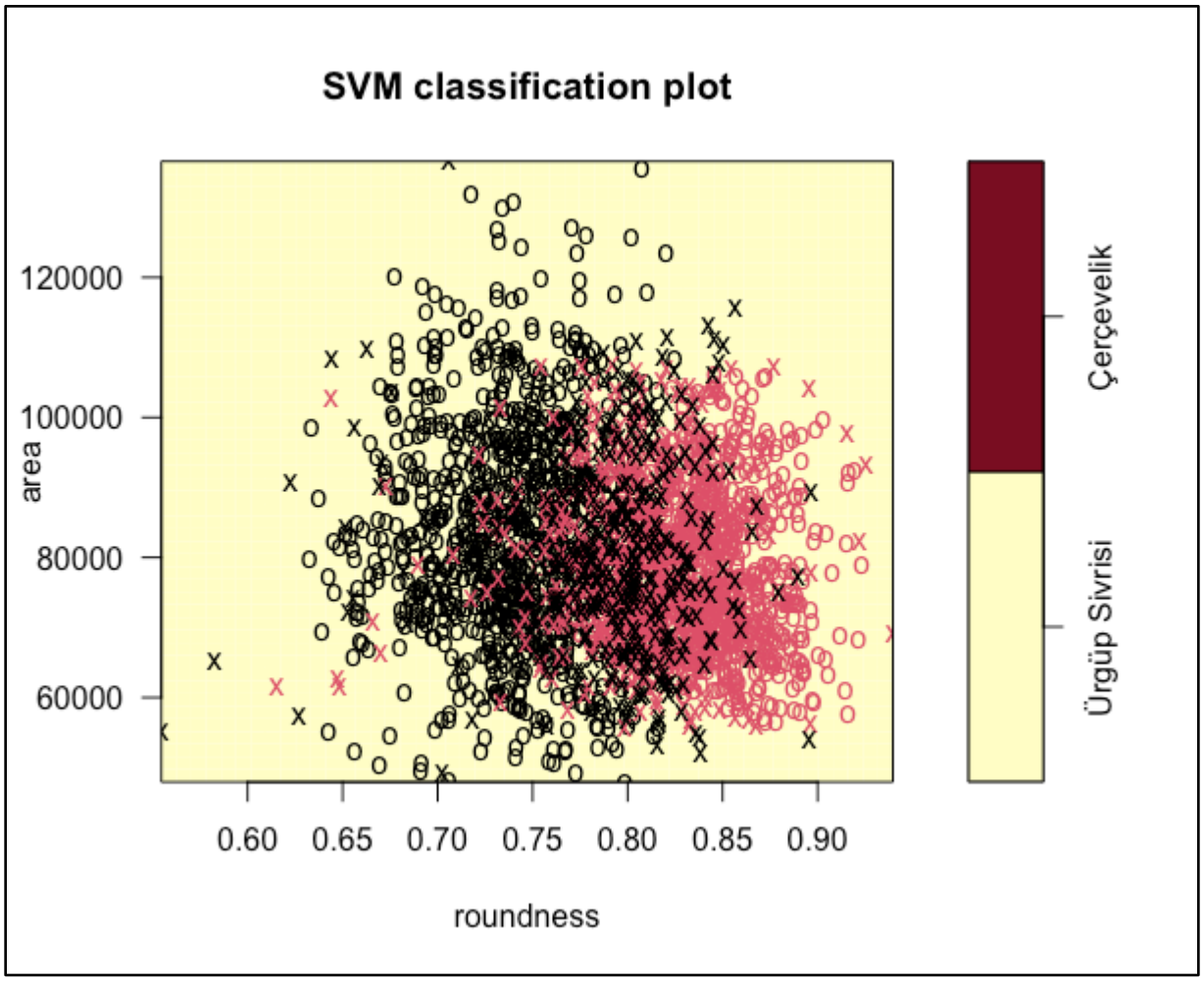

Figure 19.1: Classification plot for area and roundness features

The second plot displays seeds categorized by 'area' and 'roundness'. Here, the separation between the two classes is less distinct than in the first plot, indicating that these features alone provide a less clear basis for classification by the SVM model. The overlap suggests that roundness, across varying areas, does not distinctly differentiate the classes.

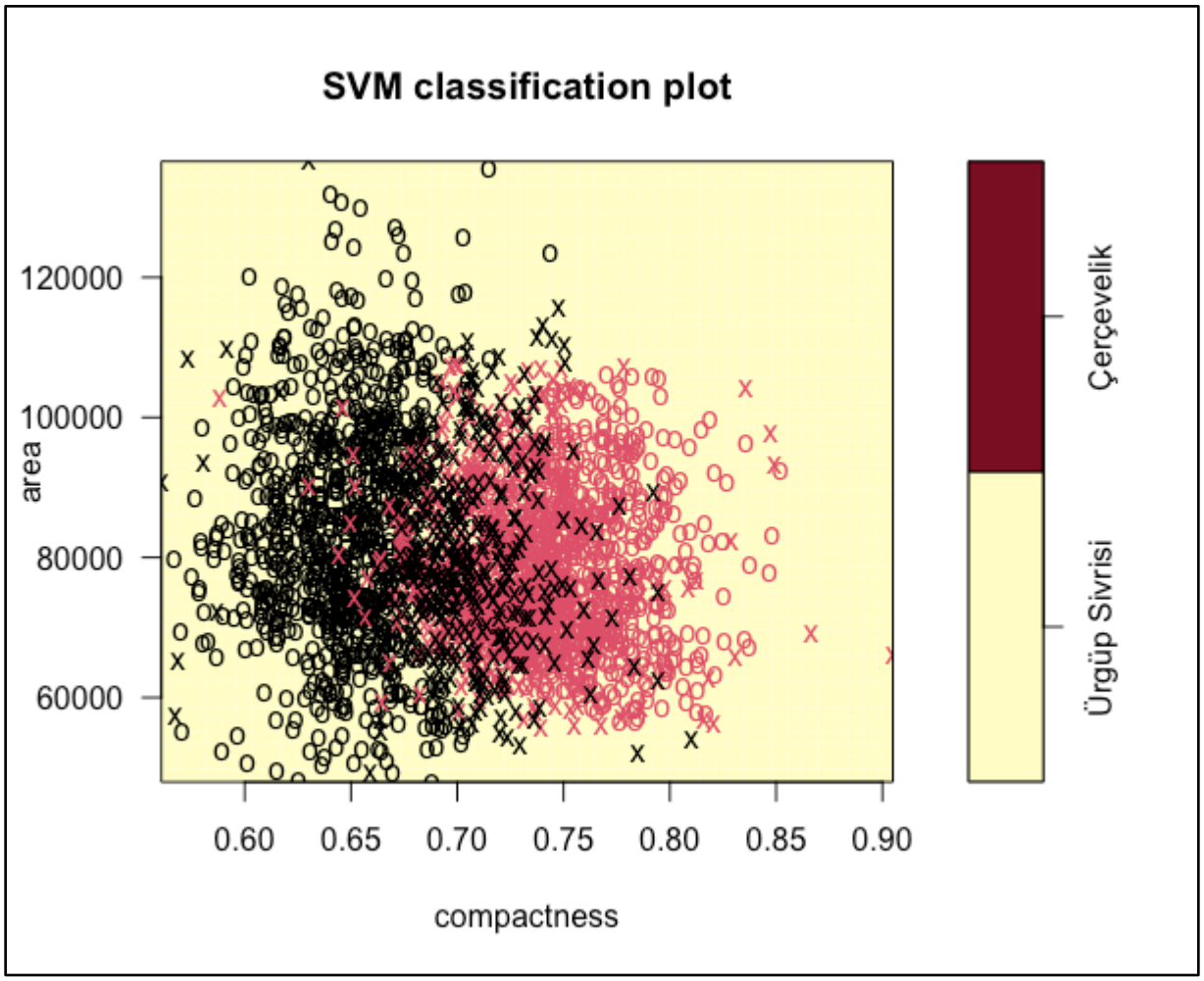

Figure 19.2: Classification plot for area and compactness features

The final plot uses 'area' and 'compactness'. Similar to the plot with roundness, this graph shows considerable overlap between the two classes, indicating that compactness combined with area doesn't sharply delineate 'Çerçevelik' from 'Ürgüp Sivrisi'.

These plots highlight how different feature combinations affect the SVM's ability to classify the data. While some features like perimeter with area provide clear separation cues, others like roundness and compactness show more overlap, suggesting that these features might be less discriminative on their own for classifying pumpkin seed types.

Figure 20 presents a function I developed to fine-tune the gamma parameter in an SVM model and evaluate its performance across different kernels. This function is based on insights from Baranyai, L. (2021) who provided guidance on SVM kernel and gamma selection in R.

```r
# Function to tune gamma and evaluate performance across different kernels
tune.gamma <- function(scale = 0.5) {
  Kernels <- c("linear", "polynomial", "radial", "sigmoid")
  gdv <- 1/12  # Default gamma value
  gDefault <- numeric(length(Kernels))  # Store default gamma results
  gLow <- numeric(length(Kernels))      # Store lower gamma results
  gHigh <- numeric(length(Kernels))     # Store higher gamma results

  # Loop through each kernel type and adjust gamma
  for (i in seq_along(Kernels)) {
    # Test default gamma
    tmp <- svm(class ~ ., data = psd, kernel = Kernels[i], gamma = gdv)
    tbl <- table(psd$class, predict(tmp))
    gDefault[i] <- sum(diag(tbl)) / sum(tbl)

    # Test gamma scaled down
    tmp <- svm(class ~ ., data = psd, kernel = Kernels[i], gamma = gdv - scale * gdv)
    tbl <- table(psd$class, predict(tmp))
    gLow[i] <- sum(diag(tbl)) / sum(tbl)

    # Test gamma scaled up
    tmp <- svm(class ~ ., data = psd, kernel = Kernels[i], gamma = gdv + scale * gdv)
    tbl <- table(psd$class, predict(tmp))
    gHigh[i] <- sum(diag(tbl)) / sum(tbl)
  }

  # Return a data frame of results for each kernel
  return(data.frame(Kernels, gLow, gDefault, gHigh))
}
```

Figure 20: Model tunning

The function's aim is to adjust the gamma parameter, which defines how much influence a single training example has. The goal is to find the optimal gamma that improves model accuracy for different kernel types.

It tests four types of kernels: 'linear', 'polynomial', 'radial', and 'sigmoid'. Each kernel represents a different way of handling the data in the feature space, from linear separations to more complex, non-linear boundaries.

The function starts with a default gamma (gdv = 1/12) and tests the classification performance of the model at this gamma, a gamma scaled down by 50% (scale = 0.5), and a gamma scaled up by the same factor.

For each kernel and gamma setting, it trains an SVM model, predicts classifications, and then computes a classification rate by comparing these predictions to the actual data. This is done through the creation of a confusion matrix and calculating the sum of the diagonal (correct predictions) divided by the total predictions.

This systematic approach allows for an empirical evaluation of how sensitive the SVM model's performance is to changes in the gamma parameter under different kernels. It helps in identifying the best combination of kernel and gamma for the given dataset, ensuring that the chosen model offers the highest possible accuracy.

Figure 21 displays the results of gamma tuning performed using different SVM kernels on the Pumpkin Seeds Dataset. This table illustrates how the performance, measured as classification rate, varies with changes in the gamma parameter for each kernel type.

```
> # Execute gamma tuning
> gamma_tuning_results <- tune.gamma(0.5)
> print(gamma_tuning_results)
      Kernels   gLow gDefault  gHigh
1      linear 0.8812   0.8812 0.8812
2 polynomial 0.8404   0.8592 0.8652
3      radial 0.8900   0.8916 0.8928
4     sigmoid 0.8156   0.7884 0.7704
```

Figure 21: Model tunning results

Linear Kernel classification rate remains unchanged at 0.8812 across different gamma values. The linear kernel does not use the gamma parameter, which explains why there is no variation in performance with changes in gamma. It only depends on the data's linear separability.

Polynomial Kernel achieves the highest classification rate of 0.8592 at the default gamma. Performance decreases when gamma is increased (0.8652) and also drops when decreased (0.8404). This suggests that the default gamma is near optimal for the polynomial kernel on this dataset.

Radial Kernel best performance at the default gamma setting, it achieves the highest classification rate of 0.8916. The classification rate for the radial kernel shows a slight increase to 0.8928 when the gamma value is increased. Conversely, when gamma is decreased, the classification rate slightly decreases to 0.8900. This indicates that the model's performance is sensitive to changes in gamma, improving slightly with an increase, which suggests that the

influence of individual support vectors becomes more finely tuned to the distinctions in the dataset at a higher gamma value.

Sigmoid Kernel has a classification rate of 0.7884 at the default gamma. This rate decreases further when gamma is increased (0.7704) and increases slightly when gamma is decreased (0.8156). This suggests that lowering gamma slightly from the default might improve performance for the sigmoid kernel.

These results underscore the crucial role of selecting the appropriate kernel and gamma settings in SVM models. Although the radial kernel performs well at the default gamma, it achieves the highest classification rate when the gamma is slightly increased. This indicates that fine-tuning the gamma value can enhance the model's ability to capture complex patterns in the data. Conversely, the performance of other kernels like polynomial and sigmoid can vary significantly with different gamma settings, showing that adjustments in gamma can lead to improvements or deteriorations in model accuracy. Optimal settings are thus highly dependent on the specific characteristics of the dataset and the kernel used, demonstrating the importance of careful parameter optimization in SVM models.

Conclusion

In this report, we embarked on a comprehensive exploration and analysis of the Pumpkin Seed dataset, using advanced data mining techniques with an emphasis on Support Vector Machines (SVM). This journey began with meticulous data preparation, including cleaning and initial data assessments to ensure the integrity and usability of the dataset for complex analytical tasks.

Through exploratory data analysis, we gained valuable insights into the distribution and relationships among various seed attributes, enabling us to understand the underlying patterns

and potential predictive factors. The use of SVM allowed us to model these relationships effectively, achieving a high classification accuracy of over 89%, demonstrating the model's robustness in distinguishing between the 'Çerçevelik' and 'Ürgüp Sivrisi' seed classes.

The model tuning phase was crucial, revealing the sensitivity of SVM performance to kernel choice and gamma settings. Adjustments in gamma particularly showed significant impacts on the accuracy of different kernels, highlighting the need for precise parameter optimization based on the dataset's characteristics.

Looking ahead, there are several avenues for further research and improvement. Enhancing the model by exploring additional or more discriminative features could potentially increase its accuracy. Moreover, experimenting with different machine learning algorithms might offer new insights or better performance. Lastly, implementing cross-validation methods would provide a more robust evaluation of the model's predictive power and generalizability.

This study not only underscores the effectiveness of SVM in classifying complex datasets but also illustrates the importance of careful model tuning and feature selection in achieving optimal results. Future work will continue to refine these approaches, further enhancing the decision-making processes in agricultural practices and beyond.

References

1. Koklu, M., Sarigil, S., & Ozbek, O. (2021). The use of machine learning methods in classification of pumpkin seeds (Cucurbita pepo L.). Genetic Resources and Crop Evolution, 68(7), 2713-2726. Doi: https://doi.org/10.1007/s10722-021-01226-0
2. Secrets of Analytical Leaders: Insights from Information Insiders by Wayne Eckerson (ISBN – 978-1935504344)
3. Sattar, S. (2024). Module 4. Canvas. https://northeastern.instructure.com/courses/174017/modules
4. Baranyai, L. (2021). [EN] R Statistics: Support Vector Machine (SVM) kernel and gamma selection. Www.youtube.com. https://www.youtube.com/watch?v=Bwb_--gVEzY
5. Support Vector Machine · UC Business Analytics R Programming Guide. (n.d.). Uc-R.github.io. https://uc-r.github.io/svm