SMOTE-Based Balancing and Decision Tree Modeling on Iris Dataset

Introduction

In this document, we will: - Balance the Iris dataset by artificially oversampling minority classes. - Apply SMOTE for more advanced balancing. - Train Decision Tree classifiers before and after SMOTE. - Compare model accuracies. - Perform hyperparameter tuning for the Decision Tree model.

Data Preparation

Load and Modify Dataset

```
# Set seed for reproducibility
set.seed(123)

# Load the iris dataset
data(iris)

# Remove 80% of 'setosa' samples to simulate imbalance
setosa_rows <- iris %>% filter(Species == "setosa")
other_rows <- iris %>% filter(Species != "setosa")
setosa_sample <- setosa_rows %>% sample_frac(0.2)  # Keep only 20% of 'setosa'

# Combine the sampled setosa rows with other classes
iris_modified <- bind_rows(setosa_sample, other_rows)

# Check class distribution
table(iris_modified$Species)
```

```
##
##     setosa versicolor  virginica
##         10         50         50
```

Manual Oversampling

```
# Oversample 'setosa' class by replicating its rows
oversampled_setosa <- setosa_sample %>% slice(rep(1:n(), each = 5))

# Combine oversampled setosa rows with other classes
iris_balanced <- bind_rows(oversampled_setosa, other_rows)
```

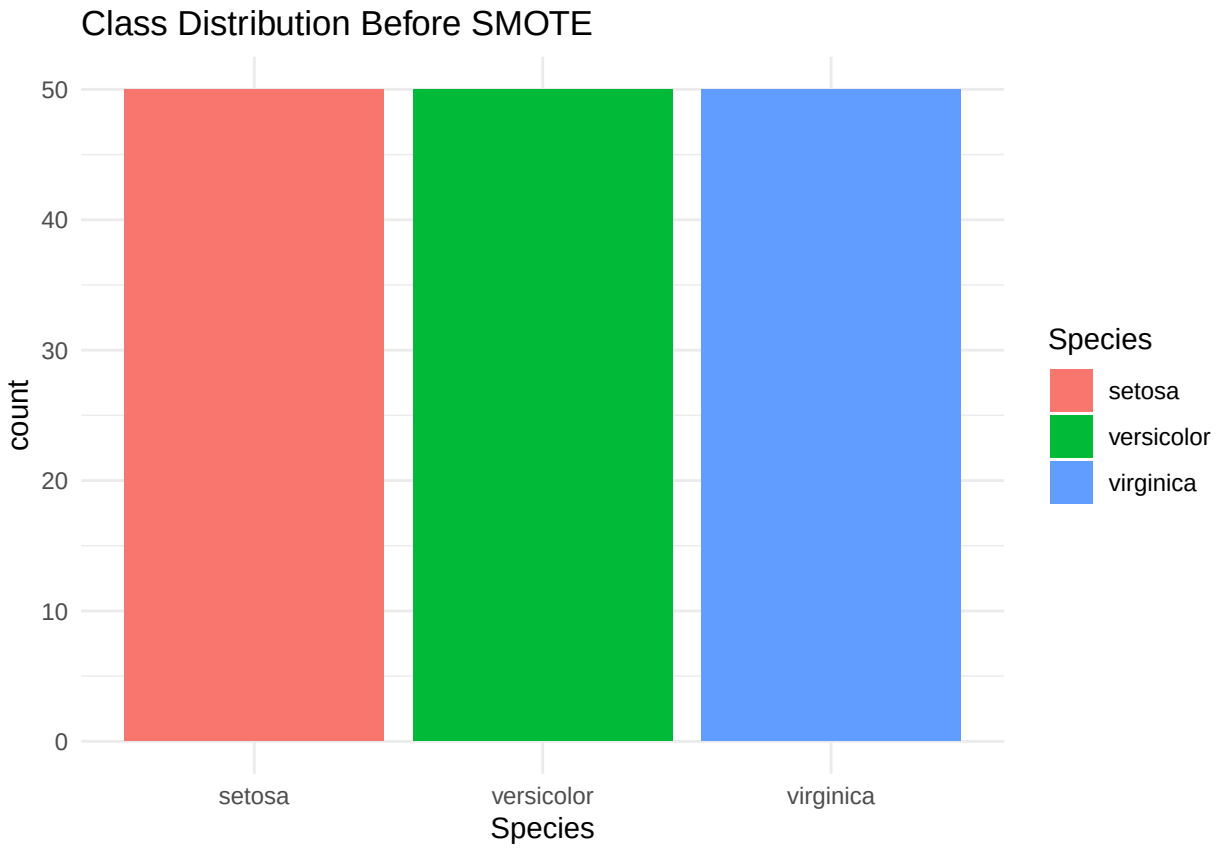
```r
# Check the new class distribution
table(iris_balanced$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

Applying SMOTE

Visualize Class Distribution Before SMOTE

```r
ggplot(iris_balanced, aes(x = Species, fill = Species)) +
  geom_bar() +
  ggtitle("Class Distribution Before SMOTE") +
  theme_minimal()
```

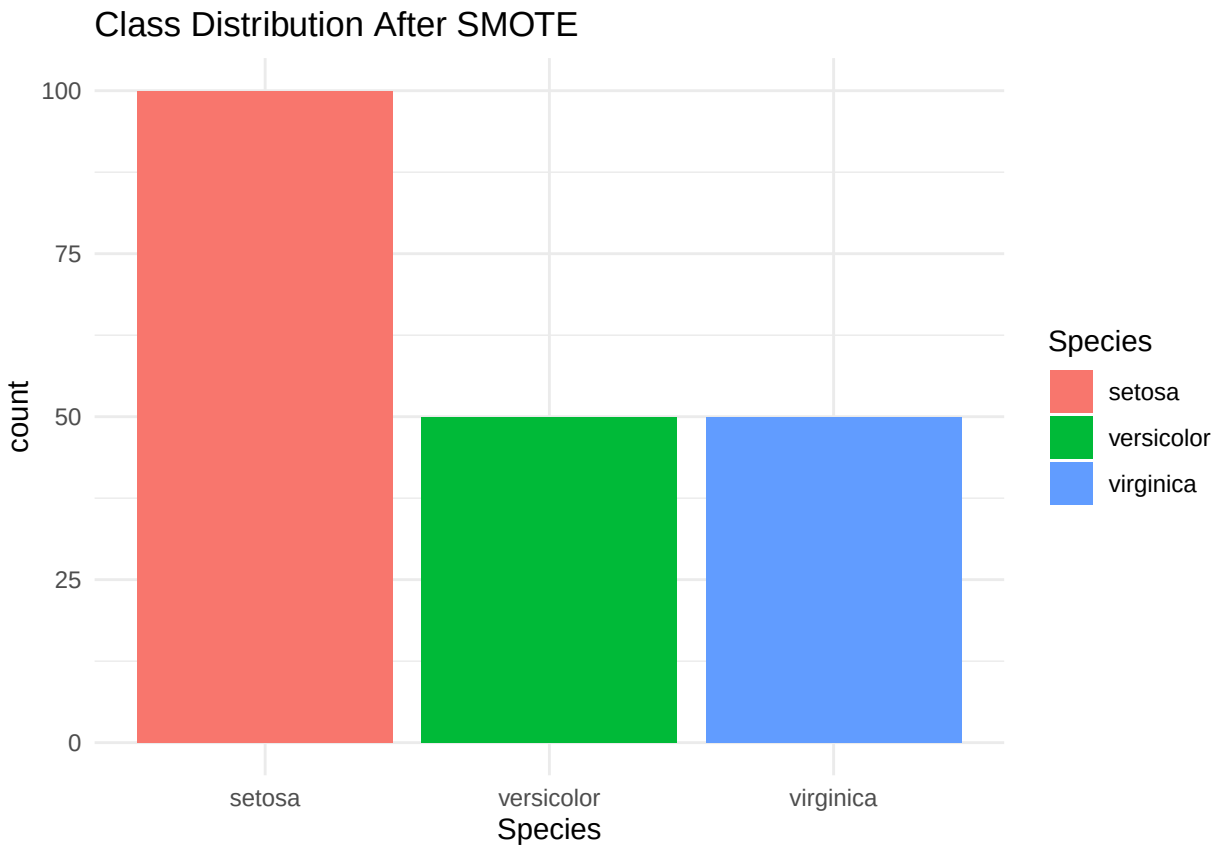


Apply SMOTE

```r
# Apply SMOTE using smotefamily
iris_smote <- SMOTE(X = iris_balanced[,-5], target = iris_balanced$Species, K = 5)

# Extract the balanced dataset
iris_smote_data <- iris_smote$data

# Rename the target column
colnames(iris_smote_data)[ncol(iris_smote_data)] <- "Species"

# Ensure it is a proper data frame
iris_smote_data <- as.data.frame(iris_smote_data)
```

Visualize Class Distribution After SMOTE

```r
ggplot(iris_smote_data, aes(x = Species, fill = Species)) +
  geom_bar() +
  ggtitle("Class Distribution After SMOTE") +
  theme_minimal()
```



Model Training and Comparison

Train Decision Tree on Imbalanced Data

```
dt_model_before <- train(
  Species ~ .,
  data = iris_balanced,
  method = "rpart",
  trControl = trainControl(method = "cv", number = 5)
)
```

Train Decision Tree on SMOTE Data

```
dt_model_after <- train(
  Species ~ .,
  data = iris_smote_data,
  method = "rpart",
  trControl = trainControl(method = "cv", number = 5)
)
```

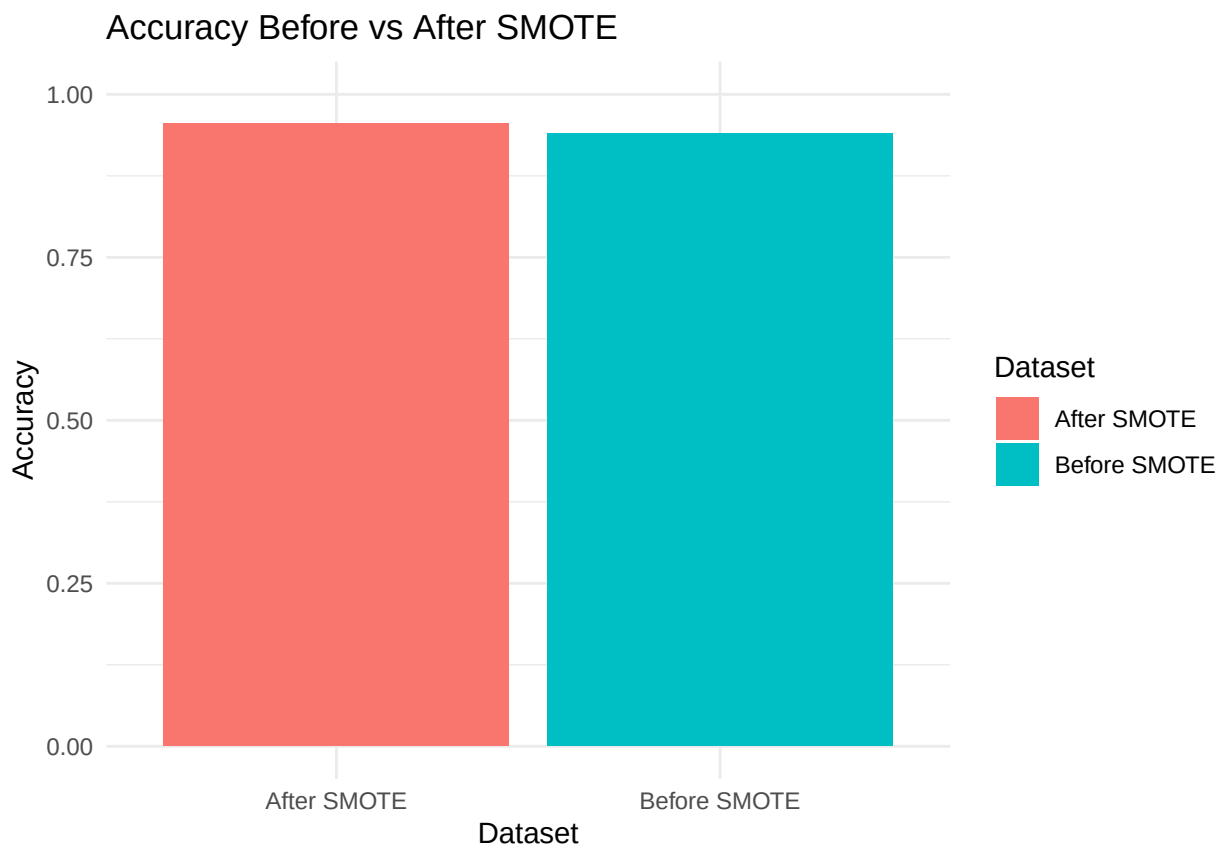
Compare Model Accuracies

```
acc_before <- max(dt_model_before$results$Accuracy)
acc_after <- max(dt_model_after$results$Accuracy)

# Create comparison dataframe
comparison <- data.frame(
  Dataset = c("Before SMOTE", "After SMOTE"),
  Accuracy = c(acc_before, acc_after)
)

# Print comparison
print(comparison)
```

```
##        Dataset Accuracy
## 1 Before SMOTE    0.940
## 2  After SMOTE    0.955
```

Plot Accuracy Comparison

```
ggplot(comparison, aes(x = Dataset, y = Accuracy, fill = Dataset)) +
  geom_bar(stat = "identity") +
  ylim(0, 1) +
  ggtitle("Accuracy Before vs After SMOTE") +
  theme_minimal()
```



Hyperparameter Tuning for Decision Tree

Preparing Dataset

```
# SMOTE already applied, reloading for safety
iris_smote <- SMOTE(X = iris_balanced[,-5], target = iris_balanced$Species, K = 5)
iris_smote_data <- iris_smote$data
colnames(iris_smote_data)[ncol(iris_smote_data)] <- "Species"
iris_smote_data <- as.data.frame(iris_smote_data)

# Check final class balance
table(iris_smote_data$Species)
```

```
##
##     setosa versicolor  virginica
##        100         50         50
```

Set Grid for Hyperparameter Search

```
# Setting a grid for 'cp' hyperparameter
tune_grid <- expand.grid(cp = seq(0.001, 0.05, by = 0.005))
```

Train Decision Tree with Cross-Validation

```
# Train with 5-fold cross-validation
dt_model <- train(
  Species ~ .,
  data = iris_smote_data,
  method = "rpart",
  trControl = trainControl(method = "cv", number = 5),
  tuneGrid = tune_grid
)
```

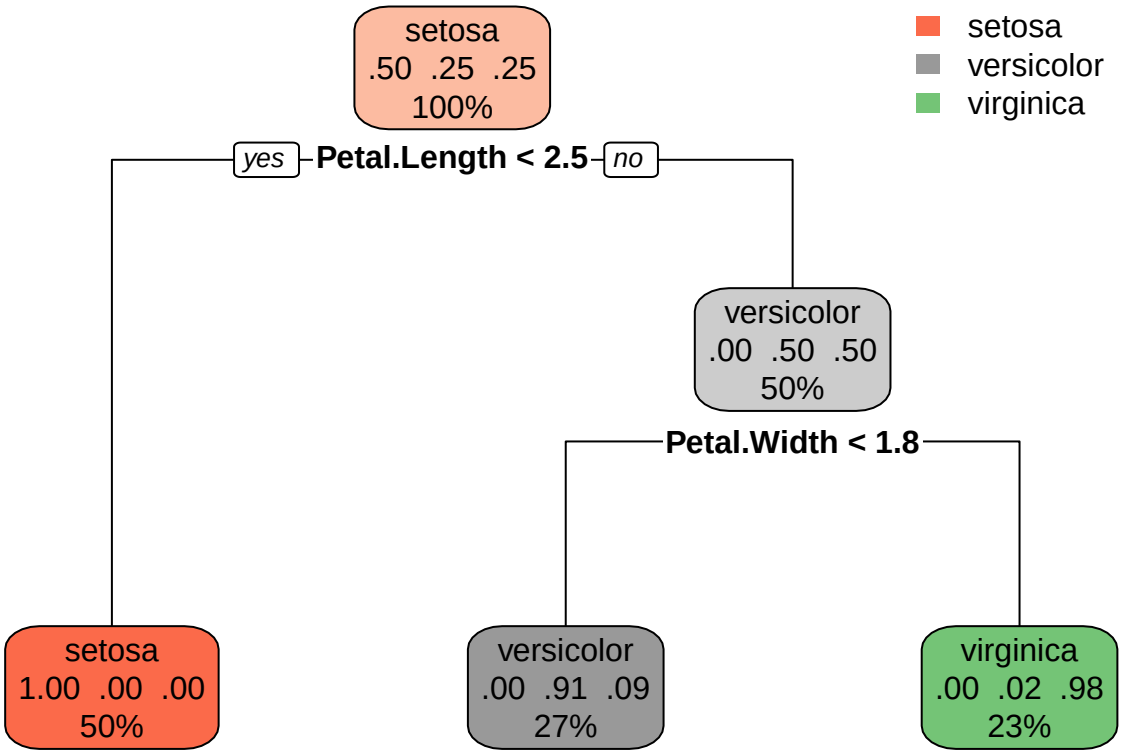
Display Best cp Value

```
# Display best cp value found
dt_model$bestTune
```

```
##       cp
## 10 0.046
```

Plot Final Decision Tree

```
# Plot the final trained Decision Tree
rpart.plot(dt_model$finalModel)
```



Conclusion

- Manual oversampling and SMOTE successfully balanced the dataset.
- SMOTE generally improved model performance compared to training on imbalanced data.
- Hyperparameter tuning further optimized the Decision Tree model for better accuracy.

‘‘‘