Data Mining Project

Gautam

2024-10-25

This Project includes healthcare cost prediction for Medicare patients specially for Heart Failure, where i am using two different model Linear regression and Decision tree to predict the cost for Heart Failure and compare among the model result.

Description of the Code Process

1. Load Libraries and Data

2. Merge Datasets and Identify Missing Values

3. Calculate and Visualize Correlation Matrix

4. Collect Categorical Variables and Perform ANOVA

5. Select Features and Filter by DRG Codes

6. Convert Categorical Variables to Factor and Split Data

7. Preprocess Data and Create Linear Model with Features

8. Make Predictions and Calculate Evaluation Metrics

9. Print Evaluation Results and Lasso Model

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v ggplot2   3.4.4     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(ggplot2)
library(maps)
```

```
##
## Attaching package: 'maps'
##
## The following object is masked from 'package:purrr':
##
##     map
```

```r
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(rpart)
library(rpart.plot)
library(rattle)
```

```
## Loading required package: bitops
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(neuralnet)
```

```
##
## Attaching package: 'neuralnet'
```

```
##
## The following object is masked from 'package:dplyr':
##
##     compute
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.3

## corrplot 0.95 loaded
```

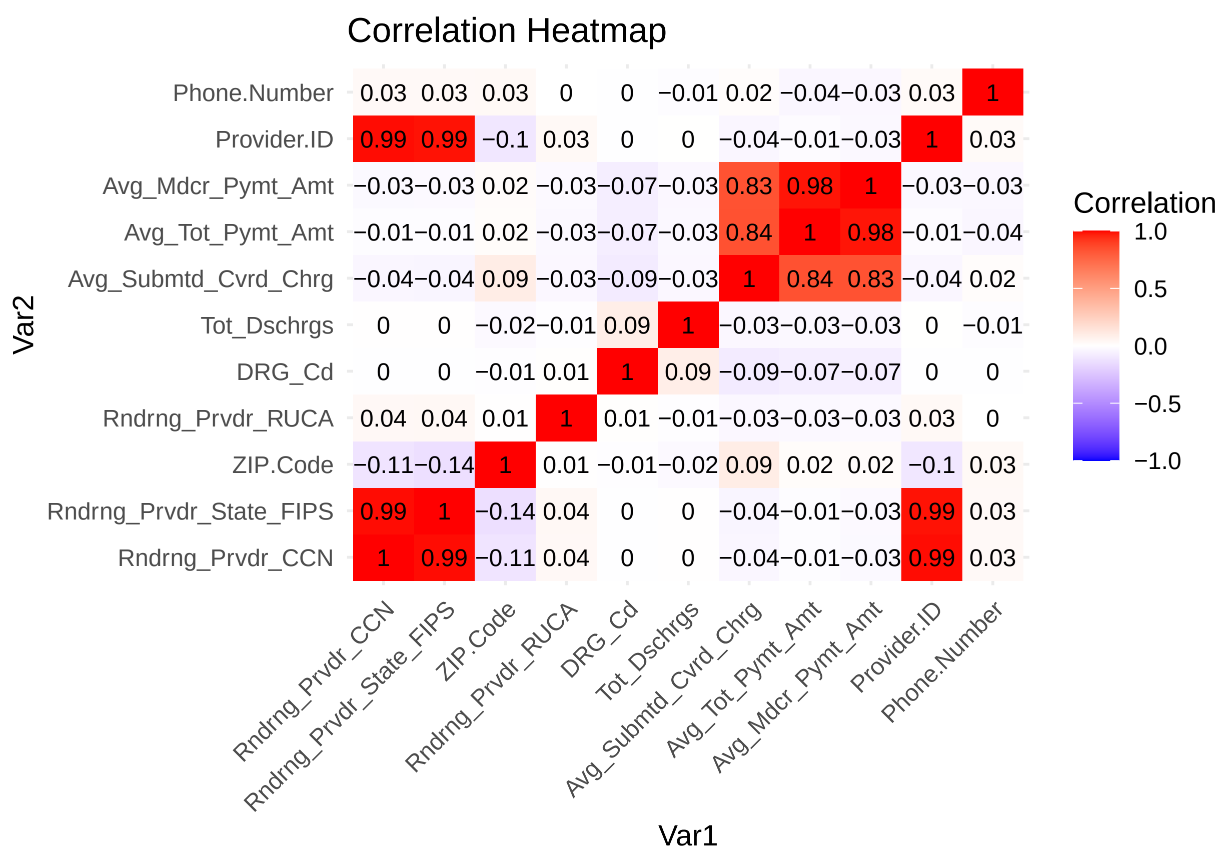
```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
# Load the data
hospital_data <- read.csv("/Users/Lord/Downloads/Hospital_General_Information.csv")
medicare_data <- read.csv("/Users/Lord/Downloads/Medicare_Inpatient_Hospital_by_Provider_and_Service_20

# Rename ZIP.Code column as it is ZIP.Code in hospital data
colnames(medicare_data)[7] <- "ZIP.Code"

# Merging datasets on ZIP.Code
merged_data <- inner_join(medicare_data, hospital_data, "ZIP.Code")
```

```
## Warning in inner_join(medicare_data, hospital_data, "ZIP.Code"): Detected an unexpected many-to-many
## i Row 489 of `x` matches multiple rows in `y`.
## i Row 1 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
# Identifying missing values
miss_val <- sum(is.na(merged_data))

# Calculating correlation matrix for numeric features to identify highly variable and related features
numeric_data <- merged_data %>% select_if(is.numeric)
correlation_matrix <- cor(numeric_data, use = "pairwise.complete.obs")


# Creating a heatmap of correlation matric
melted_corr <- melt(correlation_matrix)
ggplot(data = melted_corr, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", limit = c(-1, 1), name = "Correlation
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Correlation Heatmap") +
  geom_text(aes(label = round(value, 2)), color = "black", size = 3)
```



```r
#Collecting only categorical variables
categorical_vars <- names(merged_data)[sapply(merged_data, function(x) is.factor(x) || is.character(x))
print(categorical_vars)
```

```
##  [1] "Rndrng_Prvdr_Org_Name"
##  [2] "Rndrng_Prvdr_St"
##  [3] "Rndrng_Prvdr_City"
##  [4] "Rndrng_Prvdr_State_Abrvtn"
##  [5] "Rndrng_Prvdr_RUCA_Desc"
##  [6] "DRG_Desc"
##  [7] "Hospital.Name"
##  [8] "Address"
##  [9] "City"
## [10] "State"
## [11] "County.Name"
## [12] "Hospital.Type"
## [13] "Hospital.Ownership"
## [14] "Emergency.Services"
## [15] "Meets.criteria.for.meaningful.use.of.EHRs"
## [16] "Hospital.overall.rating"
## [17] "Hospital.overall.rating.footnote"
## [18] "Mortality.national.comparison"
## [19] "Mortality.national.comparison.footnote"
## [20] "Safety.of.care.national.comparison"
## [21] "Safety.of.care.national.comparison.footnote"
## [22] "Readmission.national.comparison"
## [23] "Readmission.national.comparison.footnote"
## [24] "Patient.experience.national.comparison"
## [25] "Patient.experience.national.comparison.footnote"
## [26] "Effectiveness.of.care.national.comparison"
## [27] "Effectiveness.of.care.national.comparison.footnote"
## [28] "Timeliness.of.care.national.comparison"
## [29] "Timeliness.of.care.national.comparison.footnote"
## [30] "Efficient.use.of.medical.imaging.national.comparison"
## [31] "Efficient.use.of.medical.imaging.national.comparison.footnote"
```

```
#Selecting only categorical variables to perform anova test for selecting highly variable feature to pr
anova_result <- aov(Avg_Mdcr_Pymt_Amt ~  Hospital.Type  + Hospital.Ownership+ State + Hospital.overall.r
summary(anova_result)
```

```
##                                                          Df    Sum Sq   Mean Sq
## Hospital.Type                                             2 3.388e+11 1.694e+11
## Hospital.Ownership                                        9 3.165e+11 3.516e+10
## State                                                    50 1.036e+12 2.072e+10
## Hospital.overall.rating                                   5 9.291e+10 1.858e+10
## Emergency.Services                                        1 1.354e+08 1.354e+08
## Meets.criteria.for.meaningful.use.of.EHRs                 1 1.323e+09 1.323e+09
## Mortality.national.comparison                             3 2.310e+11 7.699e+10
## Patient.experience.national.comparison                    3 3.105e+10 1.035e+10
## Timeliness.of.care.national.comparison                    3 3.262e+11 1.087e+11
## Efficient.use.of.medical.imaging.national.comparison      3 3.167e+10 1.056e+10
## Residuals                                            250952 4.858e+13 1.936e+08
##                                                      F value  Pr(>F)
## Hospital.Type                                        875.061 < 2e-16 ***
## Hospital.Ownership                                   181.665 < 2e-16 ***
## State                                                107.044 < 2e-16 ***
## Hospital.overall.rating                               95.998 < 2e-16 ***
## Emergency.Services                                     0.700 0.40289
## Meets.criteria.for.meaningful.use.of.EHRs              6.836 0.00893 **
## Mortality.national.comparison                        397.747 < 2e-16 ***
## Patient.experience.national.comparison                53.470 < 2e-16 ***
## Timeliness.of.care.national.comparison               561.810 < 2e-16 ***
## Efficient.use.of.medical.imaging.national.comparison  54.530 < 2e-16 ***
## Residuals
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Selecting Features based on Correlation Matrix and anova result
selected_vars <- c("Avg_Mdcr_Pymt_Amt", "Avg_Tot_Pymt_Amt", "Hospital.Type", "Hospital.Ownership",
                   "Hospital.overall.rating", "Meets.criteria.for.meaningful.use.of.EHRs",
                   "Mortality.national.comparison", "Patient.experience.national.comparison",
                   "Timeliness.of.care.national.comparison", "Efficient.use.of.medical.imaging.national


#Filtering dataset for HeartFailure DRG_Cd Code
final_data_HF <- filter(merged_data, DRG_Cd %in% c("291", "292", "293"))
selected_data_HF <- final_data_HF %>% select(all_of(selected_vars))

#Converting categorical variables to factor to pass in model- Used as optional for one-hot encoding
```

```r
variables_to_factor <- c("Hospital.Type", "Hospital.Ownership", "Hospital.overall.rating",
                         "Meets.criteria.for.meaningful.use.of.EHRs", "Mortality.national.comparison",
                         "Patient.experience.national.comparison",
                         "Timeliness.of.care.national.comparison",
                         "Efficient.use.of.medical.imaging.national.comparison")

selected_data_HF[variables_to_factor] <- lapply(selected_data_HF[variables_to_factor], as.factor)


# Train-test split
set.seed(123)
train_index <- createDataPartition(selected_data_HF$Avg_Mdcr_Pymt_Amt, p = 0.8, list = FALSE)
train_data <- selected_data_HF[train_index, ]
test_data <- selected_data_HF[-train_index, ]

# Preprocessing data
pre_process <- preProcess(train_data %>% select(-Avg_Mdcr_Pymt_Amt), method = c("center", "scale"))
train_data_normalized <- predict(pre_process, newdata = train_data %>% select(-Avg_Mdcr_Pymt_Amt))
test_data_normalized <- predict(pre_process, newdata = test_data %>% select(-Avg_Mdcr_Pymt_Amt))

# Adding the target variable back
train_data_normalized$Avg_Mdcr_Pymt_Amt <- train_data$Avg_Mdcr_Pymt_Amt
test_data_normalized$Avg_Mdcr_Pymt_Amt <- test_data$Avg_Mdcr_Pymt_Amt

# Creating linear model with interaction and polynomial features for capturing non-linearity and better
linear_model_cv <- train(Avg_Mdcr_Pymt_Amt ~ . + I(Avg_Tot_Pymt_Amt^2) +
                           Avg_Tot_Pymt_Amt:Hospital.Type +
                           Avg_Tot_Pymt_Amt:Hospital.Ownership,
                         data = train_data_normalized, method = "lm",
                         trControl = trainControl(method = "repeatedcv", number = 10,repeats = 3))
```

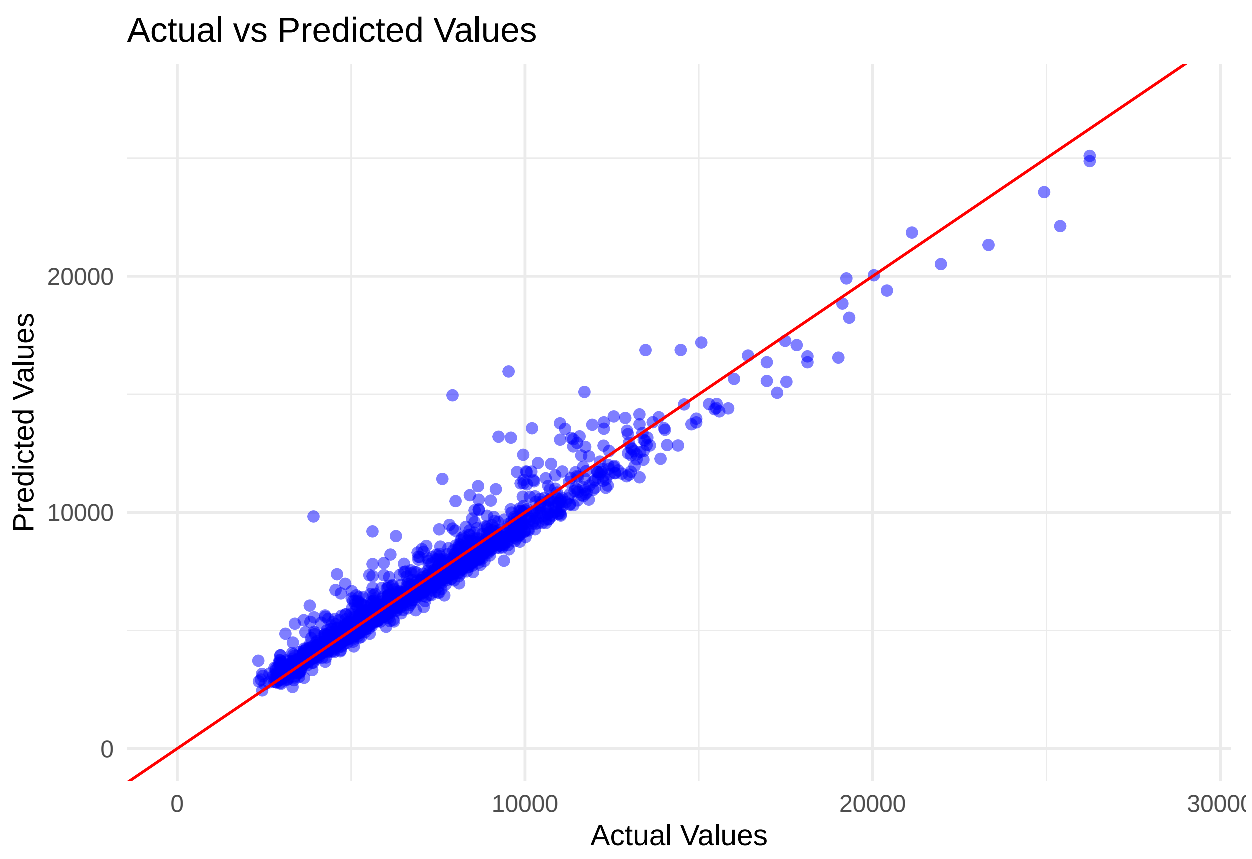
```
## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(modelFit, newdata): prediction from rank-deficient fit;
## attr(*, "non-estim") has doubtful cases
```

```r
predictions <- predict(linear_model_cv, newdata = test_data_normalized)

# Calculating evaluation metrics
train_predictions <- predict(linear_model_cv, newdata = train_data_normalized)
train_rmse <- RMSE(train_predictions, train_data_normalized$Avg_Mdcr_Pymt_Amt)
test_rmse <- RMSE(predictions, test_data_normalized$Avg_Mdcr_Pymt_Amt)
train_R2 <- R2(train_predictions, train_data_normalized$Avg_Mdcr_Pymt_Amt)
```

```r
test_R2 <- R2(predictions, test_data_normalized$Avg_Mdcr_Pymt_Amt)


# Creating Dataframe of the result
evaluation_results <- data.frame(
  RMSE = c(train_rmse, test_rmse),
  R2 = c(train_R2, test_R2),
  row.names = c("Train", "Test")
)
print(evaluation_results)
```

```
##            RMSE        R2
## Train 800.2127 0.9448077
## Test  718.0019 0.9465313
```

```r
# Creating a scatter plot of actual vs. predicted values
results_df <- data.frame(
  Actual = test_data$Avg_Mdcr_Pymt_Amt,
  Predicted = predictions
)

ggplot(results_df, aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue", alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(title = "Actual vs Predicted Values",
       x = "Actual Values",
       y = "Predicted Values") +
  theme_minimal() +
  xlim(0, max(results_df$Actual) * 1.1) +
  ylim(0, max(results_df$Predicted) * 1.1)
```



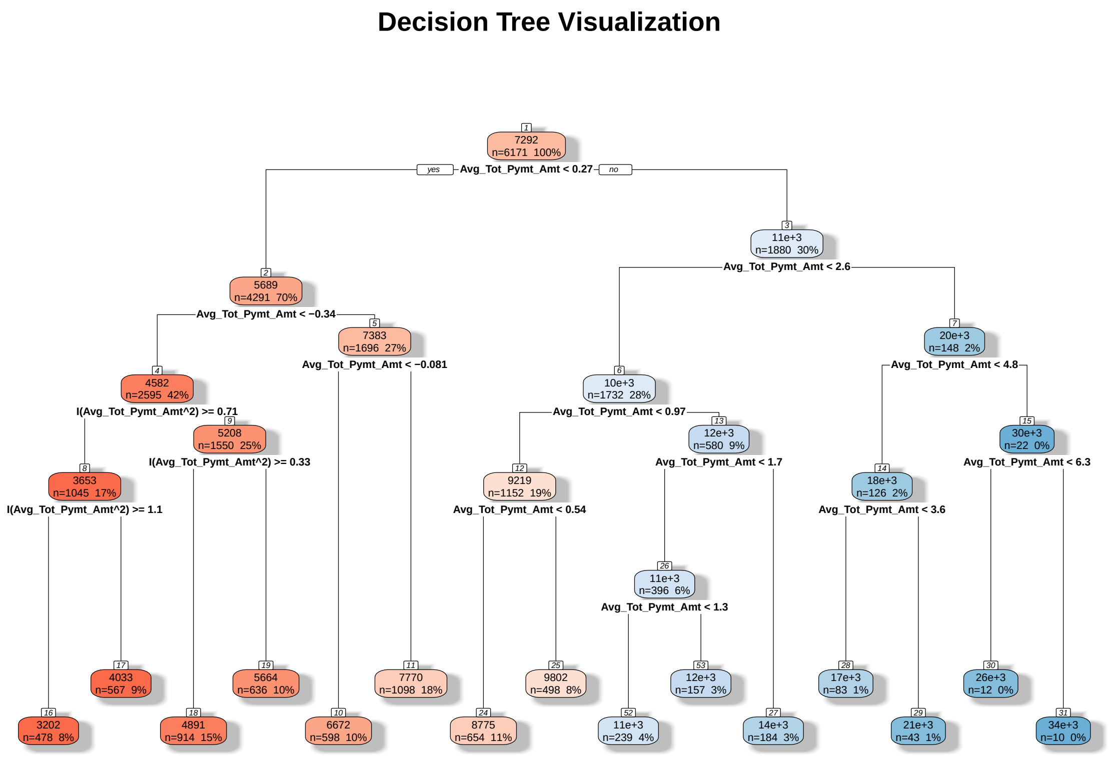
```
#Lasso Model to check overfitting or possible mutiple colinearity
lasso_model <- train(Avg_Mdcr_Pymt_Amt ~ ., data = train_data_normalized,
                                            method = "glmnet",
                                            trControl = trainControl(method = "cv", number = 10),
                                            tuneGrid = expand.grid(alpha = 1, lambda = seq(0.01, 1, leng
print(lasso_model)
```

```
## glmnet
##
## 6171 samples
##    9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5553, 5552, 5555, 5554, 5554, 5554, ...
## Resampling results across tuning parameters:
##
##   lambda  RMSE     Rsquared   MAE
##   0.01    809.231  0.9423251  494.8502
##   0.12    809.231  0.9423251  494.8502
##   0.23    809.231  0.9423251  494.8502
##   0.34    809.231  0.9423251  494.8502
##   0.45    809.231  0.9423251  494.8502
##   0.56    809.231  0.9423251  494.8502
##   0.67    809.231  0.9423251  494.8502
##   0.78    809.231  0.9423251  494.8502
##   0.89    809.231  0.9423251  494.8502
##   1.00    809.231  0.9423251  494.8502
##
## Tuning parameter 'alpha' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 1 and lambda = 1.
```

```
#Decision Tree


# Training Decision Tree Model with Cross-Validation
tree_model <- train(
  Avg_Mdcr_Pymt_Amt ~ . + I(Avg_Tot_Pymt_Amt^2) + Avg_Tot_Pymt_Amt:Hospital.Type + Avg_Tot_Pymt_Amt:Hosp
  data = train_data_normalized,
  method = "rpart",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = expand.grid(cp = seq(0.001, 0.05, by = 0.005))
)


predictions_tree <- predict(tree_model, newdata = test_data_normalized)
train_predictions_tree <- predict(tree_model, newdata = train_data_normalized)


# Calculating evaluation metrics
rmse_train_tree <- RMSE(train_predictions_tree, train_data_normalized$Avg_Mdcr_Pymt_Amt)
r2_train_tree <- R2(train_predictions_tree, train_data_normalized$Avg_Mdcr_Pymt_Amt)

rmse_test_tree <- RMSE(predictions_tree, test_data_normalized$Avg_Mdcr_Pymt_Amt)
r2_test_tree <- R2(predictions_tree, test_data_normalized$Avg_Mdcr_Pymt_Amt)

#creating dataframe of the error
tree_error_df <- data.frame(
  Metric = c("RMSE", "R-squared"),
  Training_Error = c(rmse_train_tree, r2_train_tree),
  Testing_Error = c(rmse_test_tree, r2_test_tree)
)


print(tree_error_df)
```

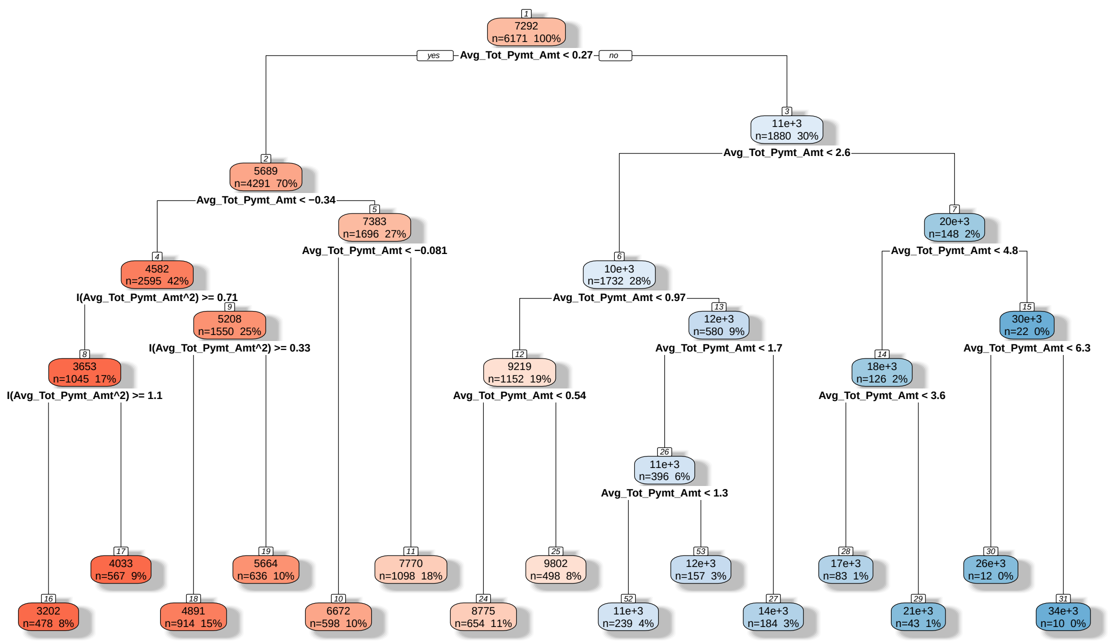
```
##      Metric Training_Error Testing_Error
## 1      RMSE    893.1954395   793.6572309
## 2 R-squared      0.9312361     0.9344725
```

```
# Plotting the decision tree for regression
rpart.plot(tree_model$finalModel, type = 2, extra = 101, fallen.leaves = TRUE,
           main = "Decision Tree Visualization",
           box.palette = "RdBu", shadow.col = "gray", nn = TRUE)
```



```r
# Checking if the cp value in the final model can be optimized and pruning the tree to avoid overfittin
optimal_cp <- tree_model$bestTune$cp
pruned_tree <- prune(tree_model$finalModel, cp = optimal_cp)

# Plotting the pruned tree if pruning improved performance
rpart.plot(pruned_tree, type = 2, extra = 101, fallen.leaves = TRUE,
           main = "Pruned Decision Tree",
           box.palette = "RdBu", shadow.col = "gray", nn = TRUE)
```

Pruned Decision Tree


1
7292
n=6171 100%
yes
Avg_Tot_Pymt_Amt < 0.27
no
2
5689
n=4291 70%
Avg_Tot_Pymt_Amt < −0.34
3
11e+3
n=1880 30%
Avg_Tot_Pymt_Amt < 2.6
4
4582
n=2595 42%
I(Avg_Tot_Pymt_Amt^2) >= 0.71
5
7383
n=1696 27%
Avg_Tot_Pymt_Amt < −0.081
6
10e+3
n=1732 28%
Avg_Tot_Pymt_Amt < 0.97
7
20e+3
n=148 2%
Avg_Tot_Pymt_Amt < 4.8
8
3653
n=1045 17%
I(Avg_Tot_Pymt_Amt^2) >= 1.1
9
5208
n=1550 25%
I(Avg_Tot_Pymt_Amt^2) >= 0.33
12
9219
n=1152 19%
Avg_Tot_Pymt_Amt < 0.54
13
12e+3
n=580 9%
Avg_Tot_Pymt_Amt < 1.7
14
18e+3
n=126 2%
Avg_Tot_Pymt_Amt < 3.6
15
30e+3
n=22 0%
Avg_Tot_Pymt_Amt < 6.3
26
11e+3
n=396 6%
Avg_Tot_Pymt_Amt < 1.3
17
4033
n=567 9%
19
5664
n=636 10%
11
7770
n=1098 18%
25
9802
n=498 8%
53
12e+3
n=157 3%
28
17e+3
n=83 1%
30
26e+3
n=12 0%
16
3202
n=478 8%
18
4891
n=914 15%
10
6672
n=598 10%
24
8775
n=654 11%
52
11e+3
n=239 4%
27
14e+3
n=184 3%
29
21e+3
n=43 1%
31
34e+3
n=10 0%