PSTAT 131 Final Project

Agneya Poduval

2025-03-18

Introduction

General

The dataset I chose is the Blood Transfusion Service Center Data Set, which was created from random samples of blood donor data in Hsin-Chu City, Taiwan. The dataset includes details about each donor's donation history, such as how recently and how frequently they have donated blood, the total volume of blood donated, and the length of time they have been donating. These factors can provide strong indicators for predicting future donation behavior. Additionally, with a binary target variable (which is whether or not a person donated blood in March 2007), this dataset is well-suited for classification. By applying machine learning techniques, we can identify patterns and build models that predict future donations, which could help optimize blood donation strategies and improve donor engagement.

Significance

Developing a strong machine learning model for classifying blood donors has significant practical applications in both healthcare and donor management. By accurately predicting whether a person is likely to donate blood, organizations can tailor their outreach efforts to encourage donations more effectively. For example, targeted reminders and personalized incentives can be directed toward individuals who have donated in the past but may need an extra push to return. Additionally, hospitals and blood banks can use these predictions to better anticipate blood supply levels, ensuring they have enough donors available to meet medical demands. This approach helps optimize resources, reducing blood shortages and making it easier for blood banks to operate without fear.

Beyond improving public health outcomes, a proper model could contribute to large public health policy changes. By utilizing any given machine learning model, policymakers could identify patterns in the behavior of blood donors, find ways to recruit new donors, and keep their old donors coming back. This could include educational campaigns targeting demographics that are less likely to donate or adjusting marketing strategies to appeal to different donor segments. In conclusion, using machine learning and marketing strategies in the public health sector could support a more efficient and data-driven healthcare system that would keep blood banks replenished and hospitals able to support their patients.

Data Loading and Cleaning

```r
data <- read.arff("/Users/agneyapoduval/Documents/Classes/PSTAT 131/Lab Files/donateblood.arff")
data$Class <- ifelse(data$Class == 2, 1, 0)
data$Class <- factor(data$Class, levels = c(0, 1))
```

Since the dataset was taken from OpenML, it was already cleaned and almost ready to go. One major change I made was converting the levels of the Class variable (which is the binary classifier or target variable) from 1/2 to 0/1 to make model prediction easier in the earlier steps.

Exploratory Data Analysis

Summary Statistics

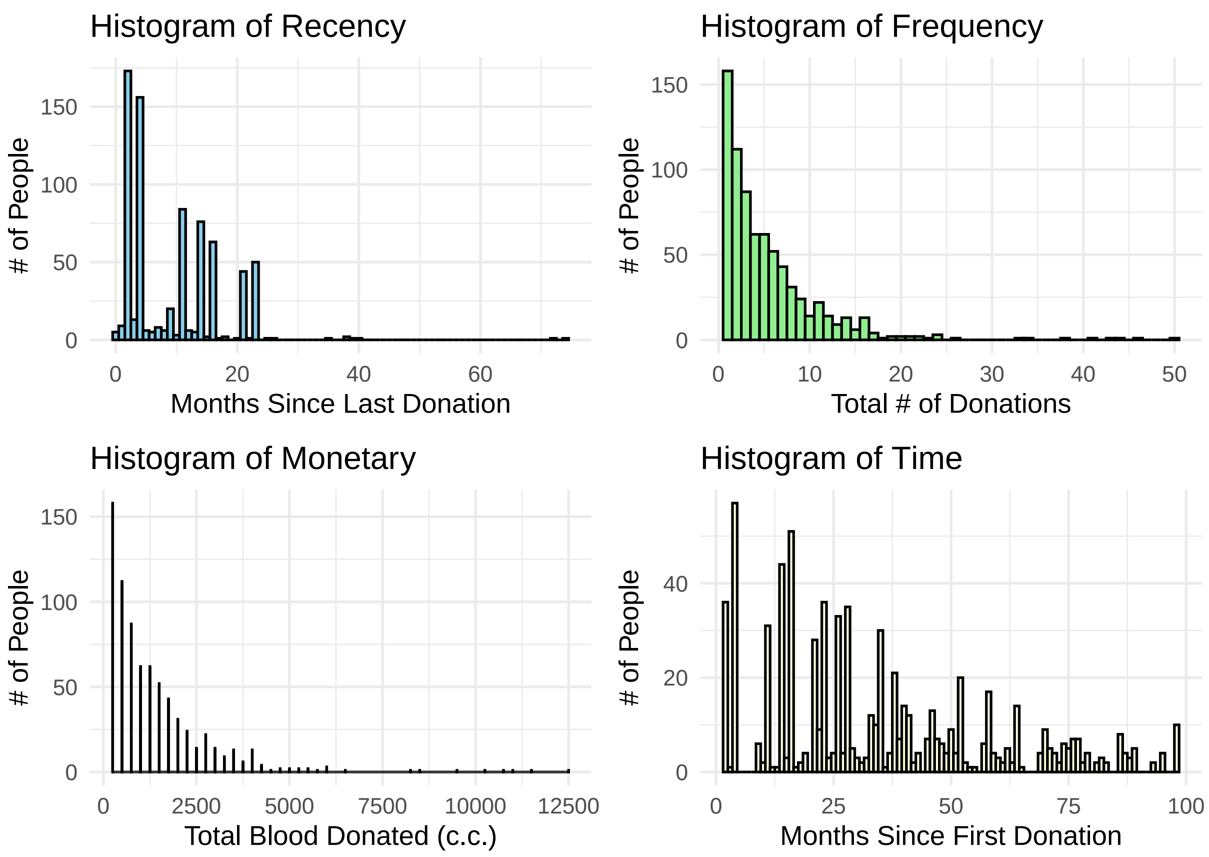
```r
summary <- data.frame(
  Variable = c("Recency", "Frequency", "Monetary", "Time"),
  Mean = c(mean(data$V1), mean(data$V2), mean(data$V3), mean(data$V4)),
  Median = c(median(data$V1), median(data$V2), median(data$V3), median(data$V4)),
  SD = c(sd(data$V1), sd(data$V2), sd(data$V3), sd(data$V4))
)
kable(summary, col.names = c("Variable", "Mean", "Median", "SD"), align = "lcc")
```

Variable	Mean	Median	SD
Recency	9.507	7	8.095
Frequency	5.515	4	5.839
Monetary	1378.677	1000	1459.827
Time	34.282	28	24.377

The Monetary variable, which is the total blood donated in cubic centimeters, has by far the largest nominal values, which makes sense as it is a measurement of volume. The Frequency variable, or total number of donations, has the lowest nominal values. Finally, the Time variable has larger values than the Recency variable, which makes sense as no one can have their last donation before their first donation. The main thing to note from this table is that for all of the variables, the mean value is above the median value, indicating right skewness. Because of this, standard linear methods may not be useful without excessive transformations, which would make it harder to interpret the result. Knowing this, we can move forward with various classification machine learning methods that can create a nonlinear prediction.

Data Visualization

```r
p1 <- ggplot(data, aes(x = V1)) +
  geom_histogram(binwidth = 1, fill = "skyblue", color = "black") +
  labs(title = "Histogram of Recency", x = "Months Since Last Donation", y = "# of People") +
  theme_minimal()

p2 <- ggplot(data, aes(x = V2)) +
  geom_histogram(binwidth = 1, fill = "lightgreen", color = "black") +
  labs(title = "Histogram of Frequency", x = "Total # of Donations", y = "# of People") +
  theme_minimal()

p3 <- ggplot(data, aes(x = V3)) +
  geom_histogram(binwidth = 1, fill = "lightcoral", color = "black") +
  labs(title = "Histogram of Monetary", x = "Total Blood Donated (c.c.)", y = "# of People") +
  theme_minimal()

p4 <- ggplot(data, aes(x = V4)) +
  geom_histogram(binwidth = 1, fill = "lightyellow", color = "black") +
  labs(title = "Histogram of Time", x = "Months Since First Donation", y = "# of People") +
  theme_minimal()

p1 + p2 + p3 + p4
```



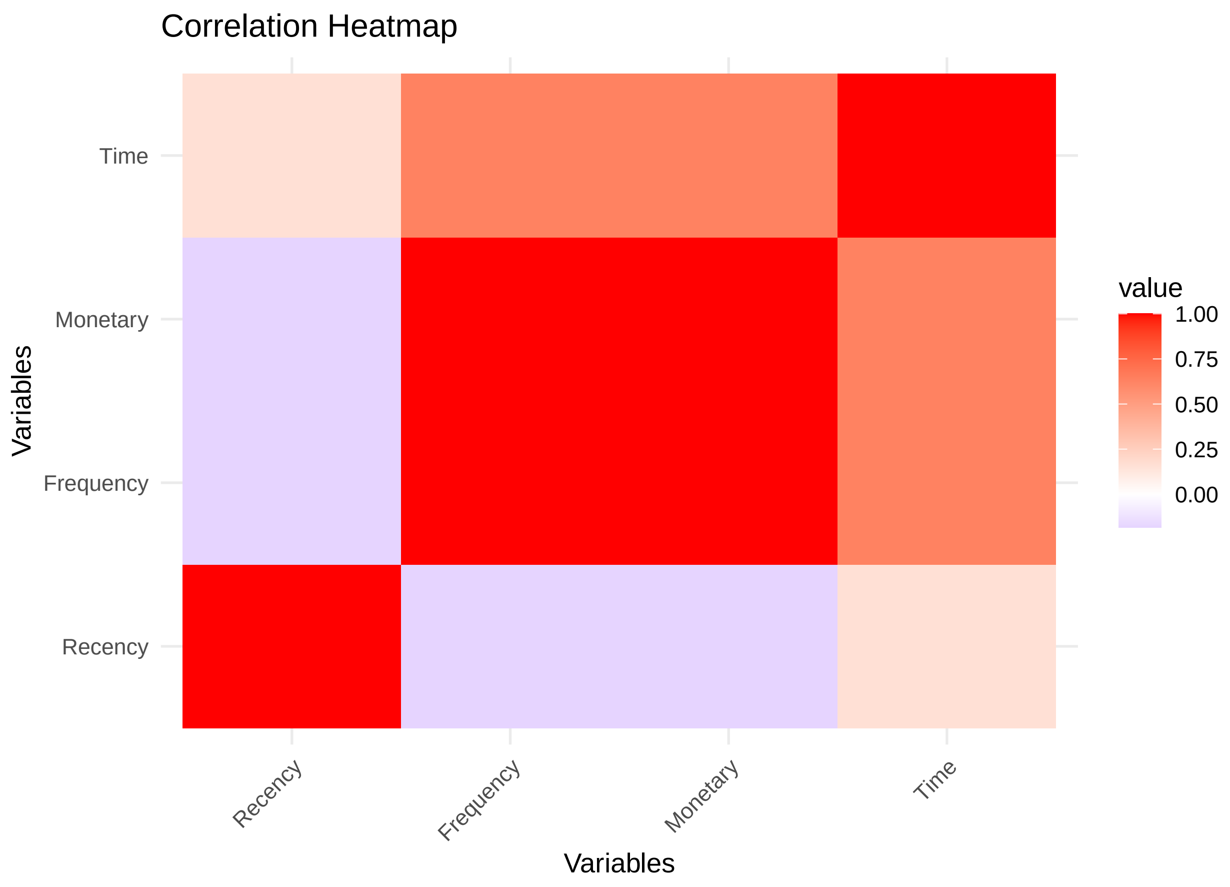
The Recency variable seems to have two main clusters of people - one around 2 - 4 months since their last donation, and another between 10 - 20 months since their last donation. This means that most of the people in the donation pool tend to donate often, while the rest of the group may have only donated once.

Both the Frequency and Monetary variable seem to have the exact same distribution, which we will explore further in the following section. Either way, we see that both of them have an extremely right skewed distribution with the amount of people decreasing as each variable increases. This makes sense because as blood is donated more times, the total amount of blood donated also increases. It seems that the amount of blood taken in each draw is standardized since both variables increase at exactly the same amount.

Finally, the Time variable has a slightly more even distribution, but still shows peaks towards the left side with lower frequencies towards the right side. This means that more people have recently joined the program, but there are also plenty of people that have logged well over 2 years since their first donation and continue to donate. This is promising for the real life significance of this model because it shows that many new people were joining the blood donation drive at the time this data was collected.

Correlation Matrix

```r
data <- data %>% rename(Recency = V1, Frequency = V2, Monetary = V3, Time = V4)
correlation_matrix <- cor(data[, c("Recency", "Frequency", "Monetary", "Time")])
melted_corr <- melt(correlation_matrix)

ggplot(melted_corr, aes(Var1, Var2, fill = value)) + geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0) +
  labs(title = "Correlation Heatmap", x = "Variables", y = "Variables") +
  theme_minimal() + theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Clearly, V2 and V3 have perfect correlation with each other. Thus, I will only consider V2: Frequency, or the total number of donations, from now on. In addition, V4 has high correlation with V2 and V3. While I am not going to completely ignore V4, I will keep this fact in mind when building the models.

```r
data <- data %>% select(-Monetary)
```

Machine Learning Models

Train/Test Split

```r
trainIndex <- createDataPartition(data$Class, p = 0.7, list = FALSE)
train_data <- data[trainIndex, ]
test_data <- data[-trainIndex, ]
```

As there is no test set, I decided to split the data into 70% training and 30% test data.

Logistic Regression

```r
set.seed(123)

log_model <- glm(Class ~ ., data = train_data, family = binomial)

log_pred <- predict(log_model, test_data, type = "response")
log_pred_class <- ifelse(log_pred > 0.5, 1, 0)
log_pred_class <- factor(log_pred_class, levels = c(0, 1))

log_cm <- confusionMatrix(log_pred_class, test_data$Class)

log_cm_table <- as.data.frame(log_cm$table)
colnames(log_cm_table) <- c("Reference", "Prediction", "Count")

kable(log_cm_table, caption = "Confusion Matrix for Logistic Regression", align = "c") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"))
```

Table 2: Confusion Matrix for Logistic Regression

Reference	Prediction	Count
0	0	168
1	0	3
0	1	47
1	1	6

We can see from the confusion matrix that even though there are very few actual donors that donated in March 2007, the model overpredicts the amount by a significant amount. This means that the Logistic Regression has a very high False Positive Rate, which is the consequence of having such a low False Negative Rate.

Random Forest

```r
set.seed(123)

rf_model <- randomForest(Class ~ ., data = train_data, importance = TRUE)
```

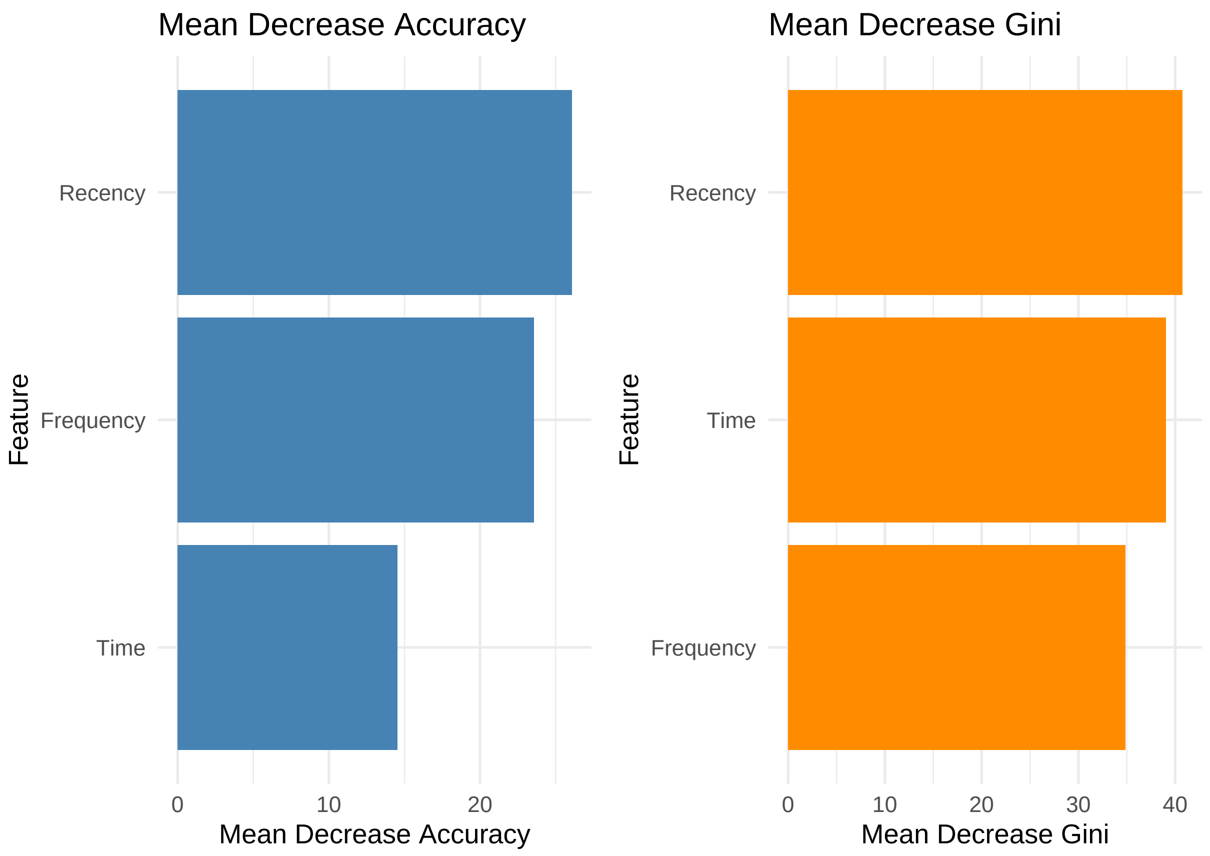
```r
rf_pred <- predict(rf_model, test_data)

rf_cm <- confusionMatrix(rf_pred, as.factor(test_data$Class))
conf_matrix_rf_df <- as.data.frame(rf_cm$table)
conf_matrix_rf_df <- conf_matrix_rf_df %>%
  select(Reference, Prediction, Freq) %>%
  slice(1, 3, 2, 4)

kable(conf_matrix_rf_df, digits = 4, caption = "Confusion Matrix for Random Forest Model") %>%
  kable_styling(full_width = FALSE, bootstrap_options = c("striped", "hover"))
```

Table 3: Confusion Matrix for Random Forest Model

Reference	Prediction	Freq
0	0	157
1	0	38
0	1	14
1	1	15

We can see from the confusion matrix that the random forest sacrifices the low False Negative rate for lower False Positive and higher True Positive Rates. This means that if using a random forest model, more actual donors will be missed, but also more donors will be classified correctly and less non-donors will be classified as donors. The model seems to be more balanced than the Logistic Regression without considering other metrics.

```r
rf_importance <- importance(rf_model)
rf_importance_df <- data.frame(Feature = rownames(rf_importance), rf_importance)

# Plot Mean Decrease Accuracy (MDA)
p1 <- ggplot(rf_importance_df, aes(x = reorder(Feature, MeanDecreaseAccuracy), y = MeanDecreaseAccuracy
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(title = "Mean Decrease Accuracy", x = "Feature", y = "Mean Decrease Accuracy") +
  theme_minimal()

# Plot Mean Decrease Gini (MDG)
p2 <- ggplot(rf_importance_df, aes(x = reorder(Feature, MeanDecreaseGini), y = MeanDecreaseGini)) +
  geom_bar(stat = "identity", fill = "darkorange") +
  coord_flip() +
  labs(title = "Mean Decrease Gini", x = "Feature", y = "Mean Decrease Gini") +
  theme_minimal()

p1 + p2
```



The Mean Decrease Accuracy (MDA) measures "per-class variable importance in terms of oob [out-of-bag] mean decrease in accuracy" (Rodriguez-Galiano et al. 2012). In other words, it measures the mean decrease in model accuracy for each feature before making predictions. This means that the features with the highest MDA are the most critical predictors. We can see that Recency has the highest MDA value, while Frequency is not far behind. Time has by far the lowest MDA value.

The Mean Decrease Gini (MDG), on the other hand, is the "average across the forest of the decrease in Gini impurity for a predictor" (Nicodemus 2011). As we saw in Lecture 13: Tree-based Methods, this implies that the features with the highest MDG are most valuable when considering the splits in the tree. We can see from the barplot that Recency still has the highest value, but now Time has a higher MDG than Frequency.

Since Recency has both the highest MDA and MDG, we can conclude that it is the most important feature when making classification decisions. As for the other two variables, they have reversed rankings when comparing MDA and MDG, meaning that both may be valuable and should be kept as part of the model.

SVM

```r
set.seed(123)
options(scipen = 999)

tune.out <- tune(svm, Class ~ ., data = train_data, kernel = "radial",
                 ranges = list(cost = c(0.1, 1, 10), gamma = c(0.01, 0.1, 1)))
tune_results <- as.data.frame(tune.out$performances)
```

```r
kable(tune_results, digits = 4, caption = "SVM Tuning Results") %>%
  kable_styling(full_width = FALSE, bootstrap_options = c("striped", "hover"))
```

Table 4: SVM Tuning Results

cost	gamma	error	dispersion
0.1	0.01	0.2387	0.0762
1.0	0.01	0.2387	0.0762
10.0	0.01	0.2292	0.0702
0.1	0.10	0.2387	0.0762
1.0	0.10	0.2291	0.0676
10.0	0.10	0.2216	0.0686
0.1	1.00	0.2387	0.0762
1.0	1.00	0.2081	0.0716
10.0	1.00	0.2157	0.0648

In lectures and Lab 9: SVM, we saw that the the value of `C` or `cost` can be used as a tuning parameter before fitting the SVM model. Since we do not know the optimal value of `cost`, I decided to use the `tune()` function as described in Lab 9: SVM, to perform 10-fold cross-validation. We see that `cost = 1` and `gamma = 1` results in the lowest cross-validation error rate and the lowest dispersion value as well. Thus I fit the SVM model with `cost = 1`, `gamma = 1`, and a radial kernel.

```r
best_cost <- tune.out$best.parameters$cost
svm_model <- svm(Class ~ ., data = train_data, kernel = "linear", cost = best_cost, probability = TRUE)
svm_pred <- predict(svm_model, test_data, probability = TRUE)

svm_cm <- confusionMatrix(svm_pred, as.factor(test_data$Class))
conf_matrix_df <- as.data.frame(svm_cm$table)
conf_matrix_df <- conf_matrix_df %>%
  select(Reference, Prediction, Freq) %>%
  slice(1, 3, 2, 4)
kable(conf_matrix_df, digits = 4, caption = "Confusion Matrix for SVM Model") %>%
  kable_styling(full_width = FALSE, bootstrap_options = c("striped", "hover"))
```

Table 5: Confusion Matrix for SVM Model

Reference	Prediction	Freq
0	0	168
1	0	48
0	1	3
1	1	5

We can see that even after tuning the model and using the best cost parameter, the prediction accuracy of the SVM is quite poor. Unlike the previous two models, the SVM model is extremely conservative at predicting anyone to be a donor. Only 7 of the 224 total predictions are predicted to be donors, and even then 3 of them are predicted wrong. The model also has 49 False Negatives compared to its 3 False Positives. This is not ideal for this specific problem since we would like to identify more people that donated blood and make sure that they come back as those that donated would have a higher chance of returning for more.

Model Evaluation

Comparison of Model Metrics

```r
# Logistic Regression
log_metrics <- c(
  log_cm$overall["Accuracy"],
  log_cm$byClass["Precision"],
  log_cm$byClass["Sensitivity"],
  log_cm$byClass["F1"],
  log_cm$byClass["Neg Pred Value"],
  log_cm$byClass["Specificity"]
)

# Random Forest
rf_metrics <- c(
  rf_cm$overall["Accuracy"],
  rf_cm$byClass["Precision"],
  rf_cm$byClass["Sensitivity"],
  rf_cm$byClass["F1"],
  rf_cm$byClass["Neg Pred Value"],
  rf_cm$byClass["Specificity"]
)

# SVM
svm_metrics <- c(
  svm_cm$overall["Accuracy"],
  svm_cm$byClass["Precision"],
  svm_cm$byClass["Sensitivity"],
  svm_cm$byClass["F1"],
  svm_cm$byClass["Neg Pred Value"],
  svm_cm$byClass["Specificity"]
)

model_comparison <- data.frame(
  Metric = c("Accuracy", "Precision (Pos)", "Recall (Pos)", "F1 Score",
             "Precision (Neg)", "Recall (Neg)"),
  Logistic_Regression = log_metrics,
  Random_Forest = rf_metrics,
  SVM = svm_metrics
)
```

```
model_comparison_table <- kable(model_comparison, caption = "Comparison of Model Performance Metrics", a
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"))
model_comparison_table
```

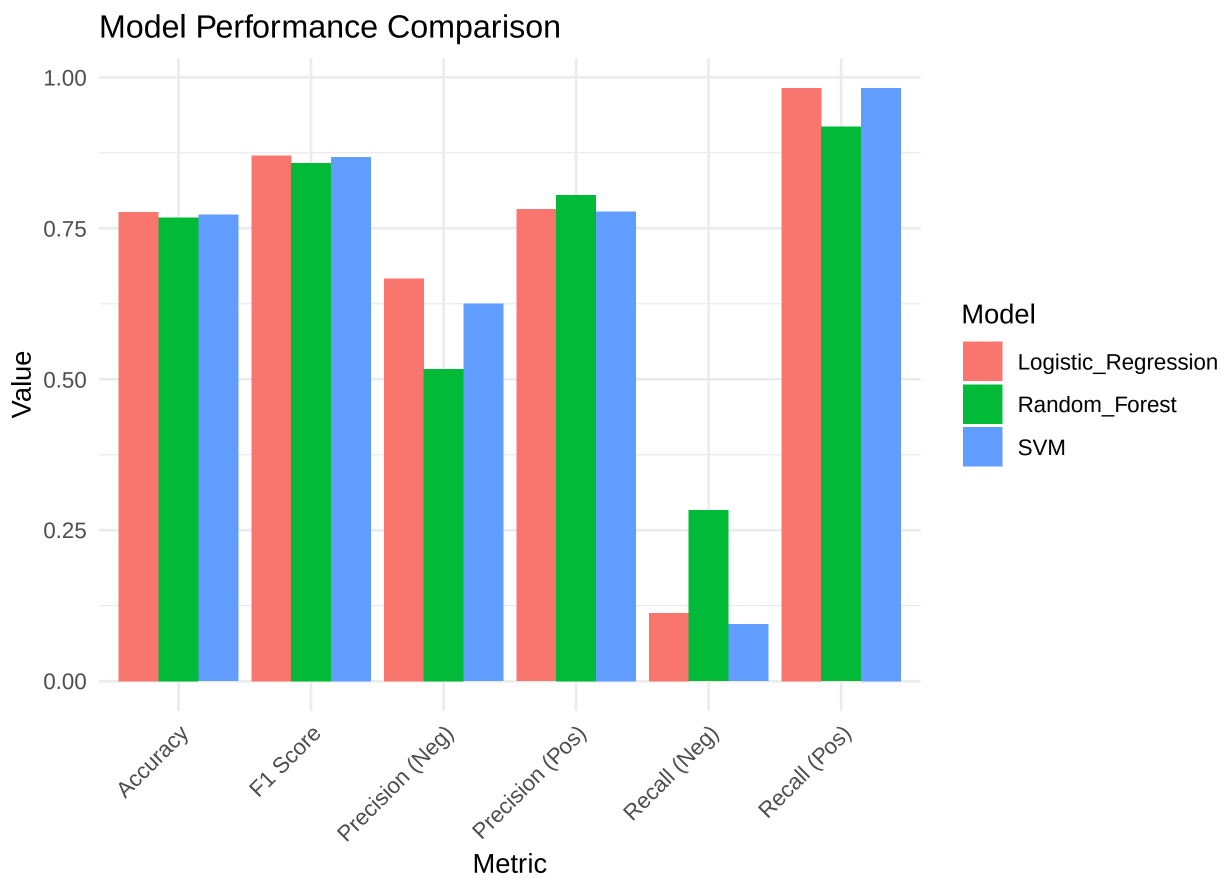
Table 6: Comparison of Model Performance Metrics

	Metric	Logistic_Regression	Random_Forest	SVM
Accuracy	Accuracy	0.7768	0.7679	0.7723
Precision	Precision (Pos)	0.7814	0.8051	0.7778
Sensitivity	Recall (Pos)	0.9825	0.9181	0.9825
F1	F1 Score	0.8705	0.8579	0.8682
Neg Pred Value	Precision (Neg)	0.6667	0.5172	0.6250
Specificity	Recall (Neg)	0.1132	0.2830	0.0943

The following results and formulas were all taken from Lecture 6: Logistic Regression.

- Note that for this table, the Negative and Positive values are switched, meaning that the Positive values are all for when the person has not donated, while the Negative values are for those that did donate.
- Predictive accuracy, or accuracy for short, is a measure of how often the model is correct overall (Bratko 1997). The predictive accuracy score is $\frac{\text{TP + TN}}{\text{P + N}}$. The three models have very similar accuracy, with logistic regression showing a slightly higher accuracy than the rest at 77.68% compared to the other two models at 76.79%.
- The positive precision score, or Positive Predictive Value (PPV) is $\frac{\text{TP}}{\text{TP + FP}}$. The positive precision score measures the accuracy of all positive results - that is, how often the model is correct when it presents a positive result. In this case, the Random Forest, which was my most aggressive model, ended up with the highest PPV at 80.51%, while the logistic regression (78.14%) and SVM (77.42%) were still close behind it. This means that even though the Random Forest was extremely aggressive, it was still able to make predictions clearly.
- The positive recall score is $\frac{\text{TP}}{\text{TP + FN}}$. The recall measures how good the models are at avoiding misclassifying donors. As we can see, all of the models have very high recall values above 90%, with the Logistic Regression and SVM both at 98.25% and the Random Forest at 91.81%. There is a tradeoff between PPV and Recall, but there are no drastic results in this case.
- The F1 score is $\frac{\text{2 * Precision * Recall}}{\text{Precision + Recall}}$. It is a balanced measure that takes into account both Precision and Recall. In our case, the Logistic Regression comes out on top with a score of 0.8705. The SVM is close behind at 0.866 due to its lower precision, while the Random Forest is not far behind as well with a score of 0.8579. All of the models are comparable when using this metric, even though it slightly favors the Logistic Regression over the others.
- The negative precision score, or Negative Predictive Value (NPV) is $\frac{\text{TN}}{\text{TN + FN}}$. As we can see in the formula, if the number of False Negatives is much larger than the number of True Negatives, this value will decrease. That is exactly what I observed when initially evaluating the models. In this case, with the definitions flipped, this score is measuring the strength of the predictions that were predicted to be donors. Since each of the NPV values are higher than 50%, the models show some promise in classifying donors. Specifically, the logistic regression does by far the best at $\frac{2}{3}$ accuracy.
- The negative recall score is $\frac{\text{TN}}{\text{TN + FP}}$. In this case, it is measuring how many donors the model predicted compared to all true donors. In this case, only the random forest has a respectable value at 28.3%, which is still quite low, while the other two models are extremely low. This shows that the models are poor at identifying true donors.

```r
# Model Comparison Barplot
model_comparison_long <- reshape2::melt(model_comparison, id.vars = "Metric")
ggplot(model_comparison_long, aes(x = Metric, y = value, fill = variable)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Model Performance Comparison", x = "Metric", y = "Value", fill = "Model") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



We can see that the models have similar accuracy, F1 score, positive precision, and positive recall, but there are slight differences in the other metrics. Namely, the logistic regression has a much higher negative precision than the other two models, while the Random Forest has a much higher negative recall than the other two models.

AUC Comparison

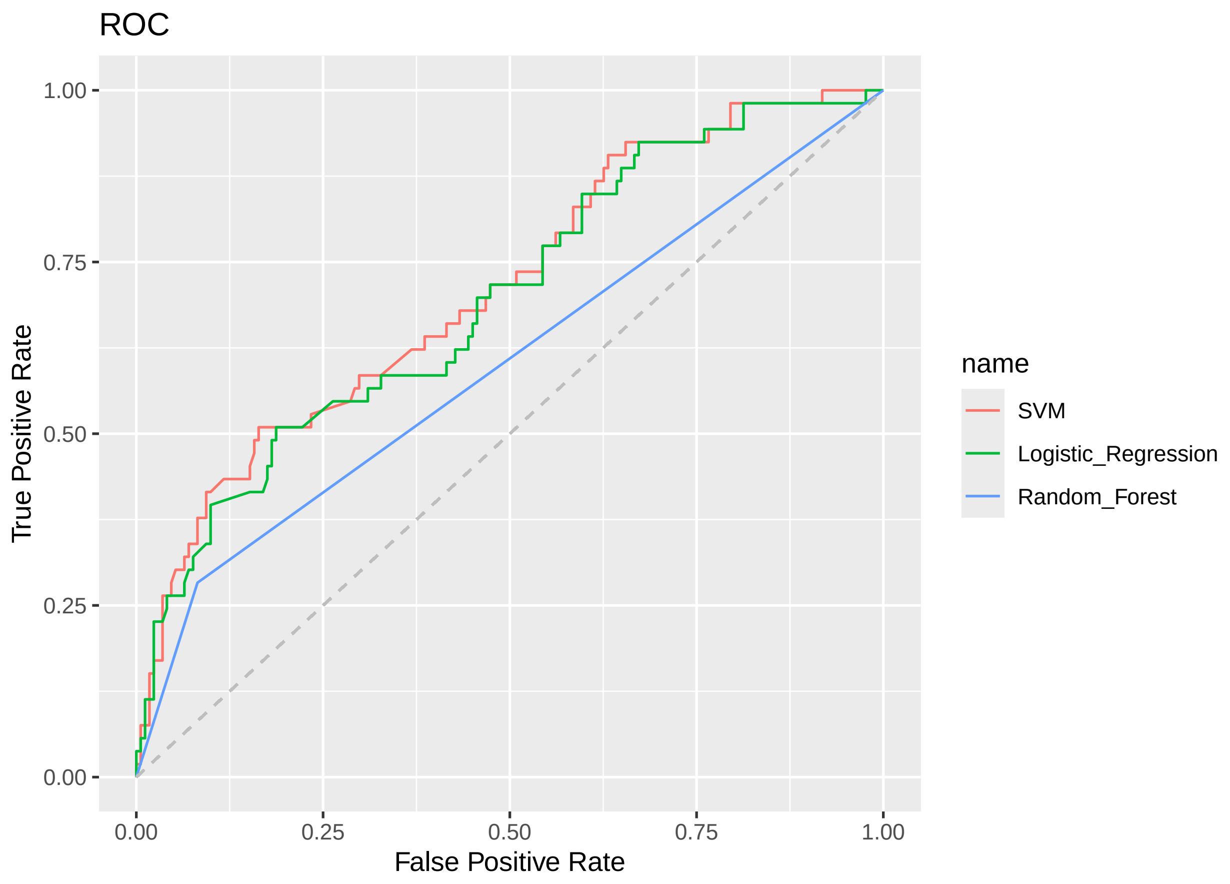
```r
log_roc <- roc(test_data$Class, log_pred)
rf_roc <- roc(test_data$Class, as.numeric(rf_pred))
svm_roc <- roc(test_data$Class, attr(svm_pred, "probabilities")[,2])

auc_values <- data.frame(
  Model = c("Logistic Regression", "Random Forest", "SVM"),
  AUC = c(log_roc$auc, rf_roc$auc, svm_roc$auc)
)
kable(auc_values, col.names = c("Model", "AUC"))
```

Model	AUC
Logistic Regression	0.6962
Random Forest	0.6006
SVM	0.7114

Surprisingly, the extremely conservative SVM model actually has the highest AUC value of 0.71, while the Logistic Regression is not far behind at 0.696. The Random Forest model suffers in this comparison due to its aggressive nature.

ROC Curves

```r
ggroc(list(SVM = svm_roc, Logistic_Regression = log_roc, Random_Forest = rf_roc), legacy.axes = TRUE) +
  labs(title="ROC", y = "True Positive Rate", x = "False Positive Rate") +
  geom_segment(aes(x=0, xend=1, y=0, yend=1), color="grey", linetype="dashed")
```



As expected, the Random Forest has the worst performance, while the SVM and Logistic Regression have almost identical curves. All three models are significantly better than the random classifier.

Training and Test MSE

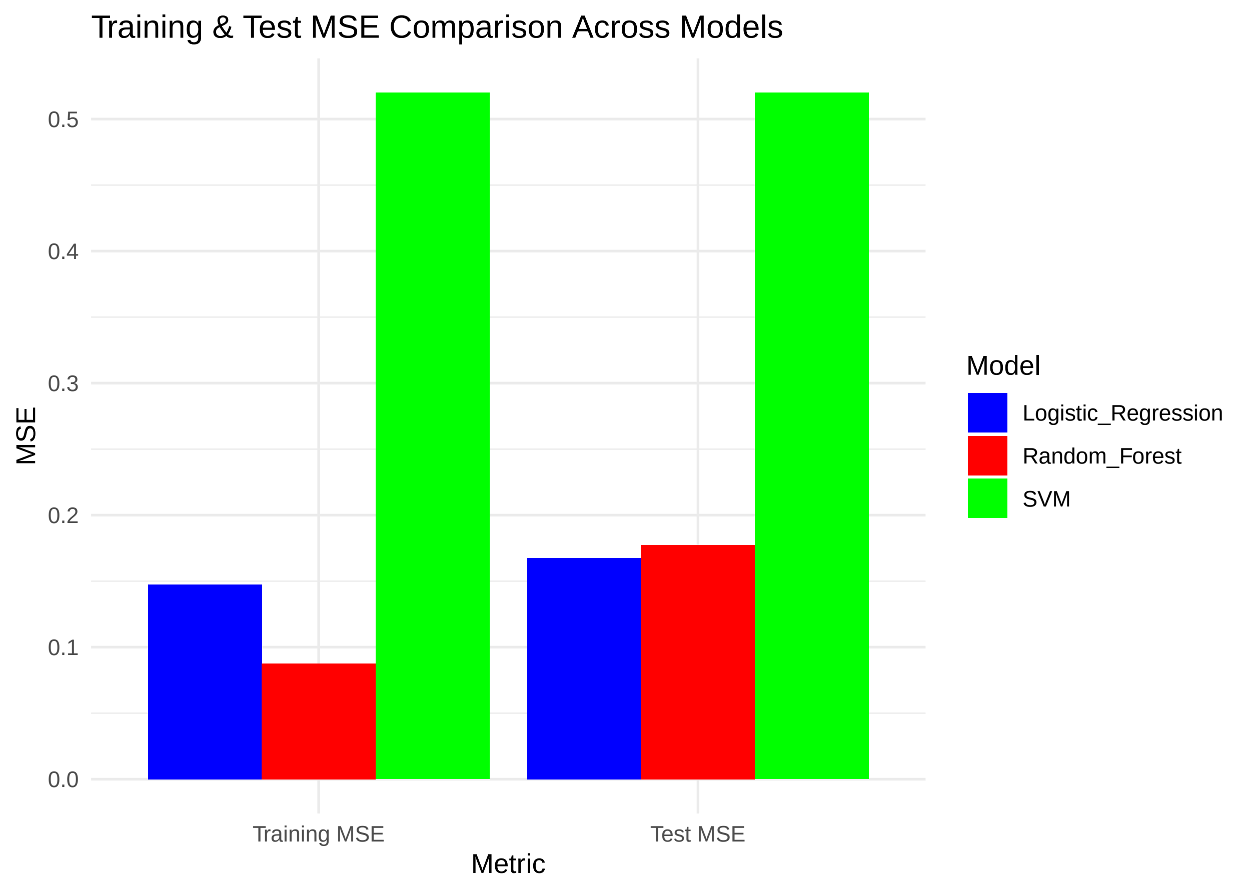
```r
train_data$Class <- as.numeric(as.character(train_data$Class))
test_data$Class <- as.numeric(as.character(test_data$Class))

# Logistic Regression Predictions
log_train_pred <- predict(log_model, train_data, type = "response")
log_test_pred <- predict(log_model, test_data, type = "response")

log_train_mse <- mean((train_data$Class - log_train_pred)^2, na.rm = TRUE)
log_test_mse <- mean((test_data$Class - log_test_pred)^2, na.rm = TRUE)

# Random Forest Predictions
rf_train_pred <- predict(rf_model, train_data, type = "prob")[,2]
rf_test_pred <- predict(rf_model, test_data, type = "prob")[,2]

rf_train_mse <- mean((train_data$Class - rf_train_pred)^2, na.rm = TRUE)
rf_test_mse <- mean((test_data$Class - rf_test_pred)^2, na.rm = TRUE)

# SVM Predictions
svm_train_pred <- predict(svm_model, train_data, probability = TRUE)
svm_train_prob <- attr(svm_train_pred, "probabilities")[,2]

svm_test_pred <- predict(svm_model, test_data, probability = TRUE)
svm_test_prob <- attr(svm_test_pred, "probabilities")[,2]

svm_train_mse <- mean((train_data$Class - svm_train_prob)^2, na.rm = TRUE)
svm_test_mse <- mean((test_data$Class - svm_test_prob)^2, na.rm = TRUE)

mse_df <- data.frame(
  Metric = c("Training MSE", "Test MSE"),
  Logistic_Regression = c(log_train_mse, log_test_mse),
  Random_Forest = c(rf_train_mse, rf_test_mse),
  SVM = c(svm_train_mse, svm_test_mse)
)

kable(mse_df, caption = "Comparison of Training and Test MSE for All Models") %>%
  kable_styling(full_width = FALSE)
```

Table 8: Comparison of Training and Test MSE for All Models

Metric	Logistic_Regression	Random_Forest	SVM
Training MSE	0.1476	0.0875	0.5199
Test MSE	0.1675	0.1775	0.5199

While the logistic regression and random forest have acceptable MSEs, the SVM shows concerning MSEs above 0.5. This means that the SVM is likely not a reliable method and is largely dependent on the random seed provided to it.

```
mse_df$Metric <- factor(mse_df$Metric, levels = c("Training MSE", "Test MSE"))
mse_long <- melt(mse_df, id.vars = "Metric")

ggplot(mse_long, aes(x = Metric, y = value, fill = variable)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Training & Test MSE Comparison Across Models",
       x = "Metric",
       y = "MSE",
       fill = "Model") +
  theme_minimal() +
  scale_fill_manual(values = c("blue", "red", "green"))
```



Outside of the obvious problem with the SVM model, we can see that the random forest may be overfitting as its training MSE is considerably lower than that of the logistic regression, but its test MSE is slightly higher. However, since the difference in test MSE is fairly insignificant, I can conclude that both models are usable, while SVM is not usable.

Conclusion

Results and Interpretation

After considering various model metrics and performance features, I can conclude that the logistic regression is likely the safest choice for this classification problem, while the random forest may also be usable. The main factors I used to make this decision were the considerable difference in AUC between the logistic regression and the Random Forest, the higher precision and accuracy without much difference in other performance metrics, and lower test MSE. The SVM is unusable due to its extremely high training and test MSEs compared to the other two models. Even though it had the highest AUC value, this is too big of an issue to ignore.

```r
log_coefficients <- summary(log_model)$coefficients
log_coeff_df <- as.data.frame(log_coefficients)
colnames(log_coeff_df) <- c("Estimate", "Std. Error", "Z value", "P-value")
kable(log_coeff_df, digits = 4, caption = "Logistic Regression Coefficients")
```

Table 9: Logistic Regression Coefficients

	Estimate	Std. Error	Z value	P-value
(Intercept)	-0.1871	0.2199	-0.8508	0.3949
Recency	-0.1367	0.0232	-5.8843	0.0000
Frequency	0.1205	0.0285	4.2273	0.0000
Time	-0.0222	0.0070	-3.1693	0.0015

We can see that each of the predictors are statistically significant, with p-values well below the common α level of 0.05. Thus, the model is reasonable to use. From the table, we see that both Recency and Time have negative effects on the probability of selecting a donor, but Recency has a much larger effect. This implies that for any given donor, their most recent donation rate will have a more significant impact on their donation decisions rather than when they first started to donate. On the other hand, the Frequency variable has a positive coefficient, meaning that a donor that donates more blood is predicted to be more likely to have donated in March 2007. This makes sense as people that donate more frequently would also be more likely to come back and donate even more. Thus, any marketing strategy should focus most heavily on contacting the most recent donors, with those that donate the most being a second priority.

Limitations

Since the dataset only had 4 potential features, two of which were correlated with each other, I was limited to just three usable predictors for this problem. This meant that the final model had to be more simple, which meant that the model was easy to interpret, but the tradeoff was poor predictive accuracy, particularly with the high False Positive Rate. In more modern applications, I would expect any given donor's decision to be influenced by other variables such as demographics, income, health conditions, or the effect of marketing campaigns on their behavior. Thus, for further research I would want to utilize a dataset which included these features and/or others so I could build a more complex model.

Another potential issue is the model finding patterns where there are none. Even though the predictors in the final model are statistically significant and the initial data visualization did show some trends in the variables, it does not mean that these particular features are the only predictors of the donation status. Also, predicting whether or not the given person donated does not necessarily mean that they will return to donate more, even though that is what the trends in the data indicate. As before, more data is required to research whether the given donors would actually return and how often they would return.

Finally, the data was collected in Hsin-Chu City in Taiwan, almost 20 years ago. This means that any policies or actions taken in light of findings derived from this data set may not apply to the modern world or other countries outside of Taiwan. However, since the models are fit generally, I believe that it would be simple enough to switch datasets and reconstruct the models.

Applications

A potential application of this model is in the intersection of health policy and marketing. The Recency, Frequency, and Monetary (RFM) marketing strategy examines "when (recency), how often (frequency) and in what purchase amount (monetary) purchase are made" (Yang 2004). In this case, the dataset had perfect correlation between the frequency and monetary variables, so I chose to omit the latter in my analysis. Either way, I found in my logistic regression model that those who donated more recently and those that donate more often are more likely to have responded to the surveyed blood drive and thus should be looked at more closely. This is a promising start for policymakers, as they know that they can approach their efforts on those that did and did not donate very differently. While those that did donate likely need simpler, quicker reminders to get them involved again, the bulk of any marketing or policy changes should be aimed at those that did not donate. One potential improvement could be monetary rewards for blood donation if a large, nationwide push is needed; however, the costs of a strategy like this might outweigh the potential benefits. More options could be TV ads or education in the workplace about the topic. By utilizing the given machine learning model, policymakers would know how to segment their policies and could predict how people would respond to their policies.

Even though the RFM marketing strategy is effective at a base level, Yang goes on to state that even though the RFM model is "a tried-and-true analysis with which almost every direct marketer begins", it is still "crude" and less effective than other potential methods. This is where the potential for improvement in this model lies. As I mentioned in the *Limitations* section, I believe that incorporating more predictors would lead to a more complex, but also more complete model. I believe that a model based on RFM strategy is a good starting point for public health policymakers, but they should investigate health effects deeper with more predictors in order to fully understand both why their policies would be effective and the potential impacts of their proposed policies. Overall, the final model in this project could be used to have a positive societal impact, namely by encouraging more people to donate blood.

Citations

- Nicodemus, Kristin K. "On the stability and ranking of predictors from random forest variable importance measures." Briefings in bioinformatics 12.4 (2011): 369-373.
- Rodriguez-Galiano, Victor Francisco, et al. "An assessment of the effectiveness of a random forest classifier for land-cover classification." ISPRS journal of photogrammetry and remote sensing 67 (2012): 93-104.
- Yang, Amoy X. "How to develop new approaches to RFM segmentation." Journal of Targeting, Measurement and Analysis for Marketing 13 (2004): 50-60.
- Yu, Guo. "Lab 9: SVM". PSTAT 131/231: Introduction to Statistical Machine Learning (2025).
- Yu, Guo. "Lecture 6: Logistic Regression". PSTAT 131/231: Introduction to Statistical Machine Learning (2025): 22.
- Yu, Guo. "Lecture 13: Tree-based Methods". PSTAT 131/231: Introduction to Statistical Machine Learning (2025): 9.