Exploring the Impact of Social Determinants of Health On Mental Well-Being Post-Pandemic Using Random Forests and Decision Trees

by

Kathryn M. Conway

Presented in Partial Fulfillment of the
Requirements of Senior Independent Study
in Statistical and Data Sciences

Advised by Dr. Marian Frazier
Department of Mathematical and Computational Sciences
Summer 2024

ABSTRACT

With data from the U.S. Census Bureau’s Household Pulse Survey, an investigation into the impact of social determinants of health on the mental well-being is launched on a post-pandemic population. The principles of random forests and decision tree statistical models working alongside machine learning algorithms highlight contributing factors to indicate a need for intervention resource mobilization.

ACKNOWLEDGEMENTS

Dedicated to my beloved grandmother Diane M. Baylis, 05/08/1944 – 12/10/2022: putting me on the right path.

This would not have been possible without Dr. Marian Frazier. Had it not been for her, this paper would not have been written; I wouldn't even be capable of writing this paper. I wouldn't be on the path I am today, studying the subject I'm discussing; I don't think I would have graduated on time. Because of Dr. Frazier's support, guidance, and dedication, I have become the person that I am today, confidently pursuing goals I once dreamed to accomplish. Thank you for your patience, your understanding, your kindness- it was what I needed.

Thank you to all my professors at the College of Wooster- yes, *all* of them. They each taught me something valuable. Many shared lessons I will carry with me for the rest of my life, even if only about the way they made me feel. I will always remember the dedication of the professors in the Mathematics and Computational Sciences Department- I will treasure my academic experience forever due to their passion.

Thank you to my family, of course; I am all that I am because of them. Moving far away for college felt like a great idea at the time, but things seem to have changed now and I wouldn't mind spending more time together. People generally tend to spend less time with their families as they grow older, but I hope we can hinder the trend a bit longer.

Thank you to my friends, my loved ones who supported me through it all. During both good and bad times, I feel incredibly lucky to be understood by those around me.

TABLE OF CONTENTS

ABSTRACT .... ***i***

ACKNOWLEDGEMENTS .... ***ii***

TABLE OF CONTENTS .... ***iii***

LIST OF FIGURES .... ***v***

1. INTRODUCTION ***1***

2. LITERATURE REVIEW ***1***

3. METHODOLOGY ***6***

3.1 LINEAR REGRESSION **6**

3.1.1 LINEAR REGRESSION EXAMPLE 8

3.2 DECISION TREES **11**

3.2.1 DECISION TREE EXAMPLE 13

3.3 RANDOM FORESTS **17**

3.3.1 RANDOM FOREST EXAMPLE 18

3.4 GINI IMPURITY **20**

3.4.1 GINI IMPURITY EXAMPLE 20

4. DATA ***22***

5. RESULTS ***28***

6. CONCLUSION ***35***

7. BIBLIOGRAPHY 38

8. APPENDIX 42

8.1 IRIS EXAMPLES 42

8.2 DATA CLEANING........ 55

8.2.1 MULTI-SELECT QUESTIONS........ 56

8.2.2 IMPUTATION........ 68

8.3 MODELING 72

8.3.1 RANDOM FORESTS........ 72

8.3.2 DECISION TREES........ 128

LIST OF FIGURES

Figure 1.3: Plot of Simple Linear Regression Model for Iris Dataset 9

Figure 2.3: Plots of Multiple Linear Regression Model for Iris Dataset 10

Figure 3.3: Diagram of a decision tree with key terminology, sourced from (Kosarenko, 2021). 12

Figure 4.3: Decision Tree for Iris Dataset Predicting Petal Length.. 14

Figure 5.3: Values of Example Iris, a Random Observation from Iris Dataset........................... 14

Figure 6.3: Decision Tree for Iris Dataset Predicting Species.. 16

Figure 7.3: Confusion Matrix of Decision Tree for Iris Dataset Predicting Species................... 17

Figure 8.3: Variable Importance Plot for Random Forest Predicting Species from Iris Dataset.. 19

Figure 9.3: Confusion Matrix for Random Forest Predicting Species from Iris Dataset 19

Figure 10.3: Matrix of Values for Random Forest Predicting Species from Iris Dataset............ 21

Figure 11.4: Description Table of Risk Response Variables.. 24

Figure 12.4: Variable Creation Charts 25

Figure 13.4 Variable Recoding Chart 27

Figure 14.5: Top Ten Variable Importance Charts.. 29

Figure 15.5: Top Variable Summary Table 30

Figure 16.5:Description of Top Predictor Variables... 31

Figure 17.5: Decision Trees for Predicting Risk (Including Mental Health Variables) 32

Figure 18.5: Decision Trees for Predicting Risk (Excluding Mental Health Variables)............. 34

1. INTRODUCTION

In 2020, the outbreak of the COVID-19 pandemic disrupted the lives of nearly every person worldwide; the United States was no exception. The shutdowns took a devastating toll on the economy and society alike, and millions of Americans were personally and devastatingly impacted by this experience. In response to the pandemic, the U.S. Census Bureau collaborated with other federal agencies to produce the Household Pulse Survey to further explore the impact of social and economic factors on households and communities. The discussion of social and economic inequalities is far from new, with widespread disparities in the accessibility and quality of healthcare increasingly common among traditionally marginalized communities. The Household Pulse Survey provides a unique source of expanding the understanding of social determinants of health and direct impacts on the population, including variables related to mental health. Applying statistical analysis techniques and using software with the data generated from the context of the pandemic, associations between social determinants of health and mental well-being can be uncovered.

2. LITERATURE REVIEW

It should come as no surprise to many that the healthcare system in the United States is riddled with inequality. The level of access to care for many individuals is hindered by a variety of factors, such as insurance, geographic isolation, or cultural attitude. The factors that affect access to healthcare are referred to as social determinants of health. Social determinants of health can be defined as social, political, and economic factors that contribute to the health of individuals and communities (Humber, 2019). These circumstances are enabled through social policy and unfair economics. Determinants can include daily living conditions, such as healthy physical environments, fair and safe employment, as well as access to social protections, particularly healthcare. A lack of equity in health programs is fueled through an unequal distribution of power and resources, which historically and disproportionately impacts the working class and communities of color (Humber, 2019). The maldistribution of services like healthcare, housing, and income has severe and impactful consequences in physical health. Inadequate living circumstances, such as diet and housing, have been shown to decrease life expectancy and increase risks of cardiovascular and respiratory diseases. These issues profoundly affect individuals and communities enduring the hardships of economic disenfranchisement or societal marginalization (Humber, 2019).

As a result of poor social policy, unequal economics, and less-than-favorable politics, the maldistribution of healthcare has led to a lack of access for those that need it (Humber, 2019). The health of a population evolves in relationship to society; as physical makeup is deeply influenced by the culture in which we live. The conditions in which individuals live and work have a direct impact on both physical and mental well-being. This is clearly illustrated when examining the effects of homelessness. Proven consequences of insufficient or unstable housing circumstances include severely decreased life expectancies, along with heightened susceptibility to a variety of diseases and a ample conditions for unchecked deterioration of mental health (Humber, 2019). Economic and employment factors contribute largely to housing circumstances and physical health, and the stress of debts can have tangible effects on those experiencing hardship.

In Kimberlé Crenshaw's foundational work *Mapping the Margins of Intersectionality,* she introduces the topic of intersectionality and identity politics. She notes how identity categories such as race, gender, class, and sexuality can converge to create circumstances in which oppression works specifically upon the coexisting identities. Crenshaw describes how broad identity politics, such as feminism or race issues, can tend to either conflate or ignore intragroup differences. Treating these issues as mutually exclusive can lead to the further marginalization of coexisting identities seen as conflicting: “the consequence of the imposition of one burden that interacts with preexisting vulnerabilities to create yet another dimension of disempowerment,” (Crenshaw, 1991).

Empowerment and reconstruction can be achieved through social discourses delineating and identifying differences to highlight overlooked needs. Basing outreach strategies on the experiences of particular people or identities can in turn limit access to those in need that face differing backgrounds and obstacles: “uniform standards of need ignore the fact that different needs often demand different priorities in terms of resource allocation,” (Crenshaw, 1991). These failures of further analysis can lead to misrepresentations of the perceived severity of a problem and can often be conflated by disingenuous attempts at justification through poor use of inaccurate data.

The emergence of the COVID-19 pandemic in 2020 offers a unique launching point for investigation into the direct influence of social determinants of health. The profound social and economic impacts on the country caused by the pandemic have illustrated the unequal distribution of power and resources in the United States. The pandemic had a tremendous effect on the economy, resulting in increased unemployment rates and leading to disastrous consequences for families and individuals, as widespread food scarcity and housing insecurity became increasingly common.

In response to the pandemic, the U.S. Census Bureau collaborated with other federal agencies to produce the Household Pulse Survey with the goal of collecting data to measure the social and economic impacts of the pandemic on U.S. households (Bureau, n.d.-a). The survey ranged from basic demographic information to questions about health, education, and food security. Additionally, questions were included with the intent to gauge the mental health of respondents (Bureau, n.d.-d). Numerous independent analyses of the survey results yielded several conclusions about the circumstances faced by specific portions of the population, in line with previous statements regarding social determinants of health.

Analyses of the survey reported that the extent and severity of economic turmoil created by the COVID-19 pandemic were disproportionately devastating to Black, Latino, Indigenous, and immigrant households. With the majority of jobs being lost in lower paying industries, 1 in 5 renters were left behind on payments, a statistic more likely to occur for renters of color (Priorities, 2021). The presence of children in the household resulted in higher rates of hardship, specifically citing food insecurity due to cost difficulty (Priorities, 2021). The prevalence of hardships among marginalized families with children is startling, as there is a large potential for long-term negative consequences in the development of youths.

Beyond economic struggles, the social effects of the pandemic had profound consequences on the mental health of the population. The complete disruption of daily life combined with the social isolation of quarantines and shutdowns served as a possible stressor, posing significant consequences for those with pre-existing mental health struggles (Ramos, 2022). These effects on daily living had the potential to be devastating, especially considering the shortage of access to medical and mental health treatment occurring at the time. Exacerbated anxiety and depression symptoms have the possibility to impact aspects of physical health and cognitive function, social isolation is now considered a public health concern (Ramos, 2022).

A 2014 publication based on a 2002/2003 community health survey conducted by Statistics Canada used statistical analyses to explore how interactions of socioeconomic factors- social determinants of health- impacted the rates at which individuals seek mental health treatment (Cairney, 2014). In comparing logistic regression models with classification and regression trees (CART), the study highlighted the shortcomings of linear models in adequately identifying complex interactions between social determinants, ultimately unhelpful in considering the intersectional nature of the issues at hand. Alternatively, CART analysis supported the existence of complex interactions of such variables. This analytical approach had the potential to identify underserved groups with low propensities to seek care (Cairney, 2014). The results were consistent with the theory of intersectionality, concluding that health outcomes "are differently affected by multiple, interacting facets of social advantage and disadvantage," (Cairney, 2014).

The 2014 study on community health surveys using classification and regression tree models serve as a launching point to explore how the impacts of the COVID-19 pandemic have affected the American population's mental health concerns. In a similar effort to identify underserved populations with a reluctance or inability to seek care, the following study will use classification and regression trees to analyze the data resulting from the Household Pulse Survey. The goal is to utilize statistical methods and machine learning software to identify factors correlating to the need/s of increased mental health interventions, with implications to the means and methods of public health policy.

3. METHODOLOGY

3.1 LINEAR REGRESSION

The following section is sourced from *STAT2: Building Models for a World of Data* by Cannon et. al. (2013). The most common method utilized to predict the relationship between variables is regression. Simple linear regression explores how two variables relate to each other, and can be modeled as follows:

$$\text{Data} = \text{Model} + \text{Error}$$

$$Y = f(x) + e$$

Here, $f(x)$ represents a function that estimates the response variable Y for a predictor variable x. Because this is *linear* regression, the function is in the form of a line with b_0 representing the intercept and b_1 representing the slope:

$$Y = \mathrm{b}_0 + \mathrm{b}_1 X + e$$

Equation 1

The machine learning software estimates the coefficient values b_0 and b_1 by minimizing the *sum of squared residuals* and fitting the least squares regression line. Essentially, this process measures how well a model predicts the actual observed outcome and selects a formula that minimizes error. The residual is equal to the difference between the observed y and predicted $\hat{y}$, as shown below

$$residual = (y - \hat{y})$$

Equation 2

The sum of squared residuals, or SSE for sum of squared errors, adds the squared values of each data point's calculated residual:

$$SSE = \sum (y - \hat{y})^2$$

Equation 3

To approximate the “typical” error in a linear regression model, the sum of squared residuals or SSE is used to estimate the standard deviation of the error term based on the fit of the least squares regression line. This is known as the regression standard error, modeled in *Equation 4*. The value in the denominator of the equation, $n - 2$, represents the degrees of freedom in the model. This adjusts the error to be correctly weighted with n number of observations.

$$\hat{s}_e = \sqrt{\frac{\sum (y - \hat{y})^2}{n - 2}} = \sqrt{\frac{SSE}{n - 2}}$$

Equation 4

Multiple Linear Regression uses multiple predictor variables to estimate the response variable Y. The formula is represented *Equation 5,* and differs from the formula for simple regression in *Equation 1* with the inclusion of multiple X variables. The addition of many variables alters the formula to approximate the regression standard error (*Equation 6*).

$$Y = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_k X_k + e$$

Equation 5

$$\hat{s}_e = \sqrt{\frac{\sum (y - \hat{y})^2}{n - k - 1}} = \sqrt{\frac{SSE}{n - k - 1}}$$

Equation 6

Another value commonly used to assess the fit of regression models is called the coefficient of determination, or R^2. The calculations for R^2, shown in *Equation 7*, are drawn from the estimated errors previously discussed, and functions to assess of how much of the variance in the response variable Y is explained by the regression model with predictors X. Essentially, R^2 measures the strength of the regression model against a baseline model, which would be the average value of Y with no predictors.

$$R^2 = \frac{Variability\ explained\ by\ the\ model}{Total\ variability\ in\ Y} = \frac{\sum(\hat{y}-\bar{y})^2}{\sum(y-\bar{y})^2}$$

$$= \frac{SSModel}{SSTotal} = \frac{SSTotal - SSE}{SSTotal} = 1 - \frac{SSE}{SSTotal}$$

Equation 7

Similarly to R^2, adjusted R^2 accounts for the number of predictor variables used in the model and observations in the dataset. This is more useful in multiple regression, given that adjusted R^2 penalizes complex models.

$$\bar{R}^2 = 1 - \left(\frac{(1-R^2)\cdot(n-1)}{n-k-1}\right)$$

Equation 8

Data must meet many conditions for regression to be a useful and accurate modeling tool. When visualized, the scatterplot of the data must follow a consistent linear pattern to meet the linearity conditions. The data and calculated residuals from least squares regression must additionally be normally distributed around zero and follow a constant and equal variance. Furthermore, all data points and observations must be assumed completely independent and random.

3.1.1 LINEAR REGRESSION EXAMPLE

The “Iris” dataset is a classic example commonly used to demonstrate a variety of machine learning techniques. The dataset contains 150 observations containing the measurements of four features of an iris flower in addition to a label classifying the species of the iris flower. The three species of iris flowers are Setosa, Versicolor, and Virginica, and the measurements of the length and width of the sepal and petal of the flower in centimeters.

Recall that Simple Linear Regression uses one response variable and one predictor variable to build a model in the form of *Equation 1*. In *Figure 1.3*, a linear model is fitted to the observations to predict petal length using sepal width of the flowers in the Iris dataset.

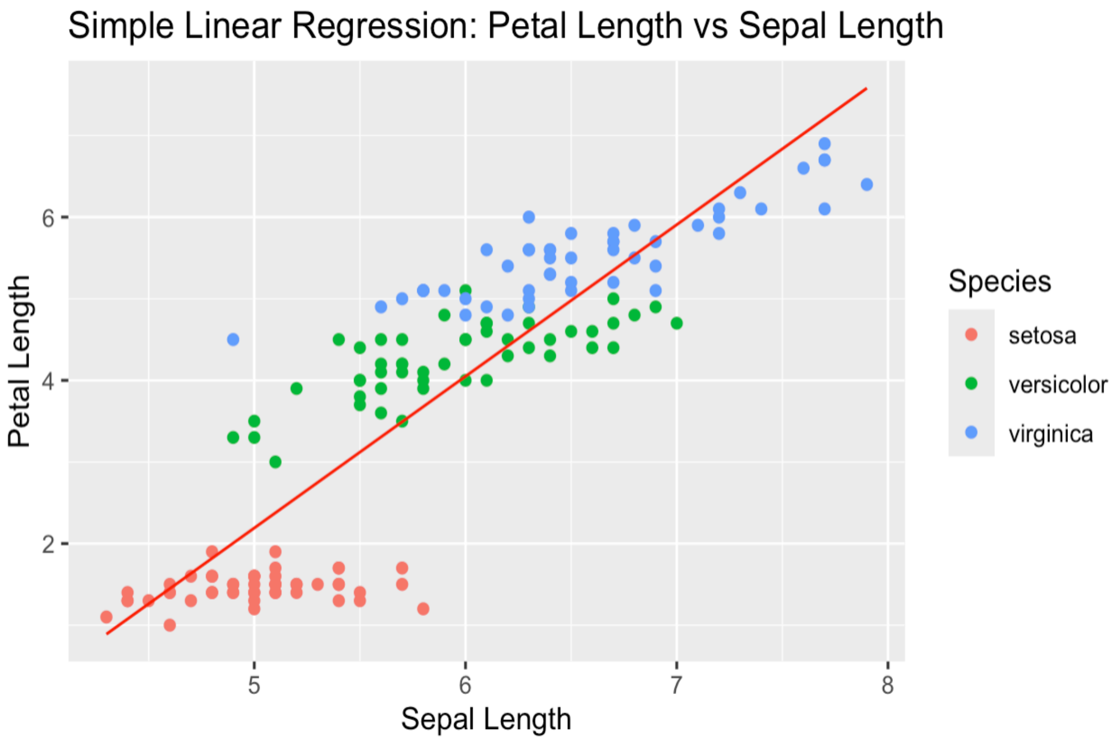


Figure 1.3: Plot of Simple Linear Regression Model for Iris Dataset

$$Petal.Length = 1.86(Sepal.Length) - 7.10 + \ e$$

The residual standard error calculated for this model is approximately 0.87 on 148 degrees of freedom. The resulting adjusted R^2 value is around 0.76. Looking at the graph, the three species of iris flowers, represented in different colors, have slightly varied characteristics. This makes a precise fit more complicated. In an effort to fit this layer of complexity, a multiple linear regression model can be built to predict petal length using the sepal length, sepal width, and petal width, as seen in *Figure 2.3*.

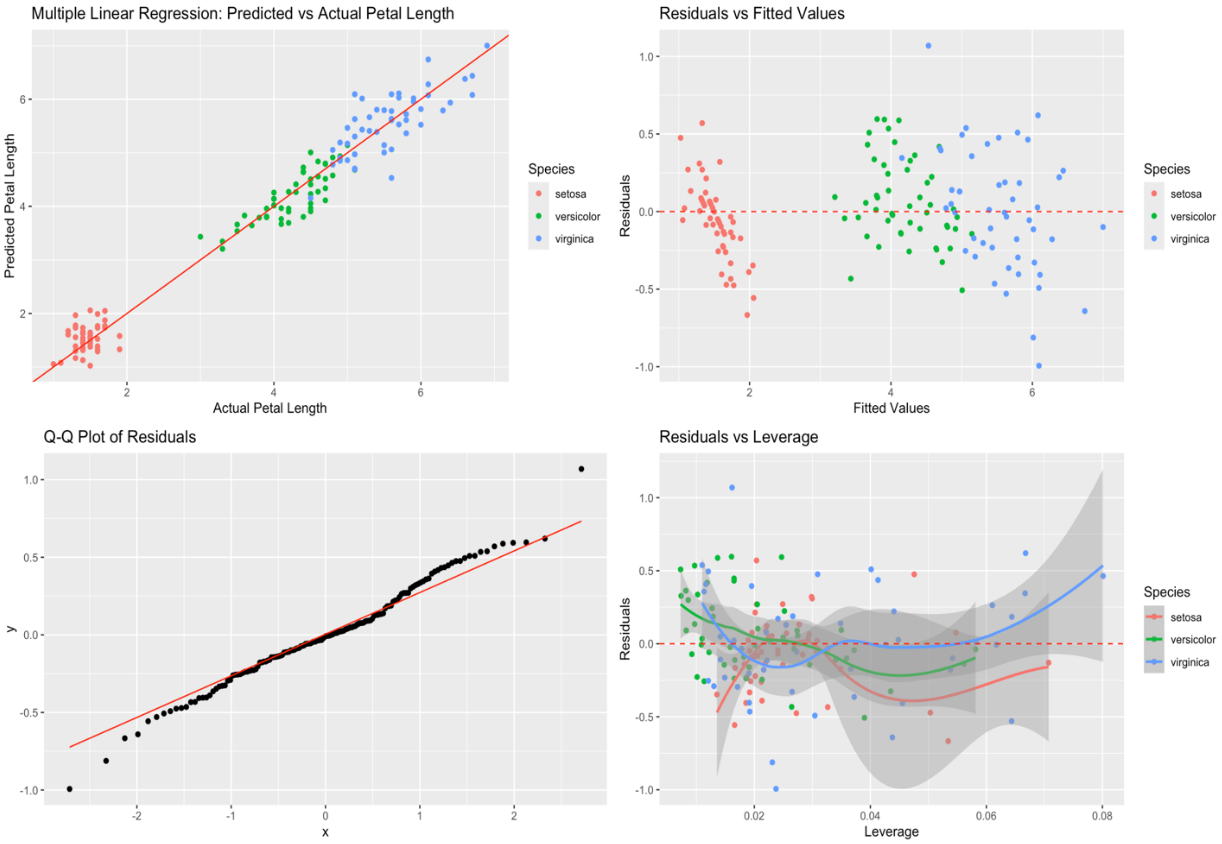


Figure 2.3: Plots of Multiple Linear Regression Model for Iris Dataset

$$Petal.Length = \ -0.26 + 0.73(Sepal.Length) - 0.65(Sepal.Width)$$
$$+ 1.45(Petal.Width) + \ e$$

For this model, the residual standard error is much smaller: approximately 0.32 on 146 degrees of freedom. Additionally, the adjusted R^2 improves significantly to about 0.97. The plots shown in *Figure 2.3* can be used to infer as to whether some of the conditions of linear regression are met. The condition for independence is assessed in the content of the data and data collection and cannot be assessed through a plot. In the residuals vs. fitted values plot, top right, the linearity condition is met if the points are randomly scattered around the horizontal line, indicating a linear relationship. In this same plot, the condition of equal variance expects that the spread of the data points around the horizontal line be approximately constant. Regarding normality, the residuals represented in the Q-Q plot, bottom left, must follow along the reference line. The plots of this example model indicate that linearity and equal variance can be accepted, as the data in the residual vs. fitted plot is evenly spread constantly along the horizontal line. The Q-Q plot, however, indicates a slight lack of normality, as the data seems to curve at the ends of the reference line.

Note that in *Figure 1.3* and *Figure 2.3* there are three clear groupings in the graph, delineated by color. The groupings slightly differ in their properties, as each represents a specific species of iris flower. The previous models based on regression techniques evaluate the overall properties of an iris flower to predict another property, without regard to the categories that differ within. This indicates a need for a method that can effectively account for categorization.

3.2 DECISION TREES

Contrary to the stringent conditions necessary for traditional regression, decision trees do not require assumptions regarding the distribution of data. The robust nature of decision trees leaves this method immune to outliers and irregular data, with the ability to determine variable relevance on its own using machine learning algorithms. Decision trees are perfect for use with complex data, with classification trees representing categorical data and regression trees focusing on numeric data.

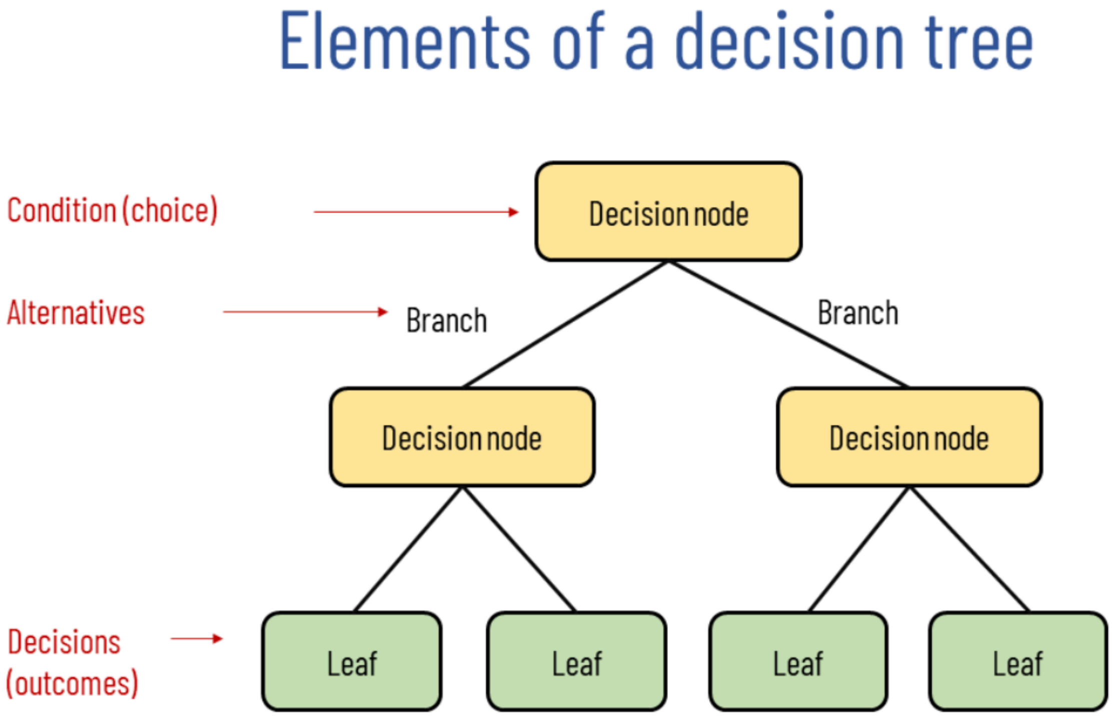


Figure 3.3: Diagram of a decision tree with key terminology, sourced from (Kosarenko, 2021).

A decision tree can be described simply as "visual map representing all paths to possible outcomes depending on a limited number of factors," (Kosarenko, 2021). This process is easily interpreted, built on binary yes/no decisions to visualize the predictive process. A binary decision is made on each variable, the node, and is partitioned into a subsequent group, or leaf, until the terminal node at which point no more splits can be made. This process is illustrated *Figure 3.3*. The decision tree methodology essentially splits the model into a piecewise function, producing a constant approximation of the outcome. Partitioning the data into these smaller groups allows for multiple unique models to be individually applied towards varieties of diverse variables to approximate the overall outcome in a single model, mathematically represented by the function in $f(x) = \sum_{i=1}^{M} c_m \, I(x \in Rm)$

Equation 9.

$$f(x) = \sum_{i=1}^{M} c_m \, I(x \in R_m)$$

Equation 9

In $f(x) = \sum_{i=1}^{M} c_m \, I(x \in Rm)$

Equation 9*,* $\mathrm{I}(x \in R_m)$ is interpreted as “for x in the subregion R_m” and c_m represents the estimated values of the response in region R_m for M nodes. The machine learning algorithm decides on the splits in the tree by minimizing the residual sum of squared errors, much like the process in regression methods. Let T represent a particular tree, so that $e_{(T)}$ is the total sum of squared errors of the tree T in *Equation 10*.

$$e_{(T)} = \sum_{i=1}^{N} [y_i - f(x_i)]^2$$

Equation 10

3.2.1 DECISION TREE EXAMPLE

Using the decision tree approach, the same variables used in the multiple regression Iris example can be fed to the machine learning algorithm to produce the model visualized in *Figure 4.3*. This model has an adjusted R^2 value of approximately 0.97, nearly equal to that of the multiple regression model. The decision tree uses only two of the three predictor variables given, indicating that the algorithm determined that a sufficient tree could be produced without one of the variables. This illustrates the ability of the algorithm to disregard unnecessary variables and highlight only those most significant.

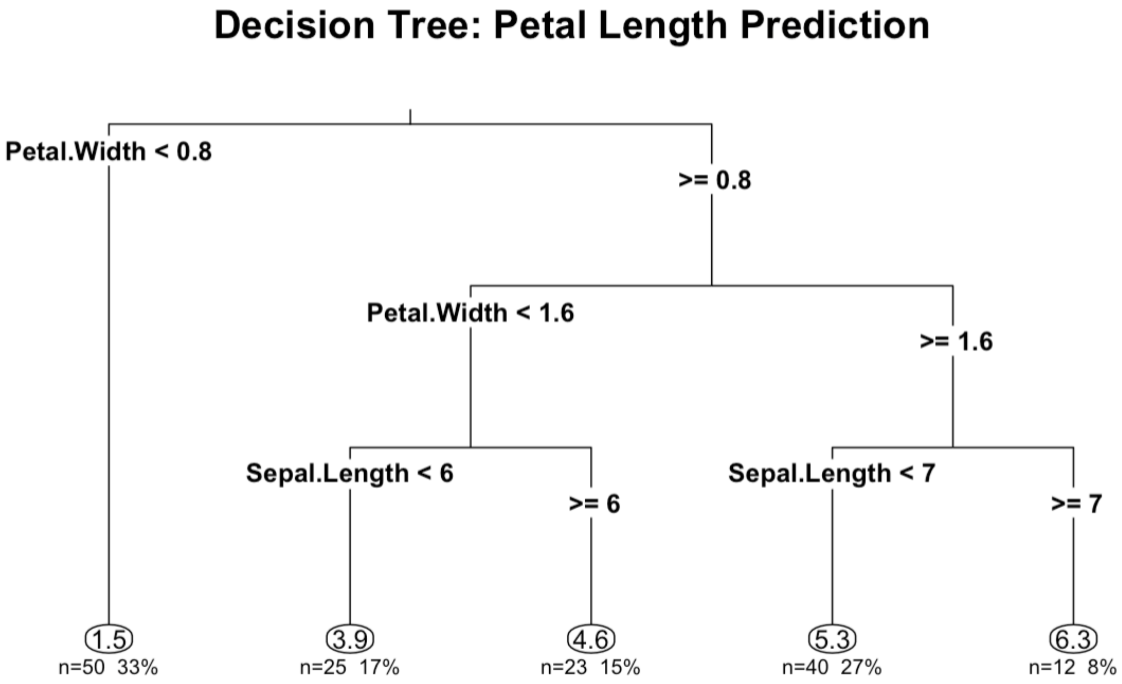
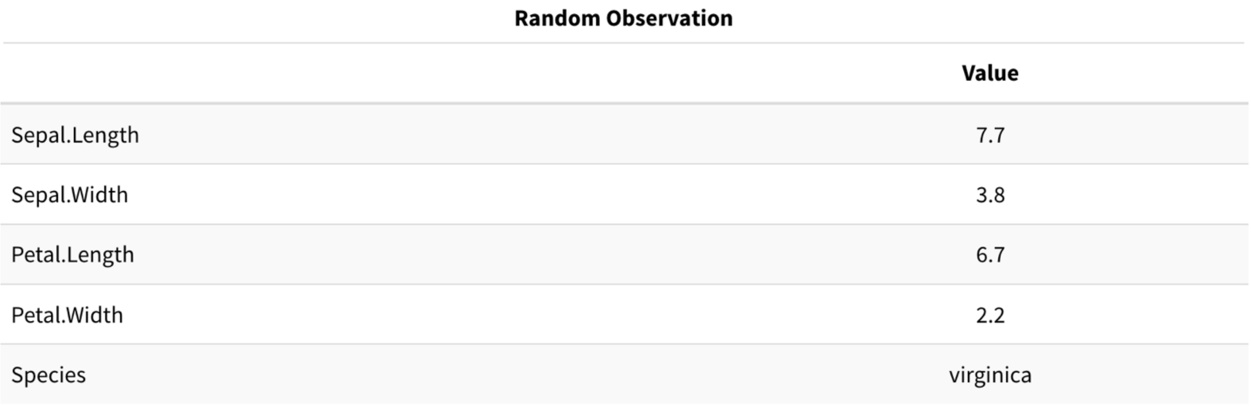


Figure 4.3: Decision Tree for Iris Dataset Predicting Petal Length

Consider the features of an example flower randomly selected observation from the Iris dataset, shown in the table in *Figure 5.3*.



Random Observation

	Value
Sepal.Length	7.7
Sepal.Width	3.8
Petal.Length	6.7
Petal.Width	2.2
Species	virginica

Figure 5.3: Values of Example Iris, a Random Observation from Iris Dataset

Starting at the root node in *Figure 4.3*, the first split occurs on the variable Petal.Width. Observations with a petal width less than 0.8 centimeters split into the left child node, and observations with a petal width greater than or equal to 0.8 centimeters go to the right child node.

The example iris in *Figure 5.3* is shown to have a petal width of 2.2 cm, so this iris will proceed to the right child node. The next split again assesses petal width. An iris with a petal width less than 1.6 cm goes to the left node, and greater than or equal to 1.6 cm goes to the right. Again, the example iris has a petal width of 2.2 cm, so it will go right. The final split refers to sepal length. An iris with a sepal length less than 7 cm goes to the left child, and greater than or equal to 7 goes to the right child. The example iris, with a sepal length of 7.7 cm, goes to the right. This is the terminal node, which predicts the petal length of the example iris to be 6.3 cm. This is relatively close to the actual petal length of the example iris, which is 6.7 cm. The values ($n = 12, 8\%$) presented under the terminal nodes in *Figure 4.3*, represent the population of the terminal node both in n number of observations and percentage of total sample size.

Figure 6.3 shows a decision tree model built to predict the species of an iris flower based off the given measurements. Linear regression models are unable to predict and classify data in this way.

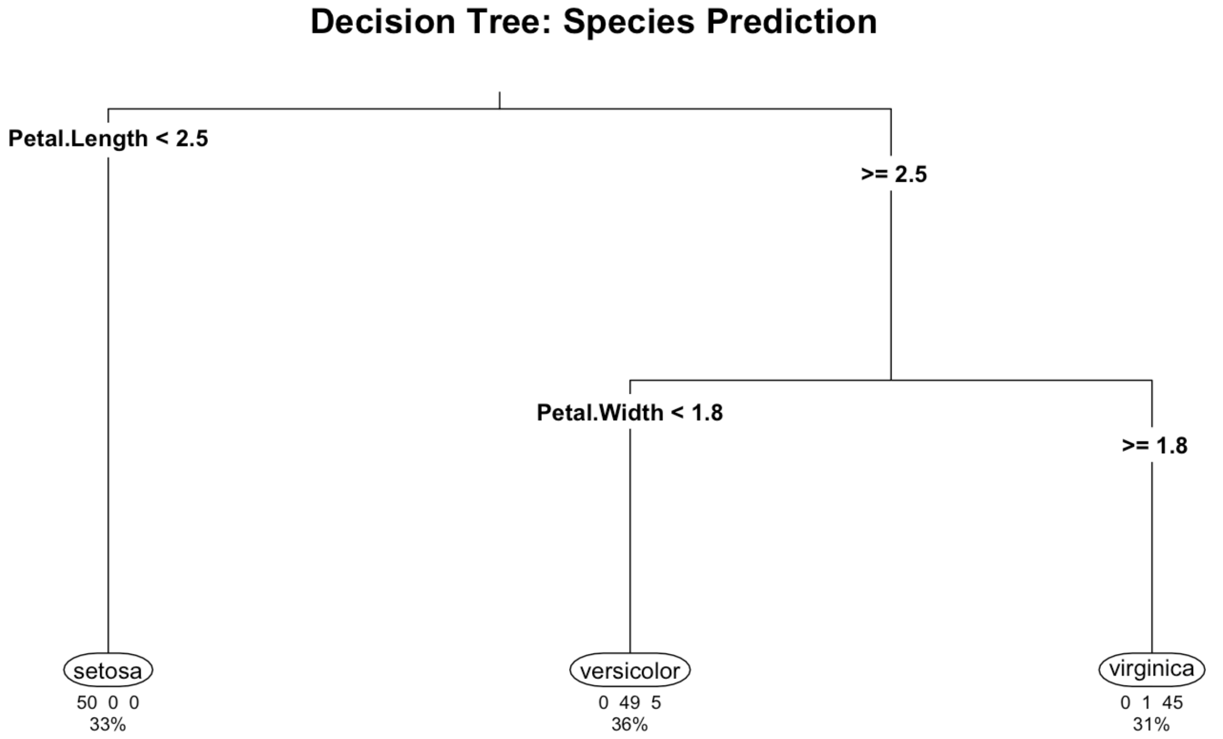
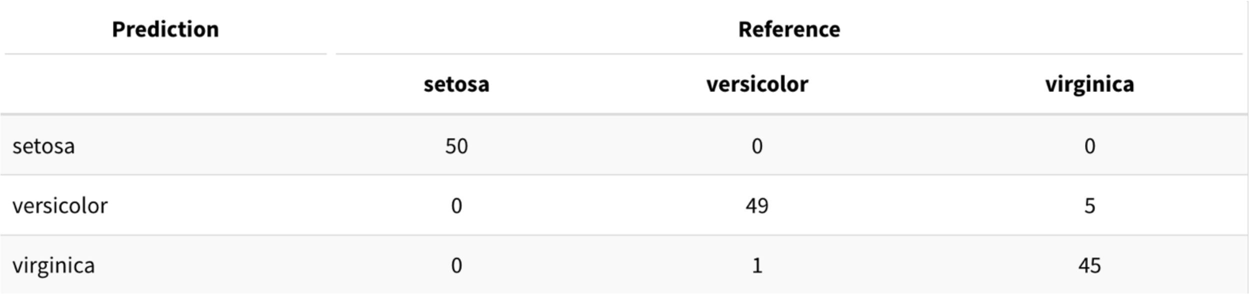


Figure 6.3: Decision Tree for Iris Dataset Predicting Species

Recall *Figure 5.3*, and the values of the example iris. The root node of the decision tree in *Figure 6.3* splits based on the value of petal length. The example iris has a petal length of 6.7 cm, so it will follow the tree to the right split, as 6.7 cm is greater than or equal to 2.5 cm. The next split is on the variable petal width. With the example iris having a petal width of 2.2 cm, it will go to the right node again as 2.2 cm is greater than or equal to 1.8 cm. This terminal node classifies the species of the example iris as virginica, which is correct. The percentages represented under the terminal nodes in *Figure 6.3* refer to the population of this node out of the total number of observations. The three numbers under each node represents the count of each species in the node. If the model was perfectly accurate, each terminal node would have exactly 50 observations of the same species and 0 of the two other species. This is the case in the leftmost node, which perfectly classifies all 50 setosa iris flowers as belonging to the setosa species. This model has an accuracy of about 96%, producing the mixed results shown in the two right nodes, versicolor and virginica.

These results are summarized by the confusion matrix in *Figure 7.3*, which compares the actual species with the predicted species and records the instances of correct and incorrect classifications.



Prediction	**Reference**		
	setosa	**versicolor**	**virginica**
setosa	50	0	0
versicolor	0	49	5
virginica	0	1	45

Figure 7.3: Confusion Matrix of Decision Tree for Iris Dataset Predicting Species

3.3 RANDOM FORESTS

The Random Forest algorithm utilizes the benefits of decision trees to maximize the scope of analysis. This method aims to reduce the inaccuracy in decision tree models that leads to difficulty in classifying new observations (StatQuest with Josh Starmer, 2018). Essentially, random forests produce a massive amount of unique decision trees and compares all of them to select the most effective tree.

The first step in building a random forest is to create a “bootstrapped” dataset, meaning randomly selecting a subset of observations from the total dataset to create a “testing” and “training” subset. Using the training dataset, numbers of decision trees are created with random subsets of variables at each step. Using different subsets of variables at each step allows for the algorithm to individually evaluate each variable and assess the optimal candidate for the root node and consider the most significant variables.

Once a multitude of decision trees have been produced, the random forest algorithm evaluates each tree and compares them against each other. In doing so, the testing data subset is fed to the model. Each tree in the forest makes a prediction for every observation in the testing subset, which is compared to the actual value of the observation and produces an accuracy for the model. The algorithm then determines the optimal decision tree, seeking to minimize misclassification while weighing complexity to avoid overfitting.

3.3.1 RANDOM FOREST EXAMPLE

Following the process of building a random forest, the iris dataset is split into testing and training subsets, representing 30% and 70% of the total observations respectively, to construct and evaluate the forest. The model generated by the random forest has an accuracy of about 98%. *Figure 8.8* shows the variable importance plot generated by the random forest algorithm. The results illustrate that petal length and width the most important variables in determining species classification. Recall *Figure 6.3: Decision Tree for Iris Dataset Predicting Species,* which based its splits on the same two variables.

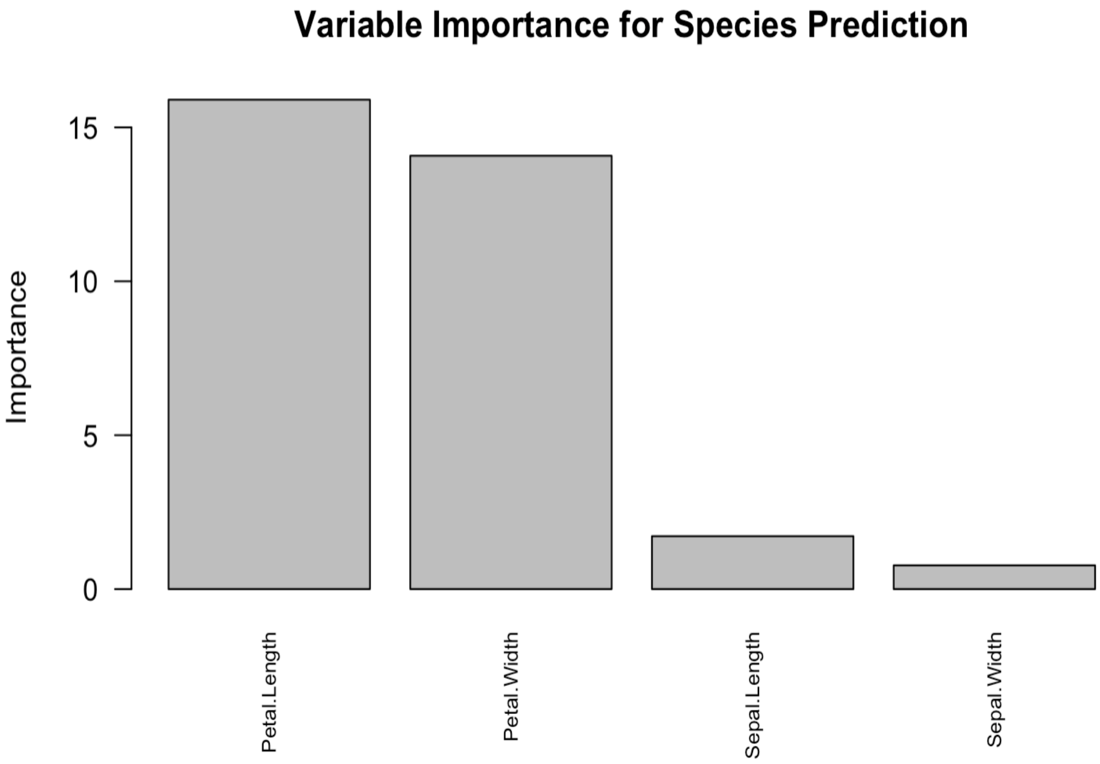
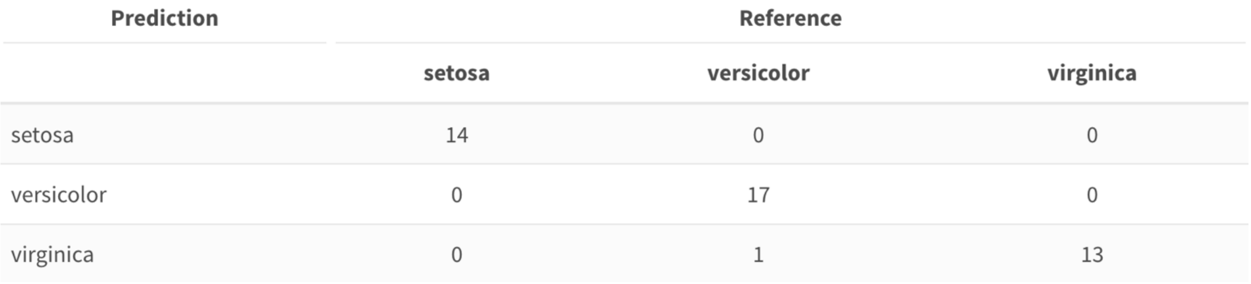


Figure 8.3: Variable Importance Plot for Random Forest Predicting Species from Iris Dataset

Figure 9.3: Confusion Matrix for Random Forest Predicting Species from Iris Dataset presents 45 flowers in the testing subset, 30% of the 150 observations in the full dataset. The higher accuracy of this model is representative of the low level of misclassification, as there is one instance in the matrix of a versicolor iris being classified as a virginica iris.



Prediction	Reference		
	setosa	**versicolor**	**virginica**
setosa	14	0	0
versicolor	0	17	0
virginica	0	1	13

Figure 9.3: Confusion Matrix for Random Forest Predicting Species from Iris Dataset

3.4 GINI IMPURITY

A common method in determining the splits in a decision tree is the measurement of Gini Impurity. Gini Impurity measures the diversity of the population in each node; it quantifies the subsets' purity or impurity level. This value assesses the likelihood of an incorrect classification when a new and random data point is given a random label according to class distribution (Karabiber, n.d.).

The Gini Impurity Index ranges in values from 0 to 0.5. A value of 0 represents a perfectly pure node, with all records belonging to the same class. 0.5 is the maximum level of impurity, representing a uniform, near random, class distribution. Decision Tree and Random Forest algorithms select the model of best fit by minimizing the impurity. Consider dataset D containing samples from k classes and the probability of samples belonging to class i at any node p_i, and Gini Impurity can be defined as shown in *Equation 11*.

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

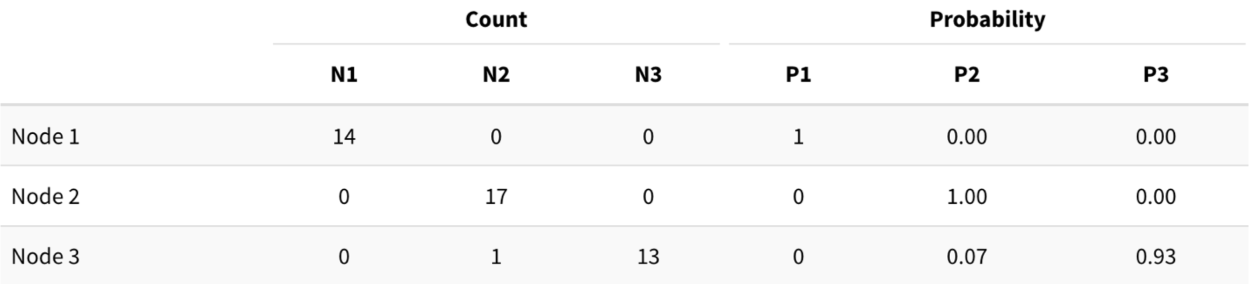
Equation 11

3.4.1 GINI IMPURITY EXAMPLE

Recall the values in *Figure 9.3: Confusion Matrix for Random Forest Predicting Species from Iris Dataset*. The same values are represented in *Figure 10.3*, below, in a slightly different way. The three prediction classifications are referred to below as terminal nodes. Node 1 represents a classification of setosa, Node 2 versicolor, and Node 3 virginica. The counts of the reference iris flowers are tallied as $N1, N2, N3$, again, for the three classes: setosa, versicolor, and virginica. The probability of each species being categorized in each node is recorded as $P1, P2, P3$. Earlier it was noted that this model has only one instance of misclassification, in

Node 3. Because Nodes 1 and 2 have no misclassifications, the nodes are considered pure. In Node 1, predicting a classification of setosa, all setosa reference iris flowers have a 100% probability of being correctly classified. The same is the case in the second node, versicolor.

In the third node, virginica, there are a total of 14 observations, of which one versicolor is incorrectly misclassified. Therefore, the probabilities $P2$ and $P3$ are $\frac{1}{14}$ and $\frac{13}{14}$, respectively.



	Count			Probability		
	N1	**N2**	**N3**	**P1**	**P2**	**P3**
Node 1	14	0	0	1	0.00	0.00
Node 2	0	17	0	0	1.00	0.00
Node 3	0	1	13	0	0.07	0.93

Figure 10.3: Matrix of Values for Random Forest Predicting Species from Iris Dataset

$$Gini(Node\ 3) = 1 - [(0)^2 + (0.07)^2 + \ (0.93)^2] = 0.1302$$

Equation 12

Equation 12 calculates the Gini Impurity for the third node, using the probability distribution produced by the confusion matrix: $P1 = 0,\ P2 = 0.07,\ P3 = 0.93$. The sum of the squared probabilities is subtracted from 1 to produce a value of 0.1302, representing the Gini Impurity of the third node.

The decision trees predicting the species of the iris produced by the random forest are compared against each other to select the “best” tree with least amount of impurity. The “best” tree may not have the lowest amount of impurity, given that complexity of the tree must be considered as well, as overfitting is a concern.

4. DATA

The first phase of the Household Pulse Survey was launched by the Census Bureau in April 2020 and included twelve week-long collection periods (Bureau, n.d.-c). Phase Two ran from August to October of 2020, and Phase Three launched in October 2020 and continued through March 2021. The data used in this analysis comes from Week 27, with observations collected between March 17th and March 29th 2021 (Bureau, n.d.-b). Despite having a two-week collection period, the products of Phases Two and Three continued to be referenced by week for the sake of continuity with Phase One.

The Census Bureau identified approximately 140 million housing units as valid for sampling. Each housing unit is listed in a Master Address File with contact information, usually an email address and a phone number, for an individual in said housing unit (Bureau, n.d.-e). The survey is designed to produce weighted sample sizes to estimate at three geographic levels, the lowest of which consists of the fifteen largest metropolitan areas in the country. The next levels of population estimates are made at the state and national level.

To achieve accurate population estimates, several adjustments are made to sampling and data. The sample size is adjusted for an anticipated response rate of approximately nine percent (Bureau, n.d.-e). Of the 140 million housing units considered valid for sampling, 1,035,000 were selected to respond to Week 27, with approximately 59,000 responses recorded. Parameters are controlled to rank sampling ratios to population estimates and account for nonresponse and undercoverage. Completed surveys are evaluated and ensured to meet a minimum completion to limit the extent of missing data (Bureau, n.d.-e).

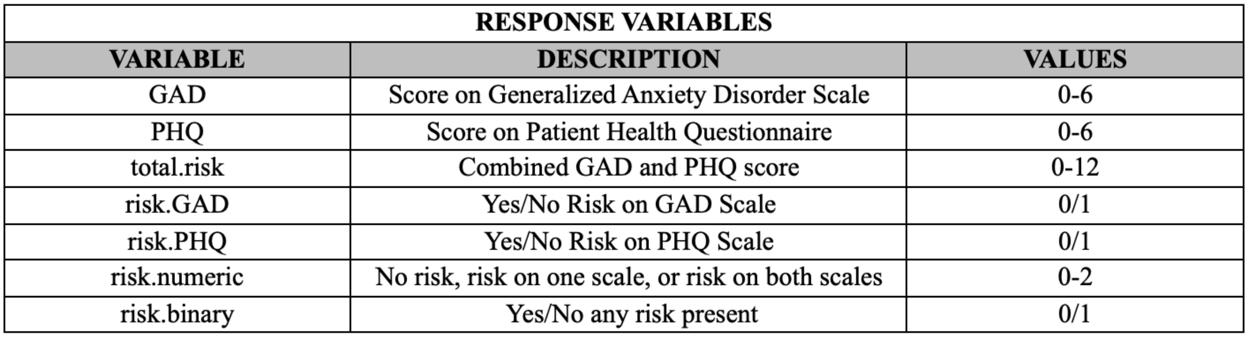
The dataset for Week 27 contained 77,140 observations and 204 variables (Bureau, n.d.-b). Each variable corresponded to a survey question; some asked basic demographic information such as age, race, gender, or level of education attained. Other variables asked respondents to rate their level of concern regarding expenses, exploring food and housing security. The Household Pulse Survey also included four questions pertaining to mental health.

The questions exploring mental health were modified from the two-part Generalized Anxiety Disorder Scale and the Patient Health Questionnaire, commonly used in healthcare as a short screening of mental health symptoms (Bureau, n.d.-d). Respondents are instructed to rate the frequency of experiencing certain emotions on a scale one, being not at all, to four nearly every day. The questions on the GAD scale gauge the frequency of experiencing anxiety or worry over the past week. The questions on the PHQ ask to rate the frequency of having little interest in things and feeling depressed. Both the GAD scale and the PHQ produce results ranging from zero to six, after recoding to a baseline of zero. A score either greater than or equal to three is associated with risk of anxiety or depression, according to the respective scale (Bureau, n.d.-d). The following are questions from the Household Pulse Survey, recording in the variables ANXIOUS, WORRY, INTEREST, and DOWN:

> Over the **last 7 days**, how often have you been bothered by the following problems
>
> … Feeling nervous, anxious, or on edge?
>
> … Not being able to stop or control worrying?
>
> … having little interest or pleasure in doing things?
>
> … feeling down, depressed, or hopeless?
>
> Would you say 1) not at all, 2) several days, 3) more than half the days, or 4) nearly every day?

The variables ANXIOUS, WORRY, INTEREST, and DOWN, represents the respondents' answers to the questions, shown above, and are originally encodes as values 1-4. For this analysis, these values will be recoded 0-3. The analysis will only focus on data including completed cases of all four of these mental health variables, reducing the number observations to 63,596. The response variables used in this study will be calculated from these four variables.

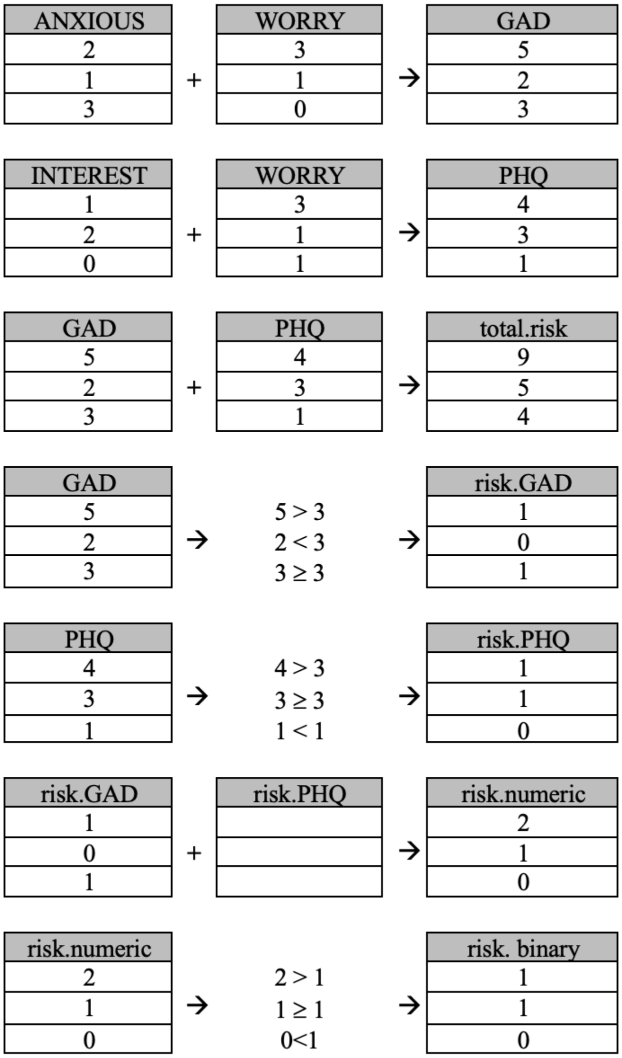
The GAD scale measures the frequency of feelings of anxiety and worry; the new variable GAD is created by adding the values of variables ANXIOUS and WORRY. Each of these variables has a maximum value of 3, making the range of values for the new variable GAD 0-6. Similarly, PHQ measures the frequency of feeling uninterested or down, using value 0-3 in variables INTEREST and DOWN. Added together, these variables will create PHQ with values ranging 0-6.



RESPONSE VARIABLES		
VARIABLE	**DESCRIPTION**	**VALUES**
GAD	Score on Generalized Anxiety Disorder Scale	0-6
PHQ	Score on Patient Health Questionnaire	0-6
total.risk	Combined GAD and PHQ score	0-12
risk.GAD	Yes/No Risk on GAD Scale	0/1
risk.PHQ	Yes/No Risk on PHQ Scale	0/1
risk.numeric	No risk, risk on one scale, or risk on both scales	0-2
risk.binary	Yes/No any risk present	0/1

Figure 11.4: Description Table of Risk Response Variables

By adding together the values GAD and PHQ, another variable is created representing total risk on a combined scale of zero to twelve. Additionally, binary variables indicating a risk on the GAD scale and the PHQ respectively were created by evaluating each score against a threshold of three, given that a score of three on both scales indicate risk. Another variable was created on a scale of zero to two indicating either no risk at all, risk on one scale, or risk on both scales. To a similar effect, a binary overall risk variable was created indicating either no risk at all or at risk on at least one scale. A description of these response variables can be seen above in *Figure 11.4*, and an example chart detailing variable creation can be found below in *Figure 12.4*.



ANXIOUS		WORRY		GAD
2		3		5
1	+	1	→	2
3		0		3

INTEREST		WORRY		PHQ
1		3		4
2	+	1	→	3
0		1		1

GAD		PHQ		total.risk
5		4		9
2	+	3	→	5
3		1		4

GAD				risk.GAD
5		5 > 3		1
2	→	2 < 3	→	0
3		3 ≥ 3		1

PHQ				risk.PHQ
4		4 > 3		1
3	→	3 ≥ 3	→	1
1		1 < 1		0

risk.GAD		risk.PHQ		risk.numeric
1				2
0	+		→	1
1				0

risk.numeric				risk. binary
2		2 > 1		1
1	→	1 ≥ 1	→	1
0		0<1		0

Figure 12.4: Variable Creation Charts

Certain variables included in the original dataset were not necessary to involve in analysis, like versions of a previous variables that had been allocated. In addition to having nonresponses in the dataset, there were further instances of null values encoded as digits -99 or -88, which were edited to represent nonresponse. The variable containing the birth year of the respondent was recalculated to represent age. A variable relating to the number of children present in the household was recoded as binary to indicate children's the presence or lack thereof in the housing unit. Variables regarding race and Hispanic ethnicity were kept separate, but for the sake of analysis recoded into a single race variable.

Several variables had proceeding variables related to them. For instance, the first question on the survey would ask if a particular situation pertained to the respondent. If yes, the following questions listed several ways in which that situation had an impact, and the respondent is asked to select all that apply. It was verified that in each case of multi-response, at least one respondent did select more than one option. In order to limit the number of variables, these questions were condensed for the sake of preliminary investigation.

Counting the frequencies of each response, the values were recoded from the original 0/1 of binary to represent the number of the response variable selected: WHYNOT1 has values 0/1, WHYNOT2 has values 0/2, WHYNOT3 has values 0/3, and so on. A new variable is created to represent the selected responses, made up of a concatenated string of selections: WHYNOT_selected would have a value of "1,2,3" if all three responses were selected. For example, consider the following questions:

Question 2: Did you receive (or do you plan to receive) all required doses of the COVID-19 vaccine?

1) Yes
2) No

Question 3: Once the vaccine is available to you would you…

1) Definitely get the vaccine
2) Probably get the vaccine
3) Probably NOT get the vaccine
4) Definitely NOT get the vaccine

If the answer to Question 2 = 2 or the answer to Question 3 = 2, 3, or 4, then the following question will be asked:

Which of the following, if any, are reasons you would not get the vaccine:

1) I'm concerned about the possible side effects
2) I don't know if the vaccine will work
3) I don't believe I need the vaccine
4) I don't like vaccines
5) My doctor has not recommended it
6) I plan to wait and see if it is safe and may get it later
7) I think other people need it more than I do right now
8) I am concerned about the cost of the vaccine
9) I don't trust the vaccine
10) I don't trust the government
11) Other

The selected responses are stored in the dataframe as follows below in *Figure 13.4*. Following the recoding process, the selections will be stored as shown below, and concatenated into a single column:

WHYNOT1	WHYNOT2	WHYNOT3	…	WHYNOT11
0	1	1	…	0
1	0	0	…	1
1	0	1	…	0

WHYNOT1	WHYNOT2	WHYNOT3	…	WHYNOT11
0	2	3	…	0
1	0	0	…	11
1	0	3	…	0

WHYNOT_selected
2, 3, …
1, …, 11
1, 3, …

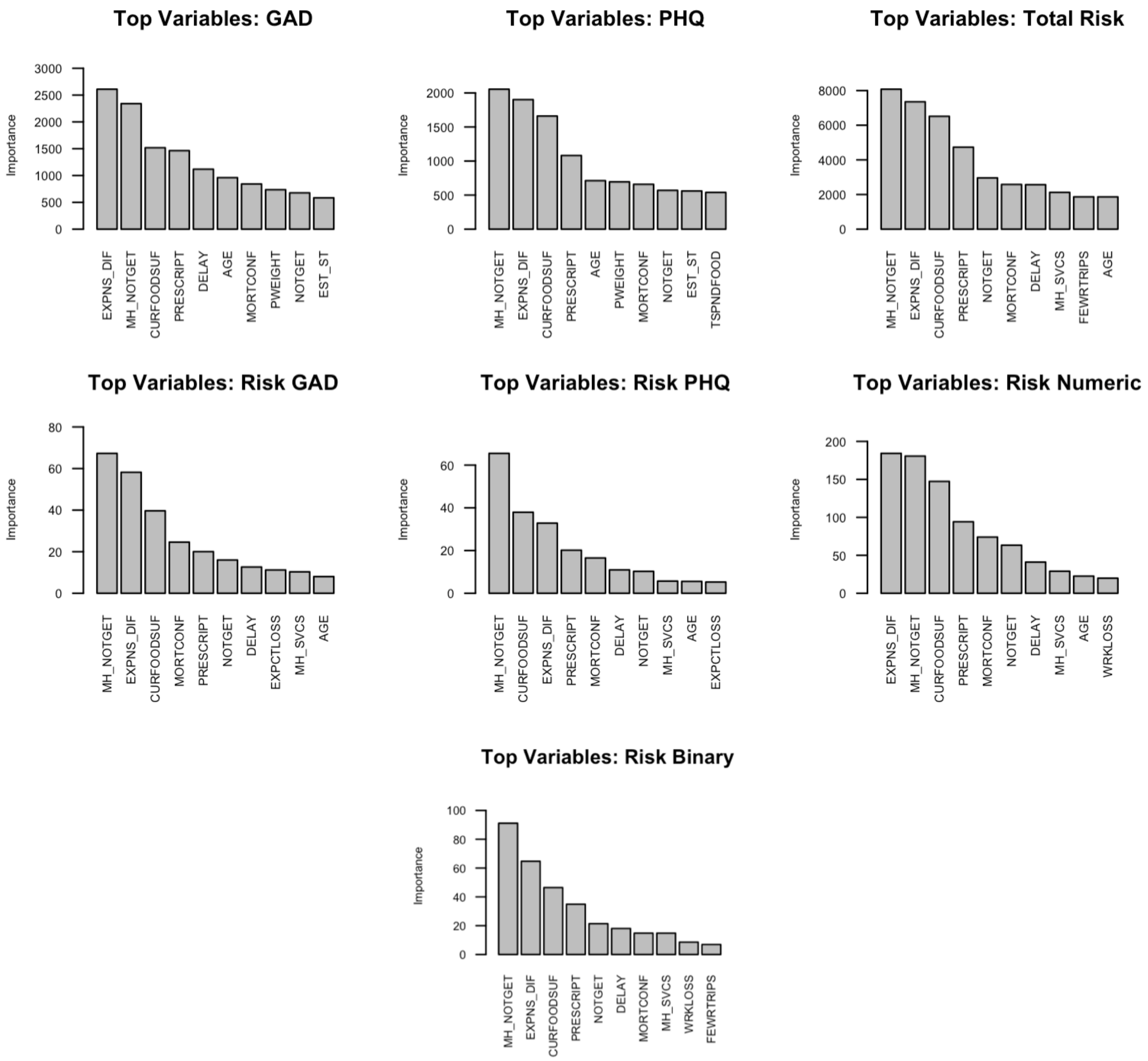
Figure 13.4 Variable Recoding Chart

These concatenated “selected” variables will be set aside during primary analysis. Each selected variable proceeds what will serve as an “indicator” variable, which will signal a need for further investigation into the multiple selections corresponding to the question. The dataset is then counted for nonresponse and the median response for each variable is calculated and imputed to minimize missingness.

5. RESULTS

The purpose of generating random forest models and fitting decision trees in this study is to highlight factors contributing to mental health struggles that can indicate a need for public intervention. By pinpointing specific circumstances that highly correlate to mental health risks, these warning signs can be used to signal a need for mental health treatment and intervention in communities and individuals otherwise underserved.

The dataset was first split into subsets for testing and training the models. For each of the seven response variables, GAD, PHQ, total.risk, risk.GAD, risk.PHQ, risk.numeric, and risk.binary (see *Figure 11.4* for variable description), Random Forests were produced using the *ranger* package (Wright et al. 2023) in R to highlight the most significant variables. This package was advantageous for the exploration of variable importance in a large dataset. The arguments implemented within the ranger function specified the use of Gini Impurity in evaluating 500 trees. Given the large number of variables in the dataset, the Random Forest approach is useful for selecting the most relevant predictors out of an expansive selection. A function was used to generate a list of the most significant predictors for each of the response variables; the top ten variable importance charts are shown in *Figure 14.5*. With seven lists of response variables’ top twenty most significant predictors, only 22 unique variables occurred; 16 variables made all seven responses’ top predictors list.


Top Variables: GAD

Top Variables: PHQ

Top Variables: Total Risk

Top Variables: Risk GAD

Top Variables: Risk PHQ

Top Variables: Risk Numeric

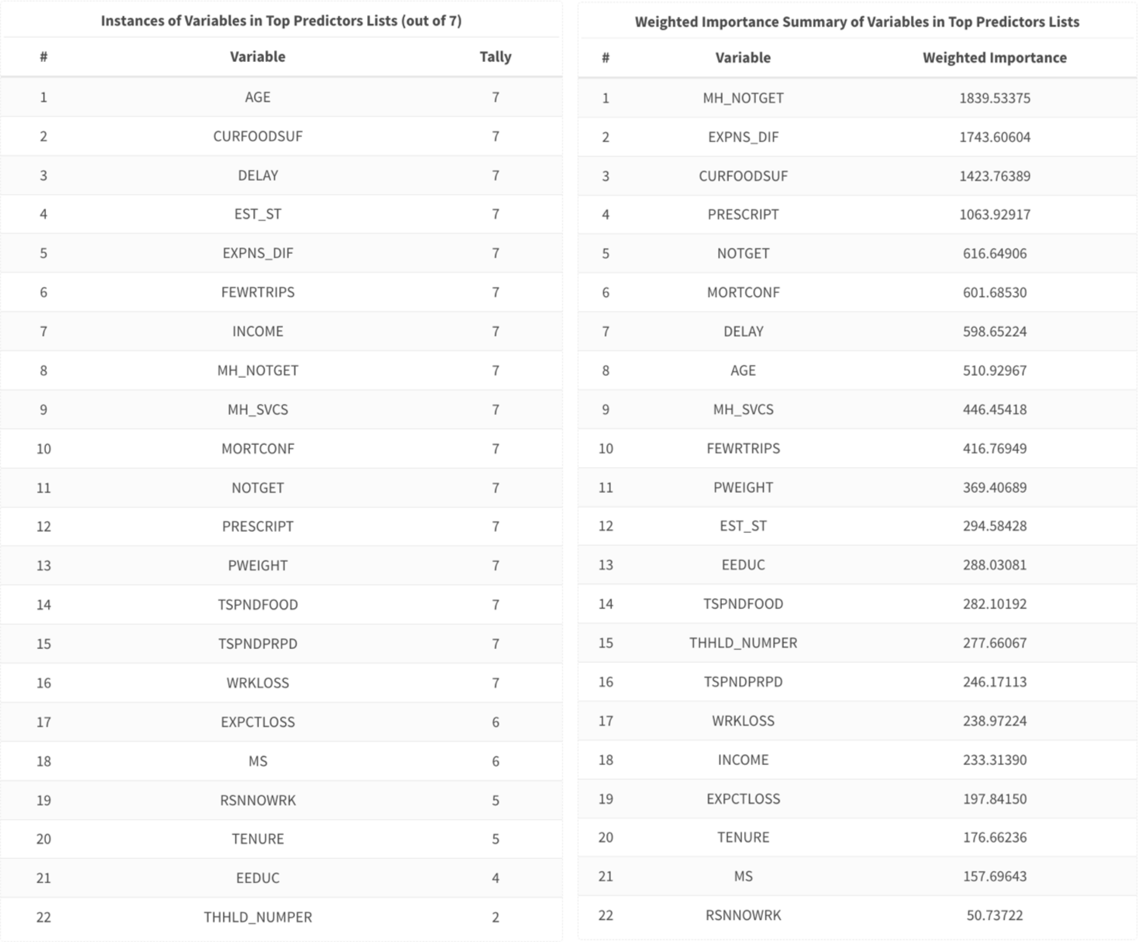
Top Variables: Risk Binary

Figure 14.5: Top Ten Variable Importance Charts

Each of these 22 predictor variables occurred in at least two lists, meaning each variable has more than one calculated importance score. By adding together each predictor's multiple importance scores and dividing this by the number of occurrences, a weighted importance score is generated for each variable.

Figure 15.5 lists the 22 predictor variables with a count of occurrences out of seven lists, sorted in descending order on one side, and the variable importance score on the other, also sorted in descending order. A list of these variables and their description is shown in *Figure*

16.5. Some of the significant variables include age, food stability, financial difficulty, and a current need of mental health treatment.



Instances of Variables in Top Predictors Lists (out of 7)

#	Variable	Tally
1	AGE	7
2	CURFOODSUF	7
3	DELAY	7
4	EST_ST	7
5	EXPNS_DIF	7
6	FEWRTRIPS	7
7	INCOME	7
8	MH_NOTGET	7
9	MH_SVCS	7
10	MORTCONF	7
11	NOTGET	7
12	PRESCRIPT	7
13	PWEIGHT	7
14	TSPNDFOOD	7
15	TSPNDPRPD	7
16	WRKLOSS	7
17	EXPCTLOSS	6
18	MS	6
19	RSNNOWRK	5
20	TENURE	5
21	EEDUC	4
22	THHLD_NUMPER	2

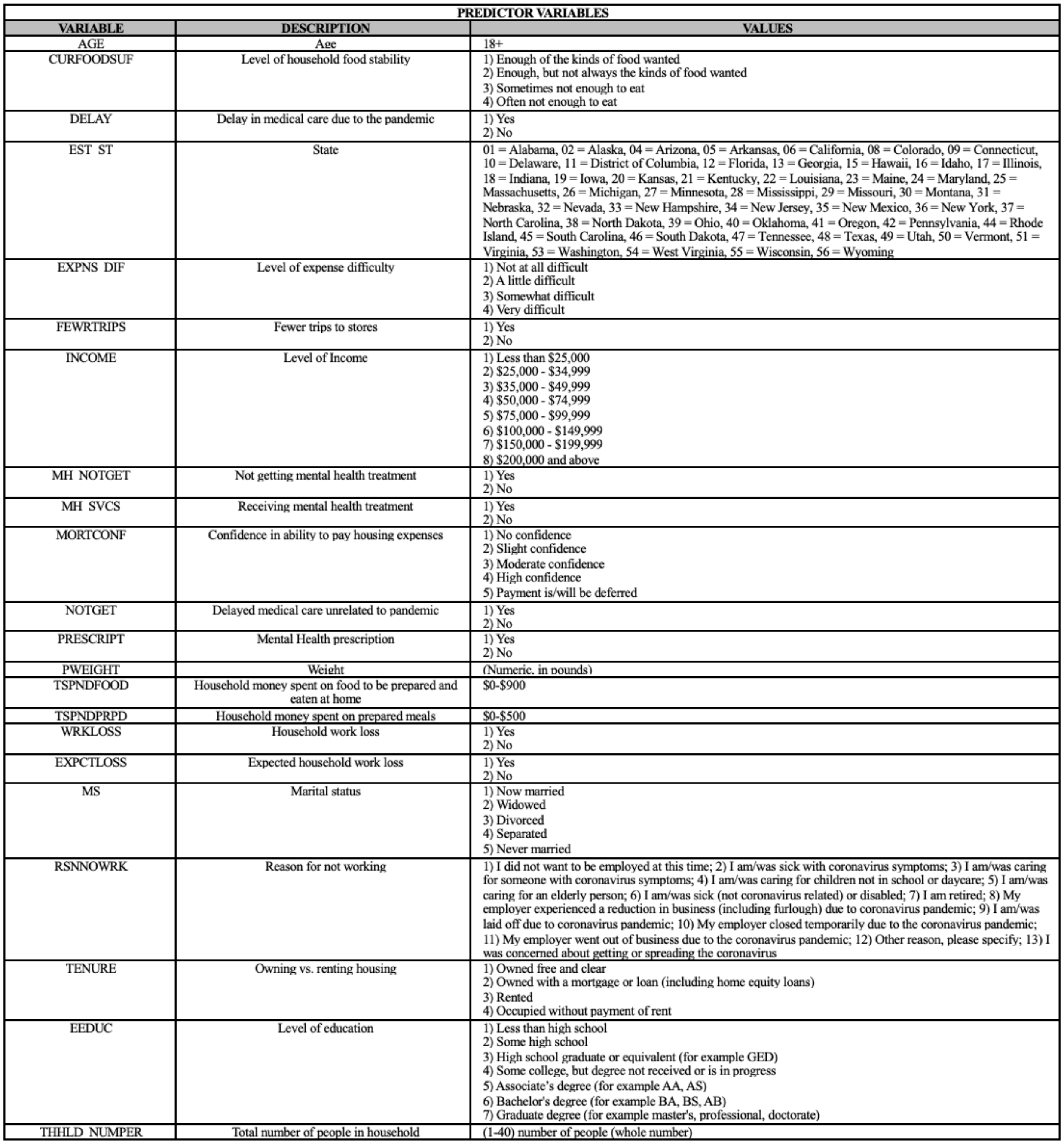
Weighted Importance Summary of Variables in Top Predictors Lists

#	Variable	Weighted Importance
1	MH_NOTGET	1839.53375
2	EXPNS_DIF	1743.60604
3	CURFOODSUF	1423.76389
4	PRESCRIPT	1063.92917
5	NOTGET	616.64906
6	MORTCONF	601.68530
7	DELAY	598.65224
8	AGE	510.92967
9	MH_SVCS	446.45418
10	FEWRTRIPS	416.76949
11	PWEIGHT	369.40689
12	EST_ST	294.58428
13	EEDUC	288.03081
14	TSPNDFOOD	282.10192
15	THHLD_NUMPER	277.66067
16	TSPNDPRPD	246.17113
17	WRKLOSS	238.97224
18	INCOME	233.31390
19	EXPCTLOSS	197.84150
20	TENURE	176.66236
21	MS	157.69643
22	RSNNOWRK	50.73722

Figure 15.5: Top Variable Summary Table

As noted, these variables were selected from seven random forests generated with the training dataset for the response variables GAD, PHQ, total risk, numeric risk, binary risk, GAD risk, and PHQ risk. Using the testing dataset, accuracies were calculated for each of the seven forests. The forests for GAD, PHQ, and total risk reported minute accuracies; 0.06%, 0.19%, and 0.05%, respectively. This contrasted with the significantly higher scores for the numeric risk, binary risk, GAD risk, and PHQ risk forests. The models for numeric and binary risk had similar accuracies of 66.2% and 66.0% respectively, while GAD risk and PHQ risk were reported at

70.8% and 69.9%. Given that the overall purpose of the study is to indicate a presence of risk rather than predict severity of risk combined with the wide range of reported accuracies, the variables GAD, PHQ, and total risk will be excluded in the modeling of decision trees.



PREDICTOR VARIABLES		
VARIABLE	DESCRIPTION	VALUES
AGE	Age	18+
CURFOODSUF	Level of household food stability	1) Enough of the kinds of food wanted 2) Enough, but not always the kinds of food wanted 3) Sometimes not enough to eat 4) Often not enough to eat
DELAY	Delay in medical care due to the pandemic	1) Yes 2) No
EST ST	State	01 = Alabama, 02 = Alaska, 04 = Arizona, 05 = Arkansas, 06 = California, 08 = Colorado, 09 = Connecticut, 10 = Delaware, 11 = District of Columbia, 12 = Florida, 13 = Georgia, 15 = Hawaii, 16 = Idaho, 17 = Illinois, 18 = Indiana, 19 = Iowa, 20 = Kansas, 21 = Kentucky, 22 = Louisiana, 23 = Maine, 24 = Maryland, 25 = Massachusetts, 26 = Michigan, 27 = Minnesota, 28 = Mississippi, 29 = Missouri, 30 = Montana, 31 = Nebraska, 32 = Nevada, 33 = New Hampshire, 34 = New Jersey, 35 = New Mexico, 36 = New York, 37 = North Carolina, 38 = North Dakota, 39 = Ohio, 40 = Oklahoma, 41 = Oregon, 42 = Pennsylvania, 44 = Rhode Island, 45 = South Carolina, 46 = South Dakota, 47 = Tennessee, 48 = Texas, 49 = Utah, 50 = Vermont, 51 = Virginia, 53 = Washington, 54 = West Virginia, 55 = Wisconsin, 56 = Wyoming
EXPNS DIF	Level of expense difficulty	1) Not at all difficult 2) A little difficult 3) Somewhat difficult 4) Very difficult
FEWRTRIPS	Fewer trips to stores	1) Yes 2) No
INCOME	Level of Income	1) Less than $25,000 2) $25,000 - $34,999 3) $35,000 - $49,999 4) $50,000 - $74,999 5) $75,000 - $99,999 6) $100,000 - $149,999 7) $150,000 - $199,999 8) $200,000 and above
MH NOTGET	Not getting mental health treatment	1) Yes 2) No
MH SVCS	Receiving mental health treatment	1) Yes 2) No
MORTCONF	Confidence in ability to pay housing expenses	1) No confidence 2) Slight confidence 3) Moderate confidence 4) High confidence 5) Payment is/will be deferred
NOTGET	Delayed medical care unrelated to pandemic	1) Yes 2) No
PRESCRIPT	Mental Health prescription	1) Yes 2) No
PWEIGHT	Weight	(Numeric, in pounds)
TSPNDFOOD	Household money spent on food to be prepared and eaten at home	$0-$900
TSPNDPRPD	Household money spent on prepared meals	$0-$500
WRKLOSS	Household work loss	1) Yes 2) No
EXPCTLOSS	Expected household work loss	1) Yes 2) No
MS	Marital status	1) Now married 2) Widowed 3) Divorced 4) Separated 5) Never married
RSNNOWRK	Reason for not working	1) I did not want to be employed at this time; 2) I am/was sick with coronavirus symptoms; 3) I am/was caring for someone with coronavirus symptoms; 4) I am/was caring for children not in school or daycare; 5) I am/was caring for an elderly person; 6) I am/was sick (not coronavirus related) or disabled; 7) I am retired; 8) My employer experienced a reduction in business (including furlough) due to coronavirus pandemic; 9) I am/was laid off due to coronavirus pandemic; 10) My employer closed temporarily due to the coronavirus pandemic; 11) My employer went out of business due to the coronavirus pandemic; 12) Other reason, please specify; 13) I was concerned about getting or spreading the coronavirus
TENURE	Owning vs. renting housing	1) Owned free and clear 2) Owned with a mortgage or loan (including home equity loans) 3) Rented 4) Occupied without payment of rent
EEDUC	Level of education	1) Less than high school 2) Some high school 3) High school graduate or equivalent (for example GED) 4) Some college, but degree not received or is in progress 5) Associate's degree (for example AA, AS) 6) Bachelor's degree (for example BA, BS, AB) 7) Graduate degree (for example master's, professional, doctorate)
THHLD NUMPER	Total number of people in household	(1-40) number of people (whole number)

Figure 16.5:Description of Top Predictor Variables

Using the 22 variables selected earlier as predictor variables, decision trees were modeled and plotted for the variables numeric.risk, binary.risk, risk.GAD, and risk.PHQ using the *rpart* package (Therneau et al. 2023). The default arguments for this package evaluate the trees using Gini Impurity, and corresponding *rpart.plot* package easily visualizes the models (Milborrow, 2024). Different control parameters were tested in an effort to add complexity to the resulting trees, with arguments specifying the minimum number of observations in a node needed to attempt a split, the minimum number of observations in a terminal node, and varying complexity parameters controlling the size of the tree. A smaller complexity parameter will produce a smaller tree, though specifying values of 0.01 and 0.001 did not result in any improvement in the trees discussed below.

The reported accuracy for risk.PHQ was the highest, at 86.9%, with risk.GAD following at 83.4%. The accuracy for risk.binary was 81.6%, with the lowest being risk.numeric at 79.6%. The decision trees can be seen in *Figure 17.5*. Only three unique variables are represented in the trees, with most of them only containing the variables relating to not getting mental health treatment and levels of expensive difficulty. The third variable, in only one of the trees, relates to not getting medical care.

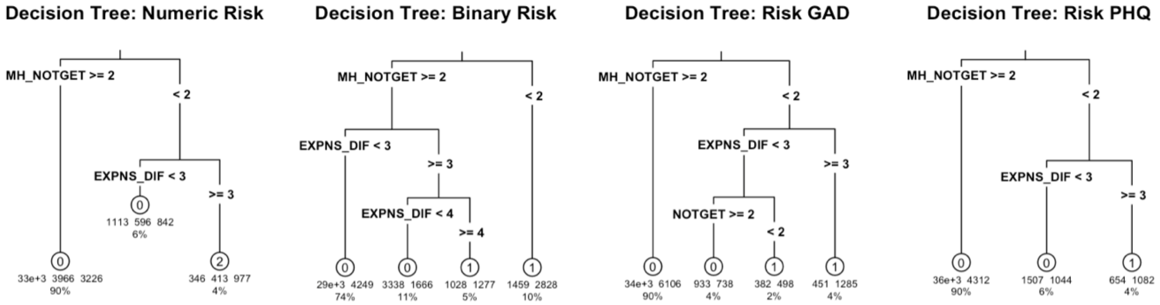


Figure 17.5: Decision Trees for Predicting Risk (Including Mental Health Variables)

The first split in each of these decision trees is made on the variable relating to not getting mental health treatment. The response variables relate to mental health, meaning there is reason to suspect confounding behavior. Given the purpose and context of the study, the exclusion of all variables relating to mental health would avoid overfitting the model in aid in focusing on the scope of variables that have the potential to indirectly impact mental health rather than the existence past or present of mental health struggles.

Moving forward, three variables relating to receiving or not receiving mental health treatment were excluded from the predictor variables used to model four new decision trees shown in *Figure 18.5*. These three variables were MH_NOTGET, MH_SVCS, and PRESCRIPT, which detailed a lack of needed mental health care, presence of active mental health treatment, or current use of a mental health prescription. The accuracies were reported at 86.4% for risk.PHQ, 82.4% for risk.GAD, 79.9% for risk.binary, and 78.9% for risk.numeric. With the exclusion of mental health variables, there was an average decrease in accuracy of about 0.975% overall, and no new variables were added to the decision trees. As seen in *Figure 17.5*, three variables were included in the decision tree models, one of which was a mental health variable. Excluding that mental health variable, the same two variables are included in the models in *Figure 18.5* with no new additions. Interestingly, the tree for numeric risk predicts only values of 0 and 2, indicating that there is either no risk present or risk present on both scales. Given the recorded instances of risk on only one scale in the dataset, the misclassification rate for this model is understandably the highest of those produced.

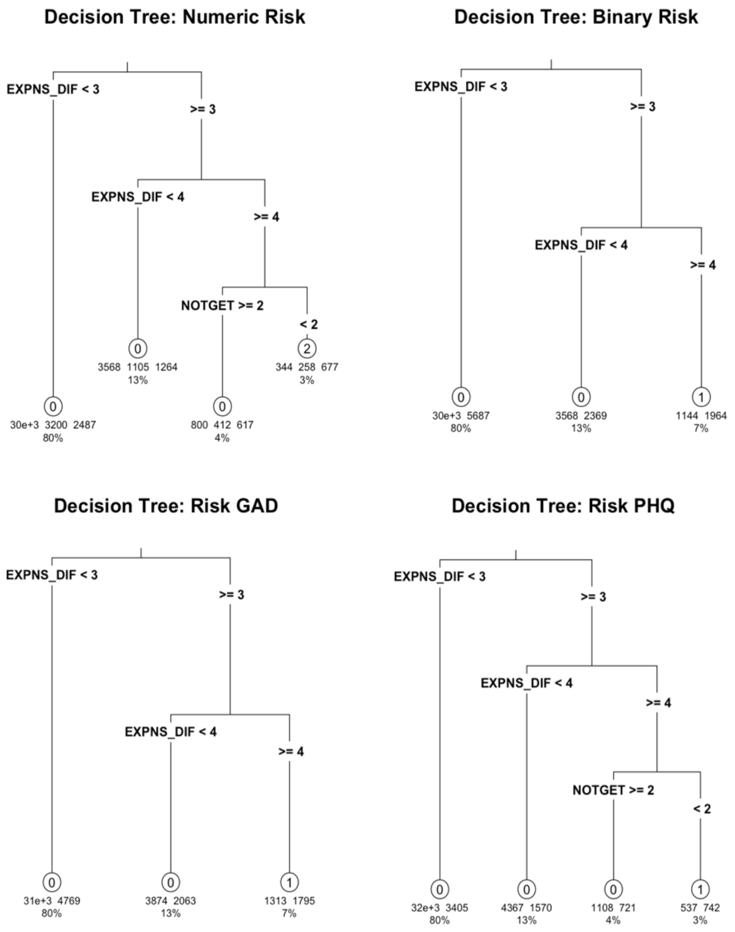


Figure 18.5: Decision Trees for Predicting Risk (Excluding Mental Health Variables)

The decision tree for risk PHQ in the bottom right of *Figure 18.5* predicts a value of either 0 or 1, indicating no risk on the PHQ scale or the presence of risk on the PHQ scale. The splits in this tree are on two variables; referring to *Figure 16.5*, EXPNS_DIF evaluates an individual's level of difficulty with expenses, 1 being no difficulty and 4 being very difficult, and NOTGET is a binary variable indicating a delay in medical care, unrelated to the pandemic. The first split is on expense difficulty, with values greater than or equal to 3 moving to the right. Again, splitting on expense difficulty, those that respond "very difficult" move right again to be split on delay in medical care. Those that did experience a delay in medical care are again split right to the terminal node, predicting that an individual experiencing a high level of difficulty with expenses and a delay in medical care would indicate risk on the PHQ scale.

The first two splits on the expense difficulty variable, separating out those experiencing a high level of difficulty, is mirrored in each response variable's tree models. As shown in the decision trees for risk GAD and binary risk in *Figure 18.5*, those experiencing high levels of expense difficulty were predicted to be at risk based off that alone. The tree for numeric risk directly mirrors that of risk PHQ. Overall, the four decision trees produced models reporting the significance of experiencing economic hardships and lack of access to medical care.

6. CONCLUSION

Inequitable healthcare systems highlight the ways in which social determinants of health create intersectional barriers that limit access to care. The historic and overarching inequalities in the U.S. result in the marginalization of the working class. The outbreak of the COVID-19 pandemic emphasized these disparities and serves as a cite of analysis to encourage outreach.

Building on regression techniques, the robust nature of decision trees can handle large amounts of complex and irregular data. The machine learning algorithm Random Forests creates large amounts of trees to select an optimal model, using a testing and training subset to measure accuracies. Decision Trees and Random Forests function to minimize impurity. The Gini Impurity measures the likelihood of an incorrect classification of a new observation. The algorithm seeks to minimize this value and produce an optimal model.

The data used for analysis comes from Week 27 of the U.S. Census Bureau's Household Pulse Survey, collected from March 17th to March 29th, 2021. The Census Bureau's database deemed 140 million household units valid for sampling; of those, about one million were randomly selected to respond to this period of the survey. Approximately 59,000 responses were recorded, and all data was adjusted for nonresponse and undercoverage.

Modified versions of the two-part Generalized Anxiety Disorder survey and Patient Health Questionnaire were included in the survey, measuring respondents' feelings of anxiety, worry, lack of interest, and depression. From these four questions, response variables were created for analysis measuring GAD, PHQ, and total scores numerically, in addition to binary indicators of risk on the GAD scale, PHQ, or any risk at all. Another variable was created to indicate the number of scales at risk.

Random Forests were created with these response variables to extract the overall most significant predictor variables, which included information ranging from age, income, and state of residence, levels of food scarcity in the household, difficulty with expenses, and access to medical care. To avoid overfitting, three variables deemed significant were dropped from use as predictors. These variables represented mental health information and were therefore highly correlated with the response variables. Of the seven models created, four yielded high accuracies: binary risk.GAD, risk.PHQ, and risk.binary, and risk.numeric with values 0-2.

Decision Tree models produced to predict a risk of mental health crises during the COVID-19 pandemic highlighted two variables as highly pertinent and warranting cause for concern: extreme difficulty with expenses and regular lack of access to medical care. Economic hardship and healthcare scarcity have been problematic and prevalent issues long before the 2020 outbreaks and government shutdowns brought the problems to a critical level.

The findings of this study aligned with existing literature, highlighting the impact of economic instability on health and well-being. Extrapolating from the results of this study, it is logical to conclude that in periods of national instability resulting in strenuous demand for economic resources or medical provisions, the mobilization of supplementary outreach regarding mental health care has the potential to be widely utilized in previously underserved communities. Economic instability, particularly during a health crisis, has tremendous potential to negatively impact mental well-being, the resulting in negative consequences in other aspects of livelihood. Gauging points of intervention in any context, national, communal, or individual, could result in the provision of critical mental health care.

7. BIBLIOGRAPHY

Bambra, C., Lynch, J., & Smith, K. E. (2021). *The Unequal Pandemic: COVID-19 and Health Inequalities* (1st ed.). Bristol University Press. https://doi.org/10.2307/j.ctv1qp9gnf

Breiman, L. (Ed.). (1984). *Classification and regression trees*. Wadsworth International Group.

Bureau, U. C. (n.d.-a). *Household Pulse Survey: Measuring Emergent Social and Economic Matters Facing U.S. Households*. Census.Gov. Retrieved February 9, 2024, from https://www.census.gov/householdpulsedata

Bureau, U. C. (n.d.-b). *Household Pulse Survey Public Use File (PUF)*. Census.Gov. Retrieved April 8, 2024, from https://www.census.gov/programs-surveys/household-pulse-survey/datasets.html

Bureau, U. C. (n.d.-c). *Household Pulse Survey Technical Documentation*. Census.Gov. Retrieved April 8, 2024, from https://www.census.gov/programs-surveys/household-pulse-survey/technical-documentation.html

Bureau, U. C. (n.d.-d). *Mental Health—Household Pulse Survey—COVID-19*. Retrieved February 9, 2024, from https://www.cdc.gov/nchs/covid19/pulse/mental-health.htm

Bureau, U. C. (n.d.-e). *Methodology*. Census.Gov. Retrieved April 8, 2024, from https://www.census.gov/programs-surveys/household-pulse-survey/technical-documentation/methodology.html

Bureau, U. C. (n.d.-f). *Questionnaires*. Census.Gov. Retrieved April 8, 2024, from https://www.census.gov/programs-surveys/household-pulse-survey/technical-documentation/questionnaires.html

Bureau, U. C. (n.d.-g). *Source and Accuracy Statements*. Census.Gov. Retrieved April 8, 2024, from https://www.census.gov/programs-surveys/household-pulse-survey/technical-documentation/source-accuracy.html

Cairney, J., Veldhuizen, S., Vigod, S., Streiner, D. L., Wade, T. J., & Kurdyak, P. (2014). Exploring the social determinants of mental health service use using intersectionality theory and CART analysis. *Journal of Epidemiology and Community Health (1979-)*, *68*(2), 145–150.

Cannon, A. R., Hartlaub, B. A., Cobb, G. W., Lock, R. H., Legler, J. M., Rossman, A. J., Moore, T. L., & Witmer, J. A. (2013). *STAT2: Building Models for a World of Data*. W.H. Freeman. https://books.google.com/books?id=P-shnwEACAAJ

Crenshaw, K. (1991). Mapping the Margins: Intersectionality, Identity Politics, and Violence against Women of Color. *Stanford Law Review*, *43*(6), 1241–1299. https://doi.org/10.2307/1229039

Frazier, Miriam. (2021, Fall). *LectureOutline_DecisionTrees.pdf*.

Huang, Qimin. (2016, March 10). *CART.pptx*.

Humber, L. (2019). *Vital Signs: The Deadly Costs of Health Inequality*. Pluto Press. https://doi.org/10.2307/j.ctvn5txst

Karabiber, F. (n.d.). *Gini Impurity*. Retrieved June 13, 2024, from https://www.learndatasci.com/glossary/gini-impurity/

Kosarenko, Y. (2021, November 14). *How to Create Decision Trees for Business Rules Analysis*. Why Change. https://why-change.com/2021/11/13/how-to-create-decision-trees-for-business-rules-analysis/

Loh, W.-Y. (2014). Fifty Years of Classification and Regression Trees. *International Statistical Review / Revue Internationale de Statistique*, *82*(3), 329–348.

Milborrow, S. (2024). *rpart.plot: Plot "rpart" Models: An Enhanced Version of "plot.rpart"* (3.1.2) [Computer software]. https://cran.r-project.org/web/packages/rpart.plot/index.html

Priorities, C. on B. and P. (2021). *Tracking the COVID-19 Recession's Effects on Food, Housing, and Employment Hardships*. Center on Budget and Policy Priorities. https://www.jstor.org/stable/resrep28464

Ramos, K. (2022). Mental Health Impacts of the COVID-19 Pandemic. *Generations: Journal of the American Society on Aging*, *46*(1), 1–8.

rpart function—RDocumentation. (n.d.). Retrieved July 4, 2024, from https://www.rdocumentation.org/packages/rpart/versions/4.1.23/topics/rpart

rpart.control function—RDocumentation. (n.d.). Retrieved July 4, 2024, from https://www.rdocumentation.org/packages/rpart/versions/4.1.23/topics/rpart.control

Sansom, K. (2019). Gardening Algorithms for the Fruitful Future: Decision Trees and Random Forest Algorithms to Predict Child Malnutrition Populations Globally. *Senior Independent Study Theses*. https://openworks.wooster.edu/independentstudy/8366

Serrano.Academy (Director). (2021, February 28). *The Gini Impurity Index explained in 8 minutes!* https://www.youtube.com/watch?v=u4IxOk2ijSs

StatQuest with Josh Starmer (Director). (2018, February 5). *StatQuest: Random Forests Part 1 - Building, Using and Evaluating*. https://www.youtube.com/watch?v=J4Wdy0Wc_xQ

StatQuest with Josh Starmer (Director). (2019, August 19). *Regression Trees, Clearly Explained!!!* https://www.youtube.com/watch?v=g9c66TUylZ4

StatQuest with Josh Starmer (Director). (2020, January 15). *StatQuest: Random Forests Part 2: Missing data and clustering*. https://www.youtube.com/watch?v=sQ870aTKqiM

StatQuest with Josh Starmer (Director). (2021, April 26). *Decision and Classification Trees, Clearly Explained!!!* https://www.youtube.com/watch?v=_L39rN6gz7Y

Therneau, T., Atkinson, B., port, B. R. (producer of the initial R., & maintainer 1999-2017). (2023). *rpart: Recursive Partitioning and Regression Trees* (4.1.23) [Computer software]. https://cran.r-project.org/web/packages/rpart/index.html

Wright, M. N., Wager, S., & Probst, P. (2023). *ranger: A Fast Implementation of Random Forests* (0.16.0) [Computer software]. https://cran.r-project.org/web/packages/ranger/index.html

Wright, M. N., & Ziegler, A. (2017). **ranger**: A Fast Implementation of Random Forests for High Dimensional Data in *C++* and *R*. *Journal of Statistical Software*, *77*(1). https://doi.org/10.18637/jss.v077.i01

Yadav, P. (2019, September 23). *Decision Tree in Machine Learning*. Medium. https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96

8. APPENDIX

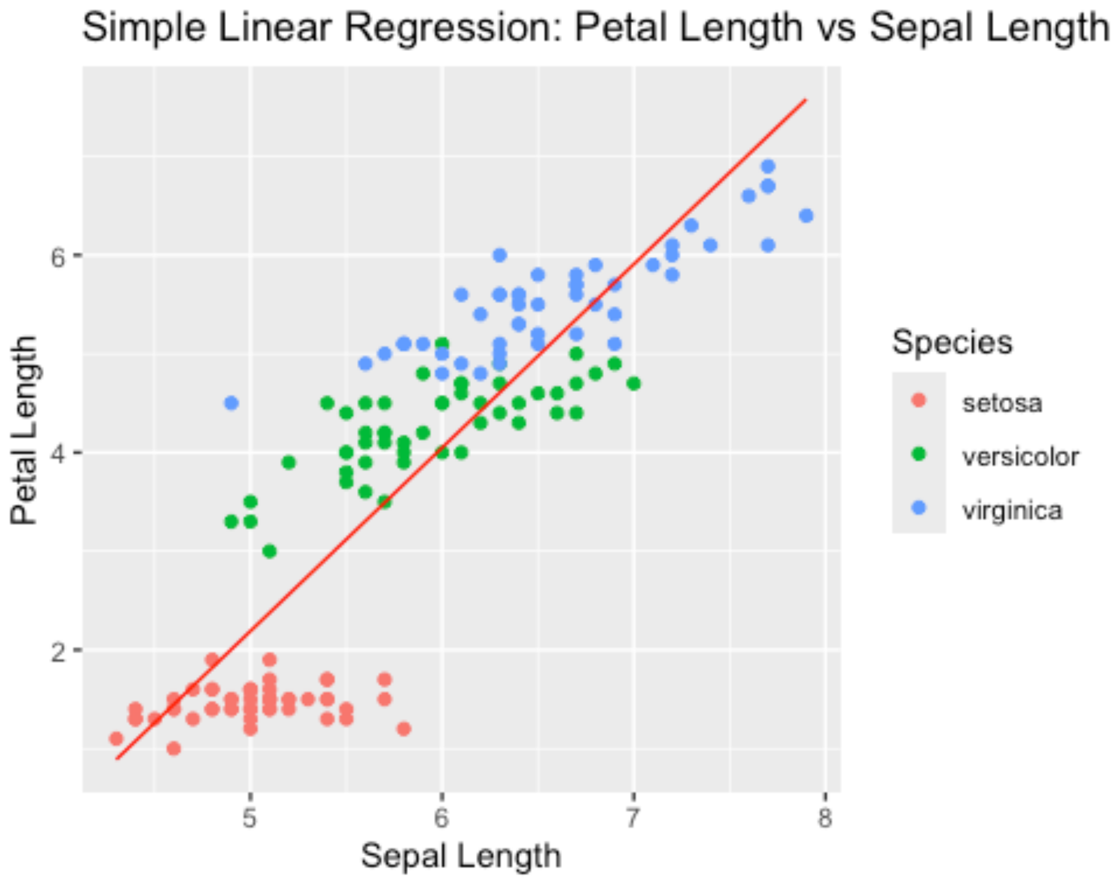
8.1 IRIS EXAMPLES

Load Iris dataset

```
data(iris)
```

Create and plot Simple Linear Model

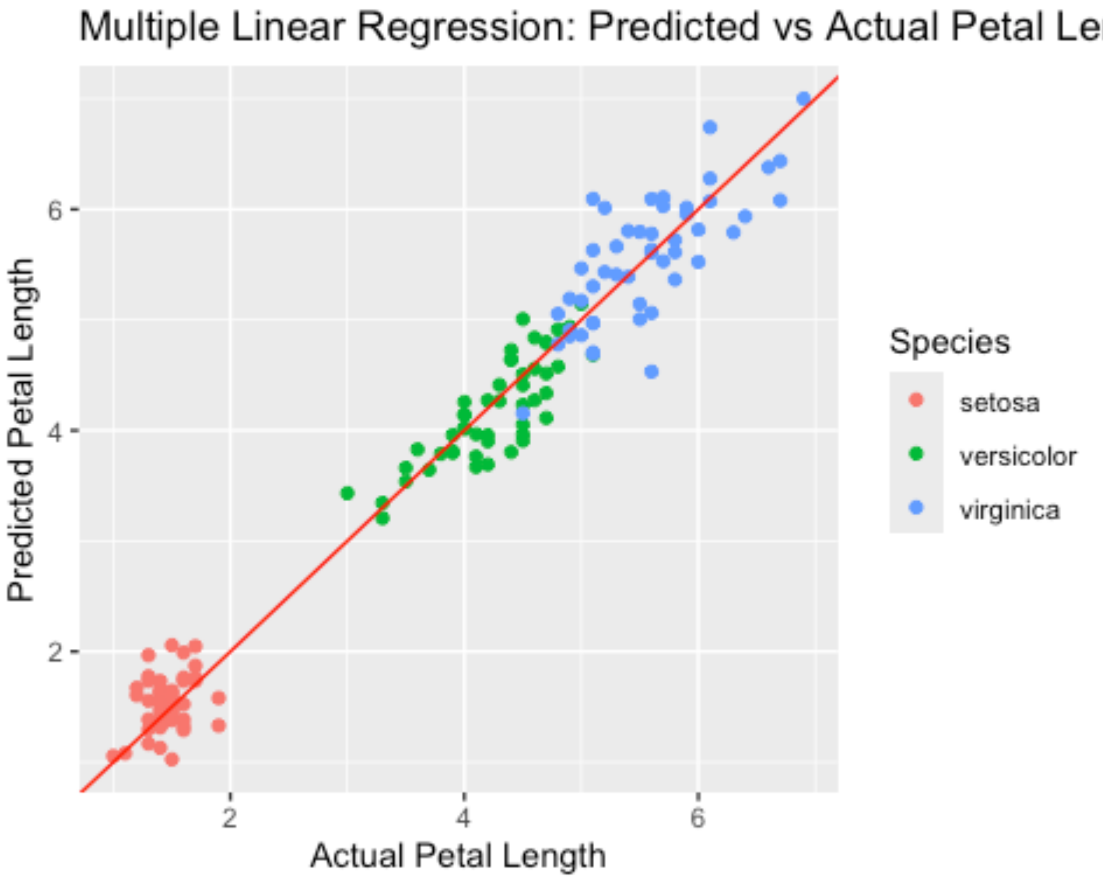
```
simple_model <- lm(Petal.Length ~ Sepal.Length, data=iris)
iris$PetalLength_Pred_Simple <- predict(simple_model)

summary(simple_model)

##
## Call:
## lm(formula = Petal.Length ~ Sepal.Length, data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.47747 -0.59072 -0.00668  0.60484  2.49512
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -7.10144    0.50666  -14.02   <2e-16 ***
## Sepal.Length  1.85843    0.08586   21.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8678 on 148 degrees of freedom
## Multiple R-squared:   0.76,  Adjusted R-squared:  0.7583
## F-statistic: 468.6 on 1 and 148 DF,  p-value: < 2.2e-16

ggplot(iris, aes(x=Sepal.Length, y=Petal.Length, color = Species)) +
  geom_point() +
  geom_line(aes(y=PetalLength_Pred_Simple), color="red") +
  ggtitle("Simple Linear Regression: Petal Length vs Sepal Length") +
  xlab("Sepal Length") +
  ylab("Petal Length")
```



Create and plot Multiple Linear Model

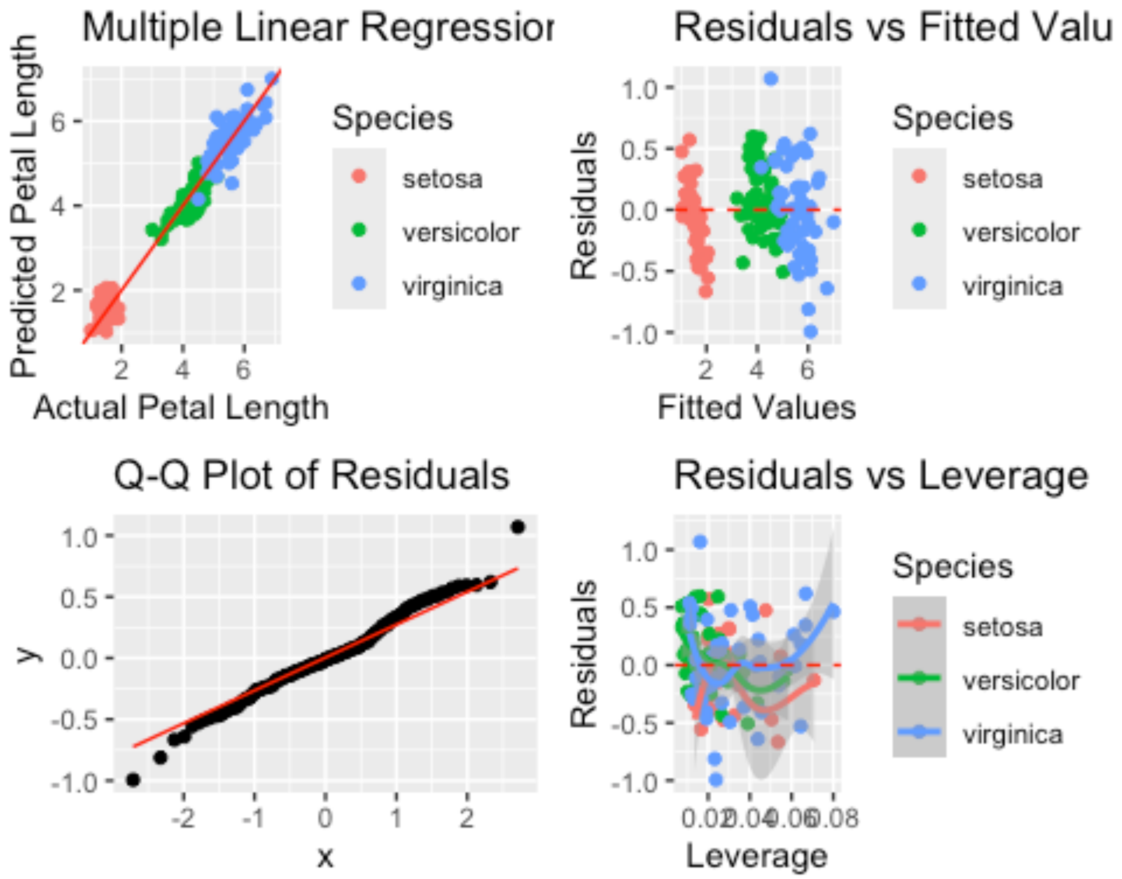
```
multiple_model <- lm(Petal.Length ~ Sepal.Length + Sepal.Width + Petal.Width,
data=iris)
iris$PetalLength_Pred_Multiple <- predict(multiple_model)

summary(multiple_model)

##
## Call:
## lm(formula = Petal.Length ~ Sepal.Length + Sepal.Width + Petal.Width,
##     data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.99333 -0.17656 -0.01004  0.18558  1.06909
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.26271    0.29741  -0.883    0.379
## Sepal.Length  0.72914    0.05832  12.502   <2e-16 ***
## Sepal.Width  -0.64601    0.06850  -9.431   <2e-16 ***
## Petal.Width   1.44679    0.06761  21.399   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.319 on 146 degrees of freedom
## Multiple R-squared:  0.968,  Adjusted R-squared:  0.9674
## F-statistic:  1473 on 3 and 146 DF,  p-value: < 2.2e-16
```

```
ggplot(iris, aes(x=Petal.Length, y=PetalLength_Pred_Multiple, color = Species
)) +
  geom_point() +
  geom_abline(slope=1, intercept=0, color="red") +
  ggtitle("Multiple Linear Regression: Predicted vs Actual Petal Length") +
  xlab("Actual Petal Length") +
  ylab("Predicted Petal Length")
```



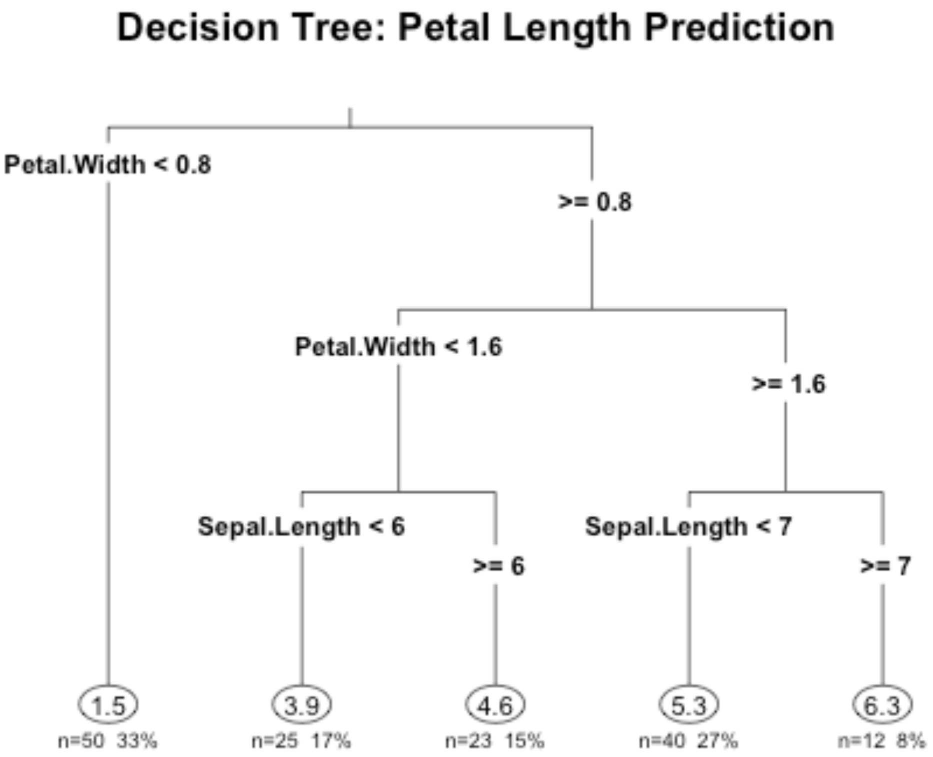
Calculate residuals and plot

```
iris$residuals <- residuals(multiple_model)

pred_vs_actual_plot <- ggplot(iris, aes(x=Petal.Length, y=PetalLength_Pred_Mu
ltiple, color=Species)) +
  geom_point() +
  geom_abline(slope=1, intercept=0, color="red") +
  ggtitle("Multiple Linear Regression: Predicted vs Actual Petal Length") +
  xlab("Actual Petal Length") +
  ylab("Predicted Petal Length")

residuals_vs_fitted_plot <- ggplot(iris, aes(x=PetalLength_Pred_Multiple, y=r
esiduals, color=Species)) +
  geom_point() +
  geom_hline(yintercept=0, linetype="dashed", color="red") +
  ggtitle("Residuals vs Fitted Values") +
  xlab("Fitted Values") +
  ylab("Residuals")

qq_plot <- ggplot(iris, aes(sample=residuals)) +
  stat_qq() +
  stat_qq_line(color="red") +
  ggtitle("Q-Q Plot of Residuals")

residuals_vs_leverage_plot <- ggplot(iris, aes(x=hatvalues(multiple_model), y
=residuals, color=Species)) +
  geom_point() +
  geom_smooth(method="loess") +
  geom_hline(yintercept=0, linetype="dashed", color="red") +
  ggtitle("Residuals vs Leverage") +
  xlab("Leverage") +
  ylab("Residuals")

grid.arrange(pred_vs_actual_plot, residuals_vs_fitted_plot, qq_plot, residual
s_vs_leverage_plot, ncol=2, nrow=2)

## `geom_smooth()` using formula = 'y ~ x'
```



Create and plot Decision Tree Model, calculate R^2

```
tree_model <- rpart(Petal.Length ~ Sepal.Length + Sepal.Width + Petal.Width,
data=iris)
iris$PetalLength_Pred_Tree <- predict(tree_model, newdata=iris)

r_squared <- cor(iris$Petal.Length, iris$PetalLength_Pred_Tree)^2
print(paste("R-squared: ", r_squared))

## [1] "R-squared:  0.967297840096059"

rpart.plot(tree_model, main="Decision Tree: Petal Length Prediction", type=3,
extra=101, under=TRUE, fallen.leaves=TRUE,
           box.palette = NULL, cex=0.7)
```



Select random example observation and print values

```
random_index <- sample(1:nrow(iris), 1)
random_observation <- iris[random_index, ]

random_observation$Species <- as.character(random_observation$Species)

random_observation_subset <- random_observation[c("Sepal.Length", "Sepal.Widt
h", "Petal.Length", "Petal.Width", "Species")]

random_observation_df <- data.frame(Value = as.character(random_observation_s
ubset))
rownames(random_observation_df) <- names(random_observation_subset)

kable(random_observation_df, align = "c", col.names = "Value") %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("Random Observation" = 2))
```

Random Observation

Value

Sepal.Length

5.7

Sepal.Width

3

Petal.Length

4.2

Petal.Width

1.2

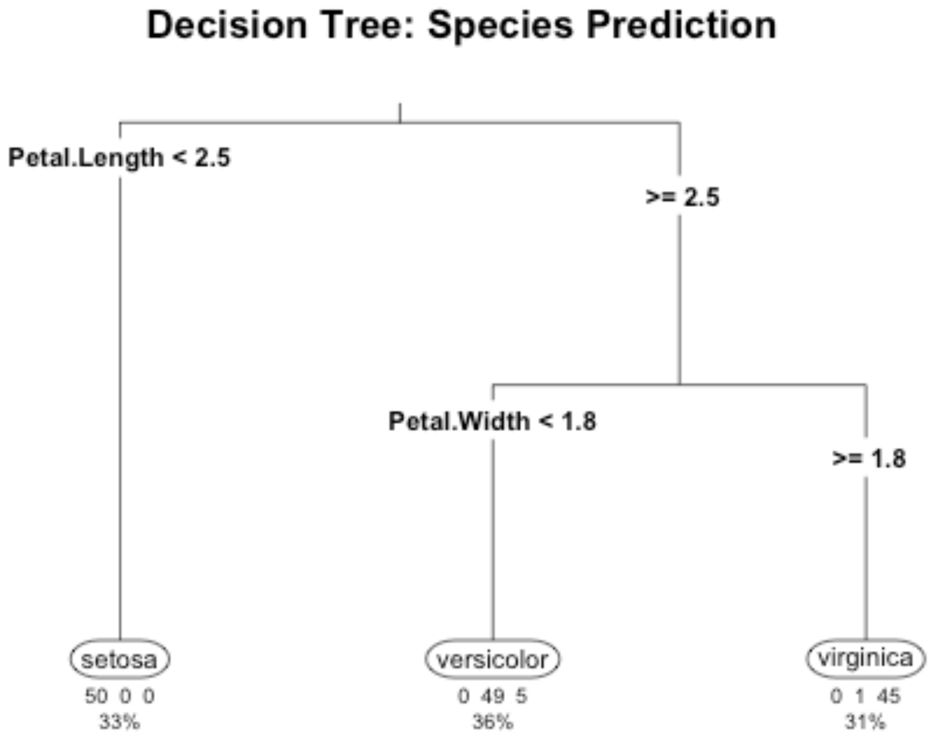
Species

versicolor

Create and plot Classification Tree

```
tree_model2 <- rpart(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Pe
tal.Width, data=iris, method="class")

iris$Species_Pred_Tree2 <- predict(tree_model2, newdata=iris, type="class")

confusion_matrix <- confusionMatrix(iris$Species_Pred_Tree2, iris$Species)
print(confusion_matrix)

## Confusion Matrix and Statistics
## 
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         49         5
##   virginica       0          1        45
## 
## Overall Statistics
## 
##                Accuracy : 0.96
##                  95% CI : (0.915, 0.9852)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.94
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9800           0.9000
## Specificity                 1.0000            0.9500           0.9900
## Pos Pred Value              1.0000          0.9074           0.9783
## Neg Pred Value              1.0000          0.9896           0.9519
## Prevalence                  0.3333          0.3333           0.3333
## Detection Rate              0.3333          0.3267           0.3000
## Detection Prevalence        0.3333          0.3600           0.3067
## Balanced Accuracy           1.0000          0.9650           0.9450
```

```
rpart.plot(tree_model2, main="Decision Tree: Species Prediction", type=3, ext
ra=101, under=TRUE, fallen.leaves=TRUE, box.palette
           = NULL, cex=0.7)
```



Create confusion matrix

```
conf_matrix <- matrix(c(50, 0, 0,
                        0, 49, 5,
                        0, 1, 45),
                      nrow = 3, byrow = TRUE,
                      dimnames = list(Prediction = c("setosa", "versicolor",
"virginica"),
                                      Reference = c("setosa", "versicolor", "
virginica")))

conf_matrix_df <- as.data.frame(conf_matrix)
```

```r
kable(conf_matrix_df, align = "c") %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("Prediction" = 1, "Reference" = 3))
```

Prediction

Reference

setosa

versicolor

virginica

setosa

50

0

0

versicolor

0

49

5

virginica

0

1

45

Build Random Forest to predict species

```r
iris <- iris %>%
  mutate(Species = as.factor(Species))

set.seed(123)
train_index <- sample(nrow(iris), 0.7 * nrow(iris))
train_data <- iris[train_index, ]
test_data <- iris[-train_index, ]

rf_model <- ranger(Species ~ .,
                   data = train_data,
                   num.trees = 500,
                   importance = "impurity")
```

```
predictions <- predict(rf_model, data = test_data)$predictions
confusion_matrix <- table(predictions, test_data$Species)
print(confusion_matrix)
```

```
##             
## predictions  setosa versicolor virginica
##   setosa         14          0         0
##   versicolor      0         17         0
##   virginica       0          1        13
```

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Accuracy:", accuracy))
```

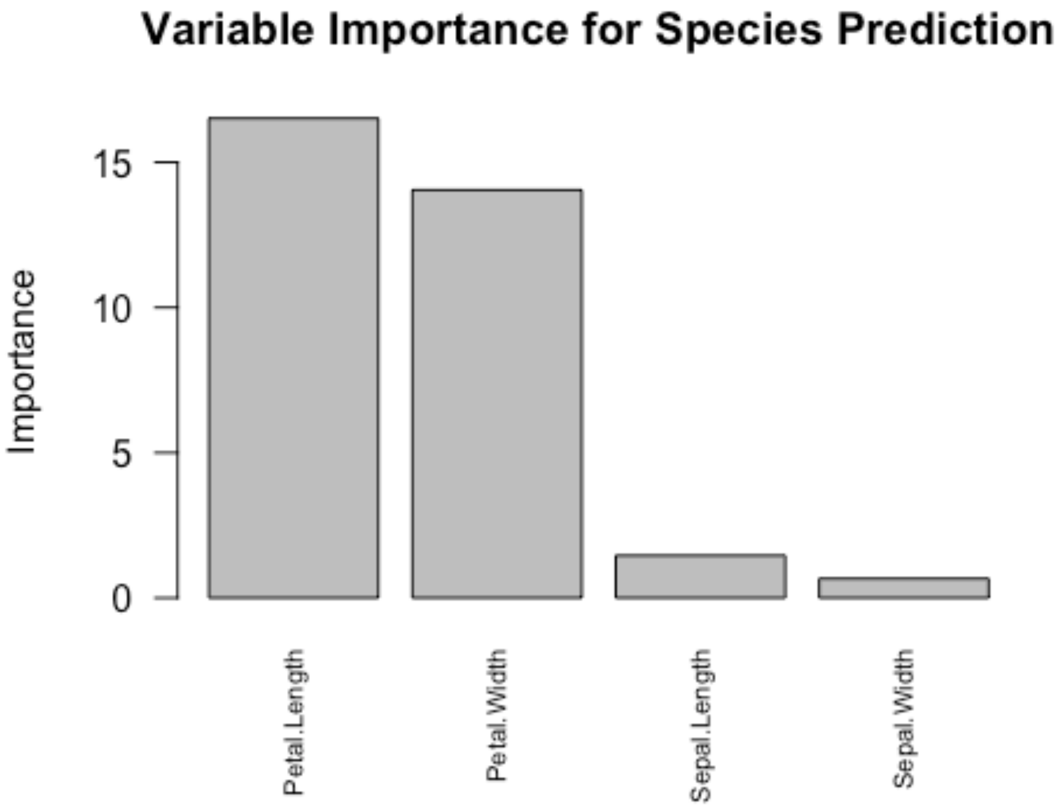
```
## [1] "Accuracy: 0.977777777777778"
```

```
var_importance <- importance(rf_model)
print(var_importance)
```

```
##              Sepal.Length               Sepal.Width              Petal.Len
gth
##                 1.4621263                 0.6659057                16.5138
538
##               Petal.Width   PetalLength_Pred_Simple PetalLength_Pred_Multi
ple
##                14.0494222                 1.3811335                10.0019
958
##                 residuals     PetalLength_Pred_Tree        Species_Pred_Tr
ee2
##                 0.8606321                12.5108281                11.7331
881
```

Variable Importance Plot

```
var_importance <- importance(rf_model)
var_importance <- as.data.frame(var_importance)
var_importance$Variable <- rownames(var_importance)
colnames(var_importance) <- c("Importance", "Variable")

var_importance <- var_importance %>%
  filter(Variable %in% c("Sepal.Length", "Sepal.Width", "Petal.Length", "Peta
l.Width"))

var_importance <- var_importance %>%
  arrange(desc(Importance))

barplot(
  var_importance$Importance,
  main = "Variable Importance for Species Prediction",
  ylab = "Importance",
  col = "gray",
  las = 2,
  names.arg = var_importance$Variable,
  cex.names = 0.7
)
```



Print formatted confusion matrix

```
conf_matrix <- matrix(c(14, 0, 0,
                        0, 17, 0,
                        0, 1, 13),
                      nrow = 3, byrow = TRUE,
                      dimnames = list(Prediction = c("setosa", "versicolor",
"virginica"),
                                      Reference = c("setosa", "versicolor", "
virginica")))

conf_matrix_df <- as.data.frame(conf_matrix)

kable(conf_matrix_df, align = "c") %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("Prediction" = 1, "Reference" = 3))
```

Prediction

Reference

setosa

versicolor

virginica

setosa

14

0

0

versicolor

0

17

0

virginica

0

1

13

```r
conf_matrix1 <- matrix(c(14, 0, 0, 1, 0, 0,
                         0, 17, 0, 0, 1, 0,
                         0, 1, 13, 0, 0.07, 0.93),
                       nrow = 3, byrow = TRUE,
                       dimnames = list(Node = c("Node 1", "Node 2", "Node 3"),
                                       Count = c("N1", "N2", "N3", "P1", "P2",
"P3")))

conf_matrix_df1 <- as.data.frame(conf_matrix1)

kable(conf_matrix_df1, align = "c") %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c(" " = 1, "Count" = 3, "Probability" = 3))
```

Count

Probability

N1

N2

N3

P1

P2

P3

Node 1

14

0

0

1

0.00

0.00

Node 2

0

17

0

0

1.00

0.00

Node 3

0

1

13

0

0.07

0.93

```r
# Calculate Gini Impurity for each node
calculate_gini <- function(probabilities) {
  return(1 - sum(probabilities^2))
}

# Extract the probabilities for each node
```

```
prob_node1 <- as.numeric(conf_matrix1["Node 1", 4:6])
prob_node2 <- as.numeric(conf_matrix1["Node 2", 4:6])
prob_node3 <- as.numeric(conf_matrix1["Node 3", 4:6])

# Calculate Gini Impurity for each node
gini_node1 <- calculate_gini(prob_node1)
gini_node2 <- calculate_gini(prob_node2)
gini_node3 <- calculate_gini(prob_node3)

# Print the Gini Impurity for each node
cat("Gini Impurity for Node 1:", gini_node1, "\n")

## Gini Impurity for Node 1: 0

cat("Gini Impurity for Node 2:", gini_node2, "\n")

## Gini Impurity for Node 2: 0

cat("Gini Impurity for Node 3:", gini_node3, "\n")

## Gini Impurity for Node 3: 0.1302
```

8.2 DATA CLEANING

Values recorded as -99 or -88 are null and must be reflected as such

```
pulse <- pulse2021_puf_27 %>%
  select(-SCRAM) %>%
  mutate_all(~as.numeric(as.character(.)))

pulse <- pulse %>%
  mutate_all(~na_if(., -99))

pulse <- pulse %>%
  mutate_all(~na_if(., -88))
```

Create age variable from birth year

```
pulse$AGE <- (2021 - pulse$TBIRTH_YEAR)
pulse <- select(pulse, -(TBIRTH_YEAR))
```

Deselect irrelevant variables from analysis

```
pulse <- select(pulse, -(WEEK), -(EST_MSA), -(ABIRTH_YEAR), -(AGENDER), -(AHI
SPANIC), -(ARACE), -(AEDUC), -(AHHLD_NUMPER), -(AHHLD_NUMKID), -(HWEIGHT))
```

Filter for complete cases of mental health variables

```
pulse <- pulse[complete.cases(pulse[c("ANXIOUS", "WORRY", "INTEREST", "DOWN")
]), ]
```

Reset mental health questions to a baseline of 0

```
pulse$ANXIOUS<- (pulse$ANXIOUS - 1)
pulse$WORRY <- (pulse$WORRY - 1)
pulse$INTEREST <- (pulse$INTEREST - 1)
pulse$DOWN <- (pulse$DOWN - 1)
```

Calculate GAD and PHQ risk scores

```
pulse$GAD <- (pulse$ANXIOUS + pulse$WORRY)
pulse$PHQ <- (pulse$INTEREST + pulse$DOWN)

pulse$total.risk <- (pulse$PHQ + pulse$GAD)
```

Create binary variables illustrating at risk vs. not at risk on GAD scale and PHQ scale and numeric risk 0-2

```
pulse$risk.GAD[pulse$GAD>=3] <- 1
pulse$risk.GAD[pulse$GAD<=3] <- 0

pulse$risk.PHQ[pulse$PHQ>=3] <- 1
pulse$risk.PHQ[pulse$PHQ<=3] <- 0

pulse$risk.numeric <- (pulse$risk.GAD + pulse$risk.PHQ)
```

Create binary risk variable: 0 = no risk, 1 = at risk on at least one scale

```
pulse$risk.binary[pulse$risk.numeric==0] <- 0
pulse$risk.binary[pulse$risk.numeric==1] <- 1
pulse$risk.binary[pulse$risk.numeric==2] <- 1

pulse <- select(pulse, -(ANXIOUS), -(WORRY), -(INTEREST), -(DOWN))
```

Recode number of children in household to binary 0/1, either no children in household or one or more children in household

```
pulse$THHLD_NUMKID[pulse$THHLD_NUMKID >= 1] <- 1
```

Recode the Race variables: 0 = White, 1 = Black, 2 = Asian, 3 = Hispanic, 4 = Mixed Race

```
pulse$RRACE[pulse$RRACE == 1] <- 0
pulse$RRACE[pulse$RRACE == 2] <- 1
pulse$RRACE[pulse$RRACE == 3] <- 2
pulse$RRACE[pulse$RHISPANIC == 2] <- 3

pulse <- select(pulse, -(RHISPANIC))
```

8.2.1 MULTI-SELECT QUESTIONS

Calculate the range and average multi-selected variables and check that in each case, more than one variable is selected at least once, and create a list of groups of multi-select questions

```
calculate_range_avg <- function(data, cols) {
  range_selected <- range(rowSums(!is.na(data[, cols])))
  avg_selected <- mean(rowSums(!is.na(data[, cols])))
  list(range = range_selected, average = avg_selected)
}

check_multiple_selection <- function(data, cols) {
  any(rowSums(!is.na(data[, cols])) > 1)
}

column_groups <- list(
  CHNGHOW = c("CHNGHOW1", "CHNGHOW2", "CHNGHOW3", "CHNGHOW4", "CHNGHOW5", "CH
NGHOW6", "CHNGHOW7", "CHNGHOW8", "CHNGHOW9", "CHNGHOW10", "CHNGHOW11", "CHNGH
OW12"),
  WHYNOT = c("WHYNOT1", "WHYNOT2", "WHYNOT3", "WHYNOT4", "WHYNOT5", "WHYNOT6"
, "WHYNOT7", "WHYNOT8", "WHYNOT9", "WHYNOT10", "WHYNOT11"),
  SSAPGM = c("SSAPGM1", "SSAPGM2", "SSAPGM3", "SSAPGM4", "SSAPGM5"),
  SSAEXPCT = c("SSAEXPCT1", "SSAEXPCT2", "SSAEXPCT3", "SSAEXPCT4", "SSAEXPCT5
"),
  EIPSPND = c("EIPSPND1", "EIPSPND2", "EIPSPND3", "EIPSPND4", "EIPSPND5", "EI
PSPND6", "EIPSPND7", "EIPSPND8", "EIPSPND9", "EIPSPND10", "EIPSPND11", "EIPSP
ND12", "EIPSPND13"),
  WHYCHNGD = c("WHYCHNGD1", "WHYCHNGD2", "WHYCHNGD3", "WHYCHNGD4", "WHYCHNGD5
", "WHYCHNGD6", "WHYCHNGD7", "WHYCHNGD8", "WHYCHNGD9", "WHYCHNGD10", "WHYCHNG
D11", "WHYCHNGD12", "WHYCHNGD13"),
  SPNDSRC = c("SPNDSRC1", "SPNDSRC2", "SPNDSRC3", "SPNDSRC4", "SPNDSRC5", "SP
NDSRC6", "SPNDSRC7", "SPNDSRC8"),
  FOODSUFRSN = c("FOODSUFRSN1", "FOODSUFRSN2", "FOODSUFRSN3", "FOODSUFRSN4",
"FOODSUFRSN5"),
  WHEREFREE = c("WHEREFREE1", "WHEREFREE2", "WHEREFREE3", "WHEREFREE4", "WHER
EFREE5", "WHEREFREE6", "WHEREFREE7"),
  HLTHINS = c("HLTHINS1", "HLTHINS2", "HLTHINS3", "HLTHINS4", "HLTHINS5", "HL
THINS6", "HLTHINS7", "HLTHINS8"),
  ENROLL = c("ENROLL1", "ENROLL2", "ENROLL3"),
  TEACH = c("TEACH1", "TEACH2", "TEACH3", "TEACH4", "TEACH5"),
  COMP = c("COMP1", "COMP2", "COMP3"),
  INTRNT = c("INTRNT1", "INTRNT2", "INTRNT3"),
  PSPLANS = c("PSPLANS1", "PSPLANS2", "PSPLANS3", "PSPLANS4", "PSPLANS5", "PS
PLANS6"),
  PSCHNG = c("PSCHNG1", "PSCHNG2", "PSCHNG3", "PSCHNG4", "PSCHNG5", "PSCHNG6"
),
  PSWHYCHG = c("PSWHYCHG1", "PSWHYCHG2", "PSWHYCHG3", "PSWHYCHG4", "PSWHYCHG5
", "PSWHYCHG6", "PSWHYCHG7", "PSWHYCHG8", "PSWHYCHG9")
)
```

Count the number of NA values in each column

```
na_count <- colSums(is.na(pulse))

print(na_count)
```

```
##        EST_ST        REGION       PWEIGHT       EGENDER         RRACE 
##             0             0             0             0             0 
##         EEDUC            MS  THHLD_NUMPER  THHLD_NUMKID THHLD_NUMADLT 
##             0           375             0             0             0 
##     RECVDVACC         DOSES       GETVACC       WHYNOT1       WHYNOT2 
##            84         26164         38318         57426         61483 
##       WHYNOT3       WHYNOT4       WHYNOT5       WHYNOT6       WHYNOT7 
##         61414         62381         62845         58266         60556 
##       WHYNOT8       WHYNOT9      WHYNOT10      WHYNOT11      WHYNOTB1 
##         63099         60642         61177         60998         63029 
##      WHYNOTB2      WHYNOTB3      WHYNOTB4      WHYNOTB5      WHYNOTB6 
##         62431         63212         62780         63234         63128 
##      HADCOVID       WRKLOSS     EXPCTLOSS       ANYWORK      KINDWORK 
##            94           107           134            77         27153 
##      RSNNOWRK      TW_START      UI_APPLY       UI_RECV      SSA_RECV 
##         37285          2487            91         54492           239 
##     SSA_APPLY       SSAPGM1       SSAPGM2       SSAPGM3       SSAPGM4 
##           407         62361         63057         63444         63351 
##       SSAPGM5     SSALIKELY     SSAEXPCT1     SSAEXPCT2     SSAEXPCT3 
##         62498         22651         62225         62957         63467 
##     SSAEXPCT4     SSAEXPCT5     SSADECISN           EIP      EIPSPND1 
##         63187         62472         20215           225         49085 
##      EIPSPND2      EIPSPND3      EIPSPND4      EIPSPND5      EIPSPND6 
##         59528         54398         60513         62297         58599 
##      EIPSPND7      EIPSPND8      EIPSPND9     EIPSPND10     EIPSPND11 
##         58072         52499         57893         50414         61134 
##     EIPSPND12     EIPSPND13     EXPNS_DIF      CHNGHOW1      CHNGHOW2 
##         53958         59458           198         37986         51780 
##      CHNGHOW3      CHNGHOW4      CHNGHOW5      CHNGHOW6      CHNGHOW7 
##         58081         47556         61214         36953         57626 
##      CHNGHOW8      CHNGHOW9     CHNGHOW10     CHNGHOW11     CHNGHOW12 
##         53452         49614         60246         62258         42106 
##     WHYCHNGD1     WHYCHNGD2     WHYCHNGD3     WHYCHNGD4     WHYCHNGD5 
##         55974         59494         38902         58458         55969 
##     WHYCHNGD6     WHYCHNGD7     WHYCHNGD8     WHYCHNGD9    WHYCHNGD10 
##         62561         59044         62233         56417         61858 
##    WHYCHNGD11    WHYCHNGD12    WHYCHNGD13      SPNDSRC1      SPNDSRC2 
##         50084         62214         61178         12847         49371 
##      SPNDSRC3      SPNDSRC4      SPNDSRC5      SPNDSRC6      SPNDSRC7 
##         50807         59992         59468         44380         61445 
##      SPNDSRC8     FEWRTRIPS     FEWRTRANS     PLNDTRIPS    CURFOODSUF 
##         61042           283           133            99           128 
##     CHILDFOOD   FOODSUFRSN1   FOODSUFRSN2   FOODSUFRSN3   FOODSUFRSN4 
##         58165         56960         61411         58860         62673 
##   FOODSUFRSN5      FREEFOOD    WHEREFREE1    WHEREFREE2    WHEREFREE3 
##         59272           155         61941         62445         63398 
##    WHEREFREE4    WHEREFREE5    WHEREFREE6    WHEREFREE7       SNAP_YN 
##         62892         63524         62893         62590           477 
##     TSPNDFOOD     TSPNDPRPD      HLTHINS1      HLTHINS2      HLTHINS3 
##          2785          2717          2718          7784          6330 
##      HLTHINS4      HLTHINS5      HLTHINS6      HLTHINS7      HLTHINS8
##          9284          9815         10061         10669         12659
##      PRIVHLTH       PUBHLTH         DELAY        NOTGET     PRESCRIPT
##             0             0          1317          1267          1288
##       MH_SVCS     MH_NOTGET        TENURE        LIVQTR       RENTCUR
##          1303          1240          1973          2176         50599
##       MORTCUR      MORTCONF         EVICT      FORCLOSE       ENROLL1
##         33321         20385         62178         61688         49561
##       ENROLL2       ENROLL3        TEACH1        TEACH2        TEACH3
##         62030         58832         60441         53825         61293
##        TEACH4        TEACH5     COMPAVAIL         COMP1         COMP2
##         61782         61965         49779         54906         55622
##         COMP3   INTRNTAVAIL       INTRNT1       INTRNT2       INTRNT3
##         63468         49876         63141         50391         63448
##       SCHLHRS     TSTDY_HRS       TCH_HRS       TNUM_PS      PSPLANS1
##         49936         50331         50469          4087         61577
##      PSPLANS2      PSPLANS3      PSPLANS4      PSPLANS5      PSPLANS6
##         60970         57791         60797         62277         61709
##       PSCHNG1       PSCHNG2       PSCHNG3       PSCHNG4       PSCHNG5
##         58556         59928         59390         62019         63365
##       PSCHNG6       PSCHNG7     PSWHYCHG1     PSWHYCHG2     PSWHYCHG3
##         63145         63078         61010         63417         62855
##     PSWHYCHG4     PSWHYCHG5     PSWHYCHG6     PSWHYCHG7     PSWHYCHG8
##         59660         62492         62397         61174         61084
##     PSWHYCHG9        INCOME           AGE           GAD           PHQ
##         62591          5259             0             0             0
##    total.risk      risk.GAD      risk.PHQ  risk.numeric   risk.binary
##             0             0             0             0             0
```

Recode the variables in the WHYNOTB group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$WHYNOTB1[pulse$WHYNOTB1 == 1] <- 1
pulse$WHYNOTB2[pulse$WHYNOTB2 == 1] <- 2
pulse$WHYNOTB3[pulse$WHYNOTB3 == 1] <- 3
pulse$WHYNOTB4[pulse$WHYNOTB4 == 1] <- 4
pulse$WHYNOTB5[pulse$WHYNOTB5 == 1] <- 5
pulse$WHYNOTB6[pulse$WHYNOTB6 == 1] <- 6

pulse$WHYNOTB_selected <- apply(pulse[, c("WHYNOTB1", "WHYNOTB2", "WHYNOTB3",
"WHYNOTB4", "WHYNOTB5", "WHYNOTB6")], 1, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})
```

```
pulse <- select(pulse, -(WHYNOTB1), -(WHYNOTB2), -(WHYNOTB3), -(WHYNOTB4), -(
WHYNOTB5), -(WHYNOTB6))
```

Recode the variables in the WHYNOT group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$WHYNOT1[pulse$WHYNOT1 == 1] <- 1
pulse$WHYNOT2[pulse$WHYNOT2 == 1] <- 2
pulse$WHYNOT3[pulse$WHYNOT3 == 1] <- 3
pulse$WHYNOT4[pulse$WHYNOT4 == 1] <- 4
pulse$WHYNOT5[pulse$WHYNOT5 == 1] <- 5
pulse$WHYNOT6[pulse$WHYNOT6 == 1] <- 6
pulse$WHYNOT7[pulse$WHYNOT7 == 1] <- 7
pulse$WHYNOT8[pulse$WHYNOT8 == 1] <- 8
pulse$WHYNOT9[pulse$WHYNOT9 == 1] <- 9
pulse$WHYNOT10[pulse$WHYNOT10 == 1] <- 10
pulse$WHYNOT11[pulse$WHYNOT11 == 1] <- 11

pulse$WHYNOT_selected <- apply(pulse[, c("WHYNOT1", "WHYNOT2", "WHYNOT3", "WH
YNOT4", "WHYNOT5", "WHYNOT6", "WHYNOT7", "WHYNOT8", "WHYNOT9", "WHYNOT10", "W
HYNOT11")], 1, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(WHYNOT1), -(WHYNOT2), -(WHYNOT3), -(WHYNOT4), -(WHYN
OT5), -(WHYNOT6), -(WHYNOT7), -(WHYNOT8), -(WHYNOT9), -(WHYNOT10), -(WHYNOT11
))
```

Recode the variables in the SSAPGM group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$SSAPGM1[pulse$SSAPGM1 == 1] <- 1
pulse$SSAPGM2[pulse$SSAPGM2 == 1] <- 2
pulse$SSAPGM3[pulse$SSAPGM3 == 1] <- 3
pulse$SSAPGM4[pulse$SSAPGM4 == 1] <- 4
pulse$SSAPGM5[pulse$SSAPGM5 == 1] <- 5

pulse$SSAPGM_selected <- apply(pulse[, c("SSAPGM1", "SSAPGM2", "SSAPGM3", "SS
APGM4", "SSAPGM5")], 1, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(SSAPGM1), -(SSAPGM2), -(SSAPGM3), -(SSAPGM4), -(SSAP
GM5))
```

Recode the variables in the SSA group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$SSAEXPCT2[pulse$SSAEXPCT2 == 1] <- 2
pulse$SSAEXPCT3[pulse$SSAEXPCT3 == 1] <- 3
pulse$SSAEXPCT4[pulse$SSAEXPCT4 == 1] <- 4
pulse$SSAEXPCT5[pulse$SSAEXPCT5 == 1] <- 5

pulse$SSAEXPCT_selected <- apply(pulse[, c("SSAEXPCT1", "SSAEXPCT2", "SSAEXPC
T3", "SSAEXPCT4", "SSAEXPCT5")], 1, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(SSAEXPCT1),-(SSAEXPCT2), -(SSAEXPCT3), -(SSAEXPCT4),
-(SSAEXPCT5))
```

Recode the variables in the EIPSPND group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$EIPSPND2[pulse$EIPSPND2 == 1] <- 2
pulse$EIPSPND3[pulse$EIPSPND3 == 1] <- 3
pulse$EIPSPND4[pulse$EIPSPND4 == 1] <- 4
pulse$EIPSPND5[pulse$EIPSPND5 == 1] <- 5
pulse$EIPSPND6[pulse$EIPSPND6 == 1] <- 6
pulse$EIPSPND7[pulse$EIPSPND7 == 1] <- 7
pulse$EIPSPND8[pulse$EIPSPND8 == 1] <- 8
pulse$EIPSPND9[pulse$EIPSPND9 == 1] <- 9
pulse$EIPSPND10[pulse$EIPSPND10 == 1] <- 10
pulse$EIPSPND11[pulse$EIPSPND11 == 1] <- 11
pulse$EIPSPND12[pulse$EIPSPND10 == 1] <- 12
pulse$EIPSPND13[pulse$EIPSPND11 == 1] <- 13

pulse$EIPSPND_selected <- apply(pulse[, c("EIPSPND1", "EIPSPND2", "EIPSPND3",
"EIPSPND4", "EIPSPND5", "EIPSPND6", "EIPSPND7", "EIPSPND8", "EIPSPND9", "EIPS
PND10", "EIPSPND11", "EIPSPND12", "EIPSPND13")], 1, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(EIPSPND1), -(EIPSPND2), -(EIPSPND3), -(EIPSPND4), -(
EIPSPND5), -(EIPSPND6), -(EIPSPND7), -(EIPSPND8), -(EIPSPND9), -(EIPSPND10),
-(EIPSPND11), -(EIPSPND12), -(EIPSPND13))
```

Recode the variables in the CHNGHOW group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$CHNGHOW2[pulse$CHNGHOW2 == 1] <- 2
pulse$CHNGHOW3[pulse$CHNGHOW3 == 1] <- 3
pulse$CHNGHOW4[pulse$CHNGHOW4 == 1] <- 4
pulse$CHNGHOW5[pulse$CHNGHOW5 == 1] <- 5
pulse$CHNGHOW6[pulse$CHNGHOW6 == 1] <- 6
pulse$CHNGHOW7[pulse$CHNGHOW7 == 1] <- 7
pulse$CHNGHOW8[pulse$CHNGHOW8 == 1] <- 8
pulse$CHNGHOW9[pulse$CHNGHOW9 == 1] <- 9
pulse$CHNGHOW10[pulse$CHNGHOW10 == 1] <- 10
pulse$CHNGHOW11[pulse$CHNGHOW11 == 1] <- 11
pulse$CHNGHOW12[pulse$CHNGHOW10 == 1] <- 12

pulse$CHNGHOW_selected <- apply(pulse[, c("CHNGHOW1", "CHNGHOW2", "CHNGHOW3",
"CHNGHOW4", "CHNGHOW5", "CHNGHOW6", "CHNGHOW7", "CHNGHOW8", "CHNGHOW9", "CHNG
HOW10", "CHNGHOW11", "CHNGHOW12")], 1, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(CHNGHOW1), -(CHNGHOW2), -(CHNGHOW3), -(CHNGHOW4), -(
CHNGHOW5), -(CHNGHOW6), -(CHNGHOW7), -(CHNGHOW8), -(CHNGHOW9), -(CHNGHOW10),
-(CHNGHOW11), -(CHNGHOW12))
```

Recode the variables in the WHYCHNGD group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$WHYCHNGD2[pulse$WHYCHNGD2 == 1] <- 2
pulse$WHYCHNGD3[pulse$WHYCHNGD3 == 1] <- 3
pulse$WHYCHNGD4[pulse$WHYCHNGD4 == 1] <- 4
pulse$WHYCHNGD5[pulse$WHYCHNGD5 == 1] <- 5
pulse$WHYCHNGD6[pulse$WHYCHNGD6 == 1] <- 6
pulse$WHYCHNGD7[pulse$WHYCHNGD7 == 1] <- 7
pulse$WHYCHNGD8[pulse$WHYCHNGD8 == 1] <- 8
pulse$WHYCHNGD9[pulse$WHYCHNGD9 == 1] <- 9
pulse$WHYCHNGD10[pulse$WHYCHNGD10 == 1] <- 10
pulse$WHYCHNGD11[pulse$WHYCHNGD11 == 1] <- 11
```

```
pulse$WHYCHNGD12[pulse$WHYCHNGD10 == 1] <- 12
pulse$WHYCHNGD13[pulse$WHYCHNGD11 == 1] <- 13

pulse$WHYCHNGD_selected <- apply(pulse[, c("WHYCHNGD1", "WHYCHNGD2", "WHYCHNG
D3", "WHYCHNGD4", "WHYCHNGD5", "WHYCHNGD6", "WHYCHNGD7", "WHYCHNGD8", "WHYCHN
GD9", "WHYCHNGD10", "WHYCHNGD11", "WHYCHNGD12", "WHYCHNGD13")], 1, function(r
ow) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(WHYCHNGD1), -(WHYCHNGD2), -(WHYCHNGD3), -(WHYCHNGD4)
, -(WHYCHNGD5), -(WHYCHNGD6), -(WHYCHNGD7), -(WHYCHNGD8), -(WHYCHNGD9), -(WHY
CHNGD10), -(WHYCHNGD11), -(WHYCHNGD12), -(WHYCHNGD13))
```

Recode the variables in the SPNDSRC group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$SPNDSRC2[pulse$SPNDSRC2 == 1] <- 2
pulse$SPNDSRC3[pulse$SPNDSRC3 == 1] <- 3
pulse$SPNDSRC4[pulse$SPNDSRC4 == 1] <- 4
pulse$SPNDSRC5[pulse$SPNDSRC5 == 1] <- 5
pulse$SPNDSRC6[pulse$SPNDSRC6 == 1] <- 6
pulse$SPNDSRC7[pulse$SPNDSRC7 == 1] <- 7
pulse$SPNDSRC8[pulse$SPNDSRC8 == 1] <- 8

pulse$SPNDSRC_selected <- apply(pulse[, c("SPNDSRC1", "SPNDSRC2", "SPNDSRC3",
"SPNDSRC4", "SPNDSRC5", "SPNDSRC6", "SPNDSRC7", "SPNDSRC8")], 1, function(row
) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(SPNDSRC1), -(SPNDSRC2), -(SPNDSRC3), -(SPNDSRC4), -(
SPNDSRC5), -(SPNDSRC6), -(SPNDSRC7), -(SPNDSRC8))
```

Recode the variables in the FOODSUFRSN group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$FOODSUFRSN2[pulse$FOODSUFRSN2 == 1] <- 2
pulse$FOODSUFRSN3[pulse$FOODSUFRSN3 == 1] <- 3
pulse$FOODSUFRSN4[pulse$FOODSUFRSN4 == 1] <- 4
```

```
pulse$FOODSUFRSN5[pulse$FOODSUFRSN5 == 1] <- 5

pulse$FOODSUFRSN_selected <- apply(pulse[, c("FOODSUFRSN1", "FOODSUFRSN2", "F
OODSUFRSN3", "FOODSUFRSN4", "FOODSUFRSN5")], 1, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(FOODSUFRSN1), -(FOODSUFRSN2), -(FOODSUFRSN3), -(FOOD
SUFRSN4), -(FOODSUFRSN5))
```

Recode the variables in the WHEREFREE group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$WHEREFREE2[pulse$WHEREFREE2 == 1] <- 2
pulse$WHEREFREE3[pulse$WHEREFREE3 == 1] <- 3
pulse$WHEREFREE4[pulse$WHEREFREE4 == 1] <- 4
pulse$WHEREFREE5[pulse$WHEREFREE5 == 1] <- 5
pulse$WHEREFREE6[pulse$WHEREFREE6 == 1] <- 6
pulse$WHEREFREE7[pulse$WHEREFREE7 == 1] <- 7

pulse$WHEREFREE_selected <- apply(pulse[, c("WHEREFREE1", "WHEREFREE2", "WHER
EFREE3", "WHEREFREE4", "WHEREFREE5", "WHEREFREE6", "WHEREFREE7")], 1, functio
n(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(WHEREFREE1), -(WHEREFREE2), -(WHEREFREE3), -(WHEREFR
EE4), -(WHEREFREE5), -(WHEREFREE6), -(WHEREFREE7))
```

Recode the variables in the HLTHINS group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$HLTHINS2[pulse$HLTHINS2 == 1] <- 2
pulse$HLTHINS3[pulse$HLTHINS3 == 1] <- 3
pulse$HLTHINS4[pulse$HLTHINS4 == 1] <- 4
pulse$HLTHINS5[pulse$HLTHINS5 == 1] <- 5
pulse$HLTHINS6[pulse$HLTHINS6 == 1] <- 6
pulse$HLTHINS7[pulse$HLTHINS7 == 1] <- 7
pulse$HLTHINS8[pulse$HLTHINS8 == 1] <- 8
```

```
pulse$HLTHINS_selected <- apply(pulse[, c("HLTHINS1", "HLTHINS2", "HLTHINS3",
"HLTHINS4", "HLTHINS5", "HLTHINS6", "HLTHINS7", "HLTHINS8")], 1, function(row
) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(HLTHINS1), -(HLTHINS2), -(HLTHINS3), -(HLTHINS4), -(
HLTHINS5), -(HLTHINS6), -(HLTHINS7), -(HLTHINS8))
```

Recode the variables in the ENROLL group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$ENROLL2[pulse$ENROLL2 == 1] <- 2
pulse$ENROLL3[pulse$ENROLL3 == 1] <- 3

pulse$ENROLL_selected <- apply(pulse[, c("ENROLL1", "ENROLL2", "ENROLL3")], 1
, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(ENROLL1), -(ENROLL2), -(ENROLL3))
```

Recode the variables in the TEACH group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$TEACH2[pulse$TEACH2 == 1] <- 2
pulse$TEACH3[pulse$TEACH3 == 1] <- 3
pulse$TEACH4[pulse$TEACH4 == 1] <- 4
pulse$TEACH5[pulse$TEACH5 == 1] <- 5

pulse$TEACH_selected <- apply(pulse[, c("TEACH1", "TEACH2", "TEACH3", "TEACH4
", "TEACH5")], 1, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})
```

```
pulse <- select(pulse, -(TEACH1), -(TEACH2), -(TEACH3), -(TEACH4), -(TEACH5))
```

Recode the variables in the COMP group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$COMP2[pulse$COMP2 == 1] <- 2
pulse$COMP3[pulse$COMP3 == 1] <- 3

pulse$COMP_selected <- apply(pulse[, c("COMP1", "COMP2", "COMP3")], 1, functi
on(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(COMP1), -(COMP2), -(COMP3))
```

Recode the variables in the INTRNT group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$INTRNT2[pulse$INTRNT2 == 1] <- 2
pulse$INTRNT3[pulse$INTRNT3 == 1] <- 3

pulse$INTRNT_selected <- apply(pulse[, c("INTRNT1", "INTRNT2", "INTRNT3")], 1
, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(INTRNT1), -(INTRNT2), -(INTRNT3))
```

Recode the variables in the PSPLANS group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$PSPLANS2[pulse$PSPLANS2 == 1] <- 2
pulse$PSPLANS3[pulse$PSPLANS3 == 1] <- 3
pulse$PSPLANS4[pulse$PSPLANS4 == 1] <- 4
pulse$PSPLANS5[pulse$PSPLANS5 == 1] <- 5
pulse$PSPLANS6[pulse$PSPLANS6 == 1] <- 6

pulse$PSPLANS_selected <- apply(pulse[, c("PSPLANS1", "PSPLANS2", "PSPLANS3",
"PSPLANS4", "PSPLANS5", "PSPLANS6")], 1, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(PSPLANS1), -(PSPLANS2), -(PSPLANS3), -(PSPLANS4), -(
PSPLANS5), -(PSPLANS6))
```

Recode the variables in the PSCHNG group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$PSCHNG2[pulse$PSCHNG2 == 1] <- 2
pulse$PSCHNG3[pulse$PSCHNG3 == 1] <- 3
pulse$PSCHNG4[pulse$PSCHNG4 == 1] <- 4
pulse$PSCHNG5[pulse$PSCHNG5 == 1] <- 5
pulse$PSCHNG6[pulse$PSCHNG6 == 1] <- 6

pulse$PSCHNG_selected <- apply(pulse[, c("PSCHNG1", "PSCHNG2", "PSCHNG3", "PS
CHNG4", "PSCHNG5", "PSCHNG6")], 1, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(PSCHNG1), -(PSCHNG2), -(PSCHNG3), -(PSCHNG4), -(PSCH
NG5), -(PSCHNG6))
```

Recode the variables in the PSWHYCHG group to only include as needed, collapse into one column, and deselect orginal variables

```
pulse$PSWHYCHG2[pulse$PSWHYCHG2 == 1] <- 2
pulse$PSWHYCHG3[pulse$PSWHYCHG3 == 1] <- 3
pulse$PSWHYCHG4[pulse$PSWHYCHG4 == 1] <- 4
pulse$PSWHYCHG5[pulse$PSWHYCHG5 == 1] <- 5
pulse$PSWHYCHG6[pulse$PSWHYCHG6 == 1] <- 6
pulse$PSWHYCHG7[pulse$PSWHYCHG7 == 1] <- 7
pulse$PSWHYCHG8[pulse$PSWHYCHG8 == 1] <- 8
pulse$PSWHYCHG9[pulse$PSWHYCHG9 == 1] <- 9

pulse$PSWHYCHG_selected <- apply(pulse[, c("PSWHYCHG1", "PSWHYCHG2", "PSWHYCH
G3", "PSWHYCHG4", "PSWHYCHG5", "PSWHYCHG6", "PSWHYCHG7", "PSWHYCHG8", "PSWHYC
HG9")], 1, function(row) {
  selected <- which(!is.na(row) & row != 0)
  if (length(selected) == 0) {
    NA
  } else {
    paste(row[selected], collapse = ",")
  }
})

pulse <- select(pulse, -(PSWHYCHG1), -(PSWHYCHG2), -(PSWHYCHG3), -(PSWHYCHG4)
, -(PSWHYCHG5), -(PSWHYCHG6), -(PSWHYCHG7), -(PSWHYCHG8), -(PSWHYCHG9))
```

8.2.2 IMPUTATION

Count NA

```
na_count <- colSums(is.na(pulse))

print(data.frame(Column = names(na_count), NA_Count = na_count))

##                                  Column NA_Count
## EST_ST                           EST_ST        0
## REGION                           REGION        0
## PWEIGHT                         PWEIGHT        0
## EGENDER                         EGENDER        0
## RRACE                             RRACE        0
## EEDUC                             EEDUC        0
## MS                                   MS      375
## THHLD_NUMPER               THHLD_NUMPER        0
## THHLD_NUMKID               THHLD_NUMKID        0
## THHLD_NUMADLT             THHLD_NUMADLT        0
## RECVDVACC                     RECVDVACC       84
## DOSES                             DOSES    26164
## GETVACC                         GETVACC    38318
## HADCOVID                       HADCOVID       94
## WRKLOSS                         WRKLOSS      107
## EXPCTLOSS                     EXPCTLOSS      134
## ANYWORK                         ANYWORK       77
## KINDWORK                       KINDWORK    27153
## RSNNOWRK                       RSNNOWRK    37285
## TW_START                       TW_START     2487
## UI_APPLY                       UI_APPLY       91
## UI_RECV                         UI_RECV    54492
## SSA_RECV                       SSA_RECV      239
## SSA_APPLY                     SSA_APPLY      407
## SSALIKELY                     SSALIKELY    22651
## SSADECISN                     SSADECISN    20215
## EIP                                 EIP      225
## EXPNS_DIF                     EXPNS_DIF      198
## FEWRTRIPS                     FEWRTRIPS      283
## FEWRTRANS                     FEWRTRANS      133
## PLNDTRIPS                     PLNDTRIPS       99
## CURFOODSUF                   CURFOODSUF      128
## CHILDFOOD                     CHILDFOOD    58165
## FREEFOOD                       FREEFOOD      155
## SNAP_YN                         SNAP_YN      477
## TSPNDFOOD                     TSPNDFOOD     2785
## TSPNDPRPD                     TSPNDPRPD     2717
## PRIVHLTH                       PRIVHLTH        0
## PUBHLTH                         PUBHLTH        0
## DELAY                             DELAY     1317
## NOTGET                           NOTGET     1267
## PRESCRIPT                     PRESCRIPT     1288
## MH_SVCS                         MH_SVCS     1303
## MH_NOTGET                     MH_NOTGET     1240
## TENURE                           TENURE     1973
## LIVQTR                           LIVQTR     2176
## RENTCUR                         RENTCUR    50599
## MORTCUR                         MORTCUR    33321
## MORTCONF                       MORTCONF    20385
## EVICT                             EVICT    62178
## FORCLOSE                       FORCLOSE    61688
## COMPAVAIL                     COMPAVAIL    49779
## INTRNTAVAIL                 INTRNTAVAIL    49876
## SCHLHRS                         SCHLHRS    49936
## TSTDY_HRS                     TSTDY_HRS    50331
## TCH_HRS                         TCH_HRS    50469
## TNUM_PS                         TNUM_PS     4087
## PSCHNG7                         PSCHNG7    63078
## INCOME                           INCOME     5259
## AGE                                 AGE        0
## GAD                                 GAD        0
## PHQ                                 PHQ        0
## total.risk                   total.risk        0
## risk.GAD                       risk.GAD        0
## risk.PHQ                       risk.PHQ        0
## risk.numeric               risk.numeric        0
## risk.binary                 risk.binary        0
## WHYNOTB_selected       WHYNOTB_selected    61442
## WHYNOT_selected         WHYNOT_selected    51133
## SSAPGM_selected         SSAPGM_selected    60960
## SSAEXPCT_selected     SSAEXPCT_selected    60788
## EIPSPND_selected       EIPSPND_selected    28596
## CHNGHOW_selected       CHNGHOW_selected      495
## WHYCHNGD_selected     WHYCHNGD_selected    23183
## SPNDSRC_selected       SPNDSRC_selected      980
## FOODSUFRSN_selected FOODSUFRSN_selected    49728
## WHEREFREE_selected   WHEREFREE_selected    59746
## HLTHINS_selected       HLTHINS_selected      597
## ENROLL_selected         ENROLL_selected    43794
## TEACH_selected           TEACH_selected    49812
## COMP_selected             COMP_selected    50015
## INTRNT_selected         INTRNT_selected    50048
## PSPLANS_selected       PSPLANS_selected    50190
## PSCHNG_selected         PSCHNG_selected    50701
## PSWHYCHG_selected     PSWHYCHG_selected    54863
```

Find and impute NA with median value, recount NA

```
median_responses <- sapply(pulse, function(x) {
  if (is.numeric(x)) {
    median(x, na.rm = TRUE)
  } else {
    NA
  }
})

pulse_imputed <- pulse
for (col in names(pulse)) {
  if (is.numeric(pulse_imputed[[col]])) {
    pulse_imputed[[col]][is.na(pulse_imputed[[col]])] <- median_responses[col
]
  }
}

na_count <- colSums(is.na(pulse_imputed))

print(na_count)
```

```
##              EST_ST              REGION             PWEIGHT              EG
ENDER
##                   0                   0                   0
0
##               RRACE               EEDUC                  MS         THHLD_N
UMPER
##                   0                   0                   0
0
##         THHLD_NUMKID       THHLD_NUMADLT           RECVDVACC
DOSES
##                   0                   0                   0
0
##             GETVACC            HADCOVID             WRKLOSS            EXPC
TLOSS
##                   0                   0                   0
0
##             ANYWORK            KINDWORK            RSNNOWRK             TW_
START
##                   0                   0                   0
0
##            UI_APPLY             UI_RECV            SSA_RECV            SSA_
APPLY
##                   0                   0                   0
0
##           SSALIKELY           SSADECISN                 EIP            EXPN
```

```
S_DIF
##                   0                   0                   0
0
##           FEWRTRIPS           FEWRTRANS           PLNDTRIPS          CURFO
ODSUF
##                   0                   0                   0
0
##           CHILDFOOD            FREEFOOD             SNAP_YN           TSPN
DFOOD
##                   0                   0                   0
0
##           TSPNDPRPD            PRIVHLTH             PUBHLTH
DELAY
##                   0                   0                   0
0
##              NOTGET           PRESCRIPT             MH_SVCS           MH_N
OTGET
##                   0                   0                   0
0
##              TENURE              LIVQTR             RENTCUR             MO
RTCUR
##                   0                   0                   0
0
##            MORTCONF               EVICT            FORCLOSE           COMP
AVAIL
##                   0                   0                   0
0
##         INTRNTAVAIL             SCHLHRS           TSTDY_HRS             TC
H_HRS
##                   0                   0                   0
0
##             TNUM_PS             PSCHNG7              INCOME
AGE
##                   0                   0                   0
0
##                 GAD                 PHQ          total.risk            ris
k.GAD
##                   0                   0                   0
0
##            risk.PHQ        risk.numeric         risk.binary    WHYNOTB_sel
ected
##                   0                   0                   0
61442
##     WHYNOT_selected     SSAPGM_selected   SSAEXPCT_selected    EIPSPND_sel
ected
##               51133               60960               60788
28596
##    CHNGHOW_selected   WHYCHNGD_selected    SPNDSRC_selected FOODSUFRSN_sel
ected
##                 495               23183                 980
```

```
49728
##  WHEREFREE_selected    HLTHINS_selected     ENROLL_selected      TEACH_sel
ected
##               59746                 597               43794               
49812
##       COMP_selected     INTRNT_selected    PSPLANS_selected     PSCHNG_sel
ected
##               50015               50048               50190               
50701
##   PSWHYCHG_selected
##               54863
```

8.3 MODELING

8.3.1 RANDOM FORESTS

Deselect multi-select variables for initial analysis

```
pulse_imputed_select <- subset(pulse_imputed, select = -c(WHYNOTB_selected, W
HYNOT_selected, SSAPGM_selected, SSAEXPCT_selected,
                                                          EIPSPND_selected, C
HNGHOW_selected, WHYCHNGD_selected, SPNDSRC_selected,
                                                          FOODSUFRSN_selected
, WHEREFREE_selected, HLTHINS_selected, ENROLL_selected,
                                                          TEACH_selected, COM
P_selected, INTRNT_selected, PSPLANS_selected, PSCHNG_selected, PSWHYCHG_sele
cted))
```

Split train and test dataset

```
set.seed(123)
train_index <- sample(nrow(pulse_imputed_select), 0.7 * nrow(pulse_imputed_se
lect))
train_pulse <- pulse_imputed_select[train_index, ]
test_pulse <- pulse_imputed_select[-train_index, ]
```

Set predictors

```
predictors <- setdiff(names(train_pulse), c("GAD", "PHQ", "total.risk", "risk
.GAD", "risk.PHQ", "risk.numeric", "risk.binary"))
predictors <- setdiff(names(test_pulse), c("GAD", "PHQ", "total.risk", "risk.
GAD", "risk.PHQ", "risk.numeric", "risk.binary"))
```

Fit random forest model and print variable importance for GAD

```
rf_gad <- ranger(GAD ~ ., data = train_pulse, importance = "impurity", num.tr
ees = 500)

predictions <- predict(rf_gad, data = test_pulse)$predictions
confusion_matrix <- table(predictions, test_pulse$GAD)
```

```r
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Accuracy:", accuracy))

## [1] "Accuracy: 0.000786204727711096"

gad_importance <- rf_gad$variable.importance

sorted_importance_gad <- sort(gad_importance, decreasing = TRUE)

print("GAD Variable Importance:")

## [1] "GAD Variable Importance:"

print(sorted_importance_gad)

##     total.risk      risk.GAD  risk.numeric   risk.binary           PHQ
##    45786.04707   33277.79889   20598.64514   19527.60978   16952.39386
##       risk.PHQ     EXPNS_DIF     MH_NOTGET    CURFOODSUF     PRESCRIPT
##     9191.73766    2608.66198    2340.11944    1518.17372    1463.73370
##          DELAY           AGE      MORTCONF       PWEIGHT        NOTGET
##     1118.31632     960.33521     842.08062     735.49797     676.89782
##         EST_ST     FEWRTRIPS     TSPNDFOOD       MH_SVCS     TSPNDPRPD
##      583.88133     560.27925     557.44317     552.37359     486.80782
##         INCOME     EXPCTLOSS       WRKLOSS         EEDUC        TENURE
##      397.31680     386.25662     356.57780     308.93839     302.66357
##   THHLD_NUMPER            MS     FEWRTRANS      RSNNOWRK        LIVQTR
##      286.84401     279.66802     277.29028     277.13383     250.28845
##         REGION           EIP      TW_START       TNUM_PS       EGENDER
##      243.22890     229.91822     229.56477     224.03469     219.81890
##  THHLD_NUMADLT     TSTDY_HRS      KINDWORK         RRACE       PUBHLTH
##      219.24560     207.08051     197.59961     178.56945     165.45080
##        GETVACC      SSA_RECV      UI_APPLY       TCH_HRS      PRIVHLTH
##      160.22285     130.45964     127.89516     121.10996     116.24922
##      RECVDVACC     PLNDTRIPS     SSADECISN       ANYWORK  THHLD_NUMKID
##      114.66729     101.86304     101.51908      92.44372      90.66594
##       HADCOVID       SCHLHRS       SNAP_YN     SSALIKELY      FREEFOOD
##       89.26171      68.31045      60.93298      60.70153      57.66880
##      SSA_APPLY   INTRNTAVAIL     COMPAVAIL         EVICT     CHILDFOOD
##       46.69333      46.11686      41.82105      41.09012      40.85797
##        UI_RECV       MORTCUR         DOSES      FORCLOSE       RENTCUR
##       35.45613      33.44547      27.97602      26.43185      24.49699
##        PSCHNG7
##        0.00000
```

Fit random forest model and print sorted variable importance for PHQ

```r
rf_phq <- ranger(PHQ ~ ., data = train_pulse, importance = "impurity", num.tr
ees = 500)

predictions <- predict(rf_phq, data = test_pulse)$predictions
confusion_matrix <- table(predictions, test_pulse$PHQ)
```

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.00167723675245034"
```

```
phq_importance <- rf_phq$variable.importance

sorted_importance_phq <- sort(phq_importance, decreasing = TRUE)

print("PHQ Variable Importance:")
```

```
## [1] "PHQ Variable Importance:"
```

```
print(sorted_importance_phq)
```

```
##    total.risk      risk.PHQ  risk.numeric   risk.binary           GAD 
##   40870.23091   25145.47798   18508.70654   15134.80211   12817.04882 
##      risk.GAD     MH_NOTGET     EXPNS_DIF    CURFOODSUF     PRESCRIPT 
##    8040.11680    2053.94381    1902.21455    1660.17243    1081.21479 
##           AGE       PWEIGHT      MORTCONF        NOTGET        EST_ST 
##     712.11729     694.18213     658.47177     569.38883     560.48990 
##     TSPNDFOOD     FEWRTRIPS     TSPNDPRPD        INCOME         DELAY 
##     539.61012     463.65956     461.13426     457.24834     422.44335 
##            MS       MH_SVCS         EEDUC  THHLD_NUMPER       WRKLOSS 
##     406.09310     388.30317     331.87863     268.47733     258.98816 
##      RSNNOWRK        LIVQTR     EXPCTLOSS        REGION           EIP 
##     240.22280     239.65155     232.49569     225.11173     216.10957 
##        TENURE THHLD_NUMADLT     TSTDY_HRS      KINDWORK      TW_START 
##     212.83370     208.55425     201.27284     189.84624     176.14970 
##     FEWRTRANS       TNUM_PS         RRACE       PUBHLTH       GETVACC 
##     171.86663     167.18502     165.45524     142.60718     138.72438 
##      PRIVHLTH       EGENDER       TCH_HRS     RECVDVACC     SSADECISN 
##     127.28661     124.88875     117.28804     114.76453      99.38700 
##       ANYWORK     PLNDTRIPS      UI_APPLY      HADCOVID  THHLD_NUMKID 
##      93.41995      90.45595      89.63692      85.31959      81.89004 
##      SSA_RECV       SCHLHRS     SSALIKELY       SNAP_YN      FREEFOOD 
##      78.34049      61.72738      61.35630      61.31838      59.70158 
##     SSA_APPLY   INTRNTAVAIL     COMPAVAIL     CHILDFOOD      FORCLOSE 
##      45.24136      43.37820      42.08728      38.12969      35.45597 
##       UI_RECV       MORTCUR         EVICT         DOSES       RENTCUR 
##      33.77353      33.42923      32.54078      25.98295      25.18650 
##       PSCHNG7 
##       0.00000
```

Fit random forest model and print sorted variable importance for Total Risk

```
rf_total_risk <- ranger(total.risk ~ ., data = train_pulse, importance = "imp
urity", num.trees = 500)

predictions <- predict(rf_total_risk, data = test_pulse)$predictions
confusion_matrix <- table(predictions, test_pulse$total.risk)
```

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.000314481891084438"
```

```
total_risk_importance <- rf_total_risk$variable.importance

sorted_importance_total <- sort(total_risk_importance, decreasing = TRUE)

print("Total Risk Variable Importance:")
```

```
## [1] "Total Risk Variable Importance:"
```

```
print(sorted_importance_total)
```

```
##           GAD           PHQ   risk.binary  risk.numeric      risk.GAD 
##  110572.72350  100237.32494   95506.24520   81588.19761   57509.30793 
##      risk.PHQ     MH_NOTGET     EXPNS_DIF    CURFOODSUF     PRESCRIPT 
##   42905.19074    8078.10193    7354.35899    6516.60580    4733.25909 
##        NOTGET      MORTCONF         DELAY       MH_SVCS     FEWRTRIPS 
##    2959.27045    2581.29823    2567.12500    2124.61134    1864.55595 
##           AGE       PWEIGHT       WRKLOSS        EST_ST     TSPNDFOOD 
##    1862.27139    1138.03231    1017.51393     904.76365     863.68906 
##     TSPNDPRPD     EXPCTLOSS        INCOME        TENURE            MS 
##     763.29833     759.15790     757.79203     565.78241     526.38925 
##         EEDUC     FEWRTRANS  THHLD_NUMPER        LIVQTR      RSNNOWRK 
##     508.07565     461.03514     434.18406     423.05691     411.54685 
##       TNUM_PS        REGION           EIP THHLD_NUMADLT     TSTDY_HRS 
##     407.35747     366.26848     357.15593     342.34040     333.82771 
##      TW_START       EGENDER      KINDWORK         RRACE       GETVACC 
##     333.80655     312.01170     302.84634     283.10391     260.59110 
##       PUBHLTH     RECVDVACC       TCH_HRS      PRIVHLTH     SSADECISN 
##     247.45951     243.48413     203.32439     201.95739     199.66239 
##       SNAP_YN      UI_APPLY      SSA_RECV     PLNDTRIPS       ANYWORK 
##     197.46894     182.29956     167.91968     162.01397     152.33224 
##      HADCOVID  THHLD_NUMKID     CHILDFOOD       SCHLHRS      FREEFOOD 
##     139.69596     135.60945     117.59600     112.16208     100.37094 
##     SSALIKELY       RENTCUR   INTRNTAVAIL         EVICT     COMPAVAIL 
##      98.48099      89.90729      84.85361      81.75150      78.99514 
##     SSA_APPLY      FORCLOSE       UI_RECV       MORTCUR         DOSES 
##      74.15119      58.43667      57.83814      54.79427      41.92924 
##       PSCHNG7 
##       0.00000
```

Fit random forest model and print sorted variable importance for GAD Risk

```
rf_risk_gad <- ranger(risk.GAD ~ ., data = train_pulse, importance = "impurity", num.trees = 500)

predictions <- predict(rf_risk_gad, data = test_pulse)$predictions
confusion_matrix <- table(predictions, test_pulse$risk.GAD)
```

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.699302898474763"
```

```
risk_gad_importance <- rf_risk_gad$variable.importance

sorted_importance_risk_gad <- sort(risk_gad_importance, decreasing = TRUE)

print("Risk GAD Variable Importance:")
```

```
## [1] "Risk GAD Variable Importance:"
```

```
print(sorted_importance_risk_gad)
```

```
##           GAD   risk.binary  risk.numeric    total.risk           PHQ
##  2115.2319933  1367.6696623  1341.8570401   995.5432372   425.4303907
##      risk.PHQ     MH_NOTGET     EXPNS_DIF    CURFOODSUF      MORTCONF
##   374.5795660    67.2942788    58.2054005    39.6560791    24.5907559
##     PRESCRIPT        NOTGET         DELAY     EXPCTLOSS       MH_SVCS
##    20.0391870    16.0068581    12.6333910    11.2150245    10.3168018
##           AGE       WRKLOSS       PWEIGHT     TSPNDFOOD        EST_ST
##     8.0251191     7.9563327     4.1800487     3.0415876     3.0318247
##     TSPNDPRPD     FEWRTRIPS        INCOME      RSNNOWRK            MS
##     2.7234495     2.5165825     2.4747479     2.3539451     2.2452319
##        TENURE  THHLD_NUMPER         EEDUC        LIVQTR THHLD_NUMADLT
##     2.2343604     2.1723839     1.8568555     1.5442986     1.4509971
##       TNUM_PS     FEWRTRANS        REGION     TSTDY_HRS           EIP
##     1.4409653     1.4058170     1.1074658     1.0865468     1.0690735
##      SSA_RECV      TW_START         RRACE      KINDWORK      UI_APPLY
##     1.0528823     1.0303957     1.0084852     0.9428295     0.9161835
##       PUBHLTH       GETVACC      PRIVHLTH       EGENDER     SSADECISN
##     0.8159340     0.7925889     0.7912502     0.7356907     0.7098178
##     RECVDVACC     CHILDFOOD       TCH_HRS       RENTCUR       ANYWORK
##     0.6518002     0.6340086     0.6088212     0.5923009     0.5681884
##     PLNDTRIPS     SSALIKELY       SNAP_YN      HADCOVID       SCHLHRS
##     0.4949954     0.4691216     0.4453140     0.4384880     0.4285502
##         EVICT     COMPAVAIL   INTRNTAVAIL      FREEFOOD      FORCLOSE
##     0.4272450     0.4186225     0.3995486     0.3907498     0.3306468
##  THHLD_NUMKID     SSA_APPLY       UI_RECV         DOSES       MORTCUR
##     0.3034722     0.2780755     0.2684817     0.1587992     0.1558872
##       PSCHNG7
##     0.0000000
```

Fit random forest model and print sorted variable importance for PHQ Risk

```
rf_risk_phq <- ranger(risk.PHQ ~ ., data = train_pulse, importance = "impurit
y", num.trees = 500)

predictions <- predict(rf_risk_phq, data = test_pulse)$predictions
confusion_matrix <- table(predictions, test_pulse$risk.PHQ)
```

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Accuracy:", accuracy))

## [1] "Accuracy: 0.693327742544159"

risk_phq_importance <- rf_risk_phq$variable.importance

sorted_importance_risk_phq <- sort(risk_phq_importance, decreasing = TRUE)

print("Risk PHQ Variable Importance:")

## [1] "Risk PHQ Variable Importance:"

print(sorted_importance_risk_phq)

##           PHQ  risk.numeric    total.risk   risk.binary           GAD
##  1760.2642133  1283.0273215   844.1396589   598.2453837   380.9099273
##      risk.GAD     MH_NOTGET    CURFOODSUF     EXPNS_DIF     PRESCRIPT
##   348.8294738    65.4944654    37.9300112    32.8398030    20.1650695
##      MORTCONF         DELAY        NOTGET       MH_SVCS           AGE
##    16.5382052    10.9543926    10.2676638     5.7093418     5.5621735
##     EXPCTLOSS       PWEIGHT        INCOME            MS     FEWRTRIPS
##     5.2515041     4.5802320     4.5643375     4.4481843     4.3184217
##     TSPNDFOOD        EST_ST       WRKLOSS         EEDUC     TSPNDPRPD
##     3.9052249     3.4835010     3.3443896     3.2305722     2.9752851
##      RSNNOWRK  THHLD_NUMPER        LIVQTR        TENURE THHLD_NUMADLT
##     2.5148260     2.3667431     2.0238710     1.9510468     1.7494414
##       TNUM_PS      TW_START     TSTDY_HRS     RECVDVACC      KINDWORK
##     1.5339537     1.5061901     1.4294712     1.3510246     1.3233061
##      PRIVHLTH        REGION           EIP         RRACE       GETVACC
##     1.3119441     1.2910233     1.2043236     1.1403977     1.1338532
##     CHILDFOOD       EGENDER       SNAP_YN     SSADECISN     FEWRTRANS
##     1.0789256     1.0591113     1.0582954     0.9799709     0.9696332
##       PUBHLTH       TCH_HRS       ANYWORK         EVICT     PLNDTRIPS
##     0.9459822     0.9397934     0.8576751     0.7508668     0.7038386
##  THHLD_NUMKID     SSALIKELY      HADCOVID      FORCLOSE       RENTCUR
##     0.5963613     0.5580786     0.5579462     0.5566599     0.5388894
##      SSA_RECV      UI_APPLY       SCHLHRS     COMPAVAIL      FREEFOOD
##     0.5282791     0.4736292     0.4686940     0.4477134     0.4472653
##   INTRNTAVAIL       UI_RECV     SSA_APPLY       MORTCUR         DOSES
##     0.3808380     0.3580441     0.3376386     0.2456618     0.1361483
##       PSCHNG7
##     0.0000000
```

Fit random forest model and print sorted variable importance for Numeric Risk

```
rf_risk_numeric <- ranger(risk.numeric ~ ., data = train_pulse, importance =
"impurity", num.trees = 500)

predictions <- predict(rf_risk_numeric, data = test_pulse)$predictions
confusion_matrix <- table(predictions, test_pulse$risk.numeric)
```

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.658263011688244"
```

```
risk_numeric_importance <- rf_risk_numeric$variable.importance

sorted_importance_risk_numeric <- sort(risk_numeric_importance, decreasing =
TRUE)

print("Risk Numeric Variable Importance:")
```

```
## [1] "Risk Numeric Variable Importance:"
```

```
print(sorted_importance_risk_numeric)
```

```
##   risk.binary      risk.GAD    total.risk           GAD      risk.PHQ
##  4294.9219034  3583.5452898  3169.1104575  3044.8634923  2698.6062608
##           PHQ     EXPNS_DIF     MH_NOTGET    CURFOODSUF     PRESCRIPT
##  2241.8582505   184.2644151   180.7004133   147.3592507    94.2088767
##      MORTCONF        NOTGET         DELAY       MH_SVCS           AGE
##    74.0054348    63.3265017    41.0312991    29.0699104    22.5970223
##       WRKLOSS     EXPCTLOSS     FEWRTRIPS        INCOME        TENURE
##    19.8518889    19.7823568    15.1217363    11.1661764     8.6772313
##       PWEIGHT      RSNNOWRK     TSPNDFOOD     TSPNDPRPD        EST_ST
##     7.6455825     6.1530016     5.7524824     5.2678318     5.2567162
##            MS     RECVDVACC         EEDUC       TNUM_PS       SNAP_YN
##     5.1987159     5.1622809     4.0915124     3.6405581     3.5447268
##        LIVQTR  THHLD_NUMPER THHLD_NUMADLT     TSTDY_HRS         RRACE
##     3.1028169     2.8409602     2.3651899     2.2389169     2.1831707
##      PRIVHLTH        REGION           EIP      TW_START       RENTCUR
##     2.0151813     1.9153172     1.8505108     1.8007550     1.7754417
##      KINDWORK      UI_APPLY     SSADECISN       PUBHLTH     FEWRTRANS
##     1.7443733     1.6590640     1.6126696     1.5795665     1.5488967
##       GETVACC      SSA_RECV       TCH_HRS         EVICT       EGENDER
##     1.4849820     1.4102114     1.3876293     1.3324928     1.3258403
##       ANYWORK      FORCLOSE      HADCOVID     PLNDTRIPS     SSALIKELY
##     0.9697954     0.9674517     0.9368846     0.8996118     0.8858997
##  THHLD_NUMKID       SCHLHRS      FREEFOOD   INTRNTAVAIL     COMPAVAIL
##     0.8173831     0.7858253     0.7813941     0.6781962     0.6319673
##     CHILDFOOD       UI_RECV     SSA_APPLY       MORTCUR         DOSES
##     0.5427475     0.5402337     0.5363644     0.4928850     0.2390267
##       PSCHNG7
##     0.0000000
```

Fit random forest model and print sorted variable importance for Binary Risk

```
rf_risk_binary <- ranger(risk.binary ~ ., data = train_pulse, importance = "i
mpurity", num.trees = 500)

predictions <- predict(rf_risk_binary, data = test_pulse)$predictions
```

```
confusion_matrix <- table(predictions, test_pulse$risk.binary)
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.653807851564547"
```

```
risk_binary_importance <- rf_risk_binary$variable.importance

sorted_importance_risk_binary <- sort(risk_binary_importance, decreasing = TR
UE)

print("Risk Numeric Variable Importance:")
```

```
## [1] "Risk Numeric Variable Importance:"
```

```
print(sorted_importance_risk_binary)
```

```
##  risk.numeric    total.risk           GAD      risk.GAD      risk.PHQ 
##  2.114630e+03  1.410829e+03  1.336618e+03  1.148202e+03  7.161498e+02 
##           PHQ     MH_NOTGET     EXPNS_DIF    CURFOODSUF     PRESCRIPT 
##  6.704147e+02  9.108191e+01  6.469717e+01  4.644995e+01  3.488347e+01 
##        NOTGET         DELAY      MORTCONF       MH_SVCS       WRKLOSS 
##  2.138530e+01  1.806191e+01  1.481211e+01  1.479510e+01  8.573206e+00 
##     FEWRTRIPS           AGE     EXPCTLOSS        TENURE        INCOME 
##  6.934917e+00  5.599457e+00  5.385584e+00  3.954236e+00  2.634829e+00 
##      RSNNOWRK            MS       PWEIGHT     TSPNDFOOD        EST_ST 
##  2.441537e+00  1.804071e+00  1.729952e+00  1.271801e+00  1.183029e+00 
##     TSPNDPRPD     RECVDVACC     SSADECISN         EVICT       TNUM_PS 
##  9.909614e-01  9.814512e-01  9.676088e-01  8.931783e-01  8.282140e-01 
##       RENTCUR         EEDUC       SNAP_YN        LIVQTR      UI_APPLY 
##  8.070659e-01  7.765975e-01  7.381996e-01  7.318456e-01  7.248703e-01 
##  THHLD_NUMPER     CHILDFOOD      FORCLOSE     TSTDY_HRS      SSA_RECV 
##  7.108520e-01  7.086362e-01  6.344502e-01  6.224735e-01  5.670717e-01 
##        REGION       MORTCUR THHLD_NUMADLT         RRACE     FEWRTRANS 
##  5.146315e-01  5.140761e-01  4.898394e-01  4.812572e-01  4.437110e-01 
##       PUBHLTH      KINDWORK           EIP      TW_START       TCH_HRS 
##  4.435426e-01  4.116665e-01  3.992818e-01  3.568575e-01  3.397080e-01 
##       GETVACC       EGENDER   INTRNTAVAIL      HADCOVID       ANYWORK 
##  3.251057e-01  3.040498e-01  2.909367e-01  2.668458e-01  2.560281e-01 
##      FREEFOOD      PRIVHLTH  THHLD_NUMKID       SCHLHRS     SSALIKELY 
##  2.464765e-01  2.343098e-01  2.314895e-01  2.213041e-01  2.045253e-01 
##     COMPAVAIL     PLNDTRIPS       UI_RECV     SSA_APPLY         DOSES 
##  1.907709e-01  1.700170e-01  1.483183e-01  1.303432e-01  6.157726e-02 
##       PSCHNG7 
##  0.000000e+00
```

Create a function to extract top ten variables and their importance scores and store in a new dataframe

```
extract_top_vars <- function(importance_df, excluded_vars = character(), N =
10) {
```

```r
  filtered_importance_df <- importance_df[!names(importance_df) %in% excluded
_vars]
  ordered_vars <- order(filtered_importance_df, decreasing = TRUE)
  top_vars <- names(filtered_importance_df)[ordered_vars[1:N]]
  top_scores <- as.numeric(filtered_importance_df[ordered_vars[1:N]])

  top_vars_df <- data.frame(
    Variable = top_vars,
    Importance = top_scores,
    stringsAsFactors = FALSE
  )

  return(top_vars_df)
}
```

Extract the top ten variables and importance scores from each of the seven response variable Random Forests and format a table for the variables and scores of each response:

```r
gad_importance <- rf_gad$variable.importance

top_10_gad_vars <- extract_top_vars(gad_importance, c("PHQ", "total.risk", "r
isk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 10)

kable(top_10_gad_vars, align = "c", col.names = c("Variable", "Importance"),
row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("GAD Variable Importance" = 2))
```

GAD Variable Importance

Variable

Importance

EXPNS_DIF

2608.6620

MH_NOTGET

2340.1194

CURFOODSUF

1518.1737

PRESCRIPT

1463.7337

DELAY

1118.3163

AGE

960.3352

MORTCONF

842.0806

PWEIGHT

735.4980

NOTGET

676.8978

EST_ST

583.8813

```r
top_20_gad_vars <- extract_top_vars(gad_importance, c("PHQ", "total.risk", "r
isk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 20)

kable(top_20_gad_vars, align = "c", col.names = c("Variable", "Importance"),
row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("GAD Variable Importance" = 2))
```

GAD Variable Importance

Variable

Importance

EXPNS_DIF

2608.6620

MH_NOTGET

2340.1194

CURFOODSUF

1518.1737

PRESCRIPT

1463.7337

DELAY

1118.3163

AGE

960.3352

MORTCONF

842.0806

PWEIGHT

735.4980

NOTGET

676.8978

EST_ST

583.8813

FEWRTRIPS

560.2793

TSPNDFOOD

557.4432

MH_SVCS

552.3736

TSPNDPRPD

486.8078

INCOME

397.3168

EXPCTLOSS

386.2566

WRKLOSS

356.5778

EEDUC

308.9384

TENURE

302.6636

THHLD_NUMPER

286.8440

Extract the top ten variables and importance scores from the GAD Random Forest and format table

```
phq_importance <- rf_phq$variable.importance

top_10_phq_vars <- extract_top_vars(phq_importance, c("GAD", "total.risk", "r
isk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 10)

kable(top_10_phq_vars, align = "c", col.names = c("Variable", "Importance"),
row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("PHQ Variable Importance" = 2))
```

PHQ Variable Importance

Variable

Importance

MH_NOTGET

2053.9438

EXPNS_DIF

1902.2145

CURFOODSUF

1660.1724

PRESCRIPT

1081.2148

AGE

712.1173

PWEIGHT

694.1821

MORTCONF

658.4718

NOTGET

569.3888

EST_ST

560.4899

TSPNDFOOD

539.6101

```r
top_20_phq_vars <- extract_top_vars(phq_importance, c("GAD", "total.risk", "r
isk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 20)

kable(top_20_phq_vars, align = "c", col.names = c("Variable", "Importance"),
row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("PHQ Variable Importance" = 2))
```

PHQ Variable Importance

Variable

Importance

MH_NOTGET

2053.9438

EXPNS_DIF

1902.2145

CURFOODSUF

1660.1724

PRESCRIPT

1081.2148

AGE

712.1173

PWEIGHT

694.1821

MORTCONF

658.4718

NOTGET

569.3888

EST_ST

560.4899

TSPNDFOOD

539.6101

FEWRTRIPS

463.6596

TSPNDPRPD

461.1343

INCOME

457.2483

DELAY

422.4433

MS

406.0931

MH_SVCS

388.3032

EEDUC

331.8786

THHLD_NUMPER

268.4773

WRKLOSS

258.9882

RSNNOWRK

240.2228

```r
total_risk_importance <- rf_total_risk$variable.importance

top_10_total_risk_vars <- extract_top_vars(total_risk_importance, c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 10)

kable(top_10_total_risk_vars, align = "c", col.names = c("Variable", "Importance"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("Total Risk Variable Importance" = 2))
```

Total Risk Variable Importance

Variable

Importance

MH_NOTGET

8078.102

EXPNS_DIF

7354.359

CURFOODSUF

6516.606

PRESCRIPT

4733.259

NOTGET

2959.270

MORTCONF

2581.298

DELAY

2567.125

MH_SVCS

2124.611

FEWRTRIPS

1864.556

AGE

1862.271

```r
top_20_total_risk_vars <- extract_top_vars(total_risk_importance, c("GAD", "P
HQ", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 20)

kable(top_20_total_risk_vars, align = "c", col.names = c("Variable", "Importa
nce"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("Total Risk Variable Importance" = 2))
```

Total Risk Variable Importance

Variable

Importance

MH_NOTGET

8078.1019

EXPNS_DIF

7354.3590

CURFOODSUF

6516.6058

PRESCRIPT

4733.2591

NOTGET

2959.2704

MORTCONF

2581.2982

DELAY

2567.1250

MH_SVCS

2124.6113

FEWRTRIPS

1864.5559

AGE

1862.2714

PWEIGHT

1138.0323

WRKLOSS

1017.5139

EST_ST

904.7636

TSPNDFOOD

863.6891

TSPNDPRPD

763.2983

EXPCTLOSS

759.1579

INCOME

757.7920

TENURE

565.7824

MS

526.3893

EEDUC

508.0757

```r
risk_gad_importance <- rf_risk_gad$variable.importance

top_10_risk_gad_vars <- extract_top_vars(risk_gad_importance, c("GAD", "PHQ", "total.risk", "risk.PHQ", "risk.numeric", "risk.binary"), 10)

kable(top_10_risk_gad_vars, align = "c", col.names = c("Variable", "Importance"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("GAD Risk Variable Importance" = 2))
```

GAD Risk Variable Importance

Variable	Importance
MH_NOTGET	67.294279
EXPNS_DIF	58.205401
CURFOODSUF	39.656079
MORTCONF	24.590756
PRESCRIPT	20.039187
NOTGET	16.006858
DELAY	12.633391
EXPCTLOSS	11.215025
MH_SVCS	10.316802
AGE	8.025119

```r
top_20_risk_gad_vars <- extract_top_vars(risk_gad_importance, c("GAD", "PHQ",
"total.risk", "risk.PHQ", "risk.numeric", "risk.binary"), 20)

kable(top_20_risk_gad_vars, align = "c", col.names = c("Variable", "Importanc
e"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("GAD Risk Variable Importance" = 2))
```

GAD Risk Variable Importance

Variable

Importance

MH_NOTGET

67.294279

EXPNS_DIF

58.205401

CURFOODSUF

39.656079

MORTCONF

24.590756

PRESCRIPT

20.039187

NOTGET

16.006858

DELAY

12.633391

EXPCTLOSS

11.215025

MH_SVCS

10.316802

AGE

8.025119

WRKLOSS

7.956333

PWEIGHT

4.180049

TSPNDFOOD

3.041588

EST_ST

3.031825

TSPNDPRPD

2.723450

FEWRTRIPS

2.516583

INCOME

2.474748

RSNNOWRK

2.353945

MS

2.245232

TENURE

2.234360

```r
risk_phq_importance <- rf_risk_phq$variable.importance

top_10_risk_phq_vars <- extract_top_vars(risk_phq_importance, c("GAD", "PHQ", "total.risk", "risk.GAD", "risk.numeric", "risk.binary"), 10)

kable(top_10_risk_phq_vars, align = "c", col.names = c("Variable", "Importance"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("PHQ Risk Variable Importance" = 2))
```

PHQ Risk Variable Importance

Variable

Importance

MH_NOTGET

65.494465

CURFOODSUF

37.930011

EXPNS_DIF

32.839803

PRESCRIPT

20.165070

MORTCONF

16.538205

DELAY

10.954393

NOTGET

10.267664

MH_SVCS

5.709342

AGE

5.562174

EXPCTLOSS

5.251504

```r
top_20_risk_phq_vars <- extract_top_vars(risk_phq_importance, c("GAD", "PHQ",
"total.risk", "risk.GAD", "risk.numeric", "risk.binary"), 20)

kable(top_20_risk_phq_vars, align = "c", col.names = c("Variable", "Importanc
e"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("PHQ Risk Variable Importance" = 2))
```

PHQ Risk Variable Importance

Variable

Importance

MH_NOTGET

65.494465

CURFOODSUF

37.930011

EXPNS_DIF

32.839803

PRESCRIPT

20.165070

MORTCONF

16.538205

DELAY

10.954393

NOTGET

10.267664

MH_SVCS

5.709342

AGE

5.562174

EXPCTLOSS

5.251504

PWEIGHT

4.580232

INCOME

4.564337

MS

4.448184

FEWRTRIPS

4.318422

TSPNDFOOD

3.905225

EST_ST

3.483501

WRKLOSS

3.344390

EEDUC

3.230572

TSPNDPRPD

2.975285

RSNNOWRK

2.514826

```r
risk_numeric_importance <- rf_risk_numeric$variable.importance

top_10_risk_numeric_vars <- extract_top_vars(risk_numeric_importance, c("GAD"
, "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.binary"), 10)

kable(top_10_risk_numeric_vars, align = "c", col.names = c("Variable", "Impor
tance"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("Numeric Risk Variable Importance" = 2))
```

Numeric Risk Variable Importance

Variable

Importance

EXPNS_DIF

184.26442

MH_NOTGET

180.70041

CURFOODSUF

147.35925

PRESCRIPT

94.20888

MORTCONF

74.00543

NOTGET

63.32650

DELAY

41.03130

MH_SVCS

29.06991

AGE

22.59702

WRKLOSS

19.85189

```r
top_20_risk_numeric_vars <- extract_top_vars(risk_numeric_importance, c("GAD"
, "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.binary"), 20)

kable(top_20_risk_numeric_vars, align = "c", col.names = c("Variable", "Impor
tance"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("Numeric Risk Variable Importance" = 2))
```

Numeric Risk Variable Importance

Variable

Importance

EXPNS_DIF

184.264415

MH_NOTGET

180.700413

CURFOODSUF

147.359251

PRESCRIPT

94.208877

MORTCONF

74.005435

NOTGET

63.326502

DELAY

41.031299

MH_SVCS

29.069910

AGE

22.597022

WRKLOSS

19.851889

EXPCTLOSS

19.782357

FEWRTRIPS

15.121736

INCOME

11.166176

TENURE

8.677231

PWEIGHT

7.645582

RSNNOWRK

6.153002

TSPNDFOOD

5.752482

TSPNDPRPD

5.267832

EST_ST

5.256716

MS

5.198716

```r
risk_binary_importance <- rf_risk_binary$variable.importance

top_10_risk_binary_vars <- extract_top_vars(risk_binary_importance, c("GAD",
"PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric"), 10)

kable(top_10_risk_binary_vars, align = "c", col.names = c("Variable", "Import
ance"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("Binary Risk Variable Importance" = 2))
```

Binary Risk Variable Importance

Variable

Importance

MH_NOTGET

91.081908

EXPNS_DIF

64.697169

CURFOODSUF

46.449952

PRESCRIPT

34.883469

NOTGET

21.385296

DELAY

18.061906

MORTCONF

14.812114

MH_SVCS

14.795102

WRKLOSS

8.573206

FEWRTRIPS

6.934917

```r
top_20_risk_binary_vars <- extract_top_vars(risk_binary_importance, c("GAD", "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric"), 20)

kable(top_20_risk_binary_vars, align = "c", col.names = c("Variable", "Importance"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(c("Binary Risk Variable Importance" = 2))
```

Binary Risk Variable Importance

Variable

Importance

MH_NOTGET

91.0819077

EXPNS_DIF

64.6971694

CURFOODSUF

46.4499516

PRESCRIPT

34.8834690

NOTGET

21.3852957

DELAY

18.0619063

MORTCONF

14.8121136

MH_SVCS

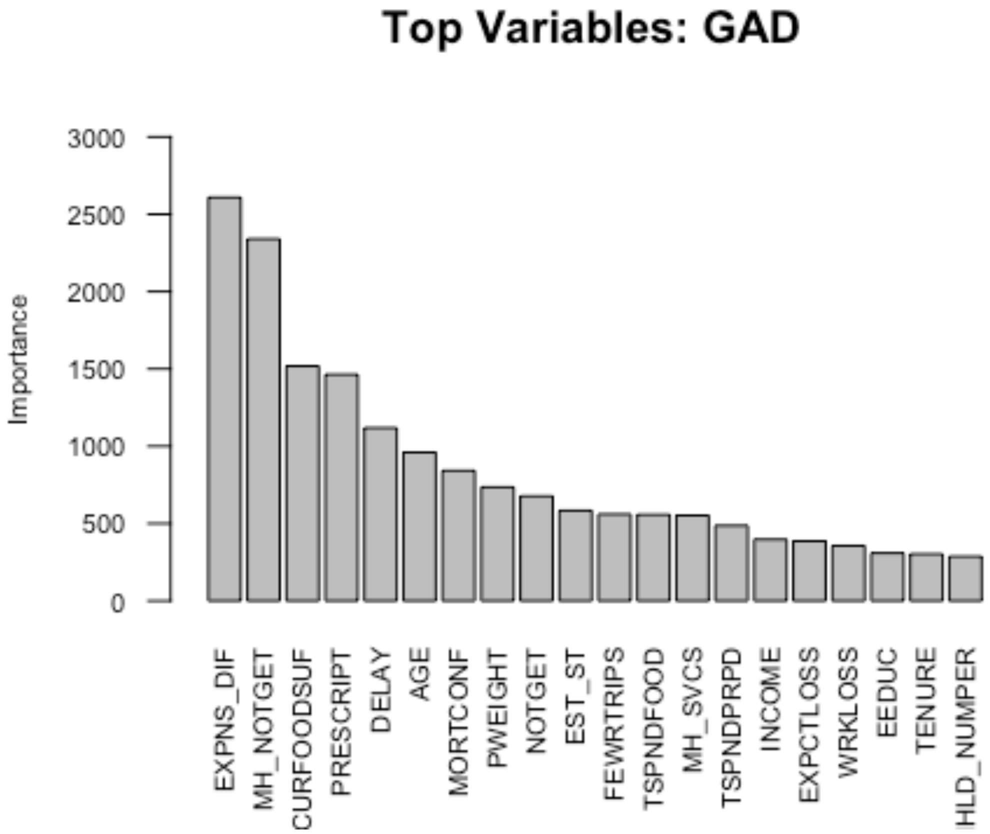
14.7951022

```
WRKLOSS

8.5732062

FEWRTRIPS

6.9349173

AGE

5.5994566

EXPCTLOSS

5.3855840

TENURE

3.9542356

INCOME

2.6348286

RSNNOWRK

2.4415375

MS

1.8040707

PWEIGHT

1.7299525

TSPNDFOOD

1.2718011

EST_ST

1.1830286

TSPNDPRPD

0.9909614
```

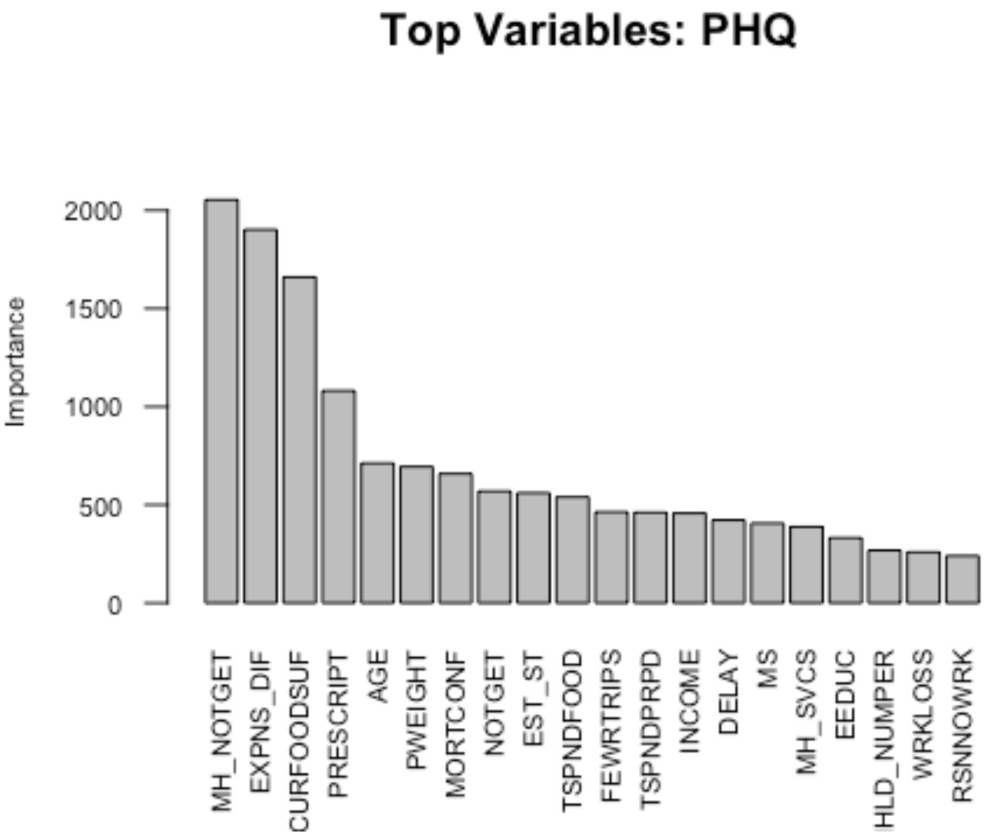
Plot the top 20 variable importance scores for each response variable

```
plot_top_vars <- function(importance_df, response_name, excluded_vars = chara
cter(), N = 20) {
  filtered_importance_df <- importance_df[!names(importance_df) %in% excluded
_vars]
```

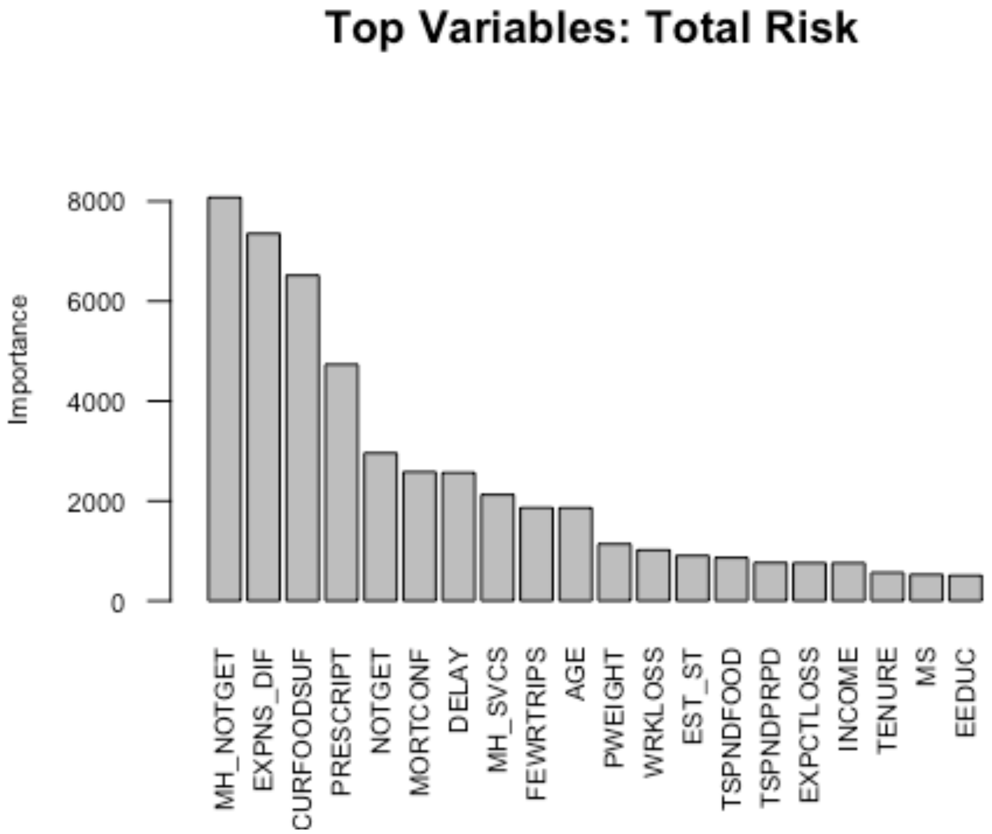
```
  ordered_vars <- order(filtered_importance_df, decreasing = TRUE)
  top_vars <- names(filtered_importance_df)[ordered_vars[1:N]]

  par(cex.axis = 0.7, cex.lab = 0.7)

  barplot(
    filtered_importance_df[ordered_vars[1:N]],
    main = paste("Top Variables:", response_name),
    ylab = "Importance",
    col = "gray",
    las = 2,
    xlim = c(0, N * 1.2),
    ylim = c(0, max(filtered_importance_df[ordered_vars[1:N]]) * 1.2),
    names.arg = top_vars,
    cex.names = 0.7
  )
}

plot_top_vars(
  gad_importance,
  "GAD",
  c("PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary
"),
  20
)
```



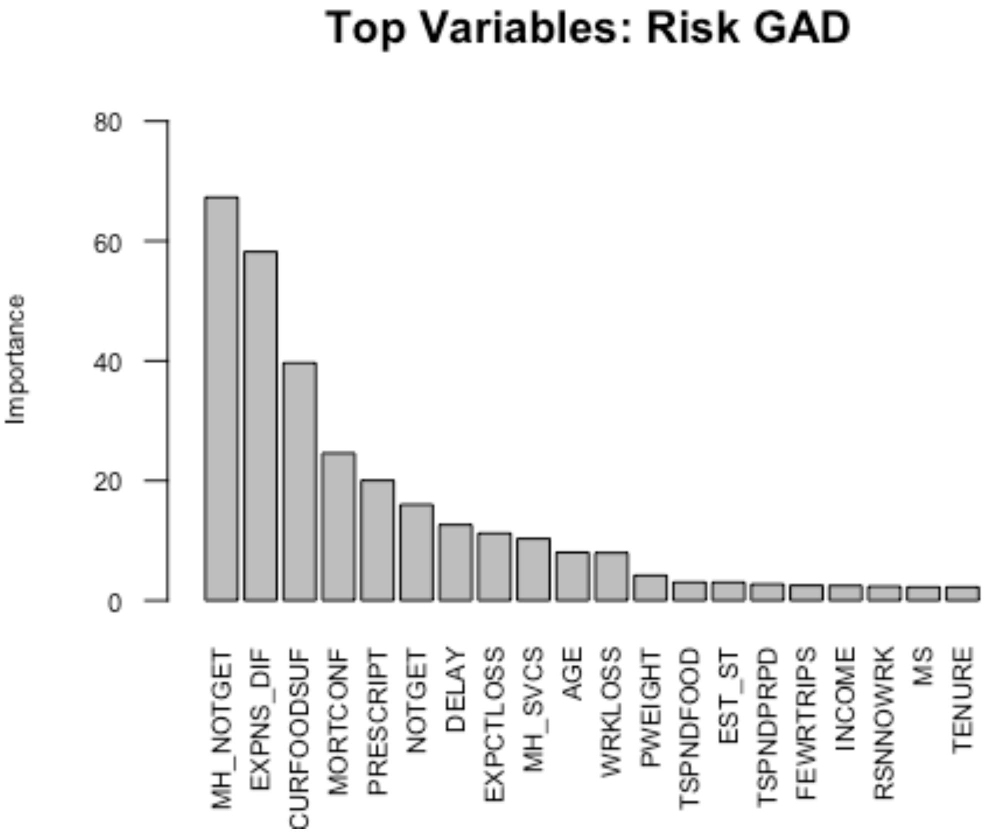
```r
plot_top_vars(
  phq_importance,
  "PHQ",
  c("GAD", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary
"),
  20
)
```



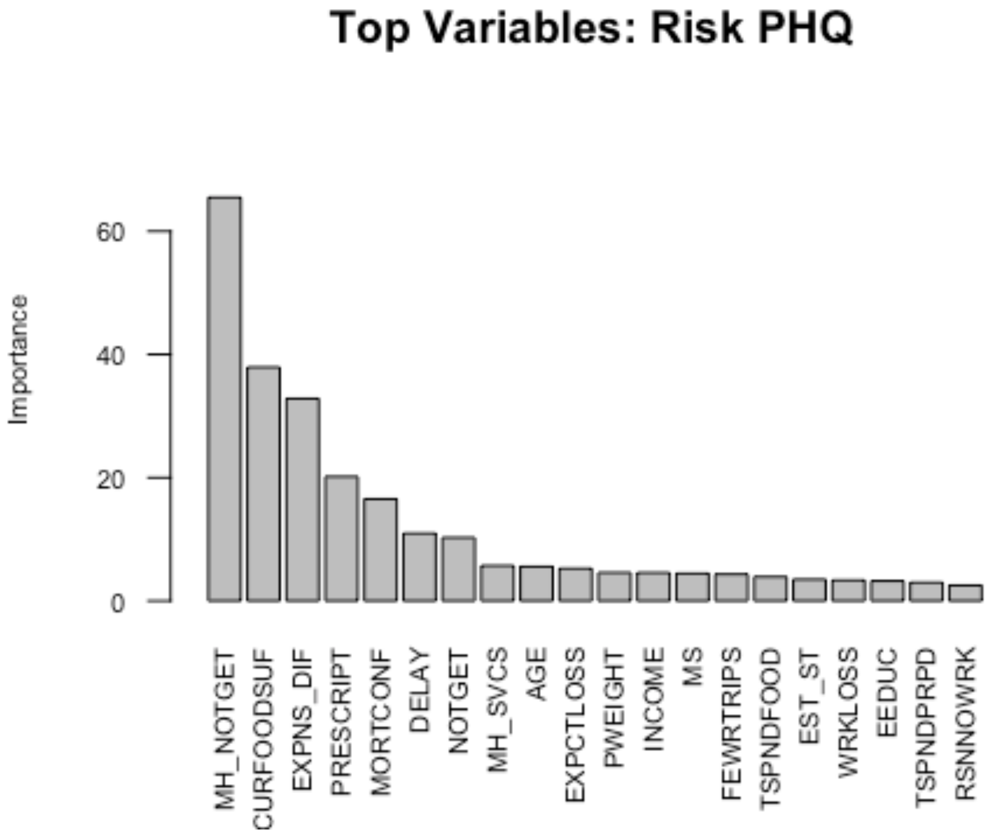
```r
plot_top_vars(
  total_risk_importance,
  "Total Risk",
  c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"),
  20
)
```



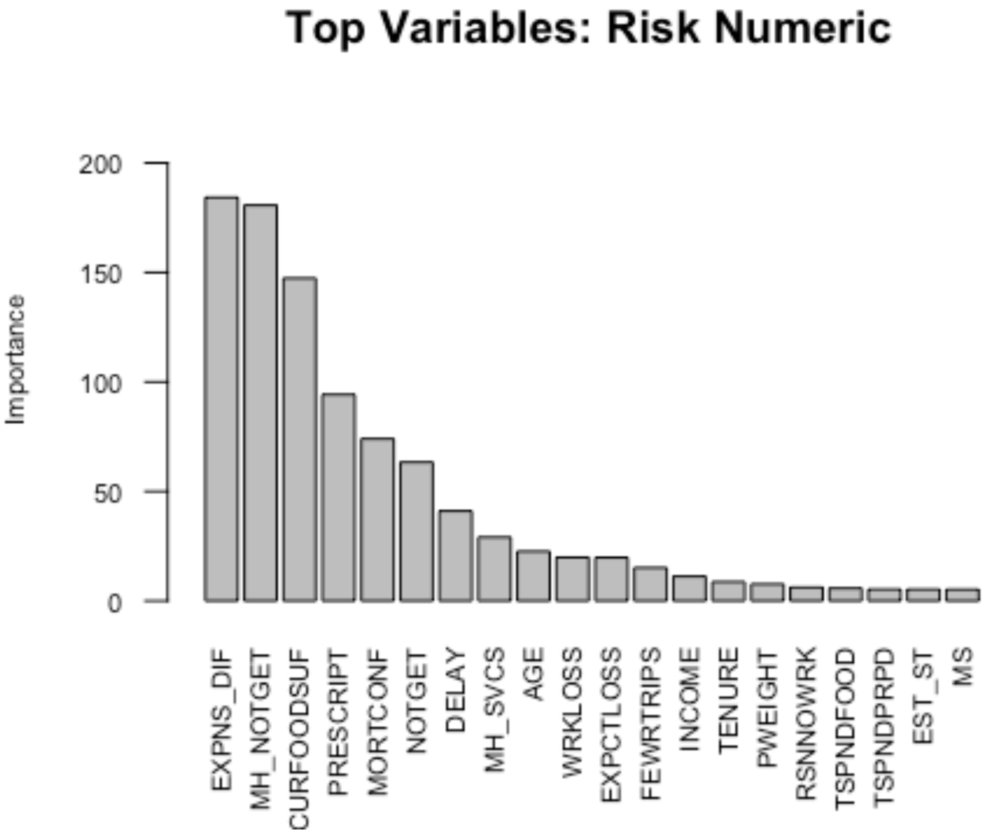
```
plot_top_vars(
  risk_gad_importance,
  "Risk GAD",
  c("GAD", "PHQ", "total.risk", "risk.PHQ", "risk.numeric", "risk.binary"),
  20
)
```



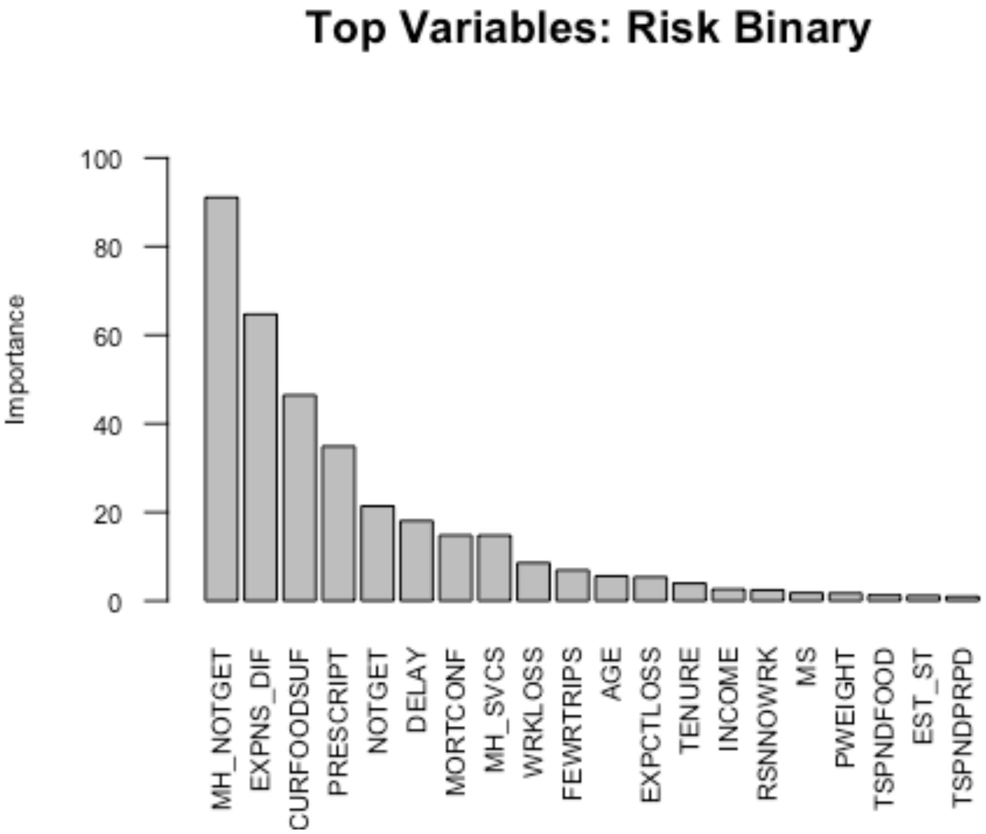
```r
plot_top_vars(
  risk_phq_importance,
  "Risk PHQ",
  c("GAD", "PHQ", "total.risk", "risk.GAD", "risk.numeric", "risk.binary"),
  20
)
```



```
plot_top_vars(
  risk_numeric_importance,
  "Risk Numeric",
  c("GAD", "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.binary"),
  20
)
```



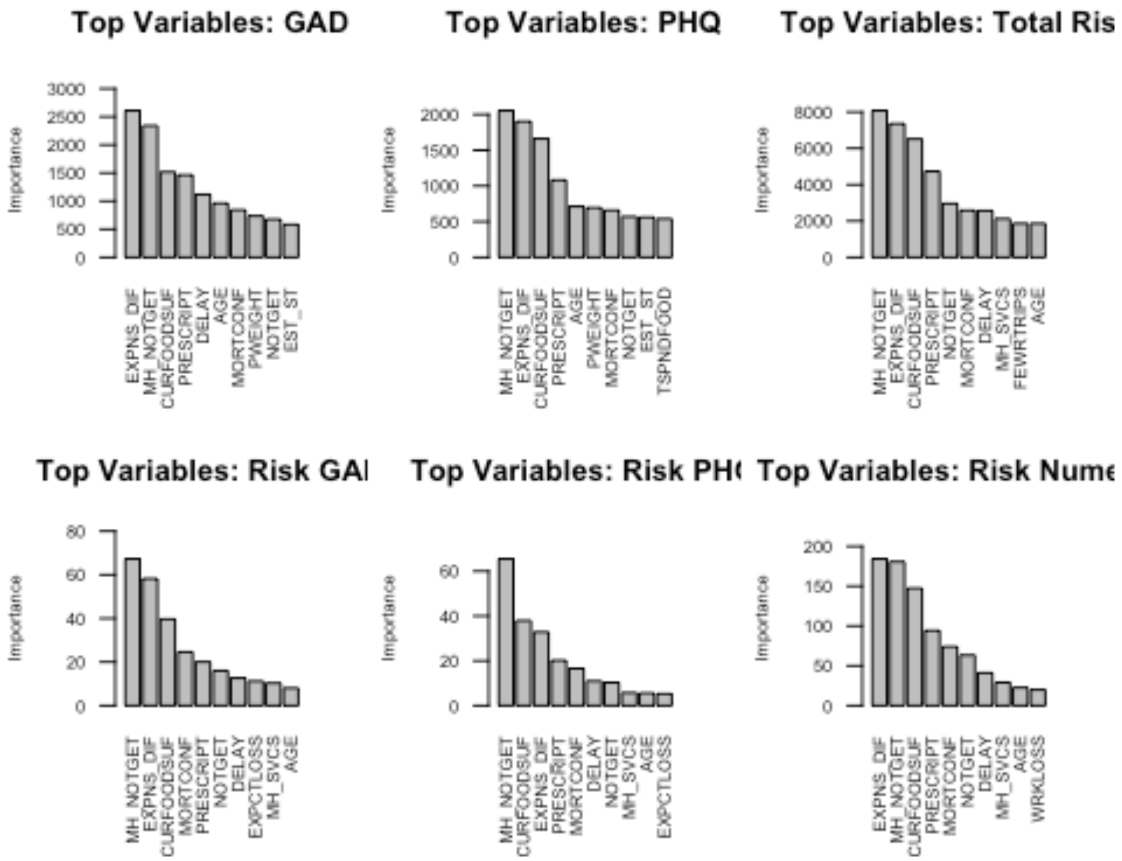
```
plot_top_vars(
  risk_binary_importance,
  "Risk Binary",
  c("GAD", "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric"),
  20
)
```



Plot the top ten variable importance scores for each response variable

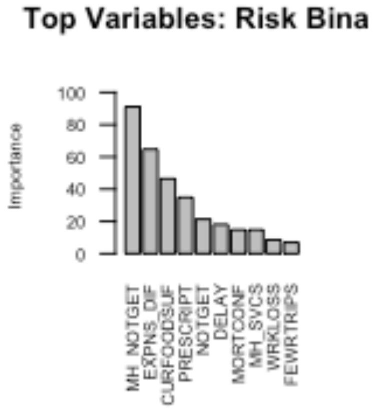
```
plot_top_vars <- function(importance_df, response_name, excluded_vars = character(), N = 10) {
  filtered_importance_df <- importance_df[!names(importance_df) %in% excluded_vars]
  ordered_vars <- order(filtered_importance_df, decreasing = TRUE)
  top_vars <- names(filtered_importance_df)[ordered_vars[1:N]]

  par(cex.axis = 0.7, cex.lab = 0.7)

  barplot(
    filtered_importance_df[ordered_vars[1:N]],
    main = paste("Top Variables:", response_name),
    ylab = "Importance",
    col = "gray",
    las = 2,
    xlim = c(0, N * 1.2),
    ylim = c(0, max(filtered_importance_df[ordered_vars[1:N]]) * 1.2),
    names.arg = top_vars,
    cex.names = 0.7
  )
}
```

```r
par(mfrow = c(2,3))

plot_top_vars(
  gad_importance,
  "GAD",
  c("PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary
"),
  10
)

plot_top_vars(
  phq_importance,
  "PHQ",
  c("GAD", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary
"),
  10
)

plot_top_vars(
  total_risk_importance,
  "Total Risk",
  c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"),
  10
)

plot_top_vars(
  risk_gad_importance,
  "Risk GAD",
  c("GAD", "PHQ", "total.risk", "risk.PHQ", "risk.numeric", "risk.binary"),
  10
)

plot_top_vars(
  risk_phq_importance,
  "Risk PHQ",
  c("GAD", "PHQ", "total.risk", "risk.GAD", "risk.numeric", "risk.binary"),
  10
)

plot_top_vars(
  risk_numeric_importance,
  "Risk Numeric",
  c("GAD", "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.binary"),
  10
)
```



```
plot_top_vars <- function(importance_df, response_name, excluded_vars = chara
cter(), N = 10) {
  filtered_importance_df <- importance_df[!names(importance_df) %in% excluded
_vars]
  ordered_vars <- order(filtered_importance_df, decreasing = TRUE)
  top_vars <- names(filtered_importance_df)[ordered_vars[1:N]]

  par(cex.axis = 0.7, cex.lab = 0.7)

  barplot(
    filtered_importance_df[ordered_vars[1:N]],
    main = paste("Top Variables:", response_name),
    ylab = "Importance",
    col = "gray",
    las = 2,
    xlim = c(0, N * 1.2),
    ylim = c(0, max(filtered_importance_df[ordered_vars[1:N]]) * 1.2),
    names.arg = top_vars,
    cex.names = 0.7
  )
}

par(mfrow = c(2,3))
```

```
plot_top_vars(
  risk_binary_importance,
  "Risk Binary",
  c("GAD", "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric"),
  10
)
```



Create a function to extract the ten variables with the highest importance scores from each response variable. Combine the variables into a single vector and count the occurrences of each variable. Sort in descending order and format into table. The table displays the number of responses out of seven that indicated each variable in its top ten.

```
extract_top_vars <- function(importance_df, excluded_vars = character(), N = 10) {
  importance_df <- as.data.frame(importance_df)
  filtered_importance_df <- importance_df[!row.names(importance_df) %in% excluded_vars, , drop = FALSE]
  filtered_importance_df <- filtered_importance_df[order(filtered_importance_df[, 1], decreasing = TRUE), , drop = FALSE]
  top_vars_df <- head(filtered_importance_df, N)
  top_vars_df$Variable <- row.names(top_vars_df)
  colnames(top_vars_df) <- c("Importance", "Variable")
```

```r
  top_vars_df <- top_vars_df[, c("Variable", "Importance")]

  return(top_vars_df)
}

top_10_gad_vars <- extract_top_vars(gad_importance, c("PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 10)
top_10_phq_vars <- extract_top_vars(phq_importance, c("GAD", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 10)
top_10_total_risk_vars <- extract_top_vars(total_risk_importance, c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 10)
top_10_risk_gad_vars <- extract_top_vars(risk_gad_importance, c("GAD", "PHQ", "total.risk", "risk.PHQ", "risk.numeric", "risk.binary"), 10)
top_10_risk_phq_vars <- extract_top_vars(risk_phq_importance, c("GAD", "PHQ", "total.risk", "risk.GAD", "risk.numeric", "risk.binary"), 10)
top_10_risk_numeric_vars <- extract_top_vars(risk_numeric_importance, c("GAD", "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.binary"), 10)
top_10_risk_binary_vars <- extract_top_vars(risk_binary_importance, c("GAD", "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric"), 10)

all_top_vars10 <- c(
  top_10_gad_vars$Variable,
  top_10_phq_vars$Variable,
  top_10_total_risk_vars$Variable,
  top_10_risk_gad_vars$Variable,
  top_10_risk_phq_vars$Variable,
  top_10_risk_numeric_vars$Variable,
  top_10_risk_binary_vars$Variable
)

variable_count10 <- table(all_top_vars10)
variable_count_df10 <- as.data.frame(variable_count10)
colnames(variable_count_df10) <- c("Variable", "Tally")

variable_count_df10 <- variable_count_df10[order(variable_count_df10$Tally, decreasing = TRUE), ]

kable(variable_count_df10[, c("Variable", "Tally")], align = "c", row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(header = c(" " = 2))
```

Variable

Tally

CURFOODSUF

7

EXPNS_DIF

7

MH_NOTGET

7

MORTCONF

7

NOTGET

7

PRESCRIPT

7

AGE

6

DELAY

6

MH_SVCS

5

EST_ST

2

EXPCTLOSS

2

FEWRTRIPS

2

PWEIGHT

2

WRKLOSS

2

TSPNDFOOD

1

Calculate total importance across all seven lists and divide total importance by count to generate weighted importance

```
extract_top_vars <- function(importance_df, excluded_vars = character(), N =
10) {
  importance_df <- as.data.frame(importance_df)
  filtered_importance_df <- importance_df[!row.names(importance_df) %in% excl
uded_vars, , drop = FALSE]
  filtered_importance_df <- filtered_importance_df[order(filtered_importance_
df[, 1], decreasing = TRUE), , drop = FALSE]
  top_vars_df <- head(filtered_importance_df, N)
  top_vars_df$Variable <- row.names(top_vars_df)
  colnames(top_vars_df) <- c("Importance", "Variable")
  top_vars_df <- top_vars_df[, c("Variable", "Importance")]

  return(top_vars_df)
}

top_10_gad_vars <- extract_top_vars(gad_importance, c("PHQ", "total.risk", "r
isk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 10)
top_10_phq_vars <- extract_top_vars(phq_importance, c("GAD", "total.risk", "r
isk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 10)
top_10_total_risk_vars <- extract_top_vars(total_risk_importance, c("GAD", "P
HQ", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 10)
top_10_risk_gad_vars <- extract_top_vars(risk_gad_importance, c("GAD", "PHQ",
"total.risk", "risk.PHQ", "risk.numeric", "risk.binary"), 10)
top_10_risk_phq_vars <- extract_top_vars(risk_phq_importance, c("GAD", "PHQ",
"total.risk", "risk.GAD", "risk.numeric", "risk.binary"), 10)
top_10_risk_numeric_vars <- extract_top_vars(risk_numeric_importance, c("GAD"
, "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.binary"), 10)
top_10_risk_binary_vars <- extract_top_vars(risk_binary_importance, c("GAD",
"PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric"), 10)

all_top_vars10 <- rbind(
  top_10_gad_vars,
  top_10_phq_vars,
  top_10_total_risk_vars,
  top_10_risk_gad_vars,
  top_10_risk_phq_vars,
  top_10_risk_numeric_vars,
  top_10_risk_binary_vars
)

variable_importance_sum10 <- aggregate(Importance ~ Variable, data = all_top_
vars10, sum)
variable_count10 <- table(all_top_vars10$Variable)
variable_count_df10 <- as.data.frame(variable_count10)
colnames(variable_count_df10) <- c("Variable", "Tally")

variable_importance_df10 <- merge(variable_importance_sum10, variable_count_d
```

```r
f10, by = "Variable")

variable_importance_df10$Weighted_Importance10 <- variable_importance_df10$Im
portance / variable_importance_df10$Tally

variable_importance_df10 <- variable_importance_df10[order(variable_importanc
e_df10$Weighted_Importance10, decreasing = TRUE), ]

kable(variable_importance_df10[, c("Variable", "Weighted_Importance10")], ali
gn = "c", col.names = c("Variable", "Weighted Importance"), row.names = FALSE
) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(header = c("Top 10 Weighted Variable Importance Summary" =
2))
```

Top 10 Weighted Variable Importance Summary

Variable

Weighted Importance

MH_NOTGET

1839.533749

EXPNS_DIF

1743.606044

CURFOODSUF

1423.763893

PRESCRIPT

1063.929170

FEWRTRIPS

935.745432

PWEIGHT

714.840050

DELAY

628.020384

NOTGET

616.649060

MORTCONF

601.685305

AGE

595.151368

EST_ST

572.185611

TSPNDFOOD

539.610125

MH_SVCS

436.900500

WRKLOSS

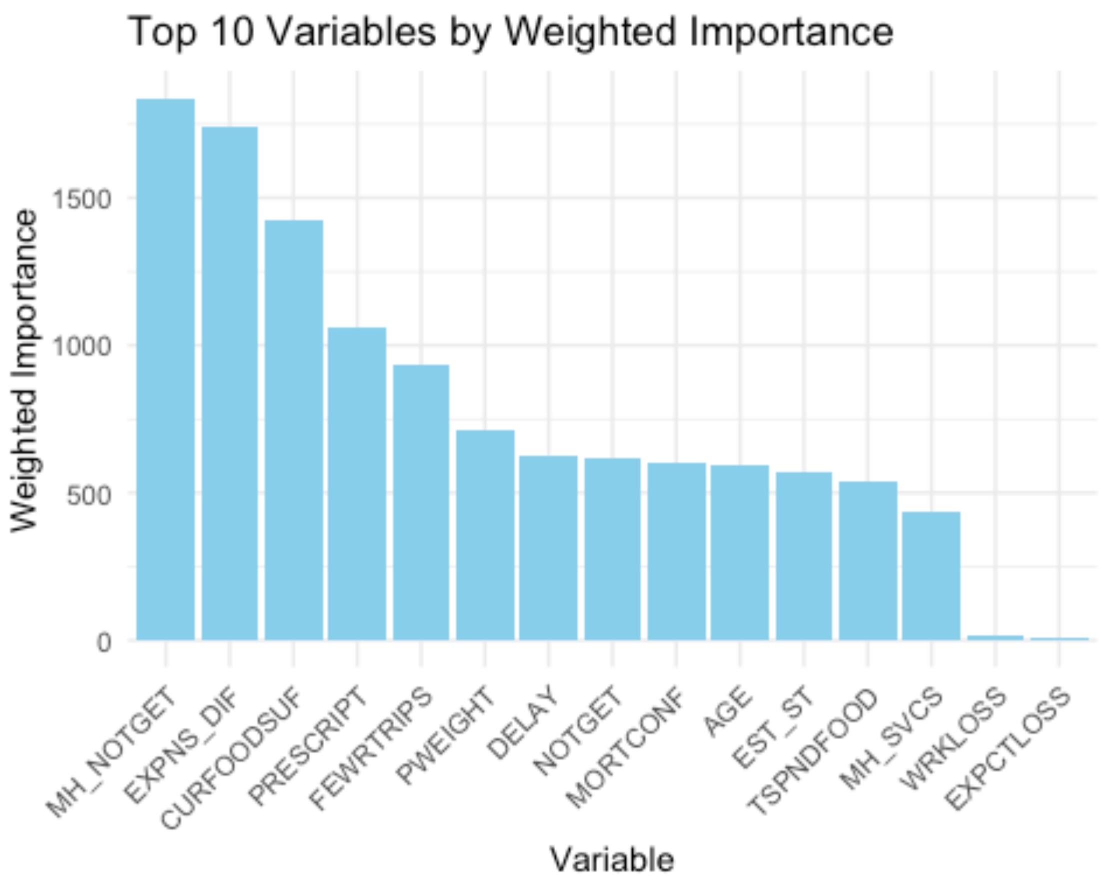
14.212548

EXPCTLOSS

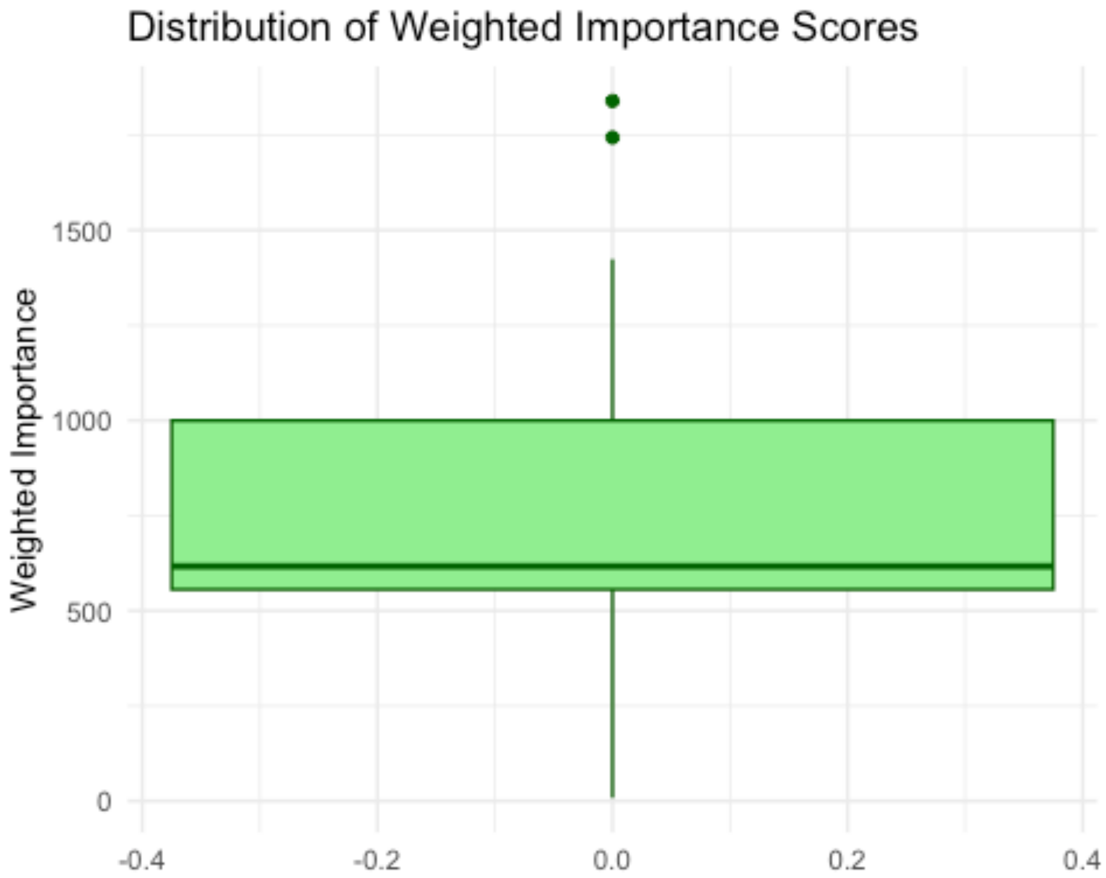
8.233264

Plot the top 10 variable importance bar chart and boxplot of score distribution

```r
ggplot(variable_importance_df10[1:15, ], aes(x = reorder(Variable, -Weighted_Importance10), y = Weighted_Importance10)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Top 10 Variables by Weighted Importance", x = "Variable", y = "Weighted Importance") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



```r
ggplot(variable_importance_df10, aes(y = Weighted_Importance10)) +
  geom_boxplot(fill = "lightgreen", color = "darkgreen") +
  labs(title = "Distribution of Weighted Importance Scores", y = "Weighted Im
portance") +
  theme_minimal()
```



Create a function to extract the 20 variables with the highest importance scores from each response variable. Combine the variables into a single vector and count the occurrences of each variable. Sort in descending order and format into table. The table displays the number of responses out of seven that indicated each variable in its top 20.

```
extract_top_vars <- function(importance_df, excluded_vars = character(), N =
20) {
  importance_df <- as.data.frame(importance_df)
  filtered_importance_df <- importance_df[!row.names(importance_df) %in% excl
uded_vars, , drop = FALSE]
  filtered_importance_df <- filtered_importance_df[order(filtered_importance_
df[, 1], decreasing = TRUE), , drop = FALSE]
  top_vars_df <- head(filtered_importance_df, N)
  top_vars_df$Variable <- row.names(top_vars_df)
  colnames(top_vars_df) <- c("Importance", "Variable")
  top_vars_df <- top_vars_df[, c("Variable", "Importance")]

  return(top_vars_df)
}

top_20_gad_vars <- extract_top_vars(gad_importance, c("PHQ", "total.risk", "r
isk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 20)
top_20_phq_vars <- extract_top_vars(phq_importance, c("GAD", "total.risk", "r
isk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 20)
top_20_total_risk_vars <- extract_top_vars(total_risk_importance, c("GAD", "P
HQ", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 20)
top_20_risk_gad_vars <- extract_top_vars(risk_gad_importance, c("GAD", "PHQ",
"total.risk", "risk.PHQ", "risk.numeric", "risk.binary"), 20)
top_20_risk_phq_vars <- extract_top_vars(risk_phq_importance, c("GAD", "PHQ",
"total.risk", "risk.GAD", "risk.numeric", "risk.binary"), 20)
top_20_risk_numeric_vars <- extract_top_vars(risk_numeric_importance, c("GAD"
, "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.binary"), 20)
top_20_risk_binary_vars <- extract_top_vars(risk_binary_importance, c("GAD",
"PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric"), 20)

all_top_vars20 <- c(
  top_20_gad_vars$Variable,
  top_20_phq_vars$Variable,
  top_20_total_risk_vars$Variable,
  top_20_risk_gad_vars$Variable,
  top_20_risk_phq_vars$Variable,
  top_20_risk_numeric_vars$Variable,
  top_20_risk_binary_vars$Variable
)

variable_count20 <- table(all_top_vars20)
variable_count_df20 <- as.data.frame(variable_count20)
colnames(variable_count_df20) <- c("Variable", "Tally")

variable_count_df20 <- variable_count_df20[order(variable_count_df20$Tally, d
ecreasing = TRUE), ]

variable_count_df20$Rank <- seq_len(nrow(variable_count_df20))

kable(variable_count_df20[, c("Rank", "Variable", "Tally")], align = "c", col
.names = c("#", "Variable", "Tally"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(header = c("Instances of Variables in Top Predictors Lists
(out of 7)" = 3))
```

Instances of Variables in Top Predictors Lists (out of 7)

#

Variable

Tally

1

AGE

7

2

CURFOODSUF

7

3

DELAY

7

4

EST_ST

7

5

EXPNS_DIF

7

6

FEWRTRIPS

7

7

INCOME

7

8

MH_NOTGET

7

9

MH_SVCS

7

10

MORTCONF

7

11

NOTGET

7

12

PRESCRIPT

7

13

PWEIGHT

7

14

TSPNDFOOD

7

15

TSPNDPRPD

7

16

WRKLOSS

7

17

EXPCTLOSS

6

18

MS

6

19

RSNNOWRK

5

20

TENURE

5

21

EEDUC

4

22

THHLD_NUMPER

2

Extract top 20 variables of importance for the seven response variables, combine all top variables and importances in a dataframe, merge and summarize importance and count data, calculate and sort weighted importance, and print a formatted summary table of weighted importance

```
extract_top_vars <- function(importance_df, excluded_vars = character(), N =
20) {
  importance_df <- as.data.frame(importance_df)
  filtered_importance_df <- importance_df[!row.names(importance_df) %in% excl
uded_vars, , drop = FALSE]
  filtered_importance_df <- filtered_importance_df[order(filtered_importance_
df[, 1], decreasing = TRUE), , drop = FALSE]
  top_vars_df <- head(filtered_importance_df, N)
  top_vars_df$Variable <- row.names(top_vars_df)
  colnames(top_vars_df) <- c("Importance", "Variable")
  top_vars_df <- top_vars_df[, c("Variable", "Importance")]

  return(top_vars_df)
}

top_20_gad_vars <- extract_top_vars(gad_importance, c("PHQ", "total.risk", "r
isk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 20)
top_20_phq_vars <- extract_top_vars(phq_importance, c("GAD", "total.risk", "r
isk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 20)
top_20_total_risk_vars <- extract_top_vars(total_risk_importance, c("GAD", "P
HQ", "risk.GAD", "risk.PHQ", "risk.numeric", "risk.binary"), 20)
top_20_risk_gad_vars <- extract_top_vars(risk_gad_importance, c("GAD", "PHQ",
"total.risk", "risk.PHQ", "risk.numeric", "risk.binary"), 20)
top_20_risk_phq_vars <- extract_top_vars(risk_phq_importance, c("GAD", "PHQ",
"total.risk", "risk.GAD", "risk.numeric", "risk.binary"), 20)
top_20_risk_numeric_vars <- extract_top_vars(risk_numeric_importance, c("GAD"
, "PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.binary"), 20)
top_20_risk_binary_vars <- extract_top_vars(risk_binary_importance, c("GAD",
```

```r
"PHQ", "total.risk", "risk.GAD", "risk.PHQ", "risk.numeric"), 20)

all_top_vars20 <- rbind(
  top_20_gad_vars,
  top_20_phq_vars,
  top_20_total_risk_vars,
  top_20_risk_gad_vars,
  top_20_risk_phq_vars,
  top_20_risk_numeric_vars,
  top_20_risk_binary_vars
)

variable_importance_sum20 <- aggregate(Importance ~ Variable, data = all_top_vars20, sum)
variable_count20 <- table(all_top_vars20$Variable)
variable_count_df20 <- as.data.frame(variable_count20)
colnames(variable_count_df20) <- c("Variable", "Tally")

variable_importance_df20 <- merge(variable_importance_sum20, variable_count_df20, by = "Variable")

variable_importance_df20$Weighted_Importance20 <- variable_importance_df20$Importance / variable_importance_df20$Tally

variable_importance_df20 <- variable_importance_df20[order(variable_importance_df20$Weighted_Importance20, decreasing = TRUE), ]

variable_importance_df20$Rank <- seq_len(nrow(variable_importance_df20))

kable(variable_importance_df20[, c("Rank", "Variable", "Weighted_Importance20")], align = "c", col.names = c("#", "Variable", "Weighted Importance"), row.names = FALSE) %>%
  kable_styling(bootstrap_options = "striped", full_width = TRUE) %>%
  add_header_above(header = c("Weighted Importance Summary of Variables in Top Predictors Lists" = 3))
```

Weighted Importance Summary of Variables in Top Predictors Lists

#

Variable

Weighted Importance

1

MH_NOTGET

1839.53375

2

EXPNS_DIF

1743.60604

3

CURFOODSUF

1423.76389

4

PRESCRIPT

1063.92917

5

NOTGET

616.64906

6

MORTCONF

601.68530

7

DELAY

598.65224

8

AGE

510.92967

9

MH_SVCS

446.45418

10

FEWRTRIPS

416.76949

11

PWEIGHT

369.40689

12

EST_ST

294.58428

13

EEDUC

288.03081

14

TSPNDFOOD

282.10192

15

THHLD_NUMPER

277.66067

16

TSPNDPRPD

246.17113

17

WRKLOSS

238.97224

18

INCOME

233.31390

19

EXPCTLOSS

197.84150

20

TENURE

176.66236

21

MS

157.69643

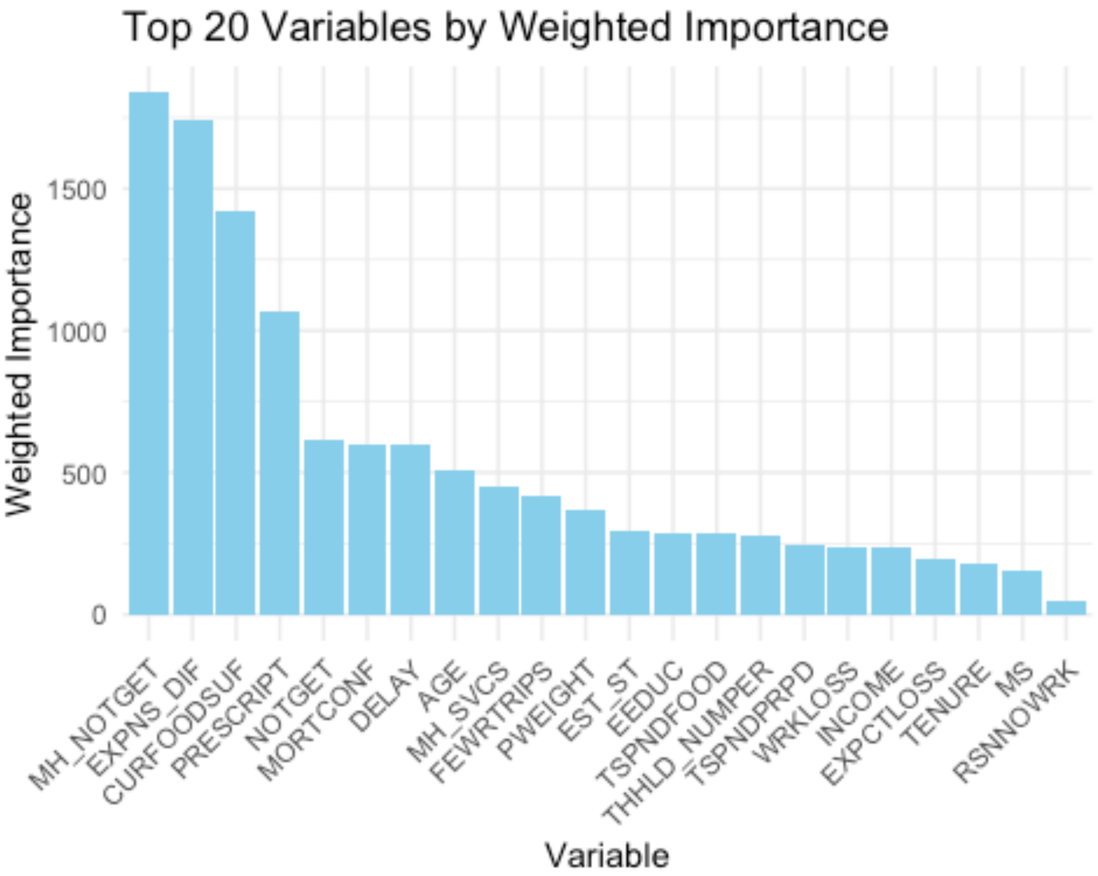
22

RSNNOWRK

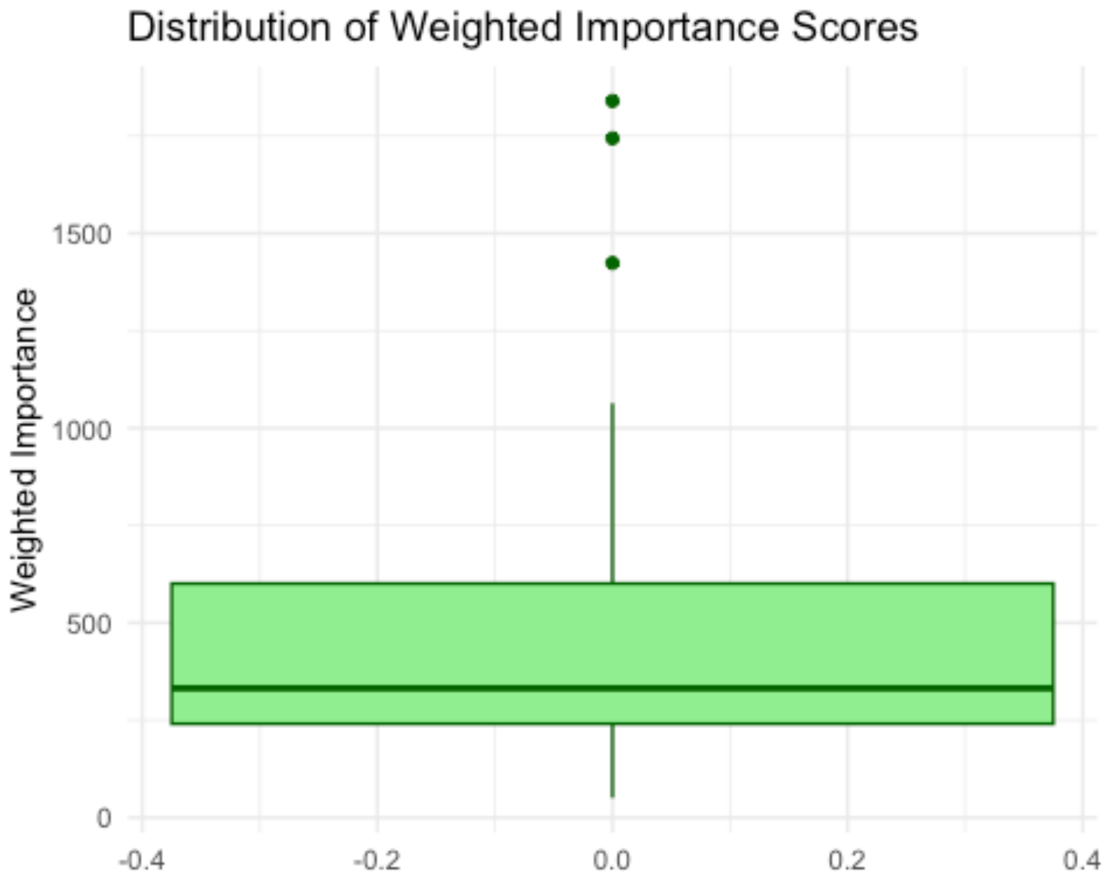
50.73722

Plot the top 20 variable importance bar chart and boxplot of score distribution

```r
ggplot(variable_importance_df20[1:22, ], aes(x = reorder(Variable, -Weighted_Importance20), y = Weighted_Importance20)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Top 20 Variables by Weighted Importance", x = "Variable", y = "Weighted Importance") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

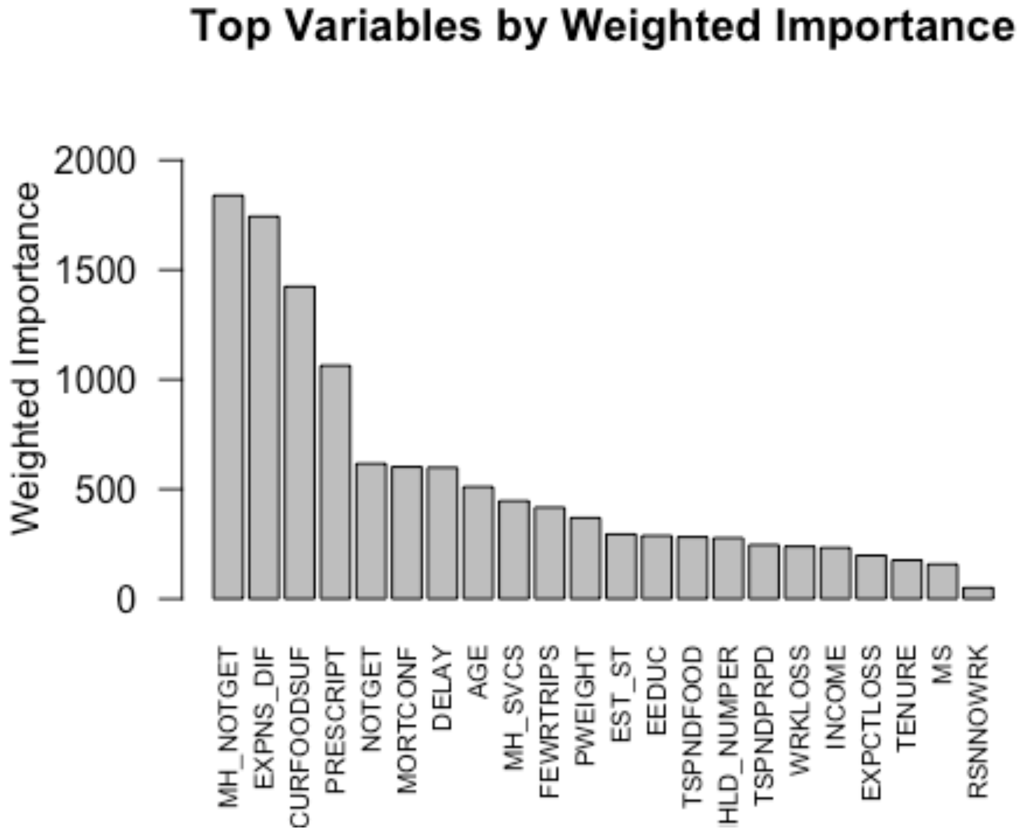


```
ggplot(variable_importance_df20, aes(y = Weighted_Importance20)) +
  geom_boxplot(fill = "lightgreen", color = "darkgreen") +
  labs(title = "Distribution of Weighted Importance Scores", y = "Weighted Im
portance") +
  theme_minimal()
```



Plot the formatted top 20 variable importance bar chart

```
N <- nrow(variable_importance_df20)

top_vars <- variable_importance_df20$Variable
top_importance <- variable_importance_df20$Weighted_Importance20

barplot(
  height = top_importance,
  main = "Top Variables by Weighted Importance",
  ylab = "Weighted Importance",
  col = "gray",
  ylim = c(0, max(top_importance) * 1.2),
  names.arg = top_vars,
  cex.names = 0.7,
  las = 2
)
```



8.3.2 DECISION TREES

Create a subset excluding mental health variables

```
train_pulse_trees <- train_pulse %>% select(c(-("MH_NOTGET"), -("MH_SVCS"), -
("PRESCRIPT")))

test_pulse_trees <- test_pulse %>% select(c(-("MH_NOTGET"), -("MH_SVCS"), -("
PRESCRIPT")))
```

Set predictor variables

```
predictor_vars <- train_pulse_trees %>% select(c("GAD", "PHQ", "risk.GAD", "r
isk.PHQ", "total.risk", "risk.numeric", "risk.binary"))
```

Model decision tree for numeric risk including mental health variables

```
tree_risk_numeric1 <- rpart(risk.numeric ~ .,
                          data = train_pulse[, !(names(train_pulse) %in% c("GAD"
, "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.binary"))],
                          method = "class")

print("Summary of decision tree model for risk.numeric")
```

```
## [1] "Summary of decision tree model for risk.numeric"
```

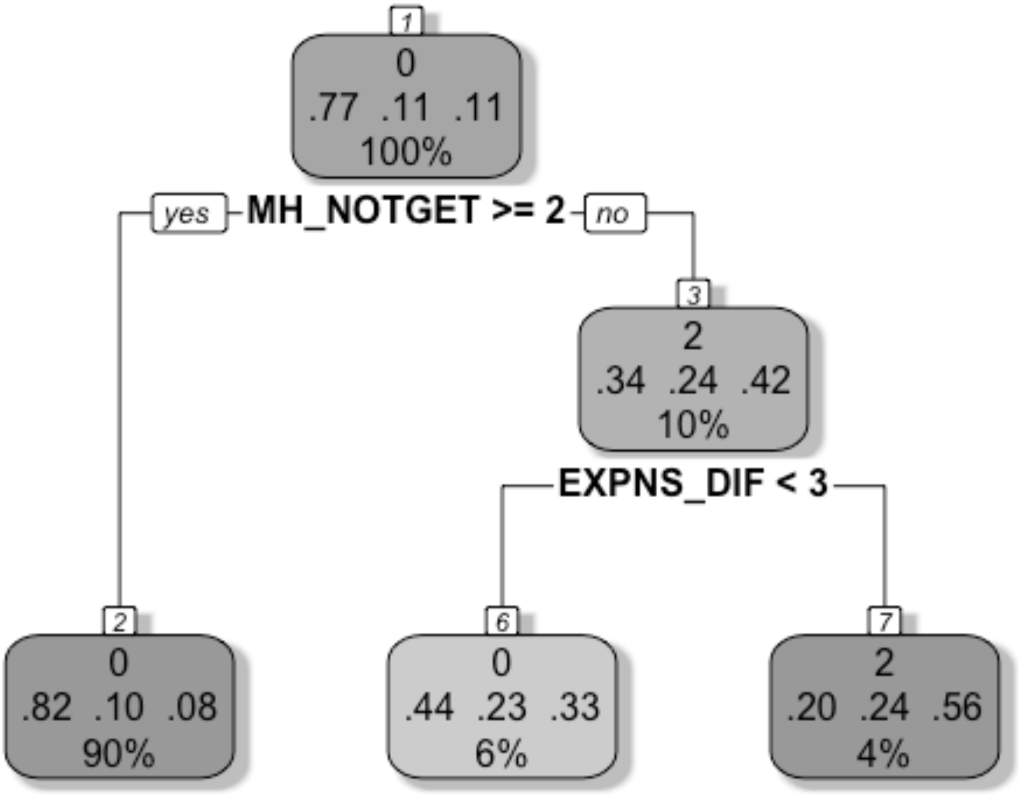
```r
summary(tree_risk_numeric1)
```

```
## Call:
## rpart(formula = risk.numeric ~ ., data = train_pulse[, !(names(train_pulse
) %in%
##     c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.binary"))]
,
##     method = "class")
##   n= 44517
## 
##           CP nsplit rel error    xerror        xstd
## 1 0.03592814      0 1.0000000 1.0000000 0.008794150
## 2 0.02704591      1 0.9640719 0.9640719 0.008679664
## 3 0.01000000      2 0.9370259 0.9370259 0.008590248
## 
## Variable importance
##  MH_NOTGET  EXPNS_DIF CURFOODSUF   MORTCONF  EXPCTLOSS     INCOME   PRIVHL
TH 
##         85          7          2          2          1          1         
1 
## 
## Node number 1: 44517 observations,    complexity param=0.03592814
##   predicted class=0  expected loss=0.2250826  P(node) =1
##     class counts: 34497  4975  5045
##    probabilities: 0.775 0.112 0.113 
##   left son=2 (40230 obs) right son=3 (4287 obs)
##   Primary splits:
##       MH_NOTGET  < 1.5 to the right, improve=1427.1900, (0 missing)
##       EXPNS_DIF  < 2.5 to the left,  improve=1139.2320, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=1021.2310, (0 missing)
##       PRESCRIPT  < 1.5 to the right, improve= 765.8515, (0 missing)
##       MORTCONF   < 3.5 to the right, improve= 718.2238, (0 missing)
## 
## Node number 2: 40230 observations
##   predicted class=0  expected loss=0.1787721  P(node) =0.9036997
##     class counts: 33038  3966  3226
##    probabilities: 0.821 0.099 0.080 
## 
## Node number 3: 4287 observations,    complexity param=0.02704591
##   predicted class=2  expected loss=0.575694  P(node) =0.09630029
##     class counts:  1459  1009  1819
##    probabilities: 0.340 0.235 0.424 
##   left son=6 (2551 obs) right son=7 (1736 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5 to the left,  improve=113.98500, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve= 90.07739, (0 missing)
##       NOTGET     < 1.5 to the right, improve= 55.44220, (0 missing)
##       MORTCONF   < 3.5 to the right, improve= 53.98824, (0 missing)
##       DELAY      < 1.5 to the right, improve= 41.76672, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 1.5 to the left,  agree=0.739, adj=0.354, (0 split)
##       MORTCONF   < 3.5 to the right, agree=0.735, adj=0.346, (0 split)
##       EXPCTLOSS  < 1.5 to the right, agree=0.680, adj=0.210, (0 split)
##       INCOME     < 3.5 to the right, agree=0.666, adj=0.174, (0 split)
##       PRIVHLTH   < 1.5 to the left,  agree=0.662, adj=0.165, (0 split)
## 
## Node number 6: 2551 observations
##   predicted class=0  expected loss=0.5637005  P(node) =0.05730395
##     class counts:  1113   596   842
##    probabilities: 0.436 0.234 0.330 
## 
## Node number 7: 1736 observations
##   predicted class=2  expected loss=0.437212  P(node) =0.03899634
##     class counts:   346   413   977
##    probabilities: 0.199 0.238 0.563
```

```
predictions_risk_numeric1 <- predict(tree_risk_numeric1, newdata = test_pulse
, type = "class")

accuracy_risk_numeric1 <- mean(predictions_risk_numeric1 == test_pulse$risk.n
umeric)
cat("Accuracy for risk.numeric:", accuracy_risk_numeric1, "\n")
```

```
## Accuracy for risk.numeric: 0.7959537
```

```
rpart.plot(tree_risk_numeric1, box.palette = "Greys", shadow.col = "gray", nn
= TRUE)
```



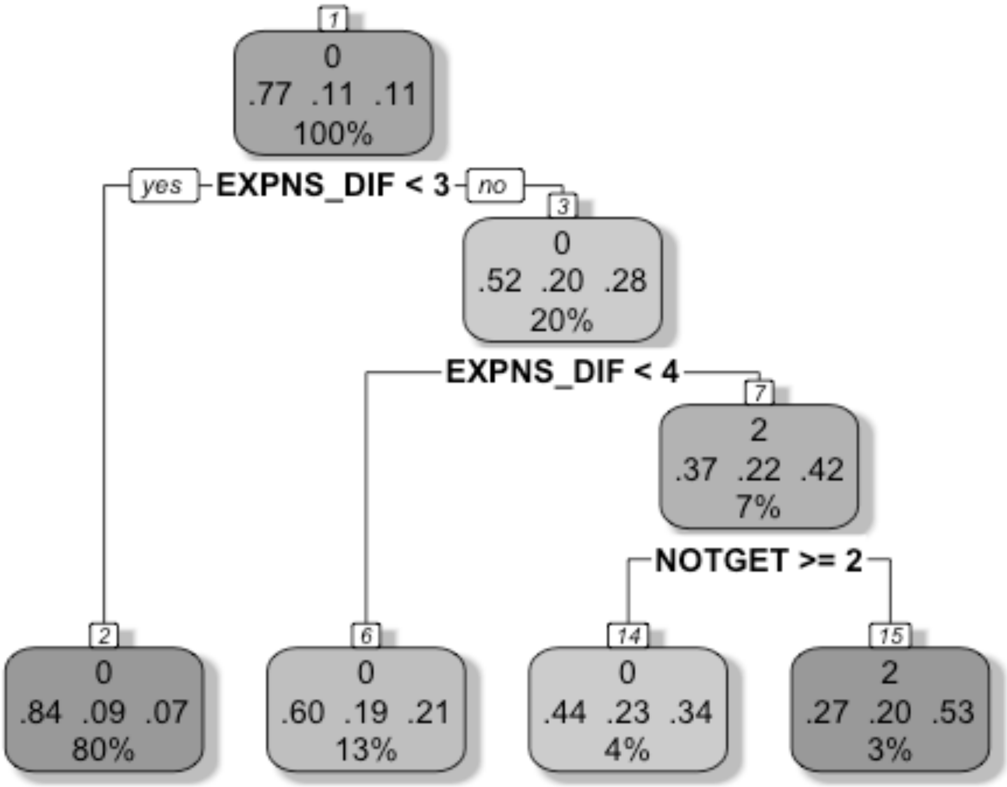
Model decision tree for numeric risk excluding mental health variables

```
tree_risk_numeric2 <- rpart(risk.numeric ~ .,
                       data = train_pulse_trees[, !(names(train_pulse_trees)
%in% c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.binary"))],
                       method = "class")

print("Summary of decision tree model for risk.numeric")

## [1] "Summary of decision tree model for risk.numeric"

summary(tree_risk_numeric2)

## Call:
## rpart(formula = risk.numeric ~ ., data = train_pulse_trees[,
##     !(names(train_pulse_trees) %in% c("GAD", "PHQ", "risk.GAD",
##         "risk.PHQ", "total.risk", "risk.binary"))], method = "class")
##   n= 44517
##
##            CP nsplit rel error    xerror        xstd
## 1 0.01107784      0 1.0000000 1.0000000 0.008794150
## 2 0.01000000      3 0.9667665 0.9721557 0.008705845
##
## Variable importance
##  EXPNS_DIF   MORTCONF CURFOODSUF  CHILDFOOD    RENTCUR  EXPCTLOSS     NOTG
ET 
##         63         11         10          4          4          3          
2 
##      DELAY      EVICT 
##          1          1 
## 
## Node number 1: 44517 observations,    complexity param=0.01107784
##   predicted class=0  expected loss=0.2250826  P(node) =1
##     class counts: 34497  4975  5045
##    probabilities: 0.775 0.112 0.113 
##   left son=2 (35472 obs) right son=3 (9045 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5 to the left,  improve=1139.2320, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=1021.2310, (0 missing)
##       MORTCONF   < 3.5 to the right, improve= 718.2238, (0 missing)
##       NOTGET     < 1.5 to the right, improve= 649.8609, (0 missing)
##       DELAY      < 1.5 to the right, improve= 627.3014, (0 missing)
##   Surrogate splits:
##       MORTCONF   < 3.5 to the right, agree=0.835, adj=0.188, (0 split)
##       CURFOODSUF < 2.5 to the left,  agree=0.829, adj=0.161, (0 split)
##       CHILDFOOD  < 2.5 to the right, agree=0.811, adj=0.068, (0 split)
##       RENTCUR    < 1.5 to the left,  agree=0.808, adj=0.054, (0 split)
##       EXPCTLOSS  < 1.5 to the right, agree=0.807, adj=0.049, (0 split)
## 
## Node number 2: 35472 observations
##   predicted class=0  expected loss=0.1603236  P(node) =0.7968192
##     class counts: 29785  3200  2487
##    probabilities: 0.840 0.090 0.070 
## 
## Node number 3: 9045 observations,    complexity param=0.01107784
##   predicted class=0  expected loss=0.4790492  P(node) =0.2031808
##     class counts:  4712  1775  2558
##    probabilities: 0.521 0.196 0.283 
##   left son=6 (5937 obs) right son=7 (3108 obs)
##   Primary splits:
##       EXPNS_DIF  < 3.5 to the left,  improve=196.85660, (0 missing)
##       NOTGET     < 1.5 to the right, improve=139.65000, (0 missing)
##       DELAY      < 1.5 to the right, improve=135.91430, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=116.53590, (0 missing)
##       MORTCONF   < 1.5 to the right, improve= 51.13054, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 2.5 to the left,  agree=0.720, adj=0.184, (0 split)
##       MORTCONF   < 1.5 to the right, agree=0.703, adj=0.137, (0 split)
##       RENTCUR    < 1.5 to the left,  agree=0.687, adj=0.088, (0 split)
##       EVICT      < 2.5 to the right, agree=0.681, adj=0.073, (0 split)
##       CHILDFOOD  < 2.5 to the right, agree=0.674, adj=0.052, (0 split)
## 
## Node number 6: 5937 observations
##   predicted class=0  expected loss=0.3990231  P(node) =0.1333648
##     class counts:  3568  1105  1264
##    probabilities: 0.601 0.186 0.213
##
## Node number 7: 3108 observations,    complexity param=0.01107784
##   predicted class=2  expected loss=0.5836551  P(node) =0.06981603
##     class counts:  1144   670  1294
##    probabilities: 0.368 0.216 0.416
##   left son=14 (1829 obs) right son=15 (1279 obs)
##   Primary splits:
##       NOTGET     < 1.5 to the right, improve=49.51079, (0 missing)
##       DELAY      < 1.5 to the right, improve=44.85452, (0 missing)
##       CURFOODSUF < 2.5 to the left,  improve=31.67681, (0 missing)
##       FEWRTRIPS  < 1.5 to the right, improve=18.33927, (0 missing)
##       RRACE      < 0.5 to the right, improve=13.04758, (0 missing)
##   Surrogate splits:
##       DELAY       < 1.5 to the right, agree=0.819, adj=0.561, (0 split)
##       TNUM_PS     < 1.5 to the left,  agree=0.613, adj=0.060, (0 split)
##       CURFOODSUF  < 3.5 to the left,  agree=0.604, adj=0.038, (0 split)
##       COMPAVAIL   < 2.5 to the left,  agree=0.604, adj=0.037, (0 split)
##       INTRNTAVAIL < 2.5 to the left,  agree=0.602, adj=0.033, (0 split)
##
## Node number 14: 1829 observations
##   predicted class=0  expected loss=0.5626025  P(node) =0.04108543
##     class counts:   800   412   617
##    probabilities: 0.437 0.225 0.337
##
## Node number 15: 1279 observations
##   predicted class=2  expected loss=0.4706802  P(node) =0.0287306
##     class counts:   344   258   677
##    probabilities: 0.269 0.202 0.529
```

```
predictions_risk_numeric2 <- predict(tree_risk_numeric2, newdata = test_pulse_trees, type = "class")

accuracy_risk_numeric2 <- mean(predictions_risk_numeric2 == test_pulse_trees$risk.numeric)
cat("Accuracy for risk.numeric:", accuracy_risk_numeric2, "\n")
```

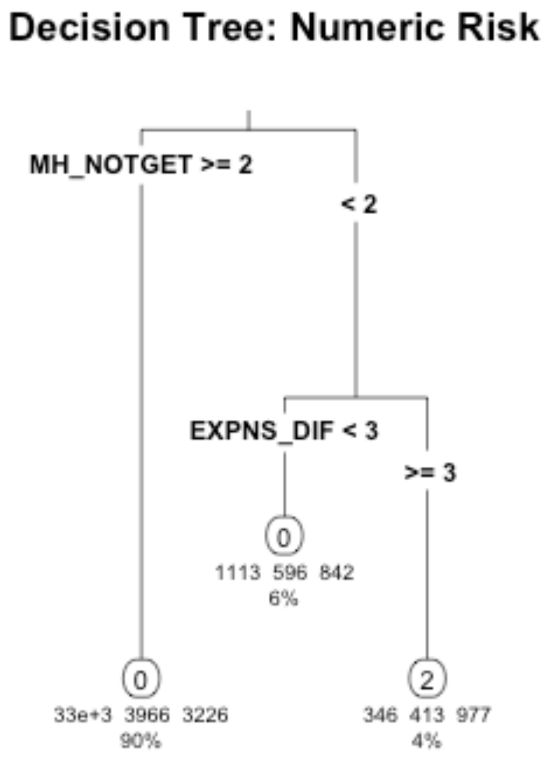
```
## Accuracy for risk.numeric: 0.7887206
```

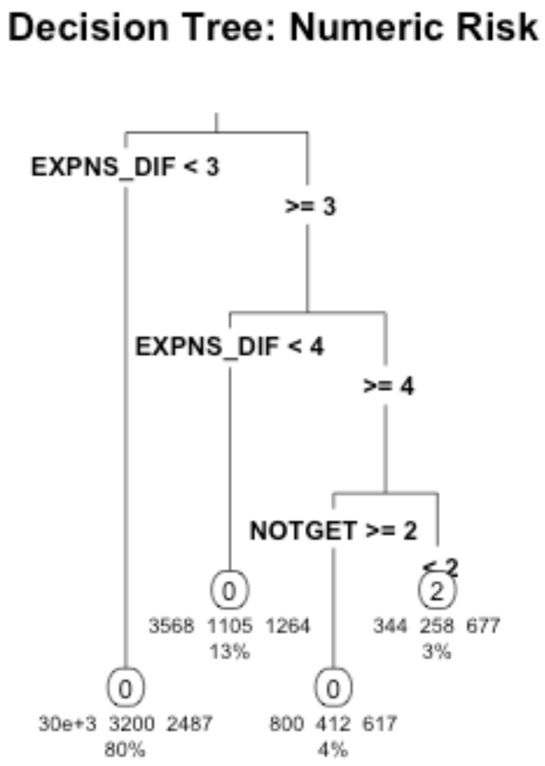
```
rpart.plot(tree_risk_numeric2, box.palette = "Greys", shadow.col = "gray", nn = TRUE)
```



Plot the two decision trees for numeric risk

```
par(mfrow = c(1,2))

rpart.plot(tree_risk_numeric1, main="Decision Tree: Numeric Risk", type=3, ex
tra=101, under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)

rpart.plot(tree_risk_numeric2, main="Decision Tree: Numeric Risk", type=3, ex
tra=101, under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)
```

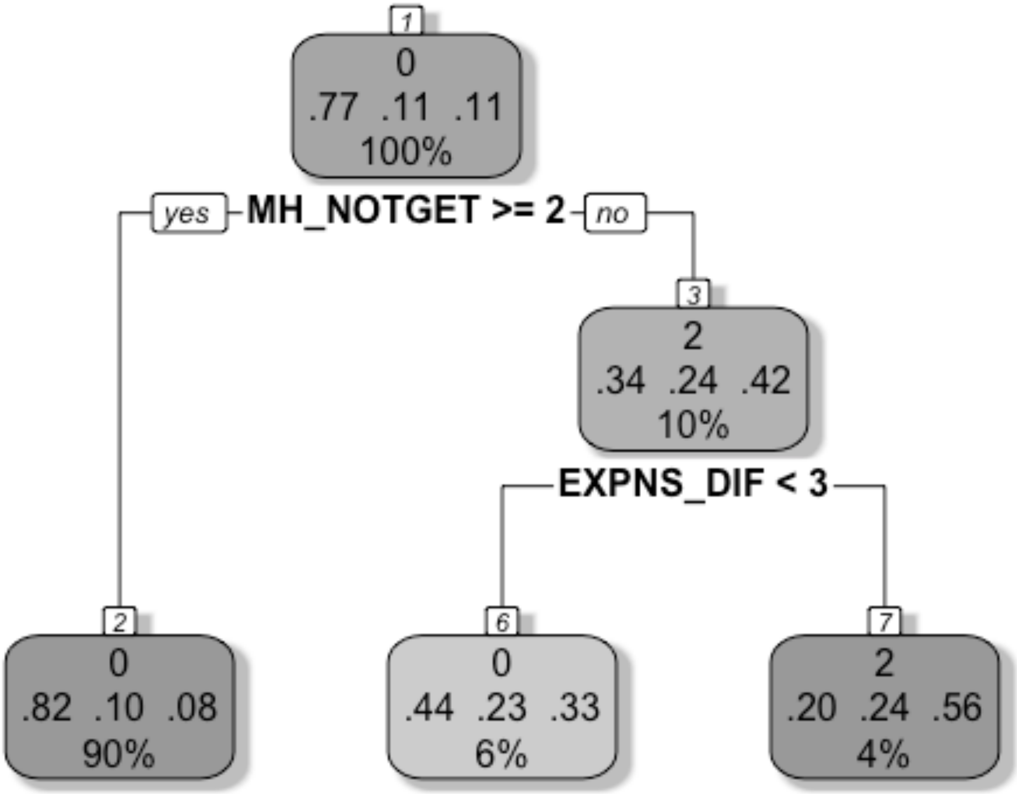




Test varying complexity parameters

```
control_params <- rpart.control(minsplit = 20, minbucket = 7, cp = 0.01)

tree_risk_numericA <- rpart(risk.numeric ~ .,
                     data = train_pulse[, !(names(train_pulse) %in% c("GAD"
, "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.binary"))],
                     method = "class")

print("Summary of decision tree model for risk.numeric")

## [1] "Summary of decision tree model for risk.numeric"

summary(tree_risk_numericA)

## Call:
## rpart(formula = risk.numeric ~ ., data = train_pulse[, !(names(train_pulse
) %in%
##     c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.binary"))]
,
##     method = "class")
##   n= 44517
##
##             CP nsplit rel error    xerror       xstd
## 1 0.03592814      0 1.0000000 1.0000000 0.008794150
## 2 0.02704591      1 0.9640719 0.9640719 0.008679664
## 3 0.01000000      2 0.9370259 0.9370259 0.008590248
## 
## Variable importance
##  MH_NOTGET  EXPNS_DIF CURFOODSUF   MORTCONF  EXPCTLOSS     INCOME   PRIVHL
TH 
##         85          7          2          2          1          1        
1 
## 
## Node number 1: 44517 observations,    complexity param=0.03592814
##   predicted class=0  expected loss=0.2250826  P(node) =1
##     class counts: 34497  4975  5045
##    probabilities: 0.775 0.112 0.113 
##   left son=2 (40230 obs) right son=3 (4287 obs)
##   Primary splits:
##       MH_NOTGET  < 1.5 to the right, improve=1427.1900, (0 missing)
##       EXPNS_DIF  < 2.5 to the left,  improve=1139.2320, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=1021.2310, (0 missing)
##       PRESCRIPT  < 1.5 to the right, improve= 765.8515, (0 missing)
##       MORTCONF   < 3.5 to the right, improve= 718.2238, (0 missing)
## 
## Node number 2: 40230 observations
##   predicted class=0  expected loss=0.1787721  P(node) =0.9036997
##     class counts: 33038  3966  3226
##    probabilities: 0.821 0.099 0.080 
## 
## Node number 3: 4287 observations,    complexity param=0.02704591
##   predicted class=2  expected loss=0.575694  P(node) =0.09630029
##     class counts:  1459  1009  1819
##    probabilities: 0.340 0.235 0.424 
##   left son=6 (2551 obs) right son=7 (1736 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5 to the left,  improve=113.98500, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve= 90.07739, (0 missing)
##       NOTGET     < 1.5 to the right, improve= 55.44220, (0 missing)
##       MORTCONF   < 3.5 to the right, improve= 53.98824, (0 missing)
##       DELAY      < 1.5 to the right, improve= 41.76672, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 1.5 to the left,  agree=0.739, adj=0.354, (0 split)
##       MORTCONF   < 3.5 to the right, agree=0.735, adj=0.346, (0 split)
##       EXPCTLOSS  < 1.5 to the right, agree=0.680, adj=0.210, (0 split)
##       INCOME     < 3.5 to the right, agree=0.666, adj=0.174, (0 split)
##       PRIVHLTH   < 1.5 to the left,  agree=0.662, adj=0.165, (0 split)
## 
## Node number 6: 2551 observations
##   predicted class=0  expected loss=0.5637005  P(node) =0.05730395
##     class counts:  1113   596   842
##    probabilities: 0.436 0.234 0.330 
## 
```

```
## Node number 7: 1736 observations
##    predicted class=2  expected loss=0.437212  P(node) =0.03899634
##      class counts:   346   413   977
##     probabilities: 0.199 0.238 0.563

predictions_risk_numericA <- predict(tree_risk_numericA, newdata = test_pulse
, type = "class")

accuracy_risk_numericA <- mean(predictions_risk_numericA == test_pulse$risk.n
umeric)
cat("Accuracy for risk.numeric:", accuracy_risk_numericA, "\n")

## Accuracy for risk.numeric: 0.7959537

rpart.plot(tree_risk_numericA, box.palette = "Greys", shadow.col = "gray", nn
= TRUE)
```



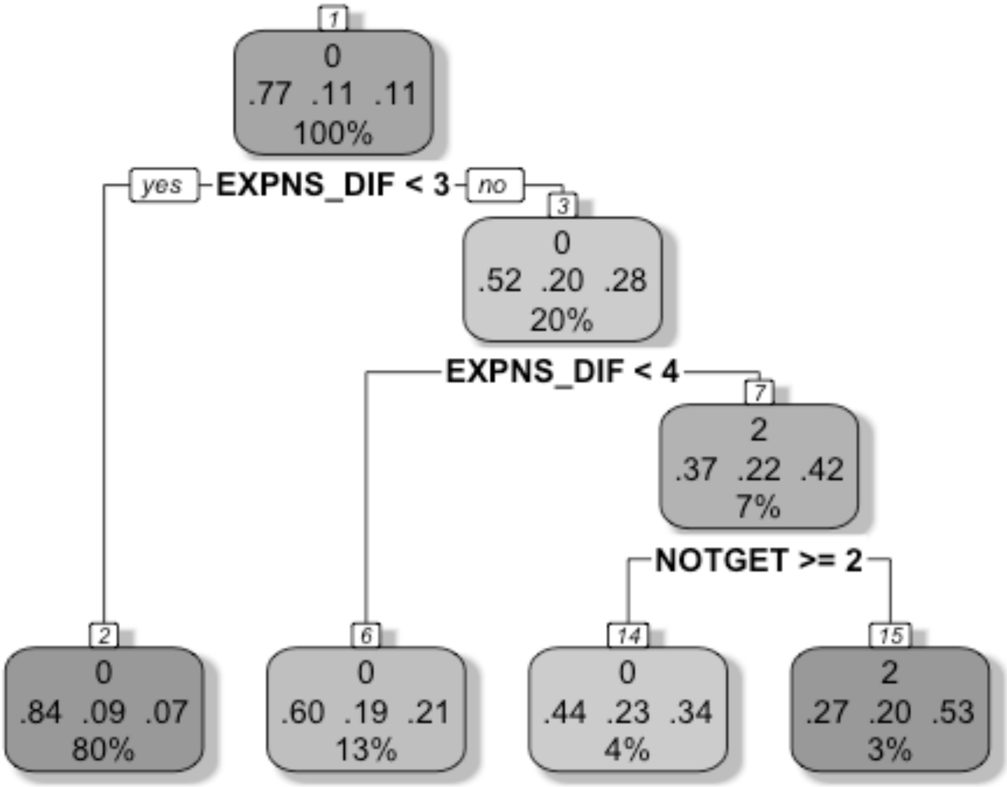
```
tree_risk_numericB <- rpart(risk.numeric ~ .,
                       data = train_pulse_trees[, !(names(train_pulse_trees)
%in% c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.binary"))],
                       method = "class")

print("Summary of decision tree model for risk.numeric")
```

```
## [1] "Summary of decision tree model for risk.numeric"
```

```r
summary(tree_risk_numericB)
```

```
## Call:
## rpart(formula = risk.numeric ~ ., data = train_pulse_trees[,
##     !(names(train_pulse_trees) %in% c("GAD", "PHQ", "risk.GAD",
##         "risk.PHQ", "total.risk", "risk.binary"))], method = "class")
##   n= 44517
##
##           CP nsplit rel error    xerror        xstd
## 1 0.01107784      0 1.0000000 1.0000000 0.008794150
## 2 0.01000000      3 0.9667665 0.9781437 0.008725079
##
## Variable importance
##  EXPNS_DIF   MORTCONF CURFOODSUF  CHILDFOOD    RENTCUR  EXPCTLOSS     NOTG
ET
##         63         11         10          4          4          3
2
##      DELAY      EVICT
##          1          1
##
## Node number 1: 44517 observations,    complexity param=0.01107784
##   predicted class=0  expected loss=0.2250826  P(node) =1
##     class counts: 34497  4975  5045
##    probabilities: 0.775 0.112 0.113
##   left son=2 (35472 obs) right son=3 (9045 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5 to the left,  improve=1139.2320, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=1021.2310, (0 missing)
##       MORTCONF   < 3.5 to the right, improve= 718.2238, (0 missing)
##       NOTGET     < 1.5 to the right, improve= 649.8609, (0 missing)
##       DELAY      < 1.5 to the right, improve= 627.3014, (0 missing)
##   Surrogate splits:
##       MORTCONF   < 3.5 to the right, agree=0.835, adj=0.188, (0 split)
##       CURFOODSUF < 2.5 to the left,  agree=0.829, adj=0.161, (0 split)
##       CHILDFOOD  < 2.5 to the right, agree=0.811, adj=0.068, (0 split)
##       RENTCUR    < 1.5 to the left,  agree=0.808, adj=0.054, (0 split)
##       EXPCTLOSS  < 1.5 to the right, agree=0.807, adj=0.049, (0 split)
##
## Node number 2: 35472 observations
##   predicted class=0  expected loss=0.1603236  P(node) =0.7968192
##     class counts: 29785  3200  2487
##    probabilities: 0.840 0.090 0.070
##
## Node number 3: 9045 observations,    complexity param=0.01107784
##   predicted class=0  expected loss=0.4790492  P(node) =0.2031808
##     class counts:  4712  1775  2558
##    probabilities: 0.521 0.196 0.283
##   left son=6 (5937 obs) right son=7 (3108 obs)
##   Primary splits:
##       EXPNS_DIF  < 3.5 to the left,  improve=196.85660, (0 missing)
##       NOTGET     < 1.5 to the right, improve=139.65000, (0 missing)
##       DELAY      < 1.5 to the right, improve=135.91430, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=116.53590, (0 missing)
##       MORTCONF   < 1.5 to the right, improve= 51.13054, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 2.5 to the left,  agree=0.720, adj=0.184, (0 split)
##       MORTCONF   < 1.5 to the right, agree=0.703, adj=0.137, (0 split)
##       RENTCUR    < 1.5 to the left,  agree=0.687, adj=0.088, (0 split)
##       EVICT      < 2.5 to the right, agree=0.681, adj=0.073, (0 split)
##       CHILDFOOD  < 2.5 to the right, agree=0.674, adj=0.052, (0 split)
##
## Node number 6: 5937 observations
##   predicted class=0  expected loss=0.3990231  P(node) =0.1333648
##     class counts:  3568  1105  1264
##    probabilities: 0.601 0.186 0.213
##
## Node number 7: 3108 observations,    complexity param=0.01107784
##   predicted class=2  expected loss=0.5836551  P(node) =0.06981603
##     class counts:  1144   670  1294
##    probabilities: 0.368 0.216 0.416
##   left son=14 (1829 obs) right son=15 (1279 obs)
##   Primary splits:
##       NOTGET     < 1.5 to the right, improve=49.51079, (0 missing)
##       DELAY      < 1.5 to the right, improve=44.85452, (0 missing)
##       CURFOODSUF < 2.5 to the left,  improve=31.67681, (0 missing)
##       FEWRTRIPS  < 1.5 to the right, improve=18.33927, (0 missing)
##       RRACE      < 0.5 to the right, improve=13.04758, (0 missing)
##   Surrogate splits:
##       DELAY       < 1.5 to the right, agree=0.819, adj=0.561, (0 split)
##       TNUM_PS     < 1.5 to the left,  agree=0.613, adj=0.060, (0 split)
##       CURFOODSUF  < 3.5 to the left,  agree=0.604, adj=0.038, (0 split)
##       COMPAVAIL   < 2.5 to the left,  agree=0.604, adj=0.037, (0 split)
##       INTRNTAVAIL < 2.5 to the left,  agree=0.602, adj=0.033, (0 split)
##
## Node number 14: 1829 observations
##   predicted class=0  expected loss=0.5626025  P(node) =0.04108543
##     class counts:   800   412   617
##    probabilities: 0.437 0.225 0.337
##
## Node number 15: 1279 observations
##   predicted class=2  expected loss=0.4706802  P(node) =0.0287306
##     class counts:   344   258   677
##    probabilities: 0.269 0.202 0.529
```

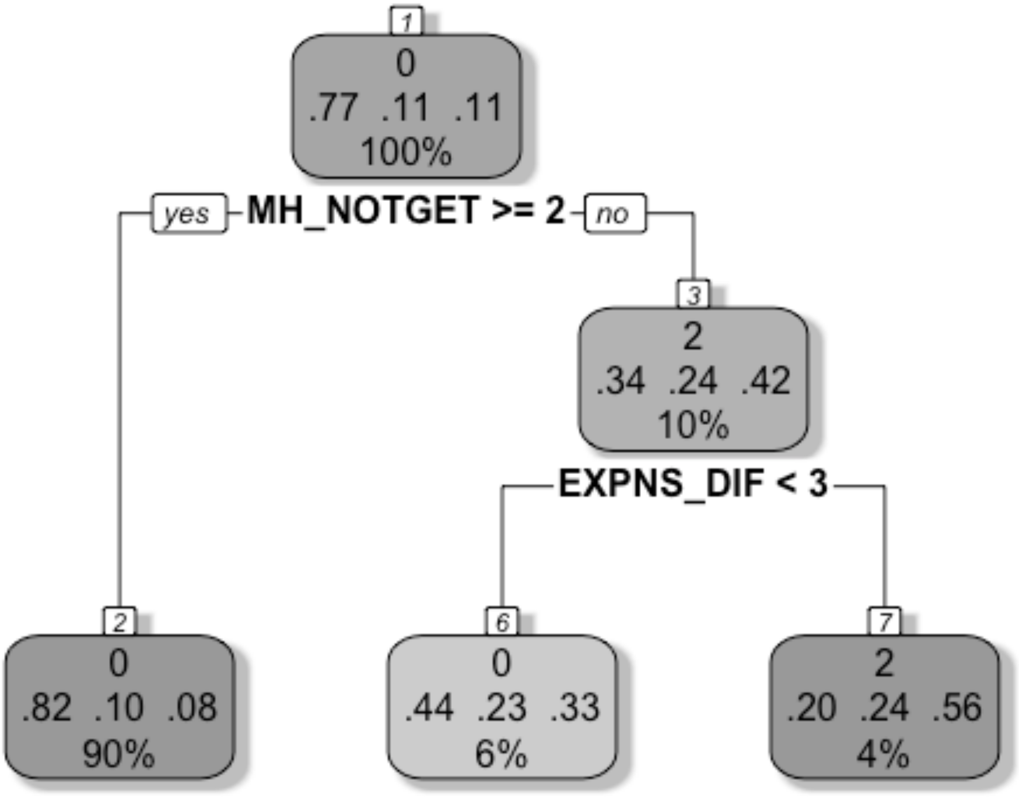
```r
predictions_risk_numericB <- predict(tree_risk_numericB, newdata = test_pulse
_trees, type = "class")

accuracy_risk_numericB <- mean(predictions_risk_numericB == test_pulse_trees$
risk.numeric)
cat("Accuracy for risk.numeric:", accuracy_risk_numericB, "\n")

## Accuracy for risk.numeric: 0.7887206

rpart.plot(tree_risk_numericB, box.palette = "Greys", shadow.col = "gray", nn
= TRUE)
```



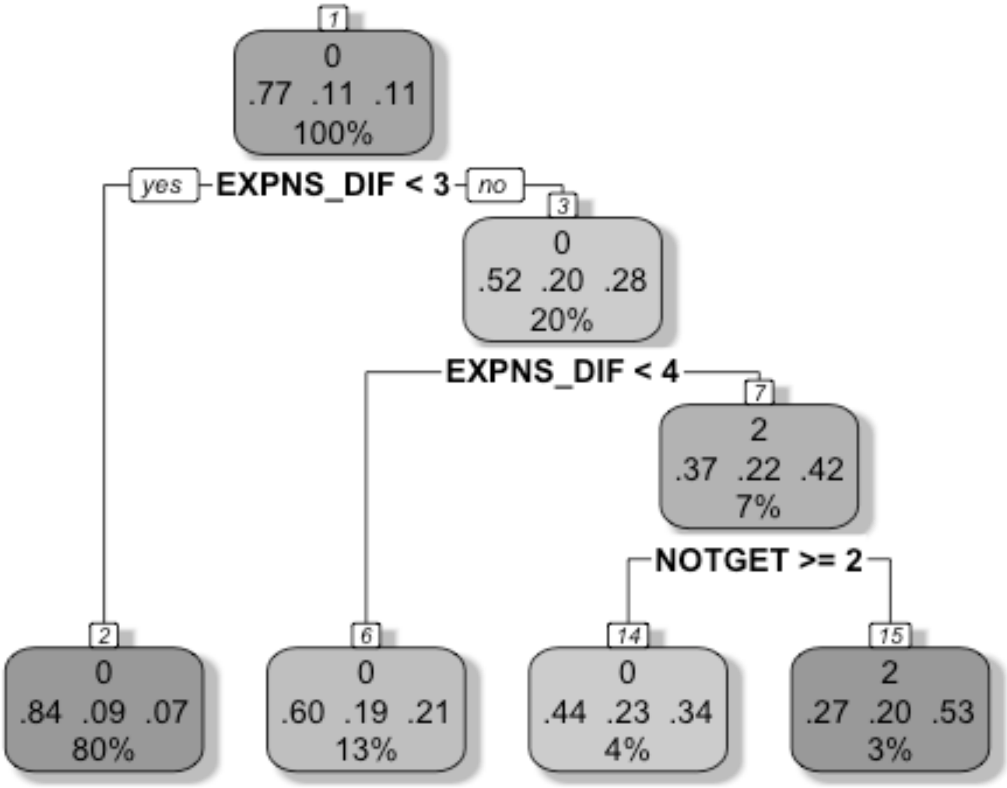
```
control_params <- rpart.control(minsplit = 10, minbucket = 5, cp = 0.001)

tree_risk_numericX <- rpart(risk.numeric ~ .,
                             data = train_pulse[, !(names(train_pulse) %in% c("GAD"
, "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.binary"))],
                             method = "class")

print("Summary of decision tree model for risk.numeric")

## [1] "Summary of decision tree model for risk.numeric"

summary(tree_risk_numericX)

## Call:
## rpart(formula = risk.numeric ~ ., data = train_pulse[, !(names(train_pulse
) %in%
##     c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.binary"))]
,
##     method = "class")
##   n= 44517
##
##           CP nsplit rel error    xerror        xstd
## 1 0.03592814      0 1.0000000 1.0000000 0.008794150
## 2 0.02704591      1 0.9640719 0.9640719 0.008679664
## 3 0.01000000      2 0.9370259 0.9370259 0.008590248
##
## Variable importance
##  MH_NOTGET  EXPNS_DIF CURFOODSUF   MORTCONF  EXPCTLOSS     INCOME   PRIVHL
TH
##         85          7          2          2          1          1
1
##
## Node number 1: 44517 observations,    complexity param=0.03592814
##   predicted class=0  expected loss=0.2250826  P(node) =1
##     class counts: 34497  4975  5045
##    probabilities: 0.775 0.112 0.113
##   left son=2 (40230 obs) right son=3 (4287 obs)
##   Primary splits:
##       MH_NOTGET  < 1.5 to the right, improve=1427.1900, (0 missing)
##       EXPNS_DIF  < 2.5 to the left,  improve=1139.2320, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=1021.2310, (0 missing)
##       PRESCRIPT  < 1.5 to the right, improve= 765.8515, (0 missing)
##       MORTCONF   < 3.5 to the right, improve= 718.2238, (0 missing)
##
## Node number 2: 40230 observations
##   predicted class=0  expected loss=0.1787721  P(node) =0.9036997
##     class counts: 33038  3966  3226
##    probabilities: 0.821 0.099 0.080
##
## Node number 3: 4287 observations,    complexity param=0.02704591
##   predicted class=2  expected loss=0.575694  P(node) =0.09630029
##     class counts:  1459  1009  1819
##    probabilities: 0.340 0.235 0.424
##   left son=6 (2551 obs) right son=7 (1736 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5 to the left,  improve=113.98500, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve= 90.07739, (0 missing)
##       NOTGET     < 1.5 to the right, improve= 55.44220, (0 missing)
##       MORTCONF   < 3.5 to the right, improve= 53.98824, (0 missing)
##       DELAY      < 1.5 to the right, improve= 41.76672, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 1.5 to the left,  agree=0.739, adj=0.354, (0 split)
##       MORTCONF   < 3.5 to the right, agree=0.735, adj=0.346, (0 split)
##       EXPCTLOSS  < 1.5 to the right, agree=0.680, adj=0.210, (0 split)
##       INCOME     < 3.5 to the right, agree=0.666, adj=0.174, (0 split)
##       PRIVHLTH   < 1.5 to the left,  agree=0.662, adj=0.165, (0 split)
```

```
##
## Node number 6: 2551 observations
##   predicted class=0  expected loss=0.5637005  P(node) =0.05730395
##     class counts:  1113   596   842
##    probabilities: 0.436 0.234 0.330
##
## Node number 7: 1736 observations
##   predicted class=2  expected loss=0.437212  P(node) =0.03899634
##     class counts:   346   413   977
##    probabilities: 0.199 0.238 0.563
```

```
predictions_risk_numericX <- predict(tree_risk_numericX, newdata = test_pulse
, type = "class")

accuracy_risk_numericX <- mean(predictions_risk_numericX == test_pulse$risk.n
umeric)
cat("Accuracy for risk.numeric:", accuracy_risk_numericX, "\n")

## Accuracy for risk.numeric: 0.7959537

rpart.plot(tree_risk_numericX, box.palette = "Greys", shadow.col = "gray", nn
= TRUE)
```



```
tree_risk_numericY <- rpart(risk.numeric ~ .,
                       data = train_pulse_trees[, !(names(train_pulse_trees)
%in% c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.binary"))],
                       method = "class")

print("Summary of decision tree model for risk.numeric")

## [1] "Summary of decision tree model for risk.numeric"

summary(tree_risk_numericY)

## Call:
## rpart(formula = risk.numeric ~ ., data = train_pulse_trees[,
##     !(names(train_pulse_trees) %in% c("GAD", "PHQ", "risk.GAD",
##         "risk.PHQ", "total.risk", "risk.binary"))], method = "class")
##   n= 44517
##
##           CP nsplit rel error    xerror        xstd
## 1 0.01107784      0 1.0000000 1.0000000 0.008794150
## 2 0.01000000      3 0.9667665 0.9748503 0.008714517
##
## Variable importance
##  EXPNS_DIF   MORTCONF  CURFOODSUF  CHILDFOOD    RENTCUR  EXPCTLOSS     NOTG
ET
##        63         11         10          4          4          3
2
##      DELAY      EVICT
##          1          1
##
## Node number 1: 44517 observations,    complexity param=0.01107784
##   predicted class=0  expected loss=0.2250826  P(node) =1
##     class counts: 34497  4975  5045
##    probabilities: 0.775 0.112 0.113
##   left son=2 (35472 obs) right son=3 (9045 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5 to the left,  improve=1139.2320, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=1021.2310, (0 missing)
##       MORTCONF   < 3.5 to the right, improve= 718.2238, (0 missing)
##       NOTGET     < 1.5 to the right, improve= 649.8609, (0 missing)
##       DELAY      < 1.5 to the right, improve= 627.3014, (0 missing)
##   Surrogate splits:
##       MORTCONF   < 3.5 to the right, agree=0.835, adj=0.188, (0 split)
##       CURFOODSUF < 2.5 to the left,  agree=0.829, adj=0.161, (0 split)
##       CHILDFOOD  < 2.5 to the right, agree=0.811, adj=0.068, (0 split)
##       RENTCUR    < 1.5 to the left,  agree=0.808, adj=0.054, (0 split)
##       EXPCTLOSS  < 1.5 to the right, agree=0.807, adj=0.049, (0 split)
##
## Node number 2: 35472 observations
##   predicted class=0  expected loss=0.1603236  P(node) =0.7968192
##     class counts: 29785  3200  2487
##    probabilities: 0.840 0.090 0.070 
## 
## Node number 3: 9045 observations,    complexity param=0.01107784
##   predicted class=0  expected loss=0.4790492  P(node) =0.2031808
##     class counts:  4712  1775  2558
##    probabilities: 0.521 0.196 0.283 
##   left son=6 (5937 obs) right son=7 (3108 obs)
##   Primary splits:
##       EXPNS_DIF  < 3.5 to the left,  improve=196.85660, (0 missing)
##       NOTGET     < 1.5 to the right, improve=139.65000, (0 missing)
##       DELAY      < 1.5 to the right, improve=135.91430, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=116.53590, (0 missing)
##       MORTCONF   < 1.5 to the right, improve= 51.13054, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 2.5 to the left,  agree=0.720, adj=0.184, (0 split)
##       MORTCONF   < 1.5 to the right, agree=0.703, adj=0.137, (0 split)
##       RENTCUR    < 1.5 to the left,  agree=0.687, adj=0.088, (0 split)
##       EVICT      < 2.5 to the right, agree=0.681, adj=0.073, (0 split)
##       CHILDFOOD  < 2.5 to the right, agree=0.674, adj=0.052, (0 split)
## 
## Node number 6: 5937 observations
##   predicted class=0  expected loss=0.3990231  P(node) =0.1333648
##     class counts:  3568  1105  1264
##    probabilities: 0.601 0.186 0.213 
## 
## Node number 7: 3108 observations,    complexity param=0.01107784
##   predicted class=2  expected loss=0.5836551  P(node) =0.06981603
##     class counts:  1144   670  1294
##    probabilities: 0.368 0.216 0.416 
##   left son=14 (1829 obs) right son=15 (1279 obs)
##   Primary splits:
##       NOTGET     < 1.5 to the right, improve=49.51079, (0 missing)
##       DELAY      < 1.5 to the right, improve=44.85452, (0 missing)
##       CURFOODSUF < 2.5 to the left,  improve=31.67681, (0 missing)
##       FEWRTRIPS  < 1.5 to the right, improve=18.33927, (0 missing)
##       RRACE      < 0.5 to the right, improve=13.04758, (0 missing)
##   Surrogate splits:
##       DELAY       < 1.5 to the right, agree=0.819, adj=0.561, (0 split)
##       TNUM_PS     < 1.5 to the left,  agree=0.613, adj=0.060, (0 split)
##       CURFOODSUF  < 3.5 to the left,  agree=0.604, adj=0.038, (0 split)
##       COMPAVAIL   < 2.5 to the left,  agree=0.604, adj=0.037, (0 split)
##       INTRNTAVAIL < 2.5 to the left,  agree=0.602, adj=0.033, (0 split)
## 
## Node number 14: 1829 observations
##   predicted class=0  expected loss=0.5626025  P(node) =0.04108543
##     class counts:   800   412   617
##    probabilities: 0.437 0.225 0.337 
## 
## Node number 15: 1279 observations
##   predicted class=2  expected loss=0.4706802  P(node) =0.0287306
##      class counts:   344   258   677
##     probabilities: 0.269 0.202 0.529

predictions_risk_numericY <- predict(tree_risk_numericY, newdata = test_pulse
_trees, type = "class")

accuracy_risk_numericY <- mean(predictions_risk_numericY == test_pulse_trees$
risk.numeric)
cat("Accuracy for risk.numeric:", accuracy_risk_numericY, "\n")

## Accuracy for risk.numeric: 0.7887206

rpart.plot(tree_risk_numericY, box.palette = "Greys", shadow.col = "gray", nn
= TRUE)
```



Model decision tree for binary risk including mental health variables

```
tree_risk_binary1 <- rpart(risk.binary ~ .,
                          data = train_pulse[, !(names(train_pulse) %in% c("GAD"
, "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.numeric"))],
                          method = "class")

print("Summary of decision tree model for risk.binary")
```

```
## [1] "Summary of decision tree model for risk.binary"
```

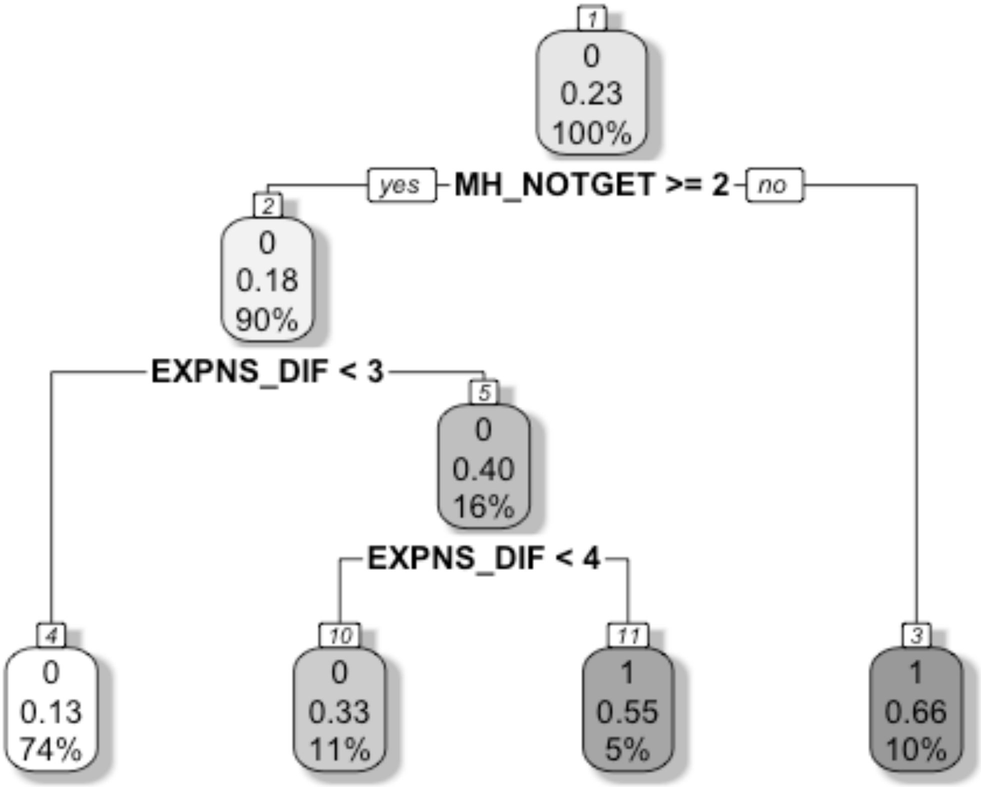
```
summary(tree_risk_binary1)
```

```
## Call:
## rpart(formula = risk.binary ~ ., data = train_pulse[, !(names(train_pulse)
%in%
##     c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.numeric"))
],
##     method = "class")
##   n= 44517
##
##           CP nsplit rel error    xerror        xstd
## 1 0.13662675      0 1.0000000 1.0000000 0.008794150
## 2 0.01242515      1 0.8633733 0.8633733 0.008331898
## 3 0.01000000      3 0.8385230 0.8441118 0.008260570
##
## Variable importance
##  MH_NOTGET  EXPNS_DIF   MORTCONF CURFOODSUF  CHILDFOOD    RENTCUR   FORCLO
SE
##         54         32          5          4          2          2
1
##
## Node number 1: 44517 observations,    complexity param=0.1366267
##   predicted class=0  expected loss=0.2250826  P(node) =1
##     class counts: 34497 10020
##    probabilities: 0.775 0.225
##   left son=2 (40230 obs) right son=3 (4287 obs)
##   Primary splits:
##       MH_NOTGET  < 1.5 to the right, improve=1791.8900, (0 missing)
##       EXPNS_DIF  < 2.5 to the left,  improve=1464.3070, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=1305.6270, (0 missing)
##       PRESCRIPT  < 1.5 to the right, improve= 993.3657, (0 missing)
##       MORTCONF   < 3.5 to the right, improve= 927.2313, (0 missing)
##
## Node number 2: 40230 observations,    complexity param=0.01242515
##   predicted class=0  expected loss=0.1787721  P(node) =0.9036997
##     class counts: 33038  7192
##    probabilities: 0.821 0.179
##   left son=4 (32921 obs) right son=5 (7309 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5 to the left,  improve=895.3732, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=760.7313, (0 missing)
##       PRESCRIPT  < 1.5 to the right, improve=567.1126, (0 missing)
##       MORTCONF   < 3.5 to the right, improve=554.8996, (0 missing)
##       MH_SVCS    < 1.5 to the right, improve=473.4015, (0 missing)
##   Surrogate splits:
##       MORTCONF   < 3.5 to the right, agree=0.846, adj=0.150, (0 split)
##       CURFOODSUF < 2.5 to the left,  agree=0.843, adj=0.137, (0 split)
##       CHILDFOOD  < 2.5 to the right, agree=0.829, adj=0.060, (0 split)
##       RENTCUR    < 1.5 to the left,  agree=0.826, adj=0.043, (0 split)
##       FORCLOSE   < 3.5 to the right, agree=0.825, adj=0.038, (0 split)
##
## Node number 3: 4287 observations
##   predicted class=1  expected loss=0.3403312  P(node) =0.09630029
##     class counts:  1459  2828
##    probabilities: 0.340 0.660
##
## Node number 4: 32921 observations
##   predicted class=0  expected loss=0.1290666  P(node) =0.7395152
##     class counts: 28672  4249
##    probabilities: 0.871 0.129
##
## Node number 5: 7309 observations,    complexity param=0.01242515
##   predicted class=0  expected loss=0.4026543  P(node) =0.1641845
##     class counts:  4366  2943
##    probabilities: 0.597 0.403
##   left son=10 (5004 obs) right son=11 (2305 obs)
##   Primary splits:
##       EXPNS_DIF  < 3.5 to the left,  improve=154.26120, (0 missing)
##       PRESCRIPT  < 1.5 to the right, improve=141.74140, (0 missing)
##       MH_SVCS    < 1.5 to the right, improve=121.92730, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve= 74.62165, (0 missing)
##       DELAY      < 1.5 to the right, improve= 63.06546, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 2.5 to the left,  agree=0.728, adj=0.137, (0 split)
##       MORTCONF   < 1.5 to the right, agree=0.721, adj=0.115, (0 split)
##       RENTCUR    < 1.5 to the left,  agree=0.710, adj=0.080, (0 split)
##       EVICT      < 2.5 to the right, agree=0.704, adj=0.063, (0 split)
##       CHILDFOOD  < 2.5 to the right, agree=0.697, adj=0.039, (0 split)
##
## Node number 10: 5004 observations
##   predicted class=0  expected loss=0.3329337  P(node) =0.1124065
##     class counts:  3338  1666
##    probabilities: 0.667 0.333
##
## Node number 11: 2305 observations
##   predicted class=1  expected loss=0.445987  P(node) =0.05177797
##     class counts:  1028  1277
##    probabilities: 0.446 0.554
```

```r
predictions_risk_binary1 <- predict(tree_risk_binary1, newdata = test_pulse,
type = "class")

accuracy_risk_binary1 <- mean(predictions_risk_binary1 == test_pulse$risk.bin
ary)
cat("Accuracy for risk.binary:", accuracy_risk_binary1, "\n")
```

```
## Accuracy for risk.binary: 0.8161853
```

```
rpart.plot(tree_risk_binary1, box.palette = "Greys", shadow.col = "gray", nn
= TRUE)
```

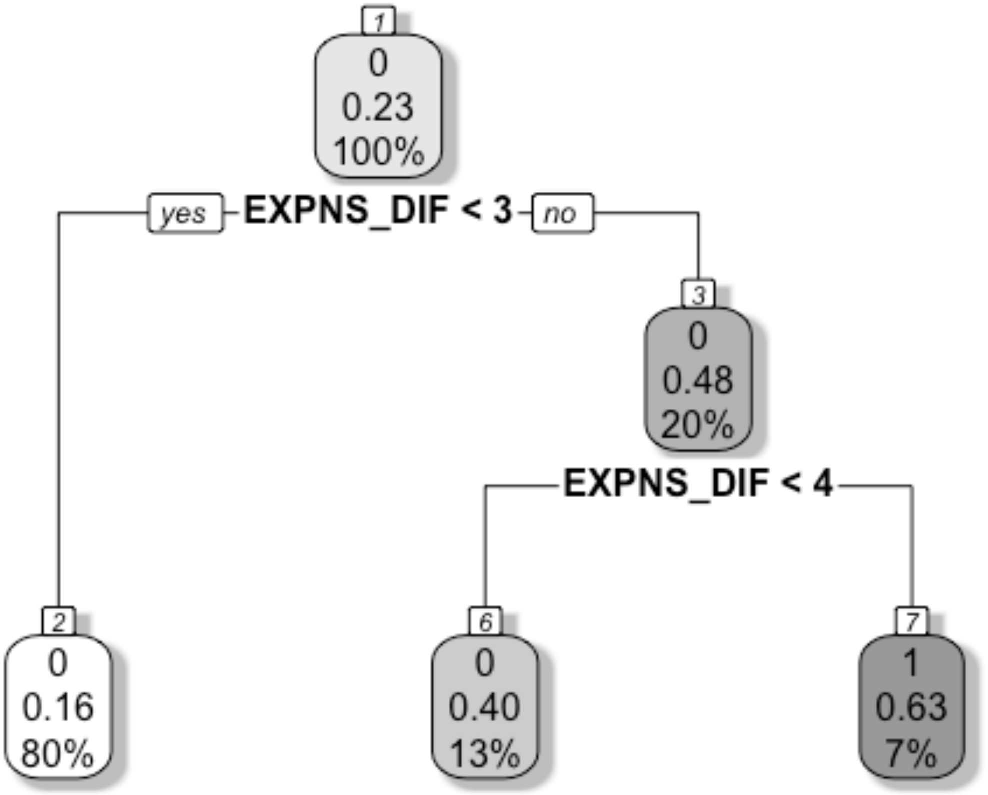


Model decision tree for binary risk including mental health variables

```
tree_risk_binary2 <- rpart(risk.binary ~ .,
                           data = train_pulse_trees[, !(names(train_pulse_trees)
%in% c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.numeric"))],
                           method = "class")

print("Summary of decision tree model for risk.binary")

## [1] "Summary of decision tree model for risk.binary"

summary(tree_risk_binary2)

## Call:
## rpart(formula = risk.binary ~ ., data = train_pulse_trees[, !(names(train_
pulse_trees) %in%
##     c("GAD", "PHQ", "risk.GAD", "risk.PHQ", "total.risk", "risk.numeric"))
],
##     method = "class")
##   n= 44517
##
```

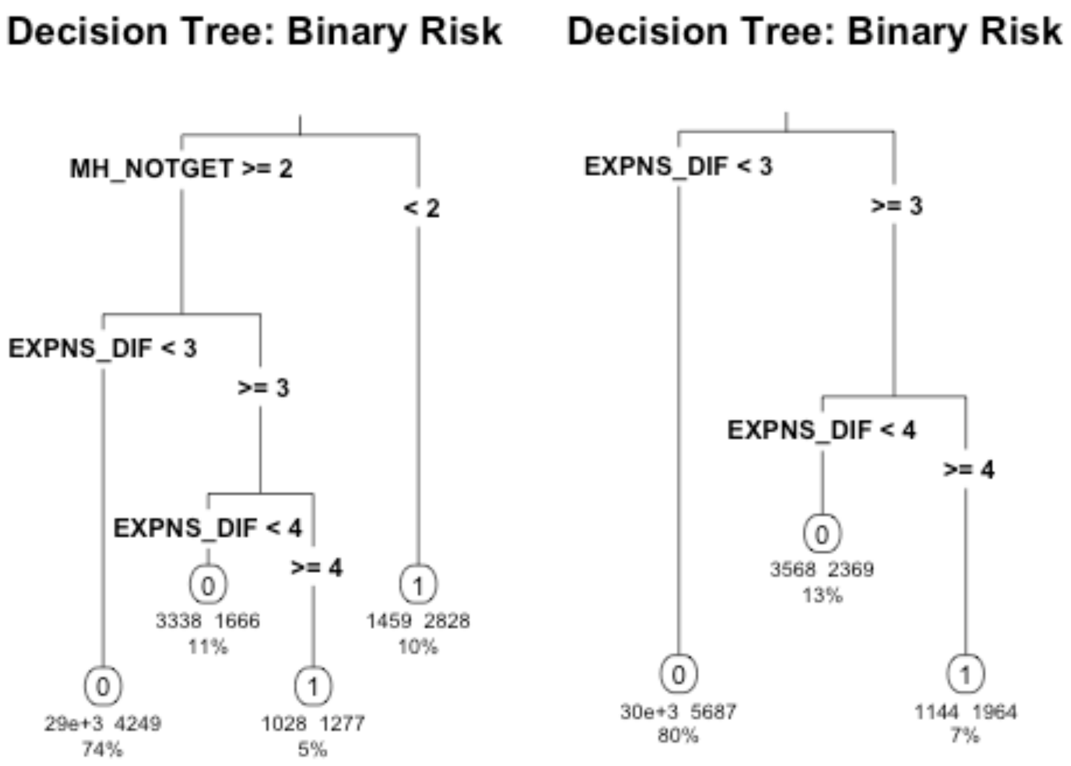
```
##           CP nsplit rel error    xerror        xstd
## 1 0.04091816      0 1.0000000 1.0000000 0.008794150
## 2 0.01000000      2 0.9181637 0.9181637 0.008526193
## 
## Variable importance
##  EXPNS_DIF   MORTCONF CURFOODSUF  CHILDFOOD    RENTCUR  EXPCTLOSS      EVI
CT 
##         66         12         11          4          4          3          
1 
## 
## Node number 1: 44517 observations,    complexity param=0.04091816
##   predicted class=0  expected loss=0.2250826  P(node) =1
##     class counts: 34497 10020
##    probabilities: 0.775 0.225 
##   left son=2 (35472 obs) right son=3 (9045 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5  to the left,  improve=1464.3070, (0 missing)
##       CURFOODSUF < 1.5  to the left,  improve=1305.6270, (0 missing)
##       MORTCONF   < 3.5  to the right, improve= 927.2313, (0 missing)
##       NOTGET     < 1.5  to the right, improve= 830.5590, (0 missing)
##       DELAY      < 1.5  to the right, improve= 815.9373, (0 missing)
##   Surrogate splits:
##       MORTCONF   < 3.5  to the right, agree=0.835, adj=0.188, (0 split)
##       CURFOODSUF < 2.5  to the left,  agree=0.829, adj=0.161, (0 split)
##       CHILDFOOD  < 2.5  to the right, agree=0.811, adj=0.068, (0 split)
##       RENTCUR    < 1.5  to the left,  agree=0.808, adj=0.054, (0 split)
##       EXPCTLOSS  < 1.5  to the right, agree=0.807, adj=0.049, (0 split)
## 
## Node number 2: 35472 observations
##   predicted class=0  expected loss=0.1603236  P(node) =0.7968192
##     class counts: 29785  5687
##    probabilities: 0.840 0.160 
## 
## Node number 3: 9045 observations,    complexity param=0.04091816
##   predicted class=0  expected loss=0.4790492  P(node) =0.2031808
##     class counts:  4712  4333
##    probabilities: 0.521 0.479 
##   left son=6 (5937 obs) right son=7 (3108 obs)
##   Primary splits:
##       EXPNS_DIF  < 3.5  to the left,  improve=221.3034, (0 missing)
##       DELAY      < 1.5  to the right, improve=156.1862, (0 missing)
##       NOTGET     < 1.5  to the right, improve=148.1898, (0 missing)
##       CURFOODSUF < 1.5  to the left,  improve=128.7954, (0 missing)
##       AGE        < 65.5 to the right, improve= 63.2674, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 2.5  to the left,  agree=0.720, adj=0.184, (0 split)
##       MORTCONF   < 1.5  to the right, agree=0.703, adj=0.137, (0 split)
##       RENTCUR    < 1.5  to the left,  agree=0.687, adj=0.088, (0 split)
##       EVICT      < 2.5  to the right, agree=0.681, adj=0.073, (0 split)
##       CHILDFOOD  < 2.5  to the right, agree=0.674, adj=0.052, (0 split)
```

```
##
## Node number 6: 5937 observations
##   predicted class=0  expected loss=0.3990231  P(node) =0.1333648
##     class counts:  3568  2369
##    probabilities: 0.601 0.399
##
## Node number 7: 3108 observations
##   predicted class=1  expected loss=0.3680824  P(node) =0.06981603
##     class counts:  1144  1964
##    probabilities: 0.368 0.632

predictions_risk_binary2 <- predict(tree_risk_binary2, newdata = test_pulse_trees, type = "class")

accuracy_risk_binary2 <- mean(predictions_risk_binary2 == test_pulse_trees$risk.binary)
cat("Accuracy for risk.binary:", accuracy_risk_binary2, "\n")

## Accuracy for risk.binary: 0.7990461

rpart.plot(tree_risk_binary2, box.palette = "Greys", shadow.col = "gray", nn = TRUE)
```



Plot the two decision trees for binary risk

```
par(mfrow = c(1,2))

rpart.plot(tree_risk_binary1, main="Decision Tree: Binary Risk", type=3, extr
a=101, under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)

rpart.plot(tree_risk_binary2, main="Decision Tree: Binary Risk", type=3, extr
a=101, under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)
```



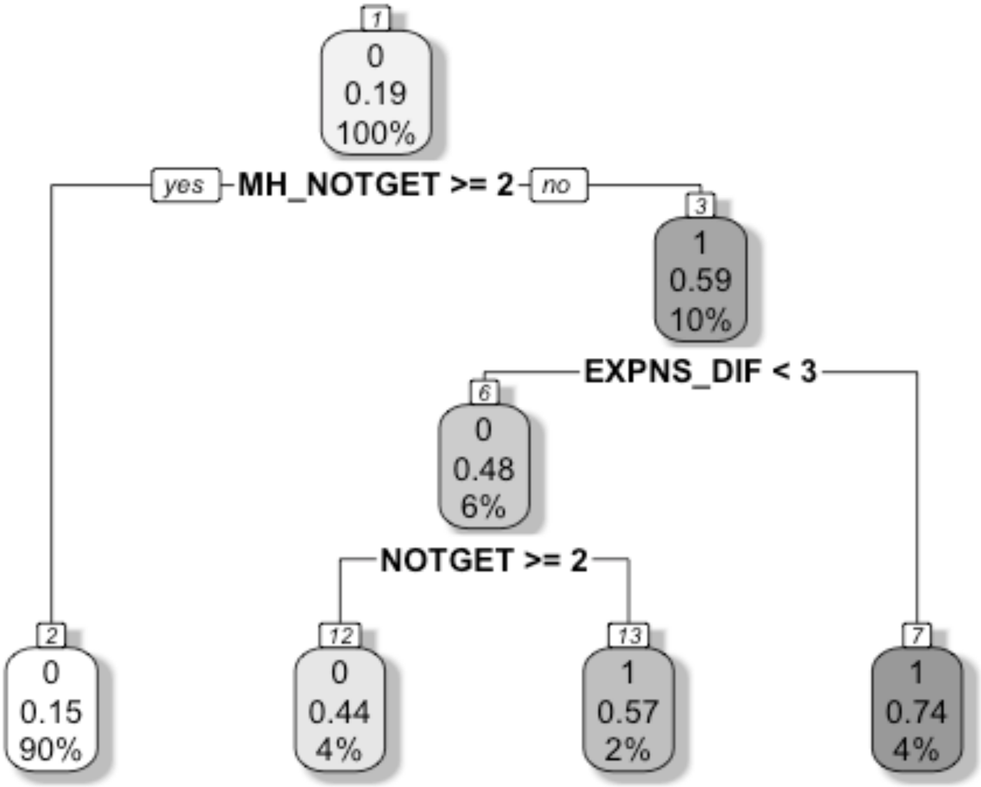
Model decision tree for GAD risk including mental health variables

```
tree_risk_gad1 <- rpart(risk.GAD ~ .,
                       data = train_pulse[, !(names(train_pulse) %in% c("GAD"
, "PHQ", "risk.binary", "risk.PHQ", "total.risk", "risk.numeric"))],
                       method = "class")

print("Summary of decision tree model for risk.gad")

## [1] "Summary of decision tree model for risk.gad"

summary(tree_risk_gad1)

## Call:
## rpart(formula = risk.GAD ~ ., data = train_pulse[, !(names(train_pulse) %i
n%
##     c("GAD", "PHQ", "risk.binary", "risk.PHQ", "total.risk",
##         "risk.numeric"))], method = "class")
##   n= 44517
##
##           CP nsplit rel error    xerror        xstd
## 1 0.08751594      0 1.0000000 1.0000000 0.009667048
## 2 0.01130173      1 0.9124841 0.9124841 0.009330978
## 3 0.01000000      3 0.8898806 0.9043700 0.009298267
##
## Variable importance
##  MH_NOTGET  EXPNS_DIF CURFOODSUF   MORTCONF  EXPCTLOSS     INCOME   PRIVHL
TH
##         82          7          3          3          2          1
1
##     NOTGET
##          1
##
## Node number 1: 44517 observations,    complexity param=0.08751594
##   predicted class=0  expected loss=0.1937911  P(node) =1
##     class counts: 35890  8627
##    probabilities: 0.806 0.194
##   left son=2 (40230 obs) right son=3 (4287 obs)
##   Primary splits:
##       MH_NOTGET  < 1.5      to the right, improve=1474.8150, (0 missing)
##       EXPNS_DIF  < 2.5      to the left,  improve=1229.7920, (0 missing)
##       CURFOODSUF < 1.5      to the left,  improve=1023.3540, (0 missing)
##       MORTCONF   < 3.5      to the right, improve= 770.7954, (0 missing)
##       PRESCRIPT  < 1.5      to the right, improve= 766.5432, (0 missing)
##
## Node number 2: 40230 observations
##   predicted class=0  expected loss=0.1517773  P(node) =0.9036997
##     class counts: 34124  6106
##    probabilities: 0.848 0.152
##
## Node number 3: 4287 observations,    complexity param=0.01130173
##   predicted class=1  expected loss=0.4119431  P(node) =0.09630029
##     class counts:  1766  2521
##    probabilities: 0.412 0.588
##   left son=6 (2551 obs) right son=7 (1736 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5      to the left,  improve=135.07320, (0 missing)
##       CURFOODSUF < 1.5      to the left,  improve= 85.84824, (0 missing)
##       MORTCONF   < 3.5      to the right, improve= 60.01275, (0 missing)
##       NOTGET     < 1.5      to the right, improve= 50.09425, (0 missing)
##       DELAY      < 1.5      to the right, improve= 43.03765, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 1.5      to the left,  agree=0.739, adj=0.354, (0 split
)
##       MORTCONF   < 3.5      to the right, agree=0.735, adj=0.346, (0 split
)
```

```
##       EXPCTLOSS  < 1.5      to the right, agree=0.680, adj=0.210, (0 split
)
##       INCOME     < 3.5      to the right, agree=0.666, adj=0.174, (0 split
)
##       PRIVHLTH   < 1.5      to the left,  agree=0.662, adj=0.165, (0 split
)
##
## Node number 6: 2551 observations,    complexity param=0.01130173
##   predicted class=0  expected loss=0.4845159  P(node) =0.05730395
##     class counts:  1315  1236
##    probabilities: 0.515 0.485
##   left son=12 (1671 obs) right son=13 (880 obs)
##   Primary splits:
##       NOTGET     < 1.5      to the right, improve=17.800130, (0 missing)
##       DELAY      < 1.5      to the right, improve=16.691530, (0 missing)
##       CURFOODSUF < 1.5      to the left,  improve=11.050070, (0 missing)
##       PWEIGHT    < 2213.879 to the left,  improve= 9.420395, (0 missing)
##       AGE        < 38.5     to the right, improve= 8.877596, (0 missing)
##   Surrogate splits:
##       DELAY      < 1.5      to the right, agree=0.824, adj=0.489, (0 split
)
##       CURFOODSUF < 2.5      to the left,  agree=0.659, adj=0.011, (0 split
)
##       MORTCONF   < 2.5      to the right, agree=0.659, adj=0.010, (0 split
)
##       CHILDFOOD  < 2.5      to the right, agree=0.657, adj=0.007, (0 split
)
##       RENTCUR    < 1.5      to the left,  agree=0.657, adj=0.007, (0 split
)
##
## Node number 7: 1736 observations
##   predicted class=1  expected loss=0.2597926  P(node) =0.03899634
##     class counts:   451  1285
##    probabilities: 0.260 0.740
##
## Node number 12: 1671 observations
##   predicted class=0  expected loss=0.4416517  P(node) =0.03753622
##     class counts:   933   738
##    probabilities: 0.558 0.442
##
## Node number 13: 880 observations
##   predicted class=1  expected loss=0.4340909  P(node) =0.01976773
##     class counts:   382   498
##    probabilities: 0.434 0.566
```

```r
predictions_risk_gad1 <- predict(tree_risk_gad1, newdata = test_pulse, type =
"class")

accuracy_risk_gad1 <- mean(predictions_risk_gad1 == test_pulse$risk.GAD)
cat("Accuracy for risk.gad:", accuracy_risk_gad1, "\n")
```

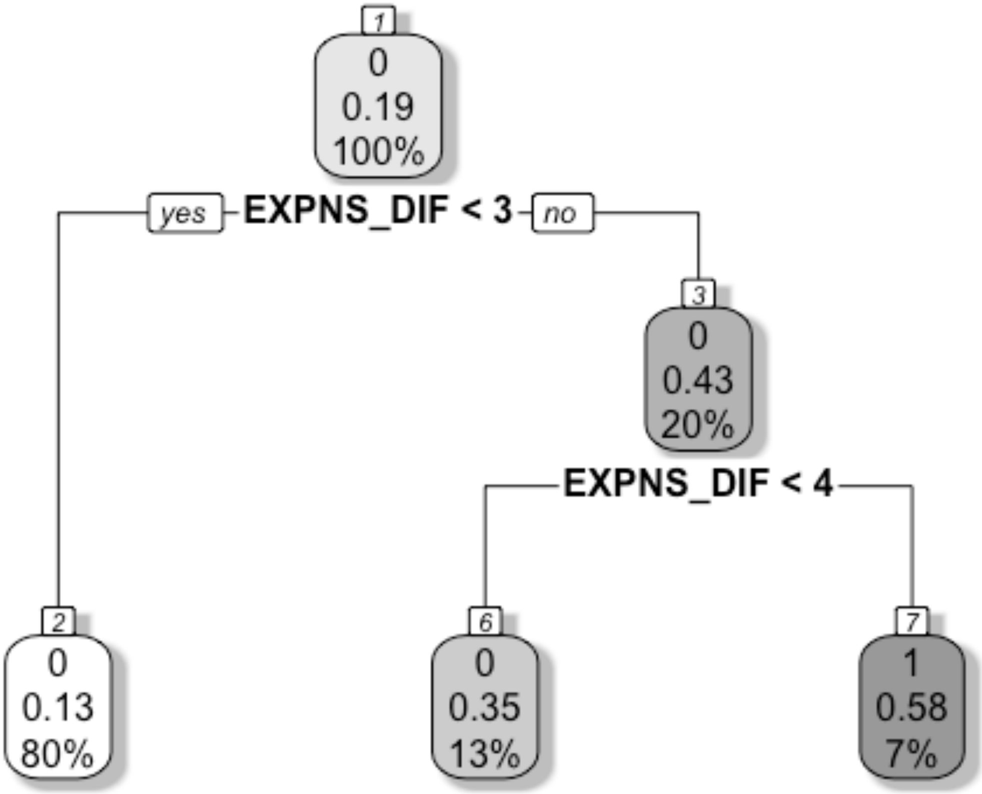
```
## Accuracy for risk.gad: 0.8339011

rpart.plot(tree_risk_gad1, box.palette = "Greys", shadow.col = "gray", nn = T
RUE)
```



Model decision tree for GAD risk excluding mental health variables

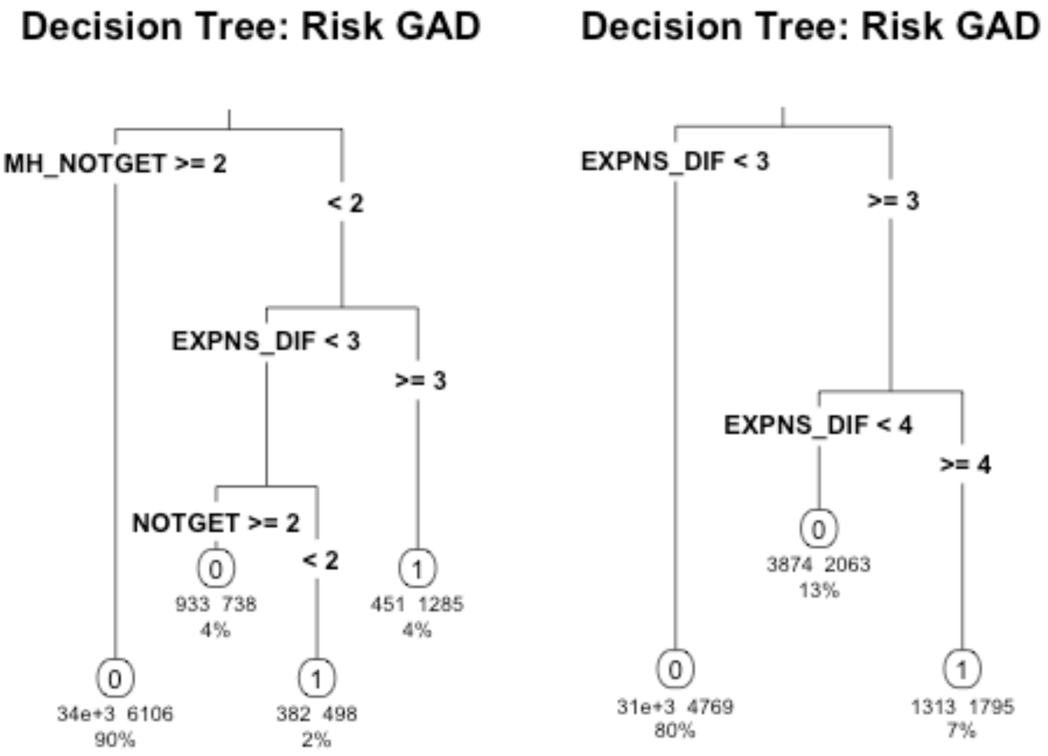
```
tree_risk_gad2 <- rpart(risk.GAD ~ .,
                        data = train_pulse_trees[, !(names(train_pulse_trees)
%in% c("GAD", "PHQ", "risk.binary", "risk.PHQ", "total.risk", "risk.numeric")
)],
                        method = "class")

print("Summary of decision tree model for risk.gad")

## [1] "Summary of decision tree model for risk.gad"

summary(tree_risk_gad2)

## Call:
## rpart(formula = risk.GAD ~ ., data = train_pulse_trees[, !(names(train_pul
se_trees) %in%
##     c("GAD", "PHQ", "risk.binary", "risk.PHQ", "total.risk",
##         "risk.numeric"))], method = "class")
##   n= 44517
##
##           CP nsplit rel error    xerror        xstd
## 1 0.02793555      0 1.0000000 1.0000000 0.009667048
## 2 0.01000000      2 0.9441289 0.9441289 0.009455977
##
## Variable importance
##  EXPNS_DIF   MORTCONF CURFOODSUF  CHILDFOOD    RENTCUR  EXPCTLOSS       EVI
CT
##         66         12         11          4          4          3
1
##
## Node number 1: 44517 observations,    complexity param=0.02793555
##   predicted class=0  expected loss=0.1937911  P(node) =1
##     class counts: 35890  8627
##    probabilities: 0.806 0.194
##   left son=2 (35472 obs) right son=3 (9045 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5  to the left,  improve=1229.7920, (0 missing)
##       CURFOODSUF < 1.5  to the left,  improve=1023.3540, (0 missing)
##       MORTCONF   < 3.5  to the right, improve= 770.7954, (0 missing)
##       NOTGET     < 1.5  to the right, improve= 704.4686, (0 missing)
##       DELAY      < 1.5  to the right, improve= 687.1613, (0 missing)
##   Surrogate splits:
##       MORTCONF   < 3.5  to the right, agree=0.835, adj=0.188, (0 split)
##       CURFOODSUF < 2.5  to the left,  agree=0.829, adj=0.161, (0 split)
##       CHILDFOOD  < 2.5  to the right, agree=0.811, adj=0.068, (0 split)
##       RENTCUR    < 1.5  to the left,  agree=0.808, adj=0.054, (0 split)
##       EXPCTLOSS  < 1.5  to the right, agree=0.807, adj=0.049, (0 split)
##
## Node number 2: 35472 observations
##   predicted class=0  expected loss=0.1344441  P(node) =0.7968192
##     class counts: 30703  4769
##    probabilities: 0.866 0.134
##
## Node number 3: 9045 observations,    complexity param=0.02793555
##   predicted class=0  expected loss=0.426534  P(node) =0.2031808
##     class counts:  5187  3858
##    probabilities: 0.573 0.427
##   left son=6 (5937 obs) right son=7 (3108 obs)
##   Primary splits:
##       EXPNS_DIF  < 3.5  to the left,  improve=215.94910, (0 missing)
##       DELAY      < 1.5  to the right, improve=147.04930, (0 missing)
##       NOTGET     < 1.5  to the right, improve=144.50300, (0 missing)
##       CURFOODSUF < 1.5  to the left,  improve=109.82480, (0 missing)
##       AGE        < 65.5 to the right, improve= 63.97985, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 2.5  to the left,  agree=0.720, adj=0.184, (0 split)
##       MORTCONF   < 1.5  to the right, agree=0.703, adj=0.137, (0 split)
##       RENTCUR    < 1.5  to the left,  agree=0.687, adj=0.088, (0 split)
##       EVICT      < 2.5  to the right, agree=0.681, adj=0.073, (0 split)
##       CHILDFOOD  < 2.5  to the right, agree=0.674, adj=0.052, (0 split)
##
## Node number 6: 5937 observations
##   predicted class=0  expected loss=0.3474819  P(node) =0.1333648
##     class counts:  3874  2063
##    probabilities: 0.653 0.347
##
## Node number 7: 3108 observations
##   predicted class=1  expected loss=0.4224582  P(node) =0.06981603
##     class counts:  1313  1795
##    probabilities: 0.422 0.578
```

```
predictions_risk_gad2 <- predict(tree_risk_gad2, newdata = test_pulse_trees,
type = "class")

accuracy_risk_gad2 <- mean(predictions_risk_gad2 == test_pulse_trees$risk.GAD
)
cat("Accuracy for risk.gad:", accuracy_risk_gad2, "\n")

## Accuracy for risk.gad: 0.8237853

rpart.plot(tree_risk_gad2, box.palette = "Greys", shadow.col = "gray", nn = T
RUE)
```



Plot the two decision trees for Risk GAD

```
par(mfrow = c(1,2))

rpart.plot(tree_risk_gad1, main="Decision Tree: Risk GAD", type=3, extra=101,
under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)

rpart.plot(tree_risk_gad2, main="Decision Tree: Risk GAD", type=3, extra=101,
under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)
```



Model decision tree for PHQ risk including mental health variables

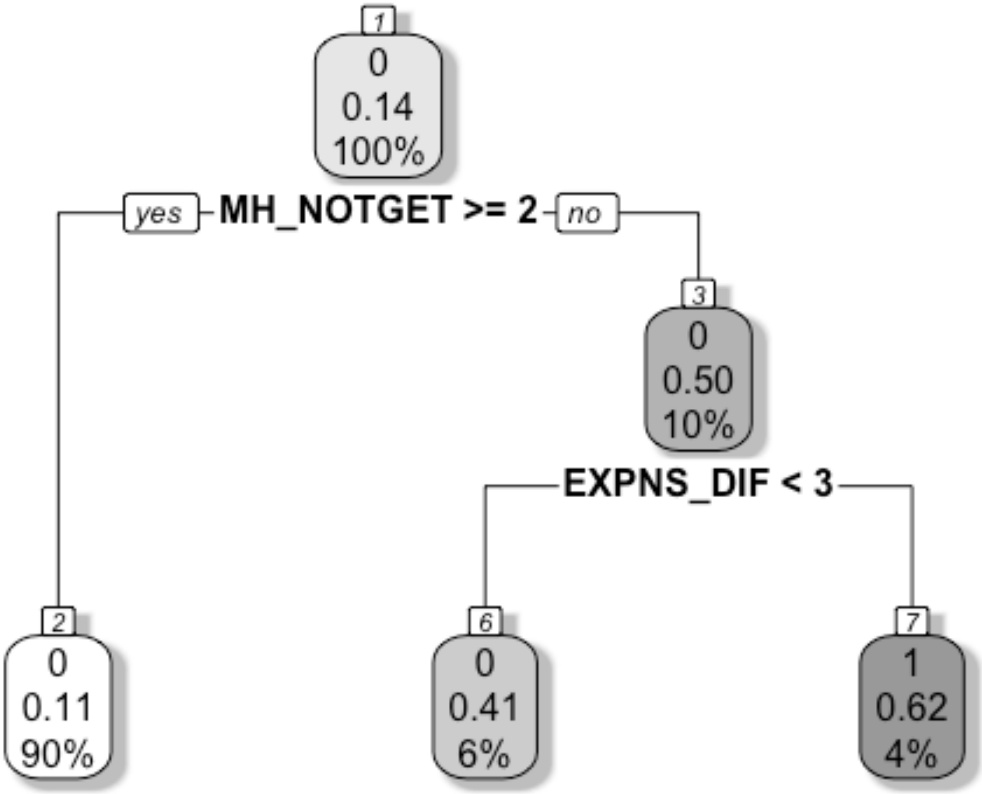
```
tree_risk_phq1 <- rpart(risk.PHQ ~ .,
                        data = train_pulse[, !(names(train_pulse) %in% c("GAD"
, "PHQ", "risk.binary", "risk.GAD", "total.risk", "risk.numeric"))],
                        method = "class")

print("Summary of decision tree model for risk.phq")

## [1] "Summary of decision tree model for risk.phq"

summary(tree_risk_phq1)
```

```
## Call:
## rpart(formula = risk.PHQ ~ ., data = train_pulse[, !(names(train_pulse) %i
n%
##     c("GAD", "PHQ", "risk.binary", "risk.GAD", "total.risk",
##         "risk.numeric"))], method = "class")
##   n= 44517
##
##           CP nsplit rel error    xerror       xstd
## 1 0.03324014      0 1.0000000 1.0000000 0.01152668
## 2 0.01000000      2 0.9335197 0.9419074 0.01124167
##
## Variable importance
##  MH_NOTGET  EXPNS_DIF CURFOODSUF   MORTCONF  EXPCTLOSS     INCOME   PRIVHL
TH
##         85          7          2          2          1          1
1
##
## Node number 1: 44517 observations,    complexity param=0.03324014
##   predicted class=0  expected loss=0.1446189  P(node) =1
##     class counts: 38079  6438
##    probabilities: 0.855 0.145
##   left son=2 (40230 obs) right son=3 (4287 obs)
##   Primary splits:
##       MH_NOTGET  < 1.5 to the right, improve=1170.8820, (0 missing)
##       EXPNS_DIF  < 2.5 to the left,  improve= 825.6587, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve= 823.2241, (0 missing)
##       PRESCRIPT  < 1.5 to the right, improve= 583.4023, (0 missing)
##       MORTCONF   < 3.5 to the right, improve= 514.7214, (0 missing)
##
## Node number 2: 40230 observations
##   predicted class=0  expected loss=0.1071837  P(node) =0.9036997
##     class counts: 35918  4312
##    probabilities: 0.893 0.107
##
## Node number 3: 4287 observations,    complexity param=0.03324014
##   predicted class=0  expected loss=0.4959179  P(node) =0.09630029
##     class counts:  2161  2126
##    probabilities: 0.504 0.496
##   left son=6 (2551 obs) right son=7 (1736 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5 to the left,  improve=94.63415, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=93.82137, (0 missing)
##       NOTGET     < 1.5 to the right, improve=59.60034, (0 missing)
##       MORTCONF   < 3.5 to the right, improve=48.24680, (0 missing)
##       FEWRTRIPS  < 1.5 to the right, improve=42.34512, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 1.5 to the left,  agree=0.739, adj=0.354, (0 split)
##       MORTCONF   < 3.5 to the right, agree=0.735, adj=0.346, (0 split)
##       EXPCTLOSS  < 1.5 to the right, agree=0.680, adj=0.210, (0 split)
##       INCOME     < 3.5 to the right, agree=0.666, adj=0.174, (0 split)
##       PRIVHLTH   < 1.5 to the left,  agree=0.662, adj=0.165, (0 split)
##
## Node number 6: 2551 observations
##   predicted class=0  expected loss=0.4092513  P(node) =0.05730395
##     class counts:  1507  1044
##    probabilities: 0.591 0.409
##
## Node number 7: 1736 observations
##   predicted class=1  expected loss=0.3767281  P(node) =0.03899634
##     class counts:   654  1082
##    probabilities: 0.377 0.623
```

```
predictions_risk_phq1 <- predict(tree_risk_phq1, newdata = test_pulse, type =
"class")

accuracy_risk_phq1 <- mean(predictions_risk_phq1 == test_pulse$risk.PHQ)
cat("Accuracy for risk.phq:", accuracy_risk_phq1, "\n")
```

```
## Accuracy for risk.phq: 0.8696473
```

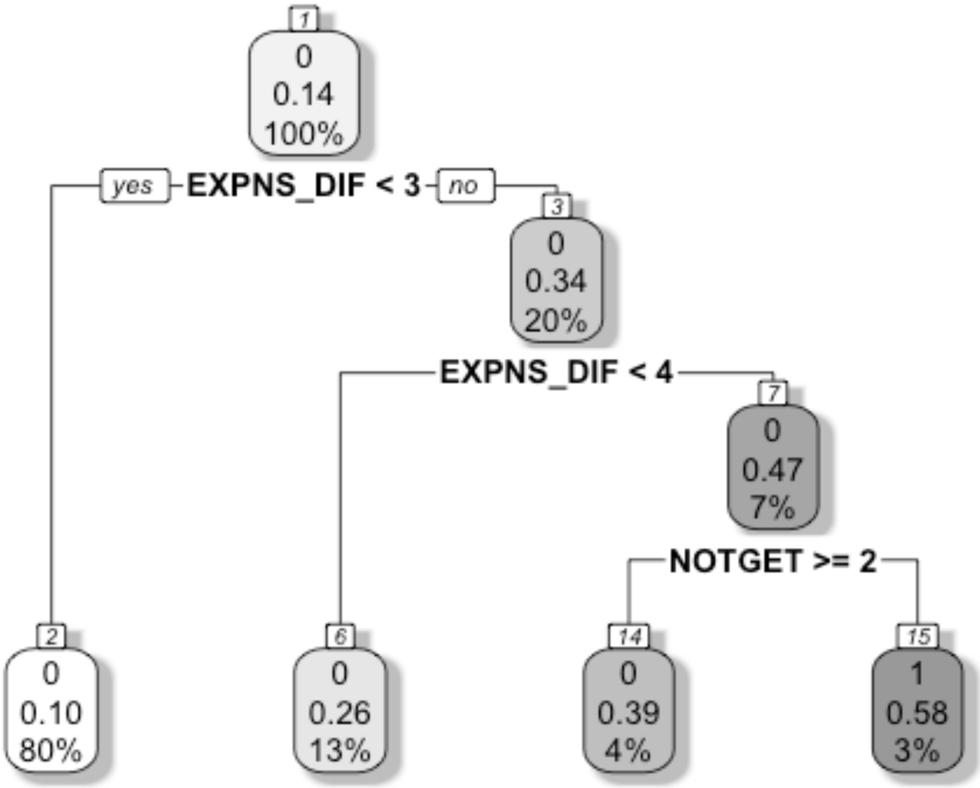
```
rpart.plot(tree_risk_phq1, box.palette = "Greys", shadow.col = "gray", nn = T
RUE)
```



Model decision tree for PHQ risk excluding mental health variables

```
tree_risk_phq2 <- rpart(risk.PHQ ~ .,
                        data = train_pulse_trees[, !(names(train_pulse_trees)
%in% c("GAD", "PHQ", "risk.binary", "risk.GAD", "total.risk", "risk.numeric")
)],
                        method = "class")

print("Summary of decision tree model for risk.phq")

## [1] "Summary of decision tree model for risk.phq"

summary(tree_risk_phq2)

## Call:
## rpart(formula = risk.PHQ ~ ., data = train_pulse_trees[, !(names(train_pul
se_trees) %in%
##     c("GAD", "PHQ", "risk.binary", "risk.GAD", "total.risk",
##         "risk.numeric"))], method = "class")
##   n= 44517
##
##           CP nsplit rel error    xerror       xstd
## 1 0.01061406      0 1.0000000 1.0000000 0.01152668
## 2 0.01000000      3 0.9681578 0.9723517 0.01139275
##
## Variable importance
##  EXPNS_DIF    MORTCONF CURFOODSUF   CHILDFOOD     RENTCUR      NOTGET   EXPCTLO
SS
##         62          11          10           4           4           3
2
##      DELAY       EVICT
##          2           1
##
## Node number 1: 44517 observations,    complexity param=0.01061406
##   predicted class=0  expected loss=0.1446189  P(node) =1
##     class counts: 38079  6438
##    probabilities: 0.855 0.145
##   left son=2 (35472 obs) right son=3 (9045 obs)
##   Primary splits:
##       EXPNS_DIF  < 2.5 to the left,  improve=825.6587, (0 missing)
##       CURFOODSUF < 1.5 to the left,  improve=823.2241, (0 missing)
##       MORTCONF   < 3.5 to the right, improve=514.7214, (0 missing)
##       NOTGET     < 1.5 to the right, improve=478.6400, (0 missing)
##       DELAY      < 1.5 to the right, improve=422.3380, (0 missing)
##   Surrogate splits:
##       MORTCONF   < 3.5 to the right, agree=0.835, adj=0.188, (0 split)
##       CURFOODSUF < 2.5 to the left,  agree=0.829, adj=0.161, (0 split)
##       CHILDFOOD  < 2.5 to the right, agree=0.811, adj=0.068, (0 split)
##       RENTCUR    < 1.5 to the left,  agree=0.808, adj=0.054, (0 split)
##       EXPCTLOSS  < 1.5 to the right, agree=0.807, adj=0.049, (0 split)
##
## Node number 2: 35472 observations
##   predicted class=0  expected loss=0.0959912  P(node) =0.7968192
##     class counts: 32067  3405
##    probabilities: 0.904 0.096 
## 
## Node number 3: 9045 observations,    complexity param=0.01061406
##   predicted class=0  expected loss=0.3353234  P(node) =0.2031808
##     class counts:  6012  3033
##    probabilities: 0.665 0.335 
##   left son=6 (5937 obs) right son=7 (3108 obs)
##   Primary splits:
##       EXPNS_DIF  < 3.5 to the left,  improve=173.60920, (0 missing)
##       NOTGET     < 1.5 to the right, improve=134.05700, (0 missing)
##       CURFOODSUF < 2.5 to the left,  improve=125.91390, (0 missing)
##       DELAY      < 1.5 to the right, improve=119.96270, (0 missing)
##       MORTCONF   < 1.5 to the right, improve= 42.51588, (0 missing)
##   Surrogate splits:
##       CURFOODSUF < 2.5 to the left,  agree=0.720, adj=0.184, (0 split)
##       MORTCONF   < 1.5 to the right, agree=0.703, adj=0.137, (0 split)
##       RENTCUR    < 1.5 to the left,  agree=0.687, adj=0.088, (0 split)
##       EVICT      < 2.5 to the right, agree=0.681, adj=0.073, (0 split)
##       CHILDFOOD  < 2.5 to the right, agree=0.674, adj=0.052, (0 split)
## 
## Node number 6: 5937 observations
##   predicted class=0  expected loss=0.2644433  P(node) =0.1333648
##     class counts:  4367  1570
##    probabilities: 0.736 0.264 
## 
## Node number 7: 3108 observations,    complexity param=0.01061406
##   predicted class=0  expected loss=0.4707207  P(node) =0.06981603
##     class counts:  1645  1463
##    probabilities: 0.529 0.471 
##   left son=14 (1829 obs) right son=15 (1279 obs)
##   Primary splits:
##       NOTGET     < 1.5 to the right, improve=52.04289, (0 missing)
##       DELAY      < 1.5 to the right, improve=47.91336, (0 missing)
##       CURFOODSUF < 2.5 to the left,  improve=41.83510, (0 missing)
##       FEWRTRIPS  < 1.5 to the right, improve=20.93129, (0 missing)
##       RRACE      < 0.5 to the right, improve=14.60492, (0 missing)
##   Surrogate splits:
##       DELAY       < 1.5 to the right, agree=0.819, adj=0.561, (0 split)
##       TNUM_PS     < 1.5 to the left,  agree=0.613, adj=0.060, (0 split)
##       CURFOODSUF  < 3.5 to the left,  agree=0.604, adj=0.038, (0 split)
##       COMPAVAIL   < 2.5 to the left,  agree=0.604, adj=0.037, (0 split)
##       INTRNTAVAIL < 2.5 to the left,  agree=0.602, adj=0.033, (0 split)
## 
## Node number 14: 1829 observations
##   predicted class=0  expected loss=0.3942045  P(node) =0.04108543
##     class counts:  1108   721
##    probabilities: 0.606 0.394 
## 
```

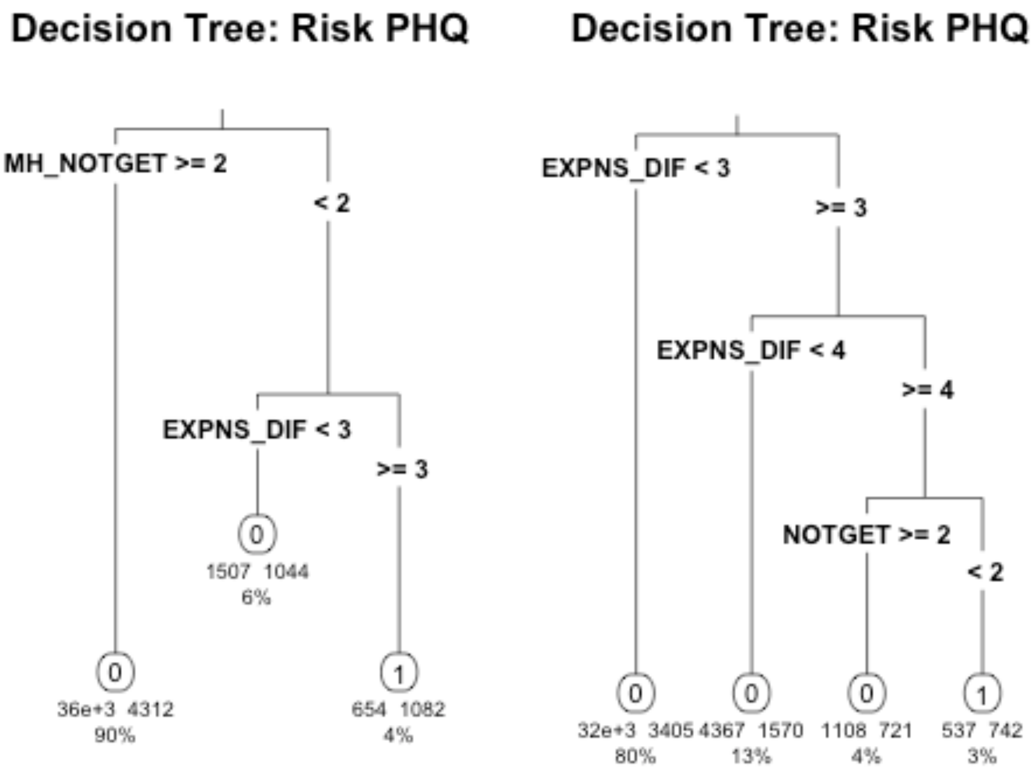
```
## Node number 15: 1279 observations
##    predicted class=1  expected loss=0.4198593  P(node) =0.0287306
##      class counts:   537   742
##     probabilities: 0.420 0.580

predictions_risk_phq2 <- predict(tree_risk_phq2, newdata = test_pulse_trees,
type = "class")

accuracy_risk_phq2 <- mean(predictions_risk_phq2 == test_pulse_trees$risk.PHQ
)
cat("Accuracy for risk.phq:", accuracy_risk_phq2, "\n")

## Accuracy for risk.phq: 0.8638818

rpart.plot(tree_risk_phq2, box.palette = "Greys", shadow.col = "gray", nn = T
RUE)
```



Plot the two decision trees for Risk PHQ

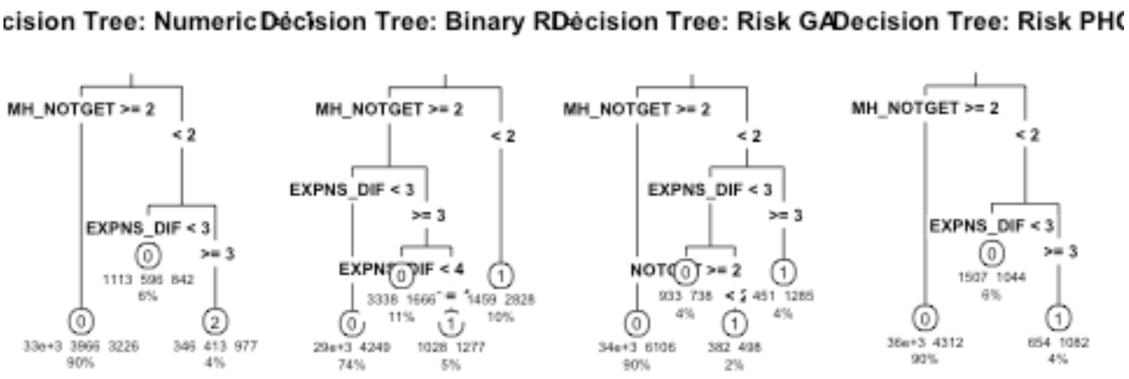
```
par(mfrow = c(1,2))

rpart.plot(tree_risk_phq1, main="Decision Tree: Risk PHQ", type=3, extra=101,
under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)
```

```
rpart.plot(tree_risk_phq2, main="Decision Tree: Risk PHQ", type=3, extra=101,
under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)
```



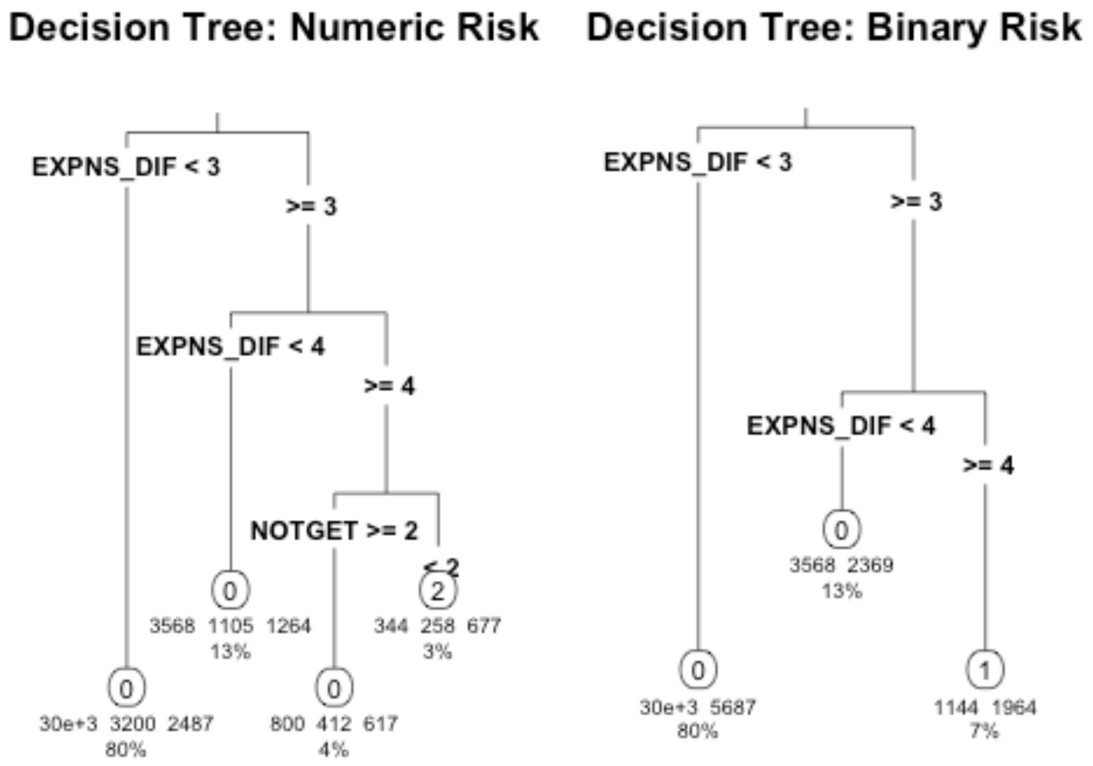
Plot the four decision trees including mental health variables

```
par(mfrow = c(2,4))

rpart.plot(tree_risk_numeric1, main="Decision Tree: Numeric Risk", type=3, ex
tra=101, under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)

rpart.plot(tree_risk_binary1, main="Decision Tree: Binary Risk", type=3, extr
a=101, under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)

rpart.plot(tree_risk_gad1, main="Decision Tree: Risk GAD", type=3, extra=101,
under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)

rpart.plot(tree_risk_phq1, main="Decision Tree: Risk PHQ", type=3, extra=101,
under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)
```



Plot the four decision trees excluding mental health variables

```
par(mfrow = c(1,2))

rpart.plot(tree_risk_numeric2, main="Decision Tree: Numeric Risk", type=3, ex
tra=101, under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)

rpart.plot(tree_risk_binary2, main="Decision Tree: Binary Risk", type=3, extr
a=101, under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)
```



```
par(mfrow = c(1,2))

rpart.plot(tree_risk_gad2, main="Decision Tree: Risk GAD", type=3, extra=101,
under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)

rpart.plot(tree_risk_phq2, main="Decision Tree: Risk PHQ", type=3, extra=101,
under=TRUE, fallen.leaves=TRUE, box.palette = NULL, cex=0.7)
```