Predictive Analysis of Costa Rican Household Poverty Index Based on Data Mining Classification Methods

Aaron Carr and Anusia Edward

Shiley-Marcos School of Engineering, University of San Diego

Abstract

The purpose of this study is to predict whether a Costa Rican household is suffering from poverty in order to correctly allocate funds to support struggling families. For this study it was hypothesized that at least one of the four classification techniques being utilized (Random Forest Decision Trees, CART Decision Tree, C5.0 Decision Tree, and *k*-nearest neighbor [KNN]) will result in a model that predicts an individual's poverty status index (PSI) based on demographic characteristics. More specifically, a secondary hypothesis is that the best predictor will be the Random Forest Decision Tree classifier, based on sensitivity being greater than 80%. The dataset was obtained from Kaggle. The predictor variables observed in this study include: access to cooking gas, access to trash truck facilities, number of dependents, access to sewage system, and total number of individuals living in a household. The study determined that the best model based on accuracy and precision was the Random Forest model, which had a precision of 74.1% and an accuracy of 73.9%, though its sensitivity was 46.6% and no other model was above 50%. Further research regarding the accuracy and sensitivity of the models could be evaluated using additional predictor variables, such as education level of individuals within a given household or location of household.

Keywords: poverty, Random Forest Decision Tree, CART Decision Tree, C5.0 Decision Tree, and K-Nearest Neighbor, classification modeling

Table of Contents

List of Tables

List of Figures

Introduction

Background and Practical Implications

Poverty is a global problem, with hundreds of millions of people being directly affected by it every year; according to the United Nations (n.d.), in 2015 10% of the entire world's population subsisted on less than $1.90 per day, which is the poverty line set by the World Bank (2015). Those whose income falls under the poverty threshold span a multitude of continents, nations, ethnicities, and religions, which makes coming up with ways to solve the issue of poverty very complex. A critical step in this process is developing robust and high-performing models to accurately predict what circumstances put individuals at risk for being impoverished. As poverty affects more than just the direct victims of it, addressing and overcoming it is of paramount importance to achieving a healthy global populace and economy. One such country that has been struggling with the effects of poverty is Costa Rica, located in Central America, where in 2020 the percent of population with an income below $5.50 per day was 13% (The World Bank, 2021).

Purpose, Objectives, and Hypothesis of Present Study

According to the Inter-American Development Bank (2018), the traditional method in Latin American countries for assessing whether an individual meets qualifications for assistance is the Proxy Means Test (PMT), which mainly uses household construction materials or assets as the predictor variables. However this test shows prediction performance loss as overall poverty proportion levels decrease. In order to provide Costa Rican economic relief agencies more stable and generalized predictive models–with a future research direction of eventually deploying them world-wide to countries in need, this study will use secondary data to perform data mining classification modeling aimed at predicting the poverty status index (PSI) of an individual based

on several features related to their living environment and family unit makeup. The general hypothesis of the study is that at least one of the four classification techniques being examined will result in a model that predicts an individual's PSI based on several characteristics that are broadly available regardless of where the data is being collected, e.g., Costa Rica, South America, Japan, the US, etc. More specifically, a secondary hypothesis is that the best predictor will be the random forest decision tree technique, focusing on sensitivity being above 80%.

Methods

Data Collection, Preprocessing, and Cleaning

This study is a retrospective data analysis based on the raw data which was sourced from an open online platform known as Kaggle. The raw data was initially obtained as an unzipped SSV file that was adjusted for processing using R in the R-Studio IDE. Initially, the dataset contained 141 different variables and one target variable which indicated whether or not a person was vulnerable to poverty. In order to reduce the dimensionality of the dataset the data was first examined for duplicate attributes. All duplicate attributes were located and removed leaving only one version of the attribute. Then the variables were categorized into the following groups: household demographics and commodities, house materials, basic necessities, and level of education. From here, irrelevant or repetitive groups were removed. The dataset was further reduced to 40 variables. Next the dataset was corrected for structural errors such as renaming the columns from the original code name to descriptive names. Additionally, all of the column names were translated to English, as they were all initially in Spanish (Inter-American Development Bank, 2018). Duplicates and null values were checked for within the dataset. Variables that had incorrectly recorded data, such as the variable of dependency, for example was recalculated using the corresponding columns of total_children and total_65. Then, age which is

a continuous variable was normalized. Finally, all variables were examined for outliers using their interquartile range.

Sample Characteristics

For the purpose of this study, the sample population being observed is Costa Rican households. The sample characteristics from this statistical study, which was determined based on Mallow's CP plots as seen in Appendix B, included the following: cooking_gas, trash_truck, dependency, sewer, and total_persons. Cooking_gas is a binary variable of zero and one which indicates whether a household uses gas (1) in order to cook their food or not (0). Trash_truck is also a binary variable of zero and one, which indicates whether a household's trash is removed via a trash truck (1) or not (0). The variable of dependency is determined using the number of individuals in a household who are below the age of 19 and over the age of 65. This is an important variable which indicates the number of people that are dependent on the heads of the household, as their income levels may be low or non-existent being that they are either too young to earn a steady income or retired. Sewer is a binary variable that shows whether or not a household has a sewer available for their waste (1) or not (0). The variable of total_persons is a discrete variable that indicates the number of people within a household. The target variable was converted to a PSI value, which is a binary variable where 0 indicates a household that is not vulnerable to being impoverished and 1 indicates a household that is vulnerable to being impoverished, moderately impoverished, or extremely impoverished. The final analytics base table (ABT) contained 9,557 instances.

Descriptive Statistics Associated with the Key Characteristics

After data pre-processing was completed, exploratory data analysis (EDA) was performed in order to examine and describe data characteristics, thereby providing important

glimpses into patterns within the data, which would then inform consequent predictive analytic efforts. Of the 40 features that remained, only seven were quantitative in nature. The descriptive statistics for each is included in Table 1, which include general measurements of location, such as mean ($\bar{x}$) and median, and dispersion, including standard deviation (SD) and range.

Table 1

Descriptive Statistics for Quantitative Variables

	total_persons	num_phones	num_children	num_65	num_adults	age	dependency
NA Count	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Mean	4.013	2.826	1.412	0.283	2.601	34.092	0.400
Median	4.000	3.000	1.000	0.000	2.000	31.000	0.400
Standard Deviation (SD)	1.766	1.486	1.368	0.597	1.173	21.728	0.254
Sample Variance	3.118	2.207	1.871	0.356	1.376	472.126	0.065
Range	12.000	10.000	9.000	3.000	9.000	97.000	1.000
Minimum	1.000	0.000	0.000	0.000	0.000	0.000	0.000
Maximum	13.000	10.000	9.000	3.000	9.000	97.000	1.000
IQR Outlier Limit	+/-3	+/-3	+/-3	+/-0	+/-1.5	+/-51	+/-0.476
25th Percentile	3.000	2.000	0.000	0.000	2.000	17.000	0.250
75th Percentile	5.000	4.000	2.000	0.000	3.000	51.000	0.567
Lower Outlier Threshold	0.000	-1.000	-3.000	0.000	0.500	-34.000	-0.226
Upper Outlier Threshold	8.000	7.000	5.000	0.000	4.500	102.000	1.044
% Outliers	3.3%	1.6%	2.7%	0.0%	6.1%	0.0%	0.0%
Mean (Outliers Excl.)	3.823	2.739	1.289	NA	2.409	34.092	0.400
Median (Outliers Excl.)	4.000	3.000	1.000	NA	2.000	31.000	0.400
SD (Outliers Excl.)	1.442	1.329	1.152	NA	0.875	21.728	0.254

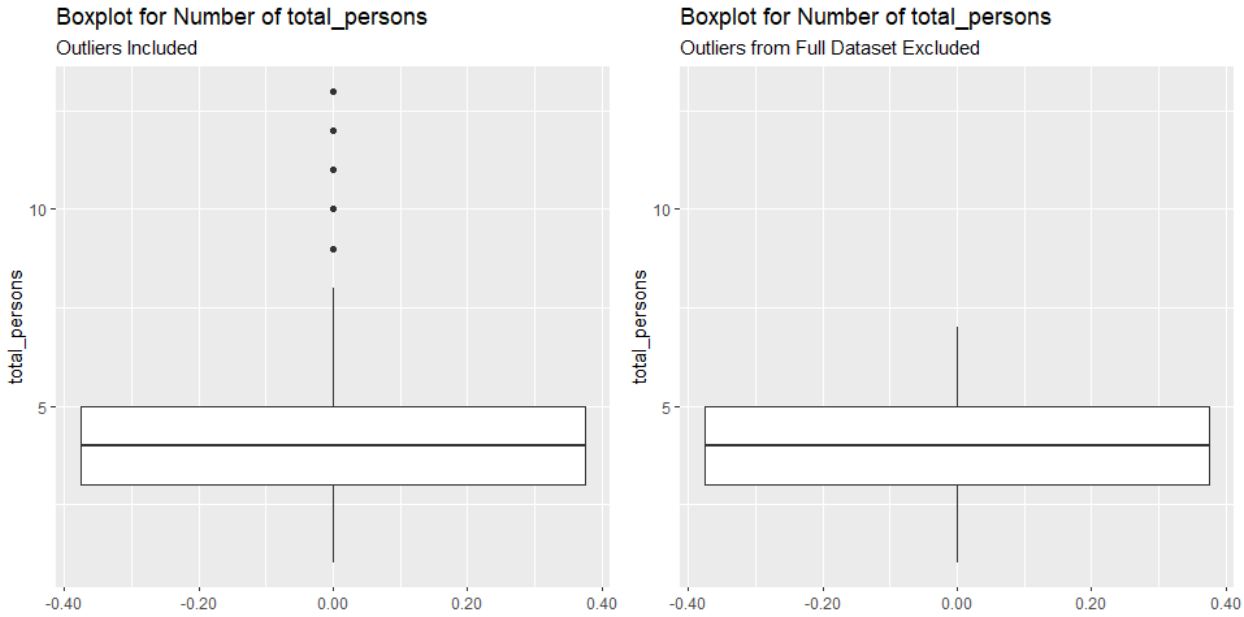
Note. n = 6,690

Also included in Table 1 are the descriptive statistics for the data set after exclusion of the outliers, assuming a threshold of 1.5 times the interquartile range (IQR). For the total_persons variable, the average number of people living in the surveyed house was 4.013, with a median of 4 and a SD of 1.766 people–obviously you cannot have a fraction of a person in reality, but the real number values represent what you would expect to observe from a probabilistic perspective. For total_persons, 3.3% of the sample (n = 224) were determined to be outliers, though $\bar{x}$ did not decrease very dramatically (3.823) and the median remained the same for the sample where they were removed. In fact, it should be noted that for every one of the seven quantitative variables, excluding the outliers did not change the median value, which is indicative of its more robust

nature as compared to mean. Figure 1 is the side-by-side comparison of boxplots for total_persons.

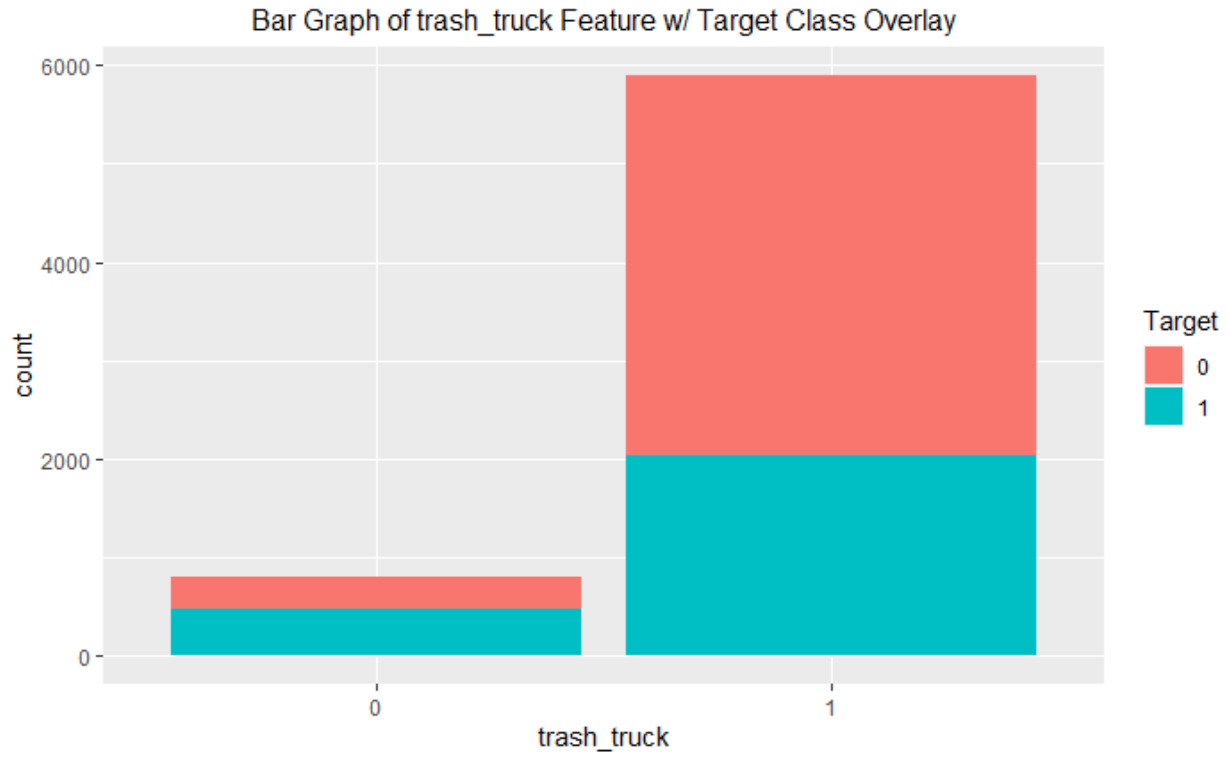
Figure 1

Boxplot Comparison for total_persons: Full Data Set vs. Data Set without Outliers



For each of the binary and categorical features, bar charts were created to visualize the frequency of each value, overlaid with the class feature, Target. Figure 2, which is the bar chart for trash_track, show that the majority of households do have their trash hauled away via truck, but there is a fair mix of both target PSI classes (0 = non-vulnerable, 1 = vulnerable, moderate, extreme) with each trash_truck value.

Figure 2



Additional EDA steps included generating a scatterplot matrix, correlation matrix, and correlation plot, as can be seen in Appendix A. The scatterplot matrix showed that very few numerical features appear to have anything close to a linear relationship; one exception was total_persons and num_children, which intuitively makes sense that more children mean more people in the household. When referencing the correlation matrix, the coefficient is very high for those two variables (r = .748), indicating a very strong positive linear relationship, though when viewing the individual scatterplot (as seen in Appendix B), it is less definite based on more points above the true linear regression line and the presence of a few outliers that may be having an outsized impact.

Modeling

A total of four different models were constructed after splitting the data into a training set and a test set. A stratified 70/30 split was carried out in order to obtain 70% of the data (n = 6,690) in the training set and 30% of the data (n = 2,867) in the test set. The splitting was done through a stratification method in order to ensure that the proportions of target variable outcomes were proportional within the sets (Appendix B).

Model 1 (M_1): Random Forest Decision Trees

A random forest model was created using cooking_gas, trash_truck, dependency, sewer, and total_persons as the predictor variables, and the binary Target variable as the outcome variable. The accuracy for the Random Forest model was found to be 73.9%. The evaluation metrics for this model can be seen in Table 2.

Model 2 (M_2): CART Decision Tree

A CART Decision Tree was created using cooking_gas, trash_truck, dependency, sewer, and total_persons as the predictor variables, and the binary Target variable as the outcome variable. The accuracy for the CART Decision Tree model was found to be 70.7%. The evaluation metrics for this model can be seen in Table 2.

Model 3 (M_3): C5.0 Decision Tree

A C5.0 Decision Tree model was created using cooking_gas, trash_truck, dependency, sewer, and total_persons as the predictor variables, and the binary Target variable as the outcome variable. The accuracy for the C5.0 Decision Tree model was found to be 72.2%. The evaluation metrics for this model can be seen in Table 2.

Model 4 (M_4): *K*-Nearest Neighbor (KNN), k = 81

A KNN-81 model was created using cooking_gas, trash_truck, dependency, sewer, and total_persons as the predictor variables and the binary Target variable as the outcome variable. The accuracy for this KNN model was found to be 70.9%. The evaluation metrics for this model can be seen in Table 2.

Model 5 (M_5): KNN, k = 3

A KNN-3 model was created using cooking_gas, trash_truck, dependency, sewer, and total_persons as the predictor variables, and the binary Target variable as the outcome variable. The accuracy for this KNN model was found to be 73.2%. The evaluation metrics for this model can be seen in Table 2.

Table 2

Summary of the Evaluation Metrics for Each Model

Model	Classifier	Accuracy	Sensitivity	Precision	F_1
M_1	Random Forest	.739	.466	.741	.572
M_2	CART	.707	.464	.655	.543
M_3	C5.0	.722	.497	.677	.573
M_4	KNN-81	.709	.375	.712	.491
M_5	KNN-3	.732	.490	.705	.578

Results

In order to determine which of the models is most efficient in using classification techniques to predict which households are vulnerable to poverty, the following metrics were

considered: accuracy, sensitivity, precision, and F_1 score. The most accurate model out of the five M_1, which used the Random Forest decision tree ensemble technique. It had an accuracy of 73.9%. The least accurate model was M_2, which was based on the CART decision tree classifier (70.7%)–note, both of these levels are above the test data set's all-negative baseline accuracy of 62.5%. The order of the classifiers with the highest level of sensitivity to the least sensitive model is as follows: C5.0 (49.7%), KNN-3 (49.0%), Random Forest (46.6%), CART (46.4%), and KNN-81 (37.5%). The most precise model was found to be M_1 (74.1%), whereas the least precise model was M_2 (65.5%). The model with the highest F_1 value was found to be M_5 (57.8%), while the model with the lowest F_1 value was M_4 (49.1%). The results are summarized in Table 2.

Additional measures were obtained for the KNN classifiers based on the fact that they can generate a probabilistic threshold score, from which a receiver operator characteristic (ROC) curve could be plotted and the area under the curve (AUC) calculated. Figures 5 and 6 are the ROC plots for M_4 (k = 81) and M_5 (k = 3), which had an AUC of .642 and .570, respectively.

Discussion

The purpose of this study was to use classification techniques to predict an individual's PSI based on several characteristics, including: whether they have access to cooking gas; whether their trash is removed via truck; number of dependents in the household; whether they have access to a sewage system; and total number of individuals living in a household. The sensitivity and precision measurements were used as the main evaluation metrics, as the overarching purpose of this study was to maximize identification of the households that are vulnerable to poverty in order to ensure that they receive the aid they need–sensitivity indicates how well the models perform in terms of correctly predicting households who are vulnerable as

vulnerable and precision relays the confidence of how well the people predicted as vulnerable or worse are actually so. The Random Forest classifier (M_1) performed the best in terms of accuracy (73.9%) and precision (74.1%), though the KNN-3 (M_5) and C5.0 (M_3) classifiers had somewhat higher sensitivities (49.0% and 49.7%, respectively). It is no surprise that M_1 was closest to the top on all measures, since the Random Forest classifier is an ensemble method that relies on aggregating results generated by running a series of decorrelated decision trees that in effect reduces prediction variance.

Though every model out-performed the all-negative baseline accuracy (62.5%), it cannot be said that any of them was overwhelmingly effective at predicting the correct PSI value. M_1 only had a sensitivity of 46.6%, which means that it performed very poorly (worse than random guessing) at correctly predicting those who were vulnerable or worse (PSI = 1). When looking at the ROC and AUC for both M_4 and M_5, it is clear that neither is significantly good, with KNN-3 being only slightly better than random. This conclusion indicates that the general hypothesis for this study was achieved, as a classification technique was able to be employed to predict an individual's PSI. For reference, this study hypothesized that at least one of the four classification techniques being utilized will result in a model that predicts an individual's PSI based on several characteristics that are broadly available regardless of where the data is being collected. However, the study was unable to meet the further specified hypothesis of achieving over 80% sensitivity with the final model.

In the future, this study could look at refining the models with the most potential in order to increase the overall sensitivity and precision of the model; specifically for the KNN models, because they are probabilistic, the threshold cutoff for assigning positive (PSI =1) versus negative (PSI = 0) could be adjusted in combination with trying other values for k. Other data

mining algorithms could also be explored, such as logistic regression. Additionally, future research could expand on this study by incorporating features such as the level of education obtained by the head of the household or location of the household in order to determine if such attributes can contribute to determining the poverty level of a given household.

References

Inter-American Development Bank. (2018). Costa Rican household poverty level prediction. *Kaggle*.https://www.kaggle.com/competitions/costa-rican-household-poverty-prediction/overview/description

The World Bank. (2015, September 30). *FAQs: Global poverty line update*. https://www.worldbank.org/en/topic/poverty/brief/global-poverty-line-faq

The World Bank. (2021, October 6). *The World Bank in Costa Rica*. https://www.worldbank.org/en/country/costarica/overview#1

United Nations. (n.d.). *Ending poverty*. https://www.un.org/en/global-issues/ending-poverty

Appendix A

Table 3

Correlation Matrix

	total_persons	num_phones	num_children	num_65	num_adults	dependency	age
total_persons	1.000	0.530	0.748	-0.063	0.633	0.199	-0.319
num_phones	0.530	1.000	0.220	-0.139	0.542	-0.154	-0.157
num_children	0.748	0.220	1.000	-0.274	-0.040	0.517	-0.469
num_65	-0.063	-0.139	-0.274	1.000	0.224	0.425	0.468
num_adults	0.633	0.542	-0.040	0.224	1.000	-0.304	0.068
dependency	0.199	-0.154	0.517	0.425	-0.304	1.000	-0.029
age	-0.319	-0.157	-0.469	0.468	0.068	-0.029	1.000

Figure 3

Correlation Plot

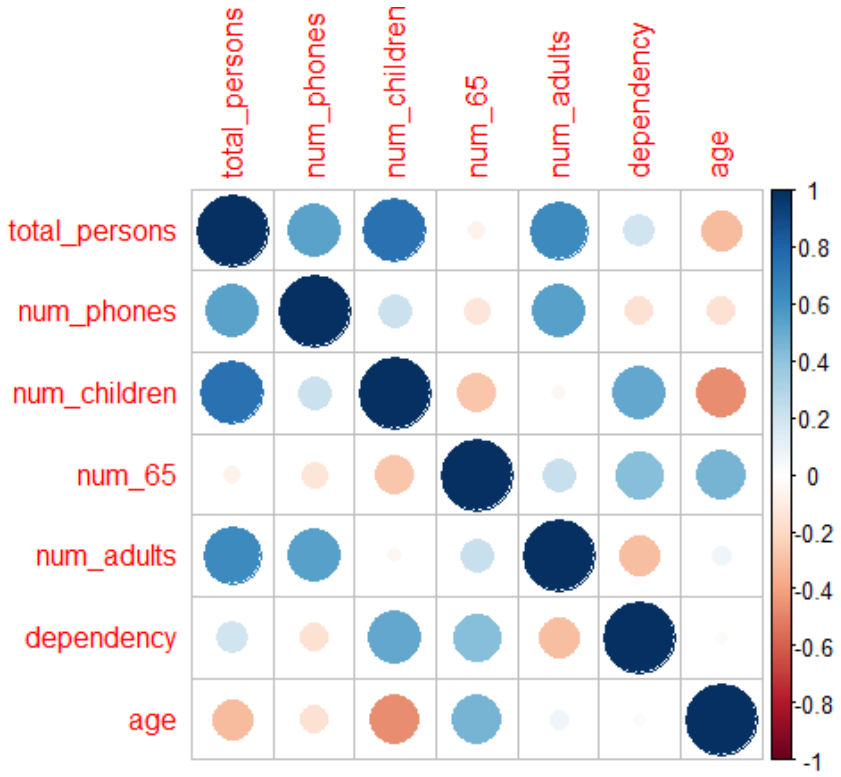


Figure 4

Scatterplot Matrix

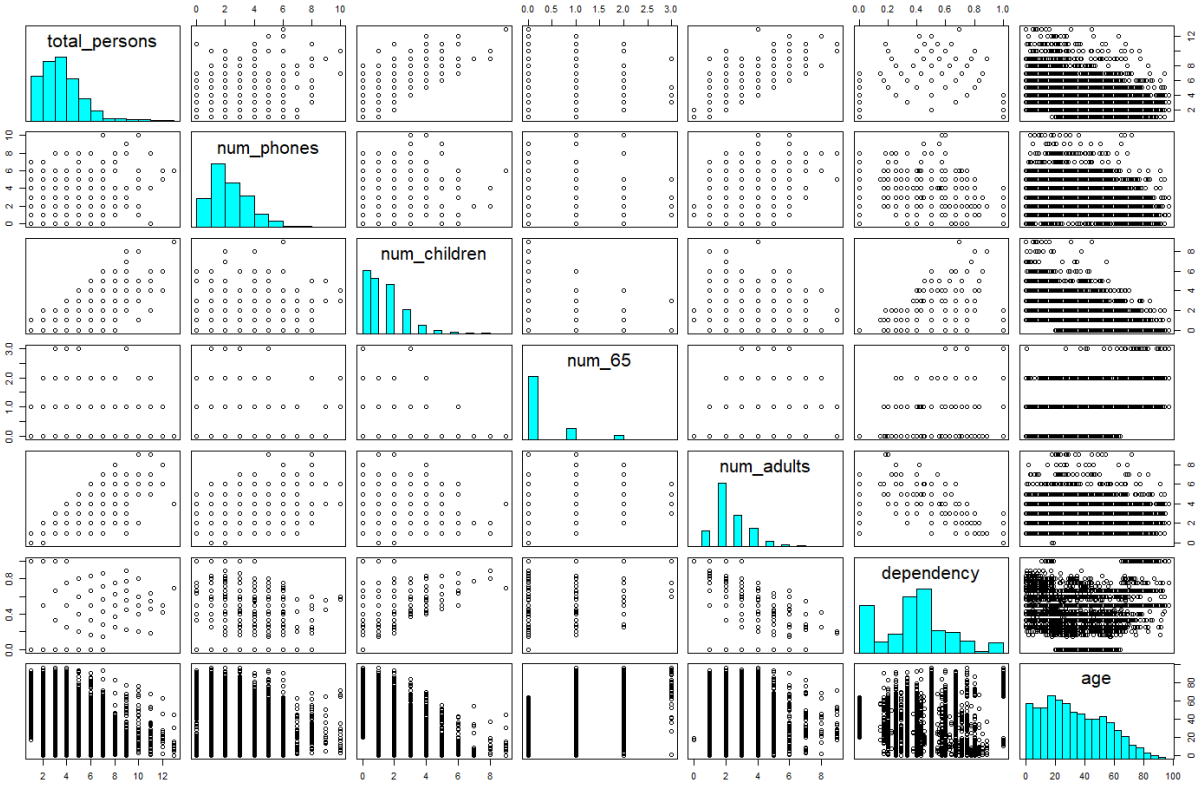


Figure 5

Receiver Operator Characteristic (ROC) Curve for KNN-81

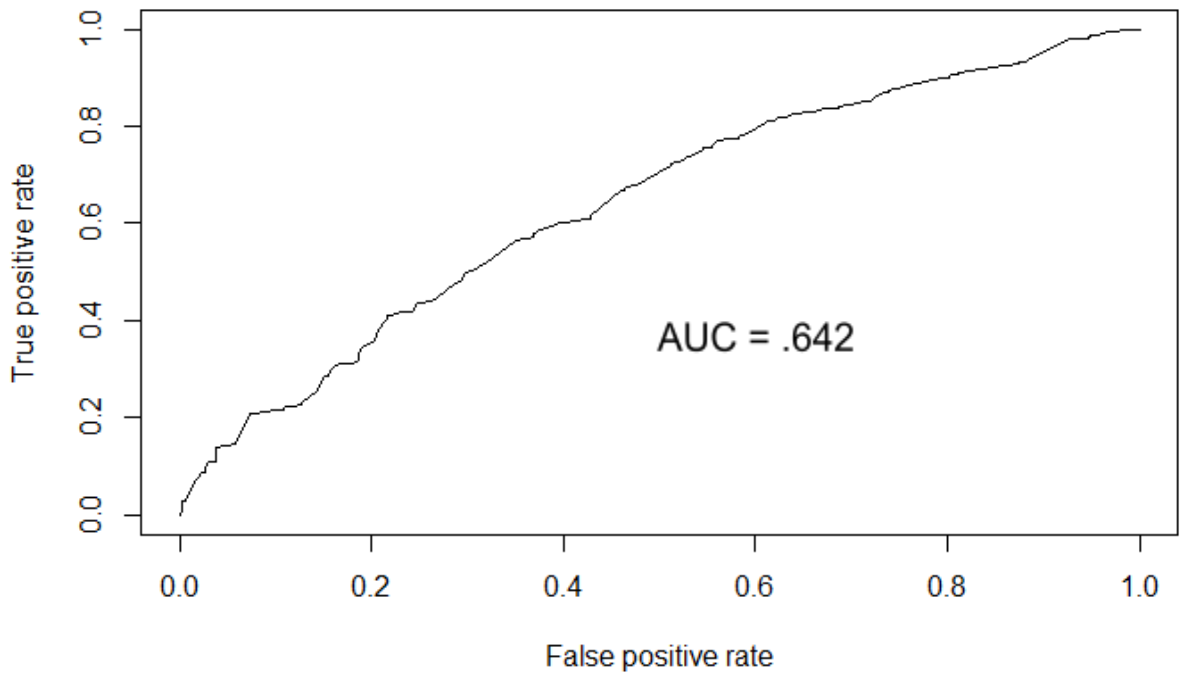
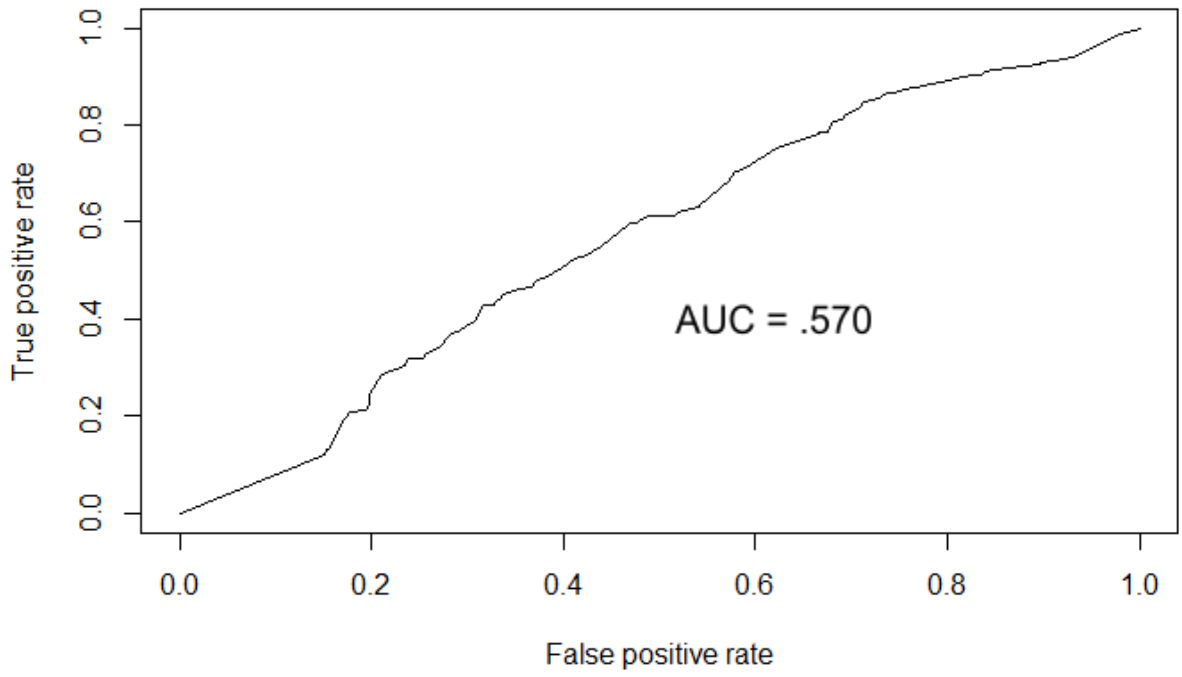


Figure 6

ROC Curve for KNN-3


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC = .570
False positive rate

Appendix B

502 Final Project - Data Cleaning

```
# loading the data set
crp_train <- read.csv("~/Desktop/train.csv")
```

```
# removal of irrelevant or repetitive data
crp_train1 <- subset(crp_train,
                     select = c(Id, v14a,refrig,
                                hogar_total,computer,
                                qmobilephone,
                                dis, estadocivil3,
                                estadocivil4,estadocivil5,
                                estadocivil6, estadocivil7,
                                hogar_nin,hogar_mayor,
                                hogar_adul,
                                dependency, abastaguadentro,
                                abastaguafuera,abastaguano,
                                public, planpri, noelec,
                                coopele, sanitario1,
                                sanitario2, sanitario3,
                                sanitario5, sanitario6,
                                energcocinar1,
                                energcocinar2,
                                energcocinar3,
                                energcocinar4,
                                elimbasu1, elimbasu2,
                                elimbasu3, elimbasu4,
                                elimbasu5, age, Target))
head(crp_train1)
```

	Id <chr>	v14a <int>	refrig <int>	hogar_total <int>	computer <int>	qmobilephone <int>	dis <int>
1	ID_279628684	1	1	1	0	1	0
2	ID_f29eb3ddd	1	1	1	0	1	0
3	ID_68de51c94	1	1	1	0	0	1
4	ID_d671db89c	1	1	4	0	3	0
5	ID_d56d6f5f5	1	1	4	0	3	0
6	ID_ec05b1a7b	1	1	4	0	3	0

6 rows | 1-8 of 40 columns

```
# fixing structural errors (renaming columns-rmv codes + translating to eng)
names(crp_train1) <- c('id', 'has_bathroom', 'has_refrig', 'total_persons',
                       'has_comp', 'num_phones', 'disabled', 'married',
                       'divorced', 'separated', 'widower', 'single',
                       'num_children', 'num_65', 'num_adults', 'dependency',
                       'water_inside', 'water_outside', 'no_water', 'gov_elec',
                       'private_elec', 'no_elec', 'coop_elec', 'no_toilet',
                       'sewer', 'septictank', 'latrine', 'toilet_other',
                       'no_cooking_energy', 'cooking_ elec', 'cooking_gas',
                       'cooking_woodcoal', 'trash_truck', 'trash_buried',
                       'trash_burning', 'trash_throwing', 'trash_river',
                       'age', 'Target')
```

```
head(crp_train1)
```

id <chr>	has_bathroom <int>	has_refrig <int>	total_persons <int>	has_comp <int>	num_phones <int>
1 ID_279628684	1	1	1	0	1
2 ID_f29eb3ddd	1	1	1	0	1
3 ID_68de51c94	1	1	1	0	0
4 ID_d671db89c	1	1	4	0	3
5 ID_d56d6f5f5	1	1	4	0	3
6 ID_ec05b1a7b	1	1	4	0	3

6 rows | 1-7 of 40 columns

```
# checking for duplicates within the data using ID
any(duplicated(crp_train1$id))
```

```
## [1] FALSE
```

```
# checking if there are any blanks or missing data
sum(crp_train1=="")
```

```
## [1] 0
```

```
sum(crp_train1=="NA")
```

```
## [1] 0
```

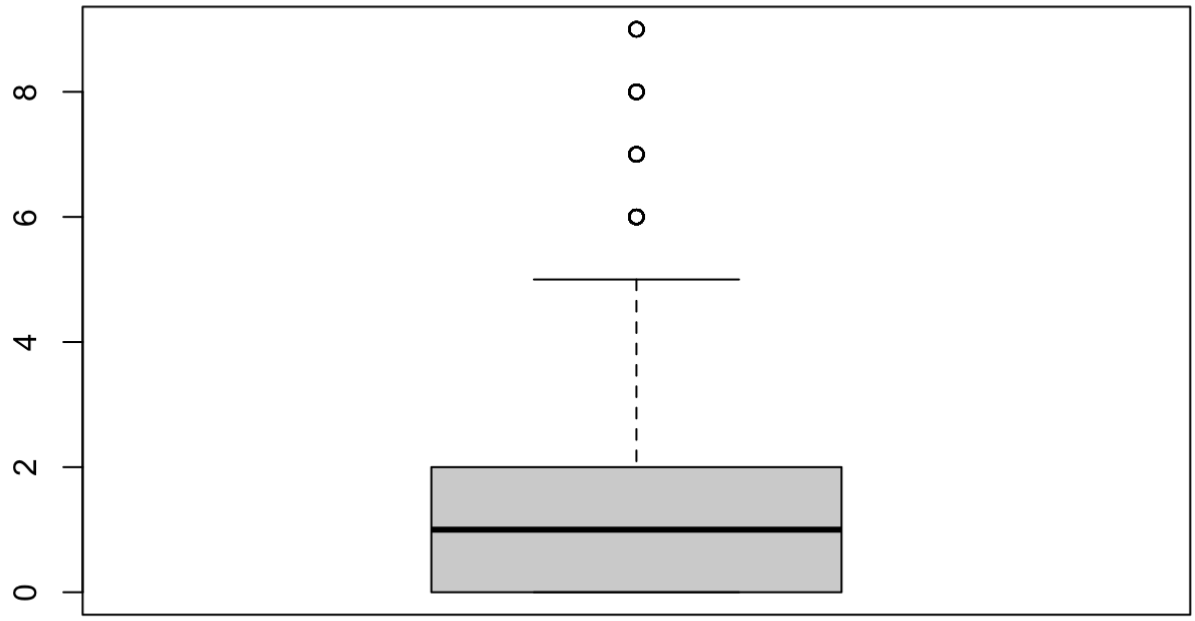
```
# checking that the max values for number of children and adults over
# 65 makes sense or if there are any outliers
max(crp_train1$num_children)
```

```
## [1] 9
```

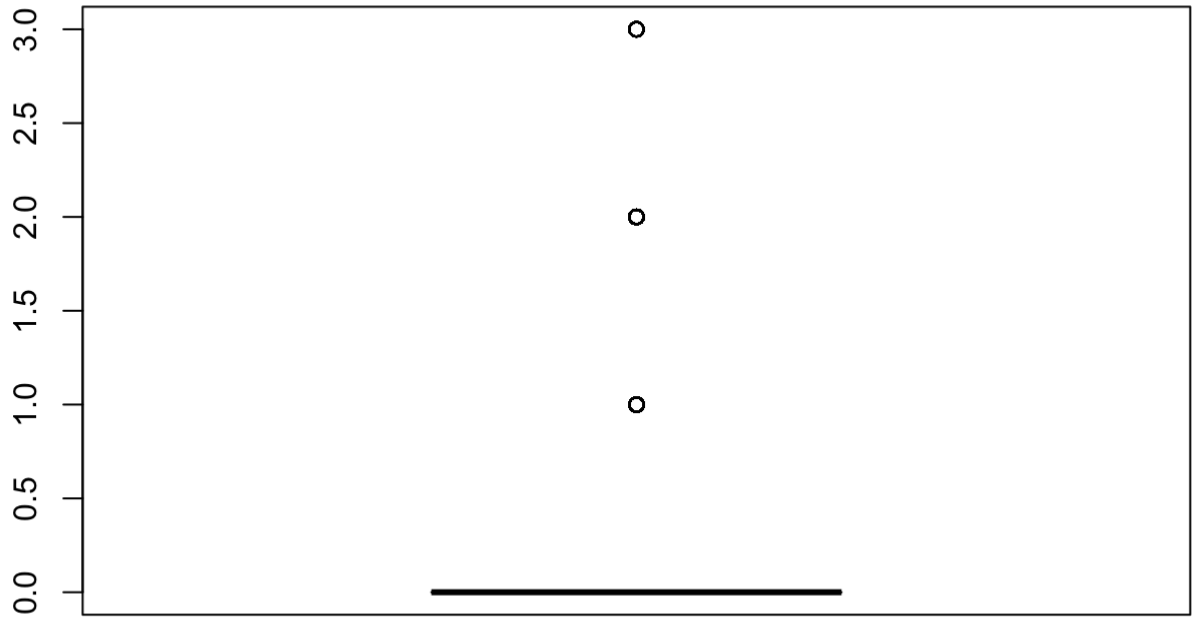
```
max(crp_train1$num_65)
```

```
## [1] 3
```

```
boxplot(crp_train1$num_children)
```




```
boxplot(crp_train1$num_65)
```




```
# The outlier values were retained for both the variables of the number of children with
in a household and the number of adults over the age of 65 within one household because
 they could be possible indicators of the differences in poverty level. Additionally, th
e outlier values are values that are realistic. For example, three people over the age o
f 65 within one household is a realistic situation considering that there can be more th
an one family under one household due to economic situations. Additionally, having a max
imum of nine children under the age of 19 within one household, is realistic as well con
sidering that there can be again a joint family situation. This is further explored with
in the EDA.
```

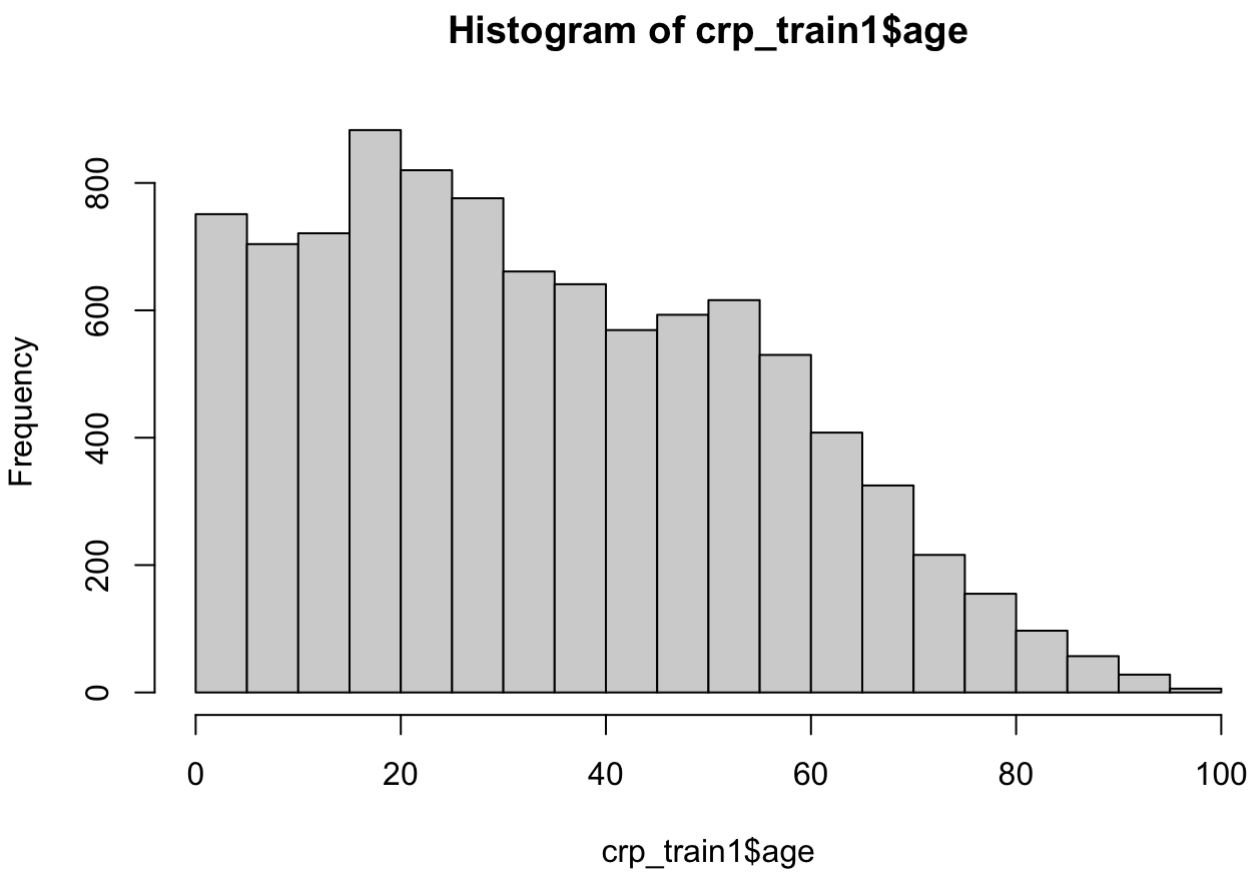
```
# fixing the yes/no responses in dependency rate
# dependency rate = (num_childre + num_65)/ total_persons
crp_train1$dependency <- (crp_train1$num_children +
                            crp_train1$num_65)/crp_train1$total_persons

head(crp_train1)
```

id <chr>	has_bathroom <int>	has_refrig <int>	total_persons <int>	has_comp <int>	num_phones <int>
1 ID_279628684	1	1	1	0	1
2 ID_f29eb3ddd	1	1	1	0	1
3 ID_68de51c94	1	1	1	0	0

	id <chr>	has_bathroom <int>	has_refrig <int>	total_persons <int>	has_comp <int>	num_phones <int>
4	ID_d671db89c	1	1	4	0	3
5	ID_d56d6f5f5	1	1	4	0	3
6	ID_ec05b1a7b	1	1	4	0	3

6 rows | 1-7 of 40 columns

```
# Histogram of Age
hist(crp_train1$age)
```




```
# The value is right skewed so the values should be normalized and the outliers
#should be dropped as when looking at the historm it indicates that the age of
#the person filling out the survey for this household is close to one hundred
#which isn't impossible, but highly unlikely.
```

```
# standardizing the numeric field of Age and adding it as an additional column
# for use in future models
crp_train1$age_z <- scale(x = crp_train1$age)
head(crp_train1)
```

	id	has_bathroom	has_refrig	total_persons	has_comp	num_phones
	<chr>	<int>	<int>	<int>	<int>	<int>
1	ID_279628684	1	1	1	0	1
2	ID_f29eb3ddd	1	1	1	0	1
3	ID_68de51c94	1	1	1	0	0
4	ID_d671db89c	1	1	4	0	3
5	ID_d56d6f5f5	1	1	4	0	3
6	ID_ec05b1a7b	1	1	4	0	3

6 rows | 1-7 of 41 columns

```
# Drop age outliers using IQR
Q1 <- quantile(crp_train1$age, .25)
Q3 <- quantile(crp_train1$age, .75)
IQR <- IQR(crp_train1$age)

crp_1 <- subset(crp_train1, crp_train1$age> (Q1 - 1.5*IQR) |
                  crp_train1$age< (Q3 + 1.5*IQR))
head(crp_1)
```

	id	has_bathroom	has_refrig	total_persons	has_comp	num_phones
	<chr>	<int>	<int>	<int>	<int>	<int>
1	ID_279628684	1	1	1	0	1
2	ID_f29eb3ddd	1	1	1	0	1
3	ID_68de51c94	1	1	1	0	0
4	ID_d671db89c	1	1	4	0	3
5	ID_d56d6f5f5	1	1	4	0	3
6	ID_ec05b1a7b	1	1	4	0	3

6 rows | 1-7 of 41 columns

```
crp_train1$Target[crp_train1$Target == 2] <- 1
crp_train1$Target[crp_train1$Target == 3] <- 1
crp_train1$Target[crp_train1$Target == 4] <- 0
```

```
# exporting csv of cleaned train
write.csv(crp_train1,
          "/Users/anusiaedward/Desktop/502_Final_Train1.csv",
          row.names=TRUE)
```

```
# loading in the test set
crp_test <- read.csv("~/Desktop/test.csv")
```

```
# removal of irrelevant data
crp_test1 <- subset(crp_test,
                    select = c(Id, v14a,refrig,
                               hogar_total,computer,
                               qmobilephone,
                               dis, estadocivil3,
                               estadocivil4,estadocivil5,
                               estadocivil6, estadocivil7,
                               hogar_nin,hogar_mayor,
                               hogar_adul,
                               dependency, abastaguadentro,
                               abastaguafuera,abastaguano,
                               public, planpri, noelec,
                               coopele, sanitario1,
                               sanitario2, sanitario3,
                               sanitario5, sanitario6,
                               energcocinar1,
                               energcocinar2,
                               energcocinar3,
                               energcocinar4,
                               elimbasu1, elimbasu2,
                               elimbasu3, elimbasu4,
                               elimbasu5))
head(crp_test1)
```

	Id <chr>	v14a <int>	refrig <int>	hogar_total <int>	computer <int>	qmobilephone <int>	dis <int>
1	ID_2f6873615	1	1	3	1	2	0
2	ID_1c78846d2	1	1	3	1	2	0
3	ID_e5442cf6a	1	1	3	1	2	0
4	ID_a8db26a79	1	1	1	1	2	0
5	ID_a62966799	1	1	1	0	1	0
6	ID_e77d38d45	1	1	2	1	1	0

6 rows | 1-8 of 38 columns

```
# fixing structural errors (renaming columns-rmv codes + changing to eng)
names(crp_test1) <- c('id', 'has_bathroom', 'has_refrig', 'total_persons', 'has_comp',
'num_phones', 'disabled', 'married', 'divorced', 'separated', 'widower', 'single', 'num_
children', 'num_65', 'num_adults', 'dependency', 'water_inside', 'water_outside', 'no_wa
ter', 'gov_elec', 'private_elec', 'no_elec', 'coop_elec', 'no_toilet', 'sewer', 'septict
ank', 'latrine', 'toilet_other', 'no_cooking_energy', 'cooking_ elec', 'cooking_gas', 'c
ooking_woodcoal', 'trash_truck', 'trash_buried', 'trash_burning', 'trash_throwing', 'tra
sh_river')

head(crp_test1)
```

id <chr>	has_bathroom <int>	has_refrig <int>	total_persons <int>	has_comp <int>	num_phones <int>
1 ID_2f6873615	1	1	3	1	2
2 ID_1c78846d2	1	1	3	1	2
3 ID_e5442cf6a	1	1	3	1	2
4 ID_a8db26a79	1	1	1	1	2
5 ID_a62966799	1	1	1	0	1
6 ID_e77d38d45	1	1	2	1	1

6 rows | 1-7 of 38 columns

```
# checking for duplicates using ID
any(duplicated(crp_test1$id))
```

```
## [1] FALSE
```

```
# checking if there are any blanks or missing data
sum(crp_test1=="")
```

```
## [1] 0
```

```
sum(crp_test1=="NA")
```

```
## [1] 0
```

```
# checking that the max values for number of children and adults over
# 65 makes sense or if there are any outliers
max(crp_test1$num_children)
```

```
## [1] 10
```

```
max(crp_test1$num_65)
```

```
## [1] 5
```

```
# fixing the yes/no responses in dependency rate
# dependency rate = (num_childre + num_65)/ total_persons
crp_test1$dependency <- (crp_test1$num_children +
                           crp_test1$num_65)/crp_test1$total_persons
```

```
# exporting csv of cleaned train set
write.csv(crp_test1,
          "~/Desktop/502_Final_Test1.csv",
          row.names=TRUE)
```

ADS502-02-SP22 - Final Project

Carr_Aaron

04/18/2022

Final Project: Costa Rica Poverty Index Prediction (Using R)

Data set up

Set global knit options

```
knitr::opts_chunk$set(
  fig.align = 'center'
  )
```

Load R libraries for global use & set seed

```
library(dplyr)
```

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

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

```
library(ggplot2)
library(datasets)
library(scales)
library(C50)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

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

```
library(e1071)
library(BioStatR)
library(rpart)
library(rpart.plot)
library(nnet)
```

```
library(NeuralNetTools)
library(randomForest)
```

```
## randomForest 4.7-1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

```
library(fastDummies)
library(caret)
```

```
## Loading required package: lattice
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

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

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:car':
##
##     logit
```

```
## The following object is masked from 'package:randomForest':
##
##     outlier
```

```
## The following objects are masked from 'package:scales':
##
##     alpha, rescale
```

```
## The following objects are masked from 'package:ggplot2':
##
```

```
##     %+%, alpha
```

```
library(class)
library (ROCR)

set.seed(333)
```

Load files into R dataframes

```
# Read training data file into R df
train_df01 <- read.table(file = "502_Final_Train.csv", header = TRUE, sep = ",")

# Review first few rows
print(head(train_df01))
```

```
##   X.1 X           id has_bathroom has_refrig total_persons has_comp num_phones
## 1   1 1 ID_279628684            1          1             1        0          1
## 2   2 2 ID_f29eb3ddd            1          1             1        0          1
## 3   5 5 ID_d56d6f5f5            1          1             4        0          3
## 4   7 7 ID_e9e0c1100            1          1             4        0          3
## 5   8 8 ID_3e04e571e            1          1             4        0          1
## 6   9 9 ID_1284f8aad            1          1             4        0          1
##   disabled married divorced separated widower single num_children num_65
## 1        0       0        1         0       0      0            0      0
## 2        0       0        1         0       0      0            0      1
## 3        0       0        0         0       0      0            2      0
## 4        0       0        0         0       0      0            2      0
## 5        0       0        0         0       0      0            2      0
## 6        0       0        0         0       0      0            2      0
##   num_adults dependency water_inside water_outside no_water gov_elec
## 1          1        0.0            1             0        0        1
## 2          1        1.0            1             0        0        1
## 3          2        0.5            1             0        0        1
## 4          2        0.5            1             0        0        1
## 5          2        0.5            1             0        0        1
## 6          2        0.5            1             0        0        1
##   private_elec no_elec coop_elec no_toilet sewer septictank latrine
## 1            0       0         0         0     1          0       0
## 2            0       0         0         0     1          0       0
## 3            0       0         0         0     1          0       0
## 4            0       0         0         0     1          0       0
## 5            0       0         0         0     1          0       0
## 6            0       0         0         0     1          0       0
##   toilet_other no_cooking_energy cooking_.elec cooking_gas cooking_woodcoal
## 1            0                 0            0           1               0
## 2            0                 0            1           0               0
## 3            0                 0            1           0               0
## 4            0                 0            1           0               0
## 5            0                 0            0           1               0
## 6            0                 0            0           1               0
##   trash_truck trash_buried trash_burning trash_throwing trash_river age Target
## 1           1            0             0              0           0  43      0
## 2           1            0             0              0           0  67      0
## 3           1            0             0              0           0  37      0
```

```
## 4           1            0             0              0           0   8      0
## 5           1            0             0              0           0   7      0
## 6           1            0             0              0           0  30      0
##        age_z
## 1  0.4023851
## 2  1.5128659
## 3  0.1247650
## 4 -1.2170660
## 5 -1.2633360
## 6 -0.1991253
```

```
# Check df length
dim(train_df01)[1]
```

```
## [1] 6690
```

```
# Check training data set features for null values
sapply(train_df01, function(x) sum(is.na(x)))
```

```
##               X.1                 X                id      has_bathroom
##                 0                 0                 0                 0
##        has_refrig     total_persons          has_comp        num_phones
##                 0                 0                 0                 0
##          disabled           married          divorced         separated
##                 0                 0                 0                 0
##           widower            single      num_children            num_65
##                 0                 0                 0                 0
##        num_adults        dependency      water_inside     water_outside
##                 0                 0                 0                 0
##          no_water          gov_elec      private_elec           no_elec
##                 0                 0                 0                 0
##         coop_elec         no_toilet             sewer        septictank
##                 0                 0                 0                 0
##           latrine      toilet_other no_cooking_energy     cooking_.elec
##                 0                 0                 0                 0
##       cooking_gas  cooking_woodcoal       trash_truck      trash_buried
##                 0                 0                 0                 0
##     trash_burning    trash_throwing       trash_river               age
##                 0                 0                 0                 0
##            Target             age_z
##                 0                 0
```

```
# Read test data file into R df
test_df01 <- read.table(file = "502_Final_Test.csv", header = TRUE, sep = ",")

# Review first few rows
print(head(test_df01))
```

```
##   X.1  X           id has_bathroom has_refrig total_persons has_comp num_phones
## 1   3  3 ID_68de51c94            1          1             1        0          0
## 2   4  4 ID_d671db89c            1          1             4        0          3
## 3   6  6 ID_ec05b1a7b            1          1             4        0          3
## 4  16 16 ID_0a39e419e            1          1             4        0          1
## 5  19 19 ID_c51938edf            1          1             4        0          1
## 6  23 23 ID_a0bff0ba7            1          1             2        0          1
```

```
##   disabled married divorced separated widower single num_children num_65
## 1        1       0        0         0       1      0            0      1
## 2        0       0        0         0       0      1            2      0
## 3        0       0        0         0       0      0            2      0
## 4        0       0        0         1       0      0            2      1
## 5        0       0        0         0       0      0            2      1
## 6        0       1        0         0       0      0            0      2
##   num_adults dependency water_inside water_outside no_water gov_elec
## 1          1       1.00            1             0        0        1
## 2          2       0.50            1             0        0        1
## 3          2       0.50            1             0        0        1
## 4          2       0.75            1             0        0        1
## 5          2       0.75            1             0        0        1
## 6          2       1.00            1             0        0        1
##   private_elec no_elec coop_elec no_toilet sewer septictank latrine
## 1            0       0         0         0     1          0       0
## 2            0       0         0         0     1          0       0
## 3            0       0         0         0     1          0       0
## 4            0       0         0         0     1          0       0
## 5            0       0         0         0     1          0       0
## 6            0       0         0         0     1          0       0
##   toilet_other no_cooking_energy cooking_.elec cooking_gas cooking_woodcoal
## 1            0                 0            1           0                0
## 2            0                 0            1           0                0
## 3            0                 0            1           0                0
## 4            0                 0            1           0                0
## 5            0                 0            1           0                0
## 6            0                 0            1           0                0
##   trash_truck trash_buried trash_burning trash_throwing trash_river age Target
## 1           1            0             0              0           0  92      0
## 2           1            0             0              0           0  17      0
## 3           1            0             0              0           0  38      0
## 4           1            0             0              0           0  19      0
## 5           1            0             0              0           0  50      0
## 6           1            0             0              0           0  66      0
##        age_z
## 1  2.6696167
## 2 -0.8006357
## 3  0.1710350
## 4 -0.7080956
## 5  0.7262754
## 6  1.4665959
```

```
# Check df length
dim(test_df01)[1]
```

```
## [1] 2867
```

```
# Check test data set features for null values
sapply(test_df01, function(x) sum(is.na(x)))
```

```
##               X.1                 X                id      has_bathroom
##                 0                 0                 0                 0
##        has_refrig     total_persons          has_comp        num_phones
##                 0                 0                 0                 0
```

```
##          disabled           married          divorced         separated
##                 0                 0                 0                 0
##           widower            single      num_children            num_65
##                 0                 0                 0                 0
##        num_adults        dependency      water_inside     water_outside
##                 0                 0                 0                 0
##          no_water          gov_elec      private_elec           no_elec
##                 0                 0                 0                 0
##         coop_elec         no_toilet             sewer        septictank
##                 0                 0                 0                 0
##           latrine      toilet_other no_cooking_energy     cooking_.elec
##                 0                 0                 0                 0
##       cooking_gas  cooking_woodcoal       trash_truck      trash_buried
##                 0                 0                 0                 0
##     trash_burning    trash_throwing       trash_river               age
##                 0                 0                 0                 0
##            Target             age_z
##                 0                 0
```

Factorize training binary/categorical data

```
train_df01$Target <- as.factor(train_df01$Target)
train_df01$has_bathroom <- as.factor(train_df01$has_bathroom)
train_df01$has_refrig <- as.factor(train_df01$has_refrig)
train_df01$has_comp <- as.factor(train_df01$has_comp)
train_df01$disabled <- as.factor(train_df01$disabled)
train_df01$married <- as.factor(train_df01$married)
train_df01$divorced <- as.factor(train_df01$divorced)
train_df01$separated <- as.factor(train_df01$separated)
train_df01$widower <- as.factor(train_df01$widower)
train_df01$single <- as.factor(train_df01$single)
train_df01$water_inside <- as.factor(train_df01$water_inside)
train_df01$water_outside <- as.factor(train_df01$water_outside)
train_df01$no_water <- as.factor(train_df01$no_water)
train_df01$gov_elec <- as.factor(train_df01$gov_elec)
train_df01$private_elec <- as.factor(train_df01$private_elec)
train_df01$no_elec <- as.factor(train_df01$no_elec)
train_df01$coop_elec <- as.factor(train_df01$coop_elec)
train_df01$no_toilet <- as.factor(train_df01$no_toilet)
train_df01$sewer <- as.factor(train_df01$sewer)
train_df01$septictank <- as.factor(train_df01$septictank)
train_df01$latrine <- as.factor(train_df01$latrine)
train_df01$toilet_other <- as.factor(train_df01$toilet_other)
train_df01$no_cooking_energy <- as.factor(train_df01$no_cooking_energy)
train_df01$cooking_.elec <- as.factor(train_df01$cooking_.elec)
train_df01$cooking_gas <- as.factor(train_df01$cooking_gas)
train_df01$cooking_woodcoal <- as.factor(train_df01$cooking_woodcoal)
train_df01$trash_truck <- as.factor(train_df01$trash_truck)
train_df01$trash_buried <- as.factor(train_df01$trash_buried)
train_df01$trash_burning <- as.factor(train_df01$trash_burning)
train_df01$trash_throwing <- as.factor(train_df01$trash_throwing)
train_df01$trash_river <- as.factor(train_df01$trash_river)
```

Factorize test binary/categorical data

```
test_df01$Target <- as.factor(test_df01$Target)
test_df01$has_bathroom <- as.factor(test_df01$has_bathroom)
test_df01$has_refrig <- as.factor(test_df01$has_refrig)
test_df01$has_comp <- as.factor(test_df01$has_comp)
test_df01$disabled <- as.factor(test_df01$disabled)
test_df01$married <- as.factor(test_df01$married)
test_df01$divorced <- as.factor(test_df01$divorced)
test_df01$separated <- as.factor(test_df01$separated)
test_df01$widower <- as.factor(test_df01$widower)
test_df01$single <- as.factor(test_df01$single)
test_df01$water_inside <- as.factor(test_df01$water_inside)
test_df01$water_outside <- as.factor(test_df01$water_outside)
test_df01$no_water <- as.factor(test_df01$no_water)
test_df01$gov_elec <- as.factor(test_df01$gov_elec)
test_df01$private_elec <- as.factor(test_df01$private_elec)
test_df01$no_elec <- as.factor(test_df01$no_elec)
test_df01$coop_elec <- as.factor(test_df01$coop_elec)
test_df01$no_toilet <- as.factor(test_df01$no_toilet)
test_df01$sewer <- as.factor(test_df01$sewer)
test_df01$septictank <- as.factor(test_df01$septictank)
test_df01$latrine <- as.factor(test_df01$latrine)
test_df01$toilet_other <- as.factor(test_df01$toilet_other)
test_df01$no_cooking_energy <- as.factor(test_df01$no_cooking_energy)
test_df01$cooking_.elec <- as.factor(test_df01$cooking_.elec)
test_df01$cooking_gas <- as.factor(test_df01$cooking_gas)
test_df01$cooking_woodcoal <- as.factor(test_df01$cooking_woodcoal)
test_df01$trash_truck <- as.factor(test_df01$trash_truck)
test_df01$trash_buried <- as.factor(test_df01$trash_buried)
test_df01$trash_burning <- as.factor(test_df01$trash_burning)
test_df01$trash_throwing <- as.factor(test_df01$trash_throwing)
test_df01$trash_river <- as.factor(test_df01$trash_river)
```

Parse columns into different sub df's for modeling

```
y01_lst <- c("Target")
x01_lst <- c("num_children",
                 "cooking_gas",
                 "trash_truck",
                 "dependency",
                 "sewer",
                 "total_persons"
                 )
xy01_lst <- c("Target",
                  "num_children",
                  "cooking_gas",
                  "trash_truck",
                  "dependency",
                  "sewer",
                  "total_persons"
                  )
```

```
train_x01_df01 <- subset(x = train_df01, select = x01_lst)
train_xy01_df01 <- subset(x = train_df01, select = xy01_lst)
test_x01_df01 <- subset(x = test_df01, select = x01_lst)
test_xy01_df01 <- subset(x = test_df01, select = xy01_lst)
```

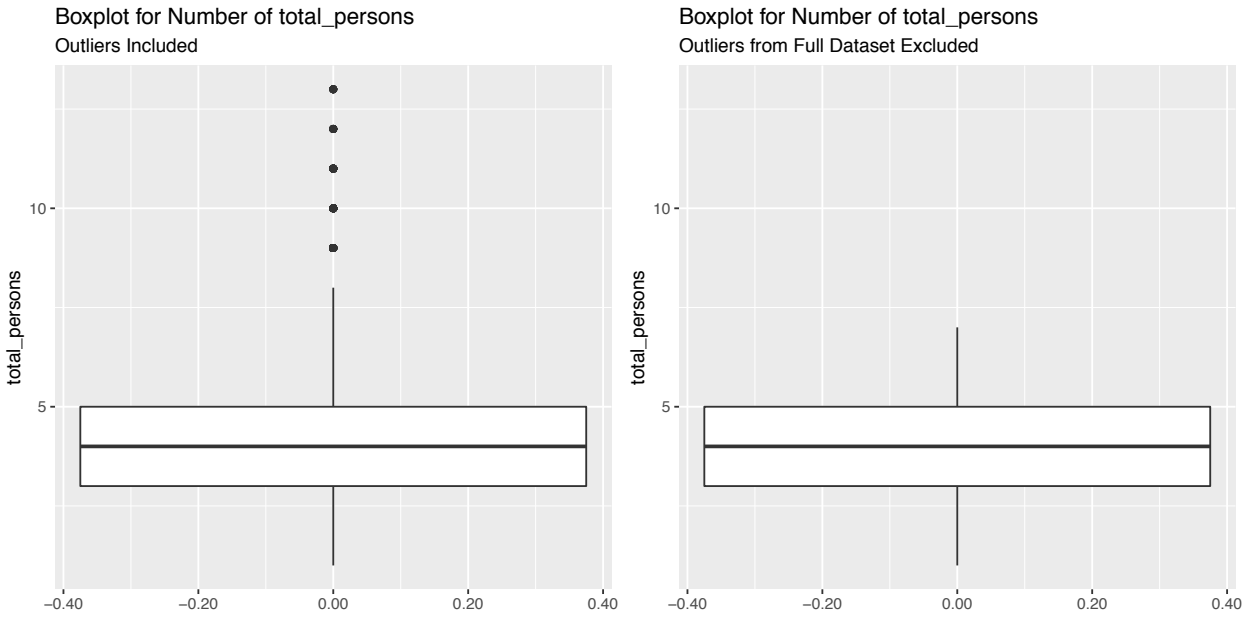
Exploratory Data Analysis (EDA)

Create boxplots for continuous variables

```
# Define function to produce formatted boxplots
box_comp <- function(xcol = NA, df = NA) {
df_s1 <- df[, xcol]

# Calculate quartiles
xcol_iqr_lim <- IQR(df_s1) * 1.5
xcol_q1 <- quantile(df_s1, probs = c(.25))
xcol_otlow <- xcol_q1 - xcol_iqr_lim

xcol_q3 <- quantile(df_s1, probs = c(.75))
xcol_othigh <- xcol_q3 + xcol_iqr_lim

# Subset non-outlier data
xcol_iqr_df01 <- subset(df, (df_s1 > xcol_otlow & df_s1 < xcol_othigh))
df_s2 <- xcol_iqr_df01[, xcol]

# Begin calculating measures of centrality & dispersion
xcol_mean <- round(mean(df_s1), 3)
xcol_iqr_df01_trunc_mean <- round(mean(df_s2), 3)
xcol_med <- median(df_s1)
xcol_iqr_df01_trunc_med <- median(df_s2)
xcol_mode <- mode(df_s1)
xcol_iqr_df01_trunc_mode <- mode(df_s2)
xcol_stde <- round(sd(df_s1), 3)
xcol_iqr_df01_trunc_stde <- round(sd(df_s2), 3)
xcol_vari <- round(var(df_s1), 3)
xcol_iqr_df01_trunc_vari <- round(var(df_s2), 3)

var01_min <- min(df[, xcol])
var01_max <- max(df[, xcol])
var01_range <- var01_max - var01_min

var02_min <- min(xcol_iqr_df01[, xcol])
var02_max <- max(xcol_iqr_df01[, xcol])
var02_range <- var02_max - var02_min

# Configure y-axis min & max to sync graphs
plot_min <- min(var01_min, var02_min)
plot_max <- max(var01_max, var02_max)

# Plot 2 boxplots as comparison of full data set vs. w/o outliers
ggp1 <- ggplot(df, aes_string(y = xcol)) +
  geom_boxplot() +
```

```
  ylim(plot_min, plot_max) +
  scale_x_continuous(labels = comma) +
  labs(title = paste0("Boxplot for Number of ", xcol), subtitle = "Outliers Included")

ggp2 <- ggplot(xcol_iqr_df01, aes_string(y = xcol)) +
  geom_boxplot() +
  ylim(plot_min, plot_max) +
  scale_x_continuous(labels = comma) +
  labs(title = paste0("Boxplot for Number of ", xcol), subtitle = "Outliers from Full
  ↪  Dataset Excluded")

# Map 2 plots to 1 grid
grid.arrange(ggp1, ggp2, ncol = 2)

# Define vectors for a measurement table df
measure_name <- c(paste0("Variable: ", xcol),
                  "Count",
                  "NA Count",
                  "Mean",
                  "Median",
                  "Mode",
                  "Standard Deviation",
                  "Variance",
                  "Range",
                  "Min",
                  "Max",
                  "IQR Outlier Limit",
                  "25th Percentile",
                  "75th Percentile",
                  "Lower Outlier Threshold",
                  "Upper Outlier Threshold"
                  )

measure_val01 <- c("All data points",
                   as.character(dim(df)[1]),
                   sum(is.na(df_s1)),
                   xcol_mean,
                   xcol_med,
                   xcol_mode,
                   xcol_stde,
                   xcol_vari,
                   var01_range,
                   var01_min,
                   var01_max,
                   paste0("+/-", xcol_iqr_lim),
                   xcol_q1,
                   xcol_q3,
                   xcol_otlow,
                   xcol_othigh
                   )

measure_val02 <- c("Outliers Excluded",
                   paste0(as.character(dim(xcol_iqr_df01)[1]), " (",
```

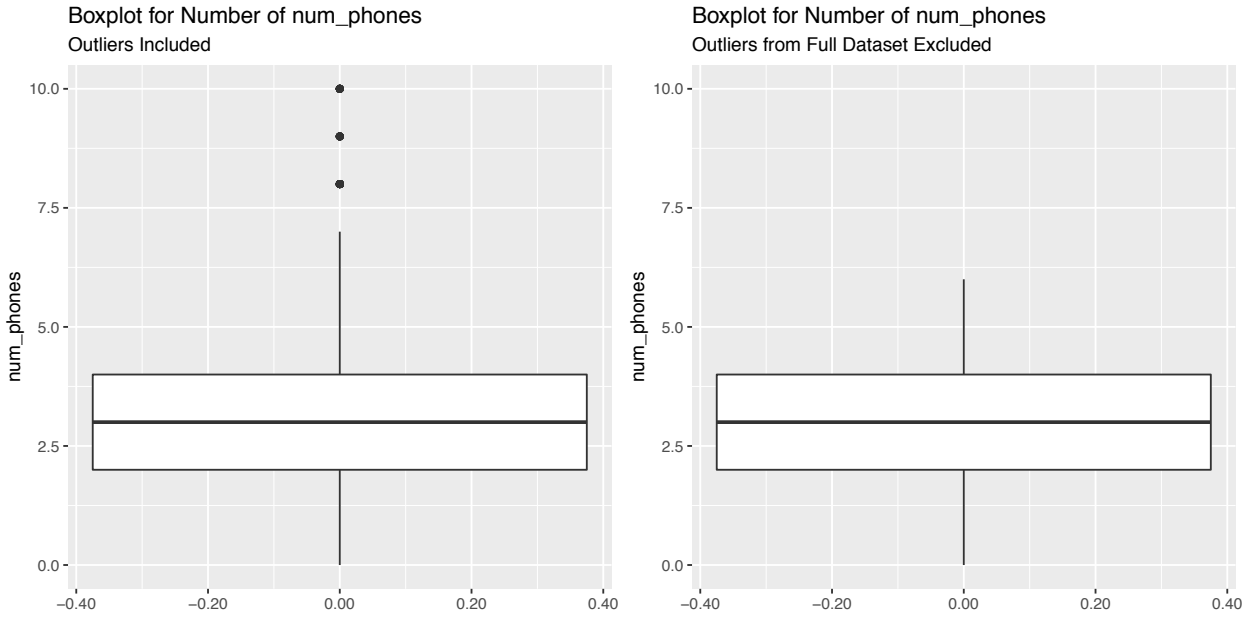
```
                        round((as.numeric(dim(xcol_iqr_df01)[1] /
                        ↪  as.numeric(dim(df)[1]))) * 100, 1),
                        "%)"
                        ),
                    sum(is.na(df_s2)),
                    xcol_iqr_df01_trunc_mean,
                    xcol_iqr_df01_trunc_med,
                    xcol_iqr_df01_trunc_mode,
                    xcol_iqr_df01_trunc_stde,
                    xcol_iqr_df01_trunc_vari,
                    var02_range,
                    var02_min,
                    var02_max,
                    "-",
                    "-",
                    "-",
                    "-",
                    "-"
                    )

measure_tbl <- data.frame(measure_name, measure_val01, measure_val02)
print(measure_tbl)
}

# Run function on all continuous variables in training data set
box_comp(xcol = "total_persons", df = train_df01)
```




```
##               measure_name   measure_val01     measure_val02
## 1  Variable: total_persons All data points Outliers Excluded
## 2                    Count            6690      6466 (96.7%)
## 3                 NA Count               0                 0
## 4                     Mean           4.013             3.823
```

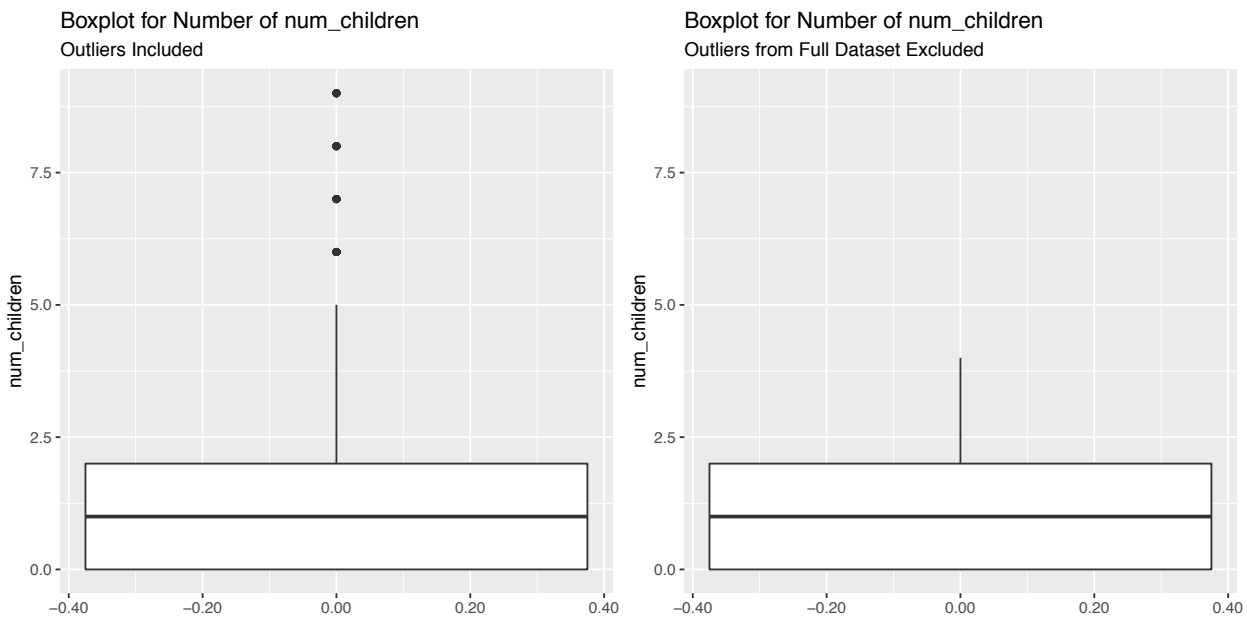
```
## 5                   Median              4                 4
## 6                     Mode          numeric           numeric
## 7       Standard Deviation            1.766             1.442
## 8                 Variance            3.118             2.078
## 9                    Range               12                 6
## 10                     Min                1                 1
## 11                     Max               13                 7
## 12       IQR Outlier Limit             +/-3                 -
## 13         25th Percentile                3                 -
## 14         75th Percentile                5                 -
## 15 Lower Outlier Threshold                0                 -
## 16 Upper Outlier Threshold                8                 -
```

```
box_comp(xcol = "num_phones", df = train_df01)
```




```
##              measure_name   measure_val01     measure_val02
## 1     Variable: num_phones All data points Outliers Excluded
## 2                   Count            6690      6582 (98.4%)
## 3                NA Count               0                 0
## 4                    Mean           2.826             2.739
## 5                  Median               3                 3
## 6                    Mode         numeric           numeric
## 7       Standard Deviation           1.486             1.329
## 8                 Variance           2.207             1.766
## 9                    Range              10                 6
## 10                     Min               0                 0
## 11                     Max              10                 6
## 12       IQR Outlier Limit            +/-3                 -
## 13         25th Percentile               2                 -
## 14         75th Percentile               4                 -
## 15 Lower Outlier Threshold              -1                 -
## 16 Upper Outlier Threshold               7                 -
```

```
box_comp(xcol = "num_children", df = train_df01)
```

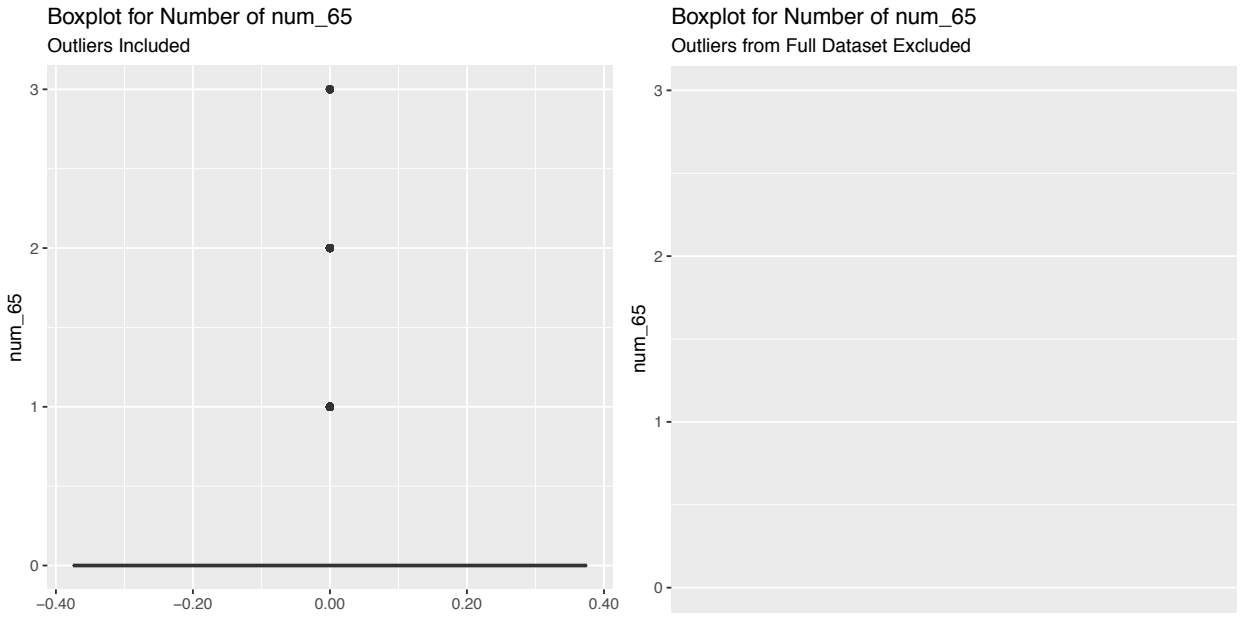



```
##               measure_name   measure_val01     measure_val02
## 1   Variable: num_children All data points Outliers Excluded
## 2                    Count            6690      6508 (97.3%)
## 3                 NA Count               0                 0
## 4                     Mean           1.412             1.289
## 5                   Median               1                 1
## 6                     Mode         numeric           numeric
## 7       Standard Deviation           1.368             1.152
## 8                 Variance           1.871             1.327
## 9                    Range               9                 4
## 10                     Min               0                 0
## 11                     Max               9                 4
## 12       IQR Outlier Limit            +/-3                 -
## 13         25th Percentile               0                 -
## 14         75th Percentile               2                 -
## 15 Lower Outlier Threshold              -3                 -
## 16 Upper Outlier Threshold               5                 -
```

```
box_comp(xcol = "num_65", df = train_df01)
```

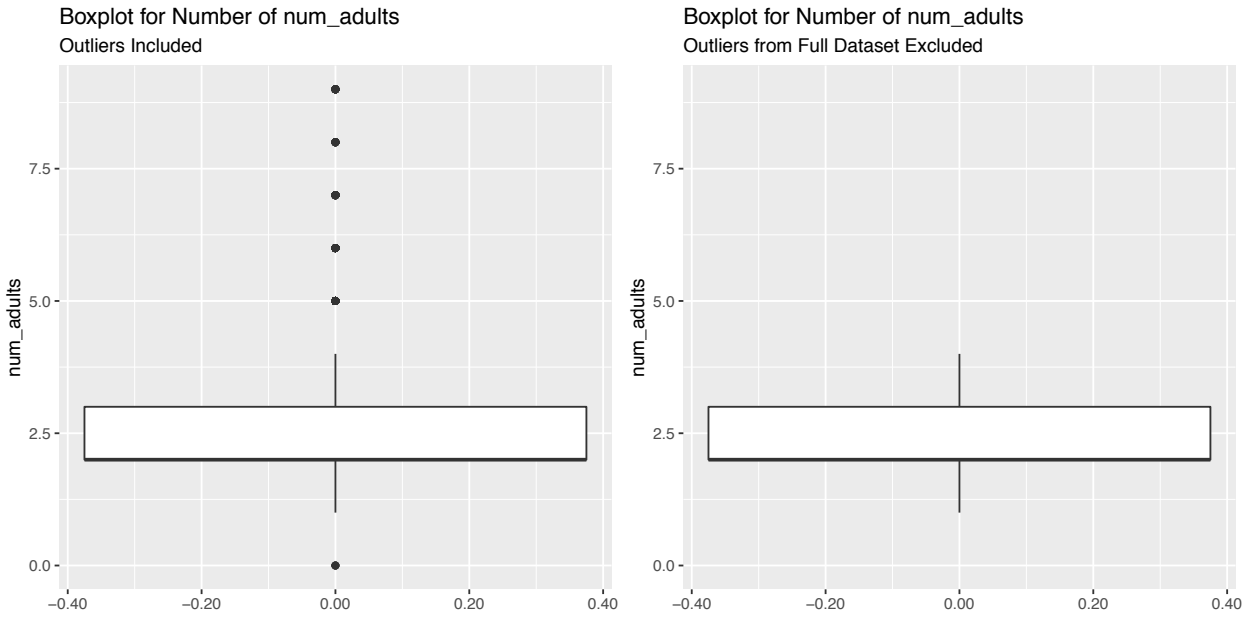
```
## Warning in min(xcol_iqr_df01[, xcol]): no non-missing arguments to min;
## returning Inf
```

```
## Warning in max(xcol_iqr_df01[, xcol]): no non-missing arguments to max;
## returning -Inf
```



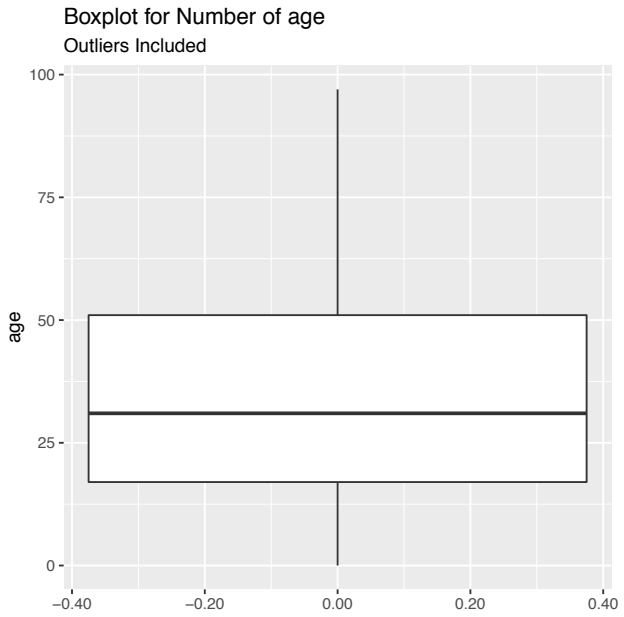
	measure_name	measure_val01	measure_val02
1	Variable: num_65	All data points	Outliers Excluded
2	Count	6690	0 (0%)
3	NA Count	0	0
4	Mean	0.283	NaN
5	Median	0	<NA>
6	Mode	numeric	numeric
7	Standard Deviation	0.597	<NA>
8	Variance	0.356	<NA>
9	Range	3	-Inf
10	Min	0	Inf
11	Max	3	-Inf
12	IQR Outlier Limit	+/-0	-
13	25th Percentile	0	-
14	75th Percentile	0	-
15	Lower Outlier Threshold	0	-
16	Upper Outlier Threshold	0	-

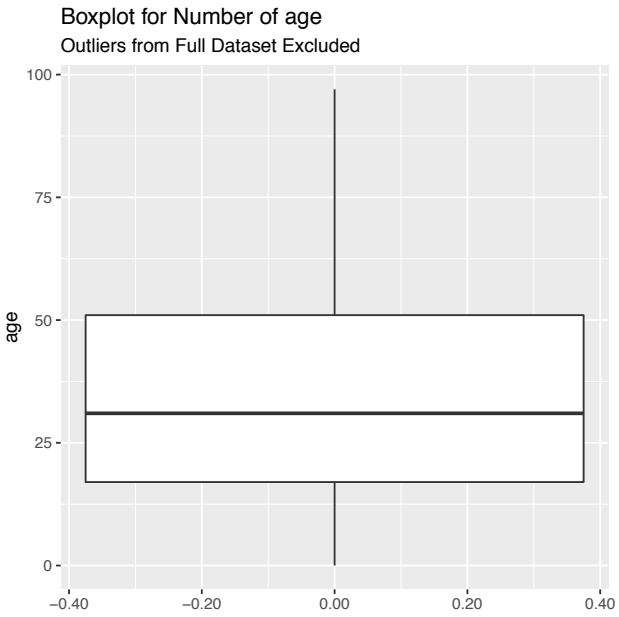
```
box_comp(xcol = "num_adults", df = train_df01)
```




```
##              measure_name   measure_val01     measure_val02
## 1     Variable: num_adults All data points Outliers Excluded
## 2                    Count            6690      6284 (93.9%)
## 3                 NA Count               0                 0
## 4                     Mean           2.601             2.409
## 5                   Median               2                 2
## 6                     Mode         numeric           numeric
## 7       Standard Deviation           1.173             0.875
## 8                 Variance           1.376             0.765
## 9                    Range               9                 3
## 10                     Min               0                 1
## 11                     Max               9                 4
## 12       IQR Outlier Limit          +/-1.5                 -
## 13         25th Percentile               2                 -
## 14         75th Percentile               3                 -
## 15 Lower Outlier Threshold             0.5                 -
## 16 Upper Outlier Threshold             4.5                 -
```

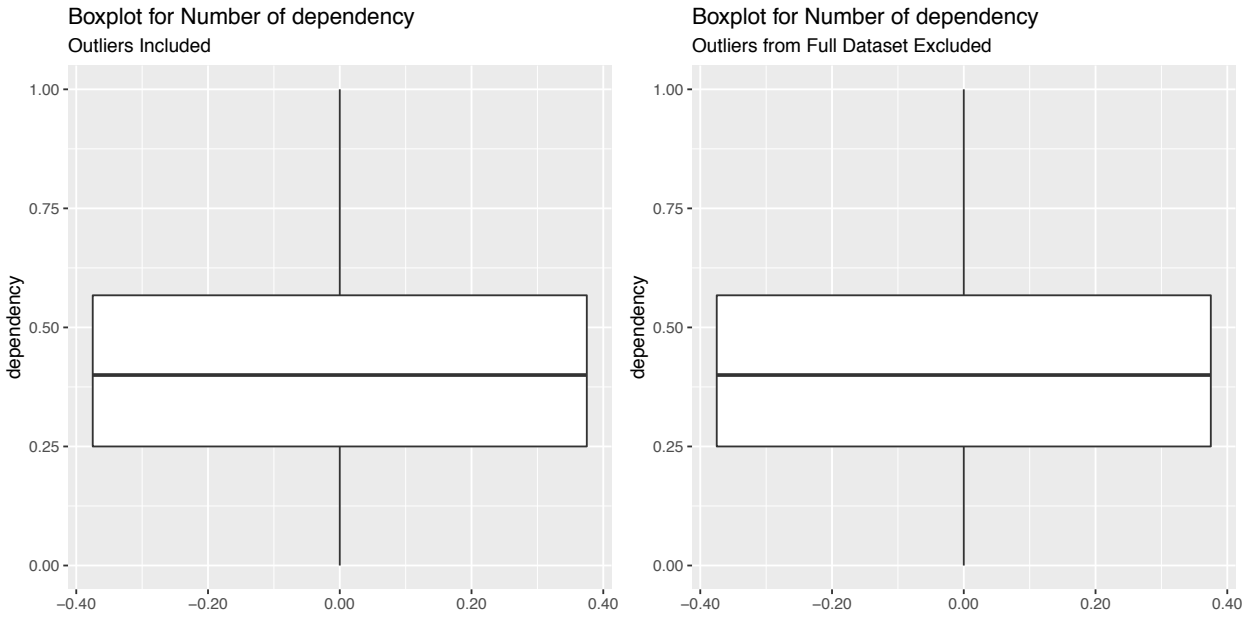
```
box_comp(xcol = "age", df = train_df01)
```






```
##                 measure_name   measure_val01     measure_val02
## 1             Variable: age All data points Outliers Excluded
## 2                     Count            6690       6690 (100%)
## 3                 NA Count               0                 0
## 4                      Mean          34.092            34.092
## 5                    Median              31                31
## 6                      Mode         numeric           numeric
## 7        Standard Deviation          21.728            21.728
## 8                  Variance         472.126           472.126
## 9                     Range              97                97
## 10                      Min               0                 0
## 11                      Max              97                97
## 12        IQR Outlier Limit           +/-51                 -
## 13          25th Percentile              17                 -
## 14          75th Percentile              51                 -
## 15 Lower Outlier Threshold             -34                 -
## 16 Upper Outlier Threshold             102                 -
```

```
box_comp(xcol = "dependency", df = train_df01)
```




```
##               measure_name        measure_val01     measure_val02
## 1     Variable: dependency      All data points Outliers Excluded
## 2                    Count                 6690       6690 (100%)
## 3                 NA Count                    0                 0
## 4                     Mean                  0.4               0.4
## 5                   Median                  0.4               0.4
## 6                     Mode              numeric           numeric
## 7       Standard Deviation                0.254             0.254
## 8                 Variance                0.065             0.065
## 9                    Range                    1                 1
## 10                     Min                    0                 0
## 11                     Max                    1                 1
## 12       IQR Outlier Limit +/-0.476190476190476                 -
## 13         25th Percentile                 0.25                 -
## 14         75th Percentile    0.567460317460317                 -
## 15 Lower Outlier Threshold   -0.226190476190476                 -
## 16 Upper Outlier Threshold     1.04365079365079                 -
```

Create bar charts for categorical variables

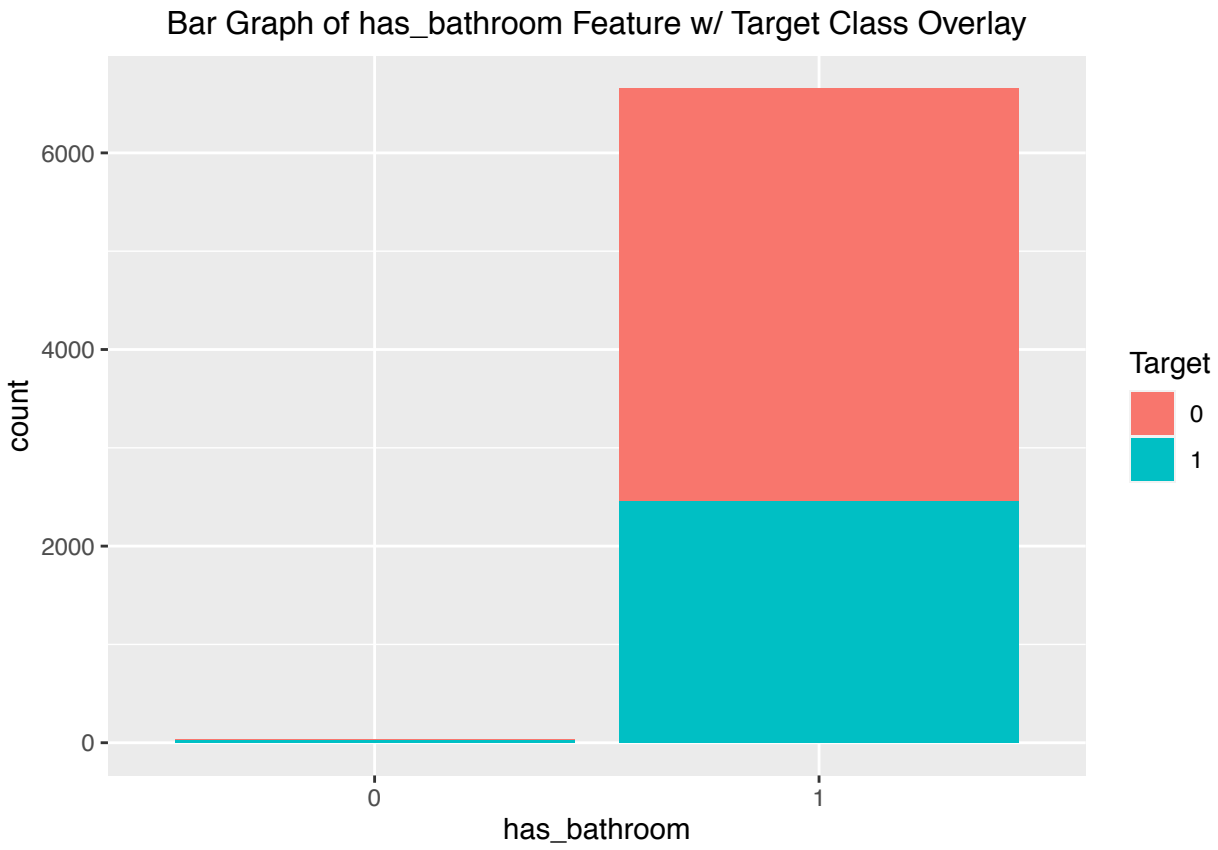
```
# Define function to produce formatted bar charts
plot_bar <- function(x = NA, y = NA, df = NA) {

# Contingency table for `marital` and `response`
cross_tab_df01 <- table(df[, y], df[, x])

cross_tab_df02 <- addmargins(A = cross_tab_df01, FUN = list(total = sum),
                             quiet = TRUE)
cross_tab_df03 <- round(prop.table(cross_tab_df01, margin = 2) * 100, 1)
cross_tab_df04 <- addmargins(A = cross_tab_df03, FUN = list(total = sum),
                             quiet = TRUE)
print(paste0("Contingency Table for ", y, " and ", x))
print(cross_tab_df02)
```

```
print(paste0("Proportions Contingency Table for ", y, " and ", x))
print(cross_tab_df04)

# Plot bar graphs
ggplot(df, aes_string(x)) +
  geom_bar(aes_string(fill = y)) +
  labs(title = paste0("Bar Graph of ", x, " Feature w/ ", y, " Class Overlay")) +
  theme(plot.title = element_text(hjust = 0.5, size = 12))
}

# Run function on all binary/categorical variables in training data set
plot_bar(x = "Target", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and Target"
##
##            0    1 total
##   0     4204    0  4204
##   1        0 2486  2486
##   total 4204 2486  6690
## [1] "Proportions Contingency Table for Target and Target"
##
##           0   1 total
##   0     100   0   100
##   1       0 100   100
##   total 100 100   200
```

```
plot_bar(x = "has_bathroom", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and has_bathroom"
##
##            0    1 total
##   0        8 4196  4204
##   1       27 2459  2486
##   total   35 6655  6690
## [1] "Proportions Contingency Table for Target and has_bathroom"
##
##             0     1 total
##   0      22.9  63.1  86.0
##   1      77.1  36.9 114.0
##   total 100.0 100.0 200.0
```



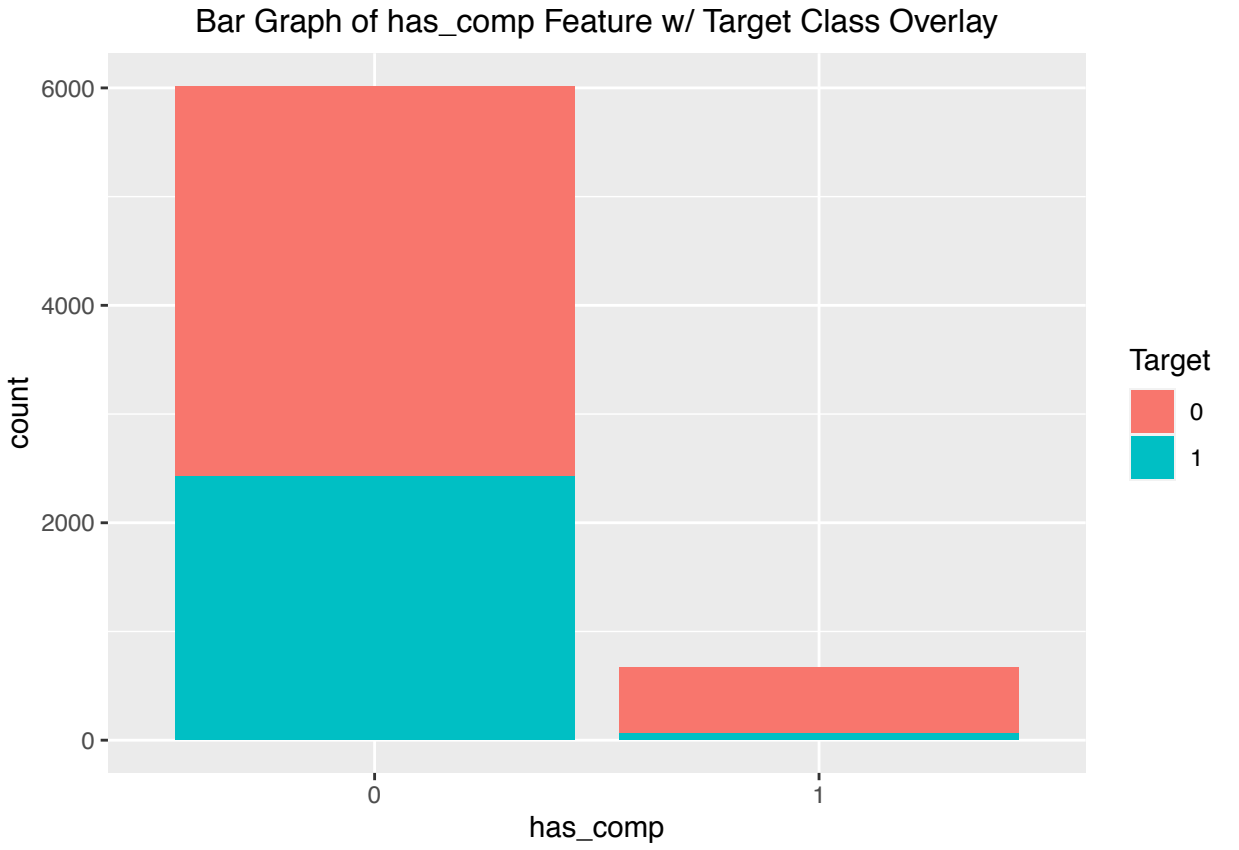

```
plot_bar(x = "has_refrig", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and has_refrig"
##
##            0    1 total
##   0      114 4090  4204
##   1      167 2319  2486
##   total  281 6409  6690
## [1] "Proportions Contingency Table for Target and has_refrig"
##
##             0     1 total
```

```
##   0      40.6  63.8 104.4
##   1      59.4  36.2  95.6
##   total 100.0 100.0 200.0
```


```
plot_bar(x = "has_comp", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and has_comp"
##
##            0    1 total
##   0     3595  609  4204
##   1     2423   63  2486
##   total 6018  672  6690
## [1] "Proportions Contingency Table for Target and has_comp"
##
##             0     1 total
##   0      59.7  90.6 150.3
##   1      40.3   9.4  49.7
##   total 100.0 100.0 200.0
```

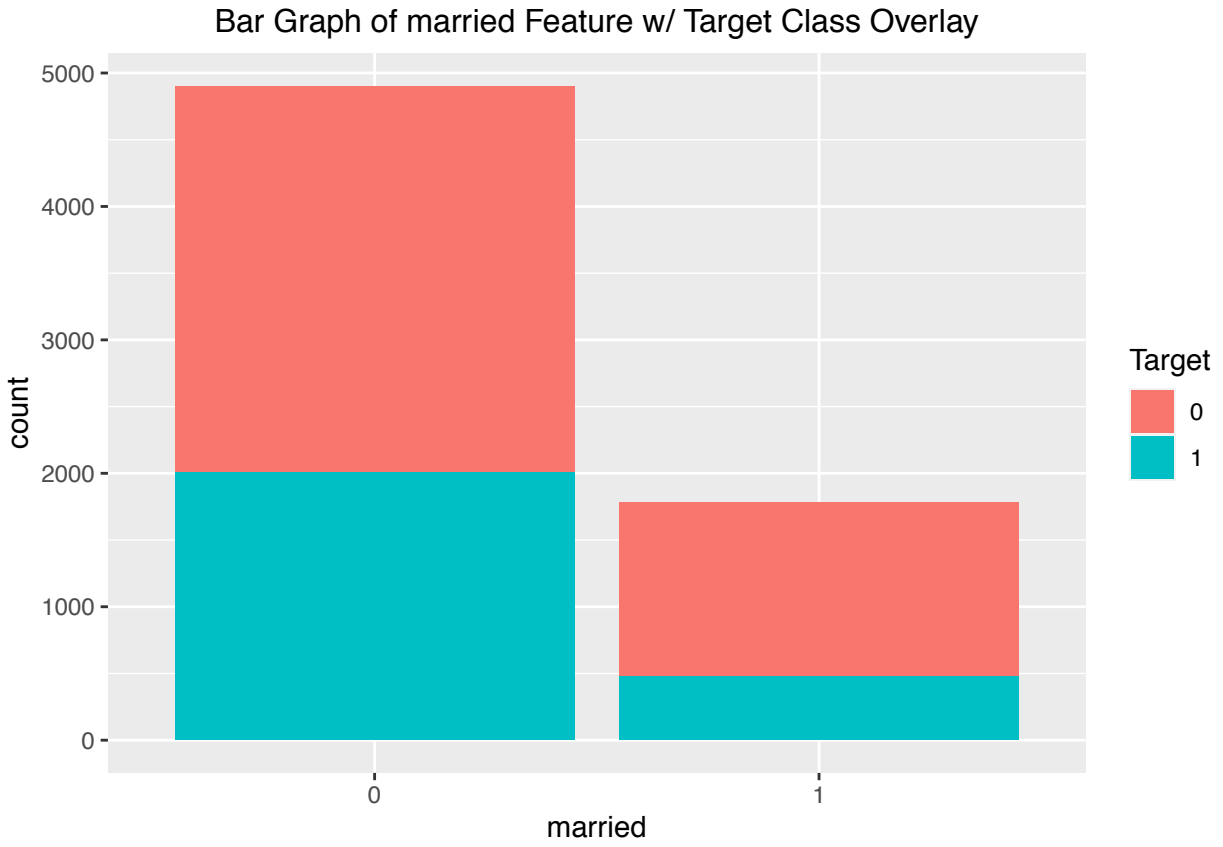



```
plot_bar(x = "disabled", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and disabled"
##
##            0    1 total
##   0     4004  200  4204
##   1     2301  185  2486
##   total 6305  385  6690
## [1] "Proportions Contingency Table for Target and disabled"
##
##              0     1 total
##   0      63.5  51.9 115.4
##   1      36.5  48.1  84.6
##   total 100.0 100.0 200.0
```


```
plot_bar(x = "married", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and married"
##
##            0    1 total
##   0     2895 1309  4204
##   1     2008  478  2486
##   total 4903 1787  6690
## [1] "Proportions Contingency Table for Target and married"
##
##              0     1 total
##   0      59.0  73.3 132.3
##   1      41.0  26.7  67.7
##   total 100.0 100.0 200.0
```

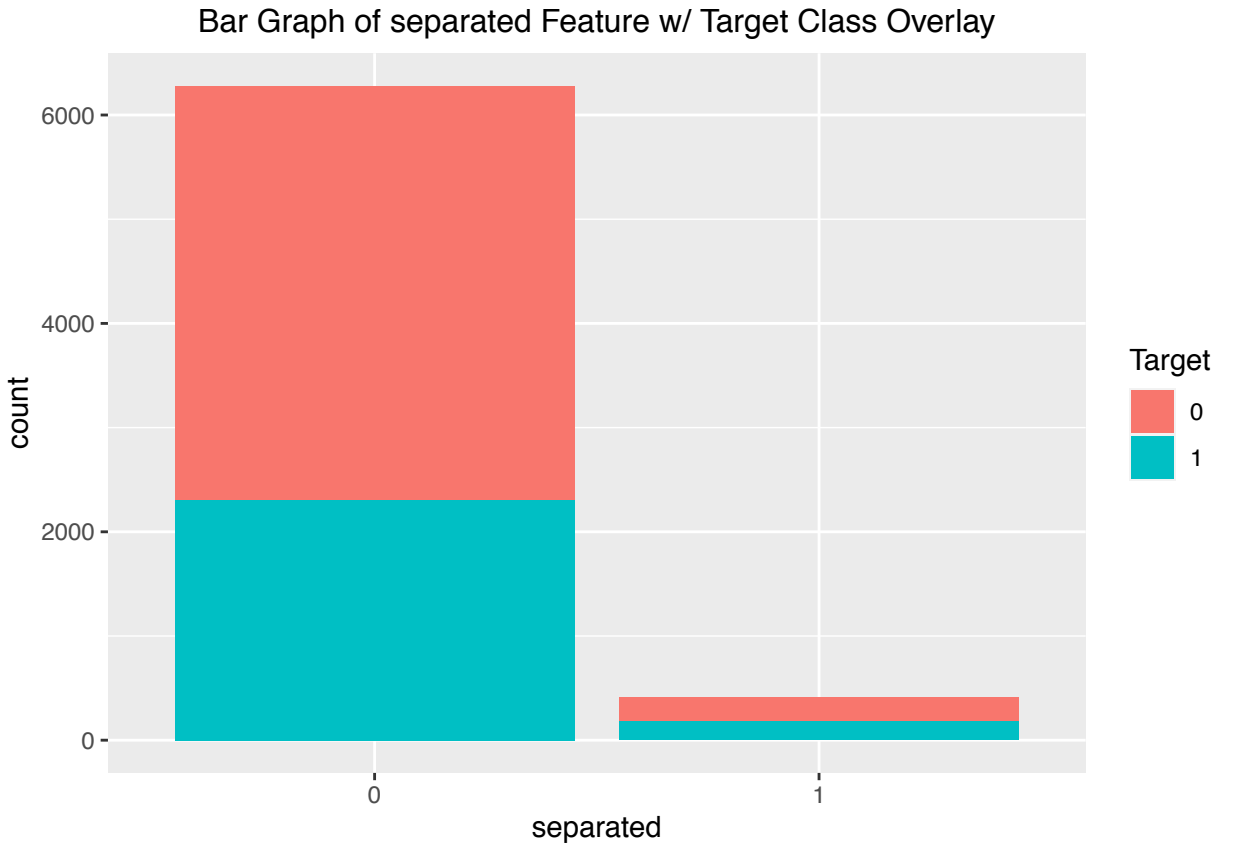



```
plot_bar(x = "divorced", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and divorced"
##
##            0    1 total
##   0     4053  151  4204
##   1     2445   41  2486
##   total 6498  192  6690
## [1] "Proportions Contingency Table for Target and divorced"
##
##              0     1 total
##   0      62.4  78.6 141.0
##   1      37.6  21.4  59.0
##   total 100.0 100.0 200.0
```


```
plot_bar(x = "separated", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and separated"
##
##            0    1 total
##   0     3973  231  4204
##   1     2306  180  2486
##   total 6279  411  6690
## [1] "Proportions Contingency Table for Target and separated"
##
##              0     1 total
##   0      63.3  56.2 119.5
##   1      36.7  43.8  80.5
##   total 100.0 100.0 200.0
```




```
plot_bar(x = "widower", y = "Target", df = train_df01)
```

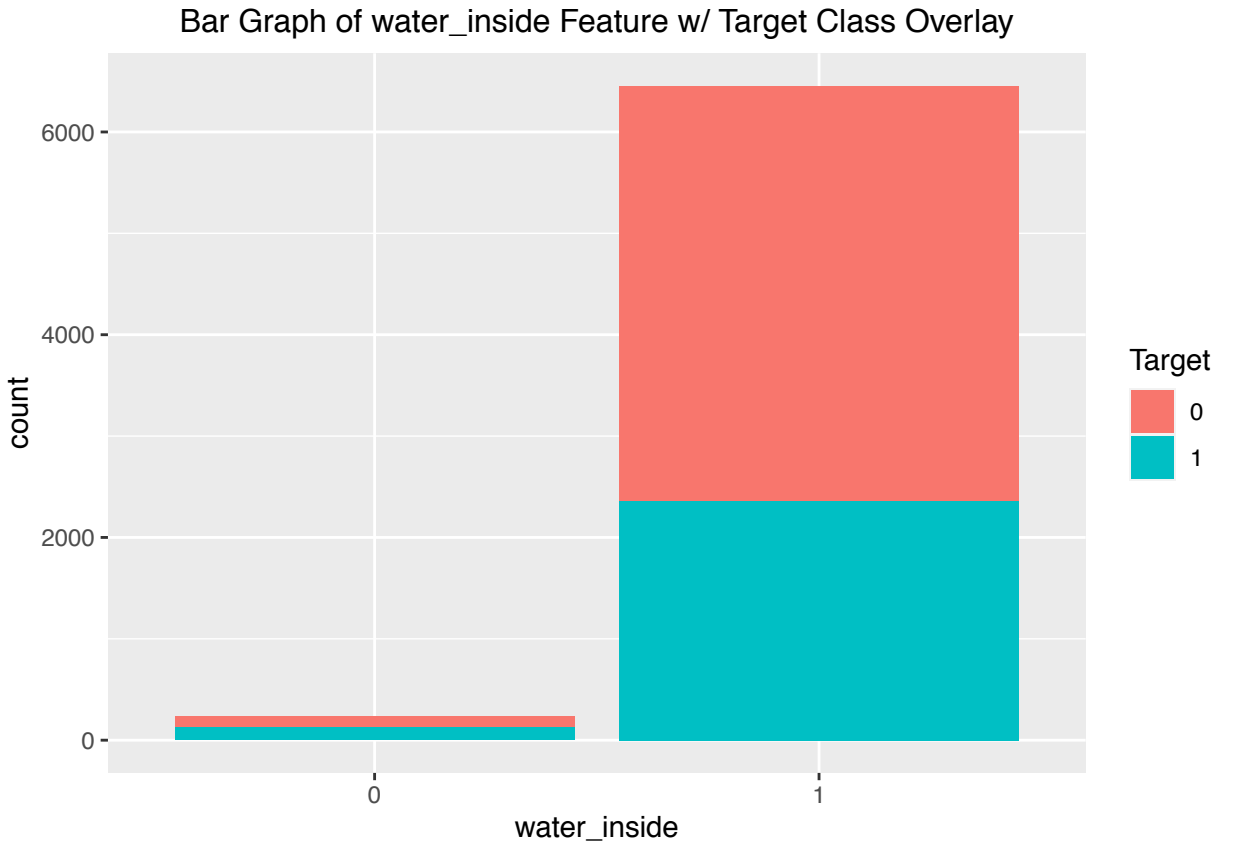
```
## [1] "Contingency Table for Target and widower"
##
##            0    1 total
##   0     4073  131  4204
##   1     2380  106  2486
##   total 6453  237  6690
## [1] "Proportions Contingency Table for Target and widower"
##
##              0     1 total
##   0      63.1  55.3 118.4
##   1      36.9  44.7  81.6
##   total 100.0 100.0 200.0
```


```
plot_bar(x = "single", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and single"
##
##            0    1 total
##   0     2747 1457  4204
##   1     1653  833  2486
##   total 4400 2290  6690
## [1] "Proportions Contingency Table for Target and single"
##
##              0     1 total
##   0      62.4  63.6 126.0
##   1      37.6  36.4  74.0
##   total 100.0 100.0 200.0
```

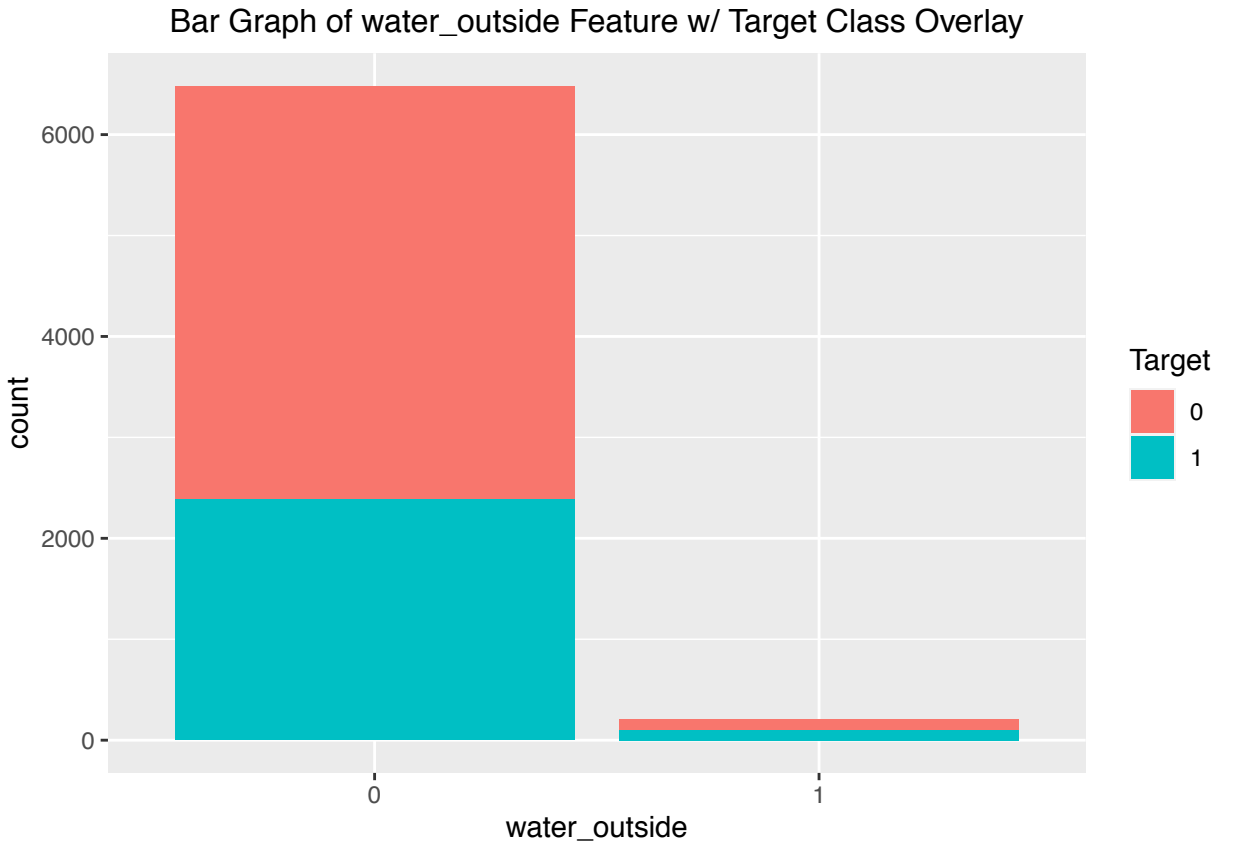

```
plot_bar(x = "water_inside", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and water_inside"
##
##            0    1 total
##   0      109 4095  4204
##   1      127 2359  2486
##   total  236 6454  6690
## [1] "Proportions Contingency Table for Target and water_inside"
##
##              0     1 total
##   0      46.2  63.4 109.6
##   1      53.8  36.6  90.4
##   total 100.0 100.0 200.0
```



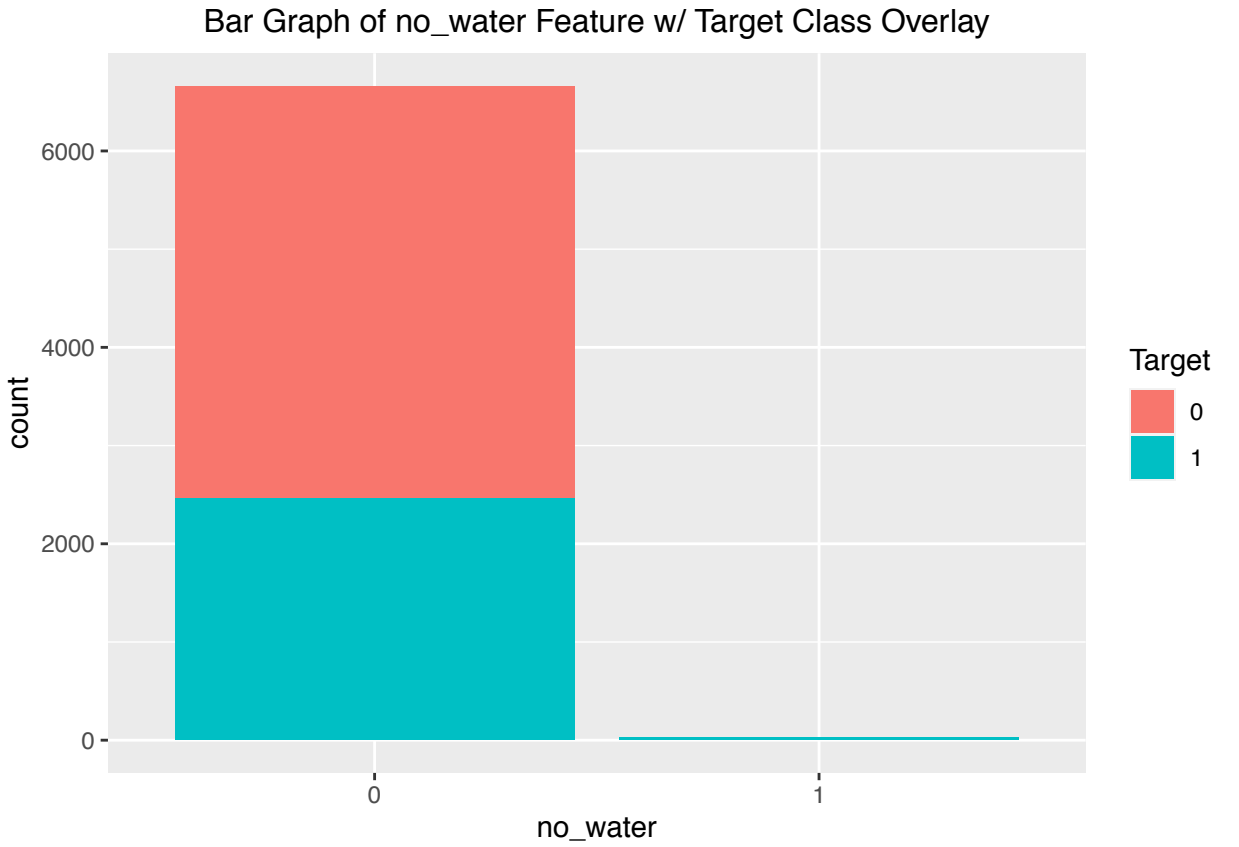

```
plot_bar(x = "water_outside", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and water_outside"
##
##            0    1 total
##   0     4097  107  4204
##   1     2385  101  2486
##   total 6482  208  6690
## [1] "Proportions Contingency Table for Target and water_outside"
##
##              0     1 total
##   0      63.2  51.4 114.6
##   1      36.8  48.6  85.4
##   total 100.0 100.0 200.0
```




```
plot_bar(x = "no_water", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and no_water"
##
##            0    1 total
##   0     4202    2  4204
##   1     2460   26  2486
##   total 6662   28  6690
## [1] "Proportions Contingency Table for Target and no_water"
##
##              0     1 total
##   0      63.1   7.1  70.2
##   1      36.9  92.9 129.8
##   total 100.0 100.0 200.0
```

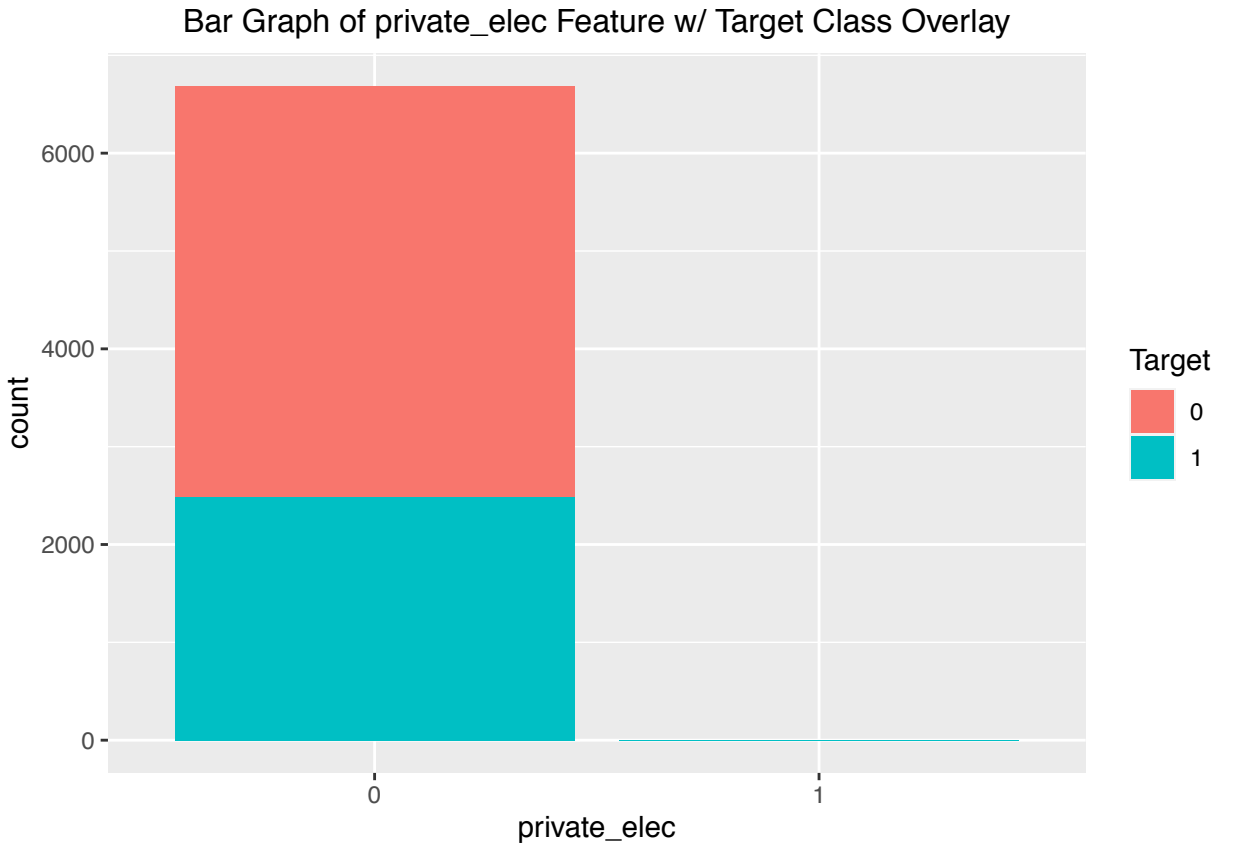



```
plot_bar(x = "gov_elec", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and gov_elec"
##
##            0    1 total
##   0      483 3721  4204
##   1      302 2184  2486
##   total  785 5905  6690
## [1] "Proportions Contingency Table for Target and gov_elec"
##
##              0     1 total
##   0      61.5  63.0 124.5
##   1      38.5  37.0  75.5
##   total 100.0 100.0 200.0
```

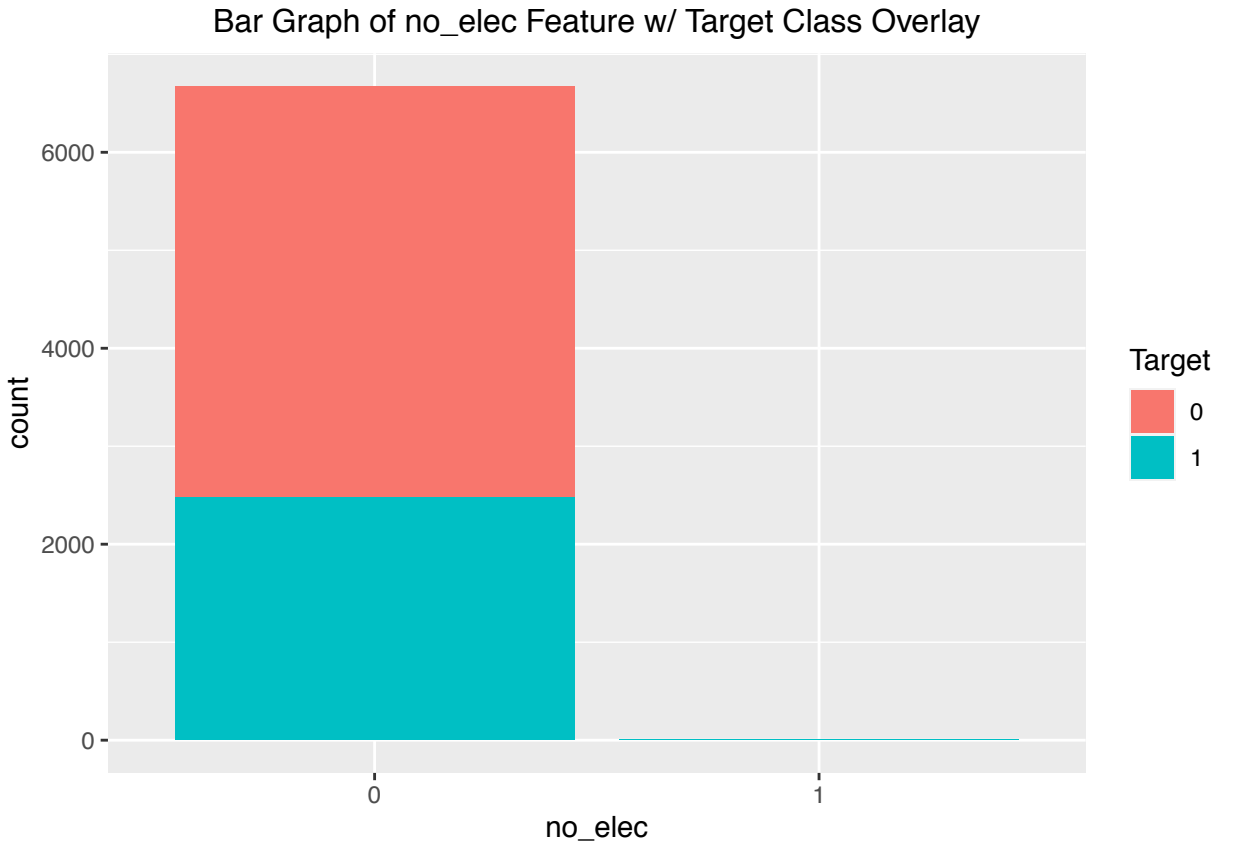

```
plot_bar(x = "private_elec", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and private_elec"
##
##            0    1 total
##   0     4202    2  4204
##   1     2485    1  2486
##   total 6687    3  6690
## [1] "Proportions Contingency Table for Target and private_elec"
##
##              0     1 total
##   0      62.8  66.7 129.5
##   1      37.2  33.3  70.5
##   total 100.0 100.0 200.0
```



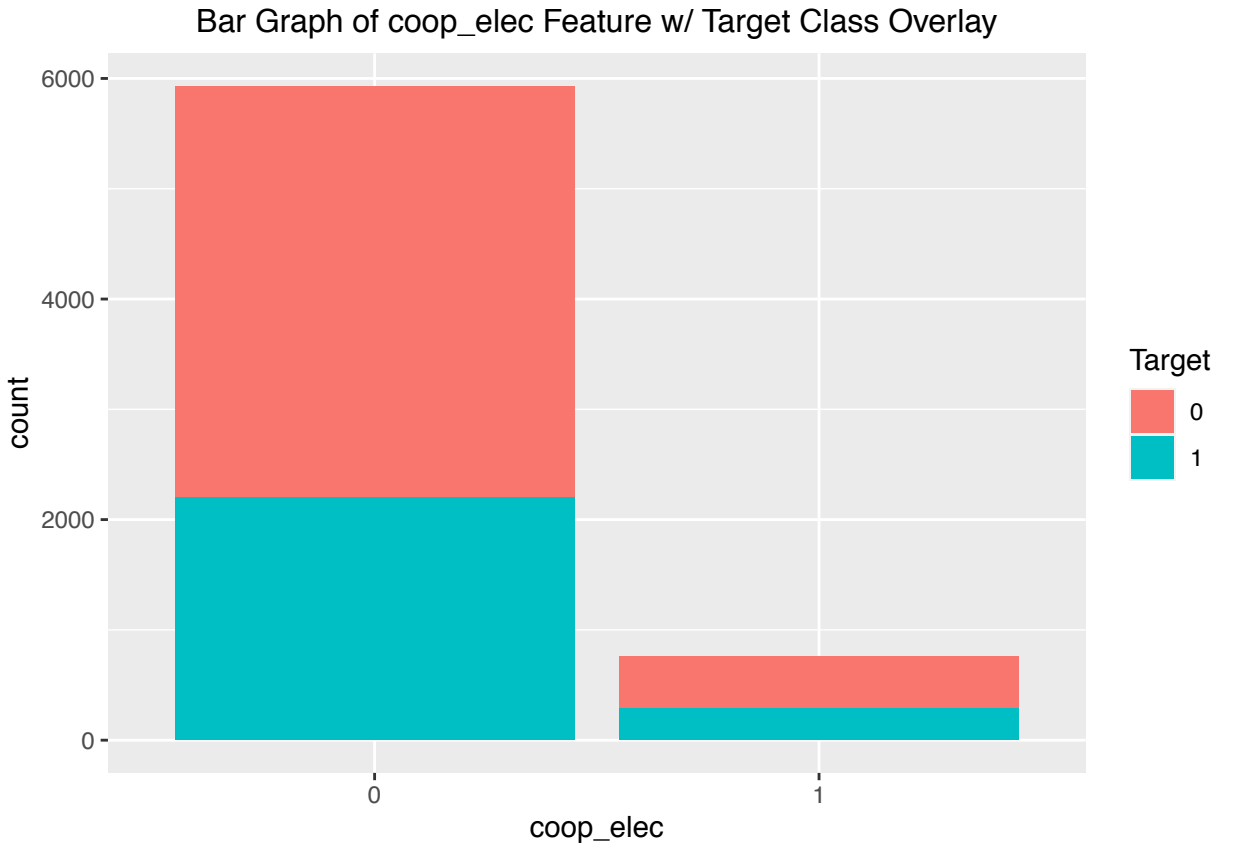

```
plot_bar(x = "no_elec", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and no_elec"
##
##            0    1 total
##   0     4198    6  4204
##   1     2479    7  2486
##   total 6677   13  6690
## [1] "Proportions Contingency Table for Target and no_elec"
##
##              0     1 total
##   0      62.9  46.2 109.1
##   1      37.1  53.8  90.9
##   total 100.0 100.0 200.0
```



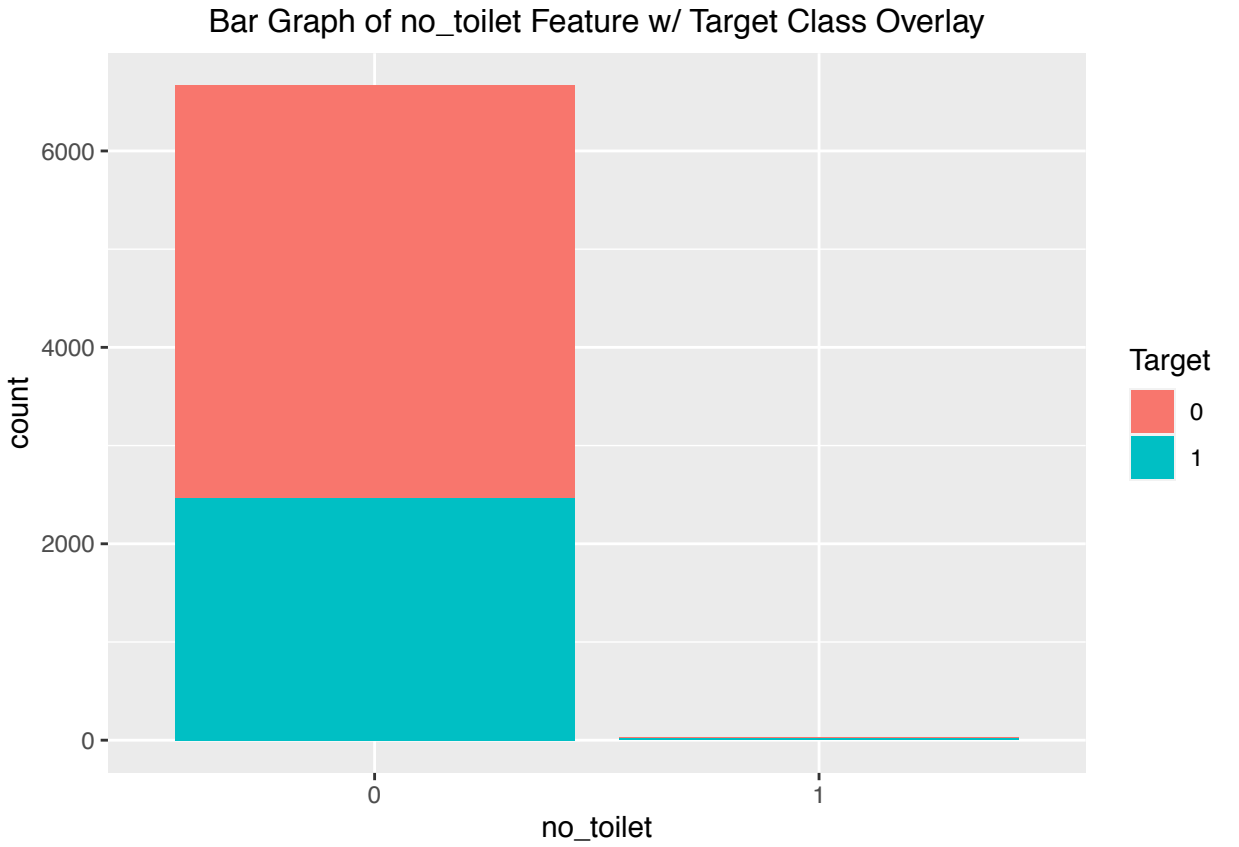

```
plot_bar(x = "coop_elec", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and coop_elec"
##
##            0    1 total
##   0     3731  473  4204
##   1     2201  285  2486
##   total 5932  758  6690
## [1] "Proportions Contingency Table for Target and coop_elec"
##
##              0     1 total
##   0      62.9  62.4 125.3
##   1      37.1  37.6  74.7
##   total 100.0 100.0 200.0
```



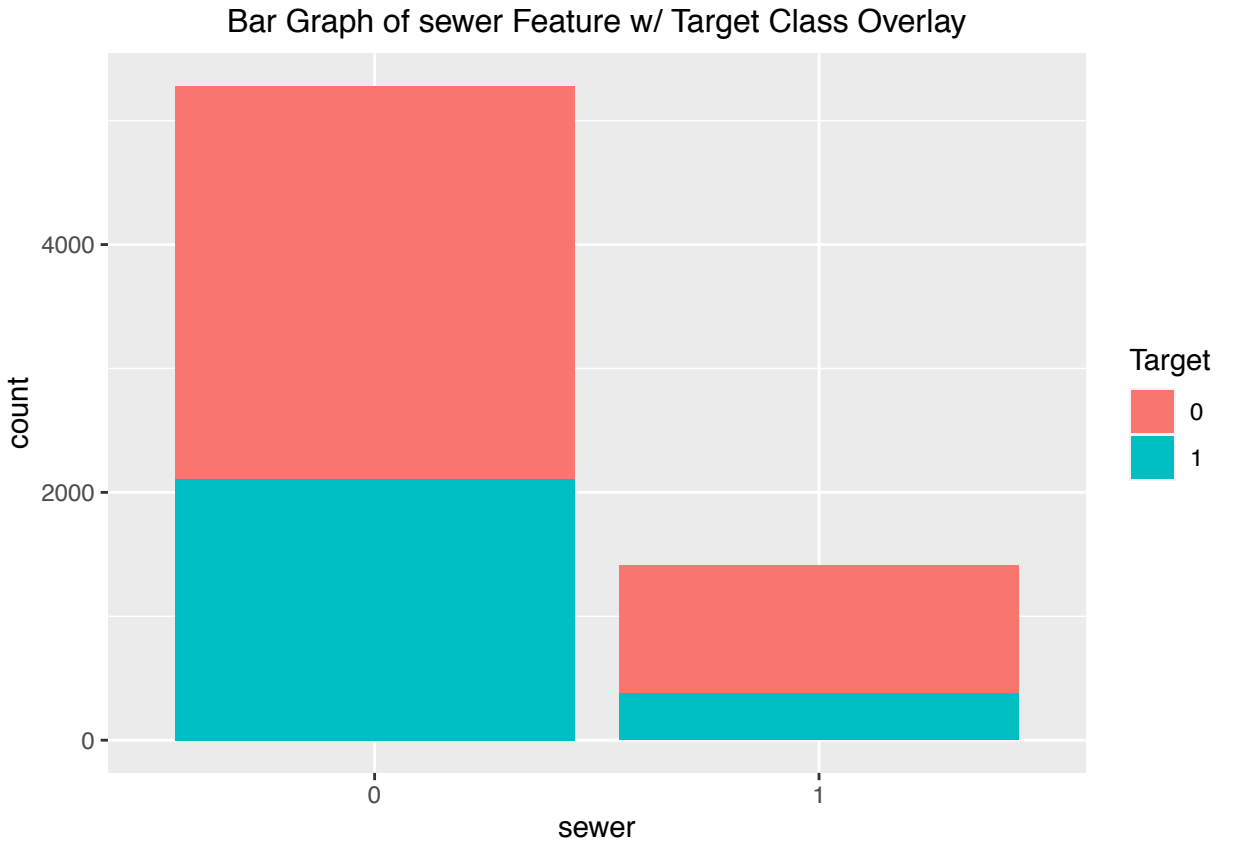

```
plot_bar(x = "no_toilet", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and no_toilet"
##
##            0    1 total
##   0     4196    8  4204
##   1     2466   20  2486
##   total 6662   28  6690
## [1] "Proportions Contingency Table for Target and no_toilet"
##
##              0     1 total
##   0      63.0  28.6  91.6
##   1      37.0  71.4 108.4
##   total 100.0 100.0 200.0
```




```
plot_bar(x = "sewer", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and sewer"
##
##            0    1 total
##   0     3171 1033  4204
##   1     2109  377  2486
##   total 5280 1410  6690
## [1] "Proportions Contingency Table for Target and sewer"
##
##              0     1 total
##   0      60.1  73.3 133.4
##   1      39.9  26.7  66.6
##   total 100.0 100.0 200.0
```




```
plot_bar(x = "septictank", y = "Target", df = train_df01)
```

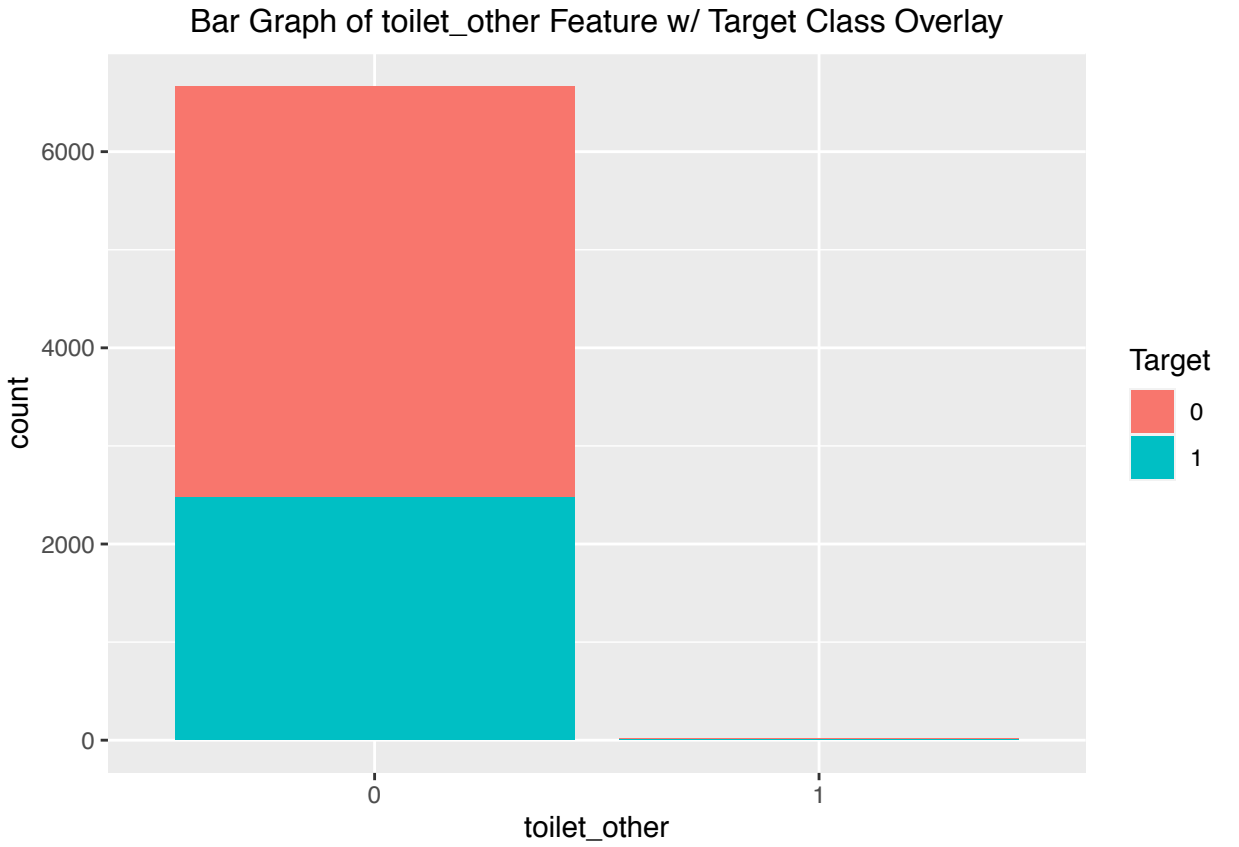
```
## [1] "Contingency Table for Target and septictank"
##
##            0    1 total
##   0     1077 3127  4204
##   1      483 2003  2486
##   total 1560 5130  6690
## [1] "Proportions Contingency Table for Target and septictank"
##
##           0   1 total
##   0      69  61   130
##   1      31  39    70
##   total 100 100   200
```


```
plot_bar(x = "latrine", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and latrine"
##
##            0    1 total
##   0     4179   25  4204
##   1     2409   77  2486
##   total 6588  102  6690
## [1] "Proportions Contingency Table for Target and latrine"
##
##              0     1 total
##   0      63.4  24.5  87.9
##   1      36.6  75.5 112.1
##   total 100.0 100.0 200.0
```

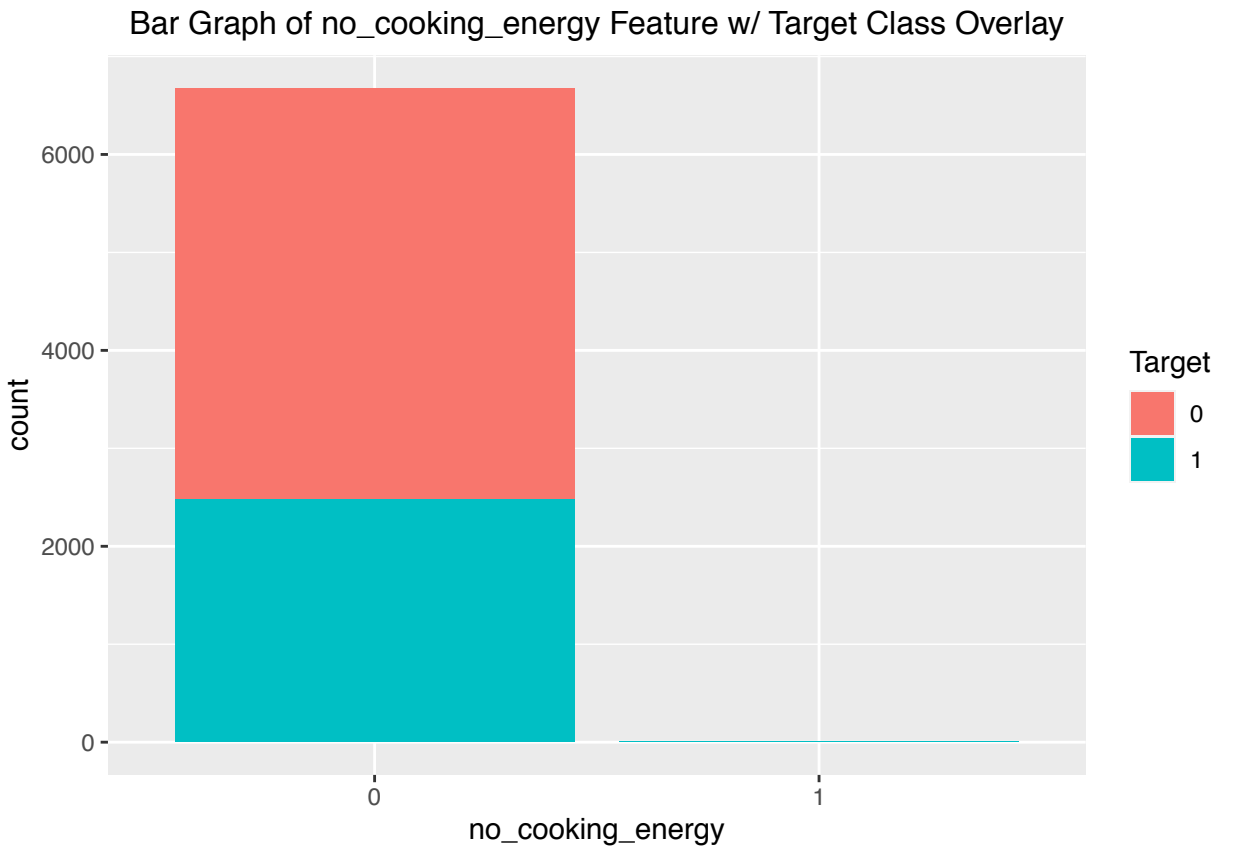

```
plot_bar(x = "toilet_other", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and toilet_other"
##
##            0    1 total
##   0     4193   11  4204
##   1     2477    9  2486
##   total 6670   20  6690
## [1] "Proportions Contingency Table for Target and toilet_other"
##
##              0     1 total
##   0      62.9  55.0 117.9
##   1      37.1  45.0  82.1
##   total 100.0 100.0 200.0
```




```
plot_bar(x = "no_cooking_energy", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and no_cooking_energy"
##
##            0    1 total
##   0     4200    4  4204
##   1     2479    7  2486
##   total 6679   11  6690
## [1] "Proportions Contingency Table for Target and no_cooking_energy"
##
##              0     1 total
##   0      62.9  36.4  99.3
##   1      37.1  63.6 100.7
##   total 100.0 100.0 200.0
```




```
plot_bar(x = "cooking_.elec", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and cooking_.elec"
##
##            0    1 total
##   0     1915 2289  4204
##   1     1522  964  2486
##   total 3437 3253  6690
## [1] "Proportions Contingency Table for Target and cooking_.elec"
##
##              0     1 total
##   0      55.7  70.4 126.1
##   1      44.3  29.6  73.9
##   total 100.0 100.0 200.0
```


```
plot_bar(x = "cooking_gas", y = "Target", df = train_df01)
```

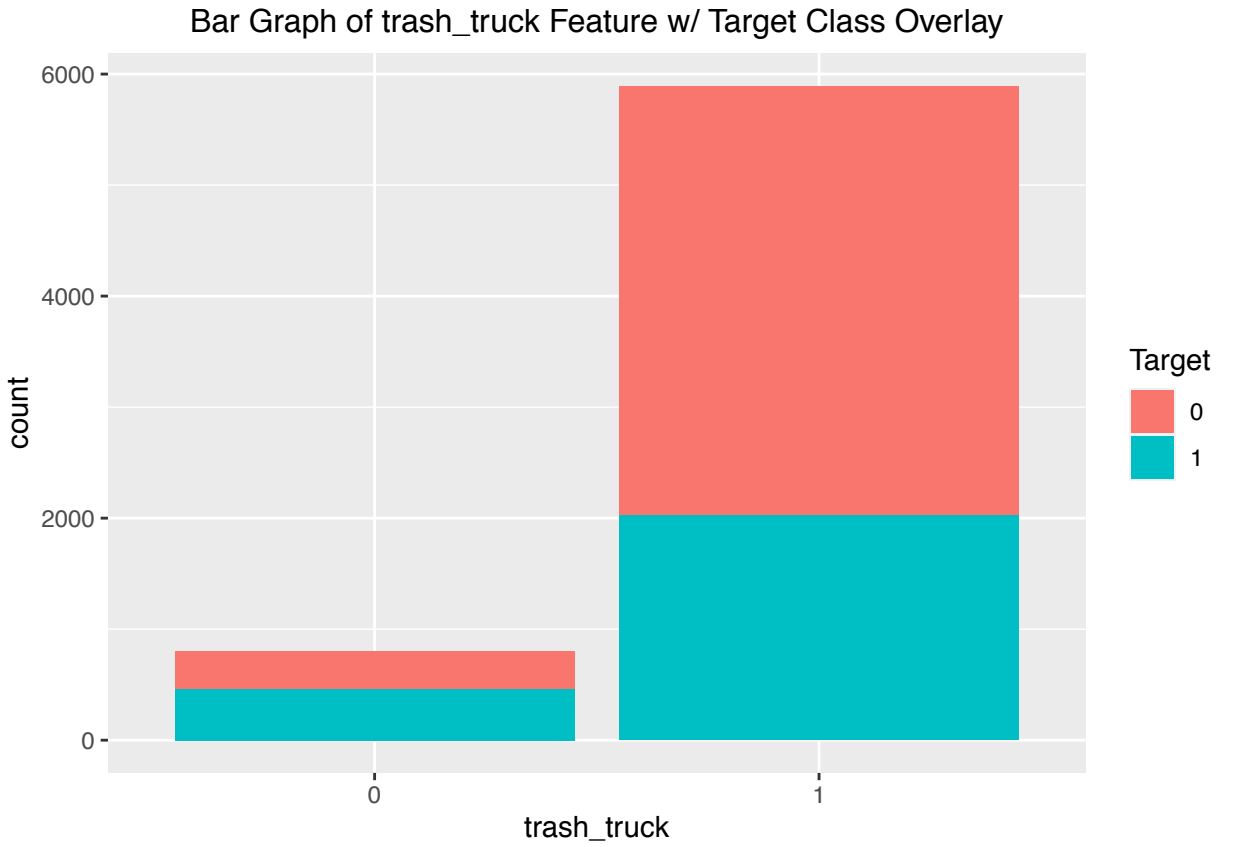
```
## [1] "Contingency Table for Target and cooking_gas"
##
##            0    1 total
##   0     2411 1793  4204
##   1     1203 1283  2486
##   total 3614 3076  6690
## [1] "Proportions Contingency Table for Target and cooking_gas"
##
##              0     1 total
##   0      66.7  58.3 125.0
##   1      33.3  41.7  75.0
##   total 100.0 100.0 200.0
```


```
plot_bar(x = "cooking_woodcoal", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and cooking_woodcoal"
##
##            0    1 total
##   0     4086  118  4204
##   1     2254  232  2486
##   total 6340  350  6690
## [1] "Proportions Contingency Table for Target and cooking_woodcoal"
##
##              0     1 total
##   0      64.4  33.7  98.1
##   1      35.6  66.3 101.9
##   total 100.0 100.0 200.0
```


```
plot_bar(x = "trash_truck", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and trash_truck"
##
##            0    1 total
##   0      336 3868  4204
##   1      461 2025  2486
##   total  797 5893  6690
## [1] "Proportions Contingency Table for Target and trash_truck"
##
##              0     1 total
##   0      42.2  65.6 107.8
##   1      57.8  34.4  92.2
##   total 100.0 100.0 200.0
```

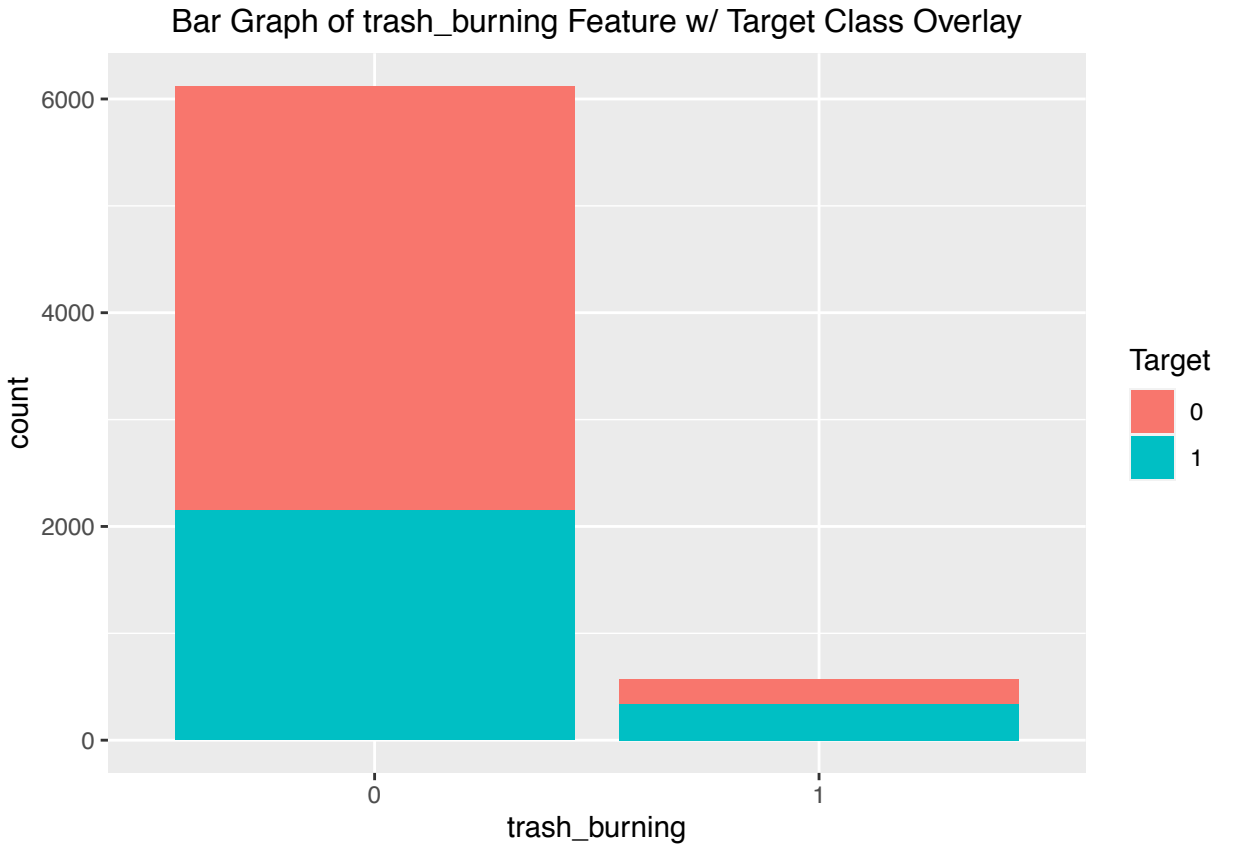



```
plot_bar(x = "trash_buried", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and trash_buried"
##
##            0    1 total
##   0     4106   98  4204
##   1     2373  113  2486
##   total 6479  211  6690
## [1] "Proportions Contingency Table for Target and trash_buried"
##
##              0     1 total
##   0      63.4  46.4 109.8
##   1      36.6  53.6  90.2
##   total 100.0 100.0 200.0
```

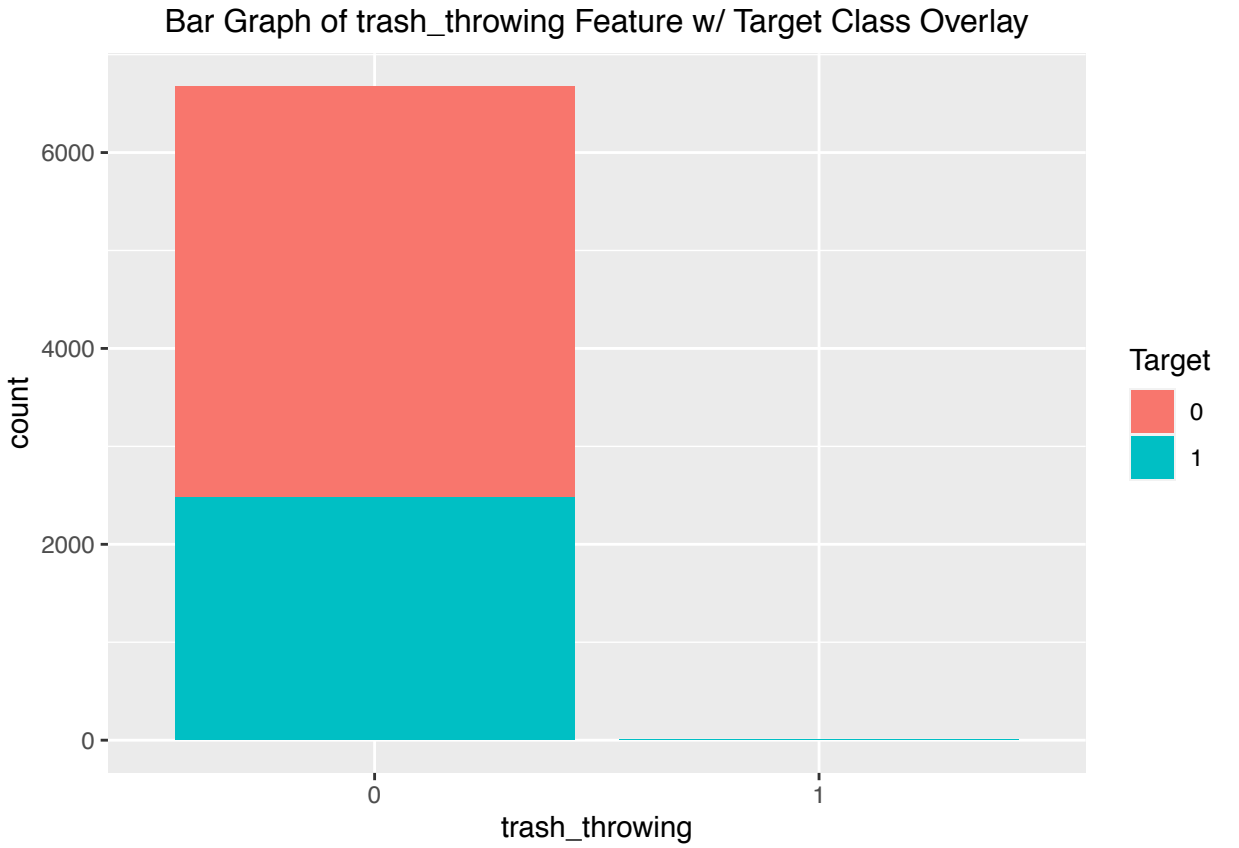

```
plot_bar(x = "trash_burning", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and trash_burning"
##
##            0    1 total
##   0     3974  230  4204
##   1     2148  338  2486
##   total 6122  568  6690
## [1] "Proportions Contingency Table for Target and trash_burning"
##
##              0     1 total
##   0      64.9  40.5 105.4
##   1      35.1  59.5  94.6
##   total 100.0 100.0 200.0
```




```
plot_bar(x = "trash_throwing", y = "Target", df = train_df01)
```

```
## [1] "Contingency Table for Target and trash_throwing"
##
##            0    1 total
##   0     4204    0  4204
##   1     2476   10  2486
##   total 6680   10  6690
## [1] "Proportions Contingency Table for Target and trash_throwing"
##
##              0     1 total
##   0      62.9   0.0  62.9
##   1      37.1 100.0 137.1
##   total 100.0 100.0 200.0
```




```
plot_bar(x = "trash_river", y = "Target", df = train_df01)
```

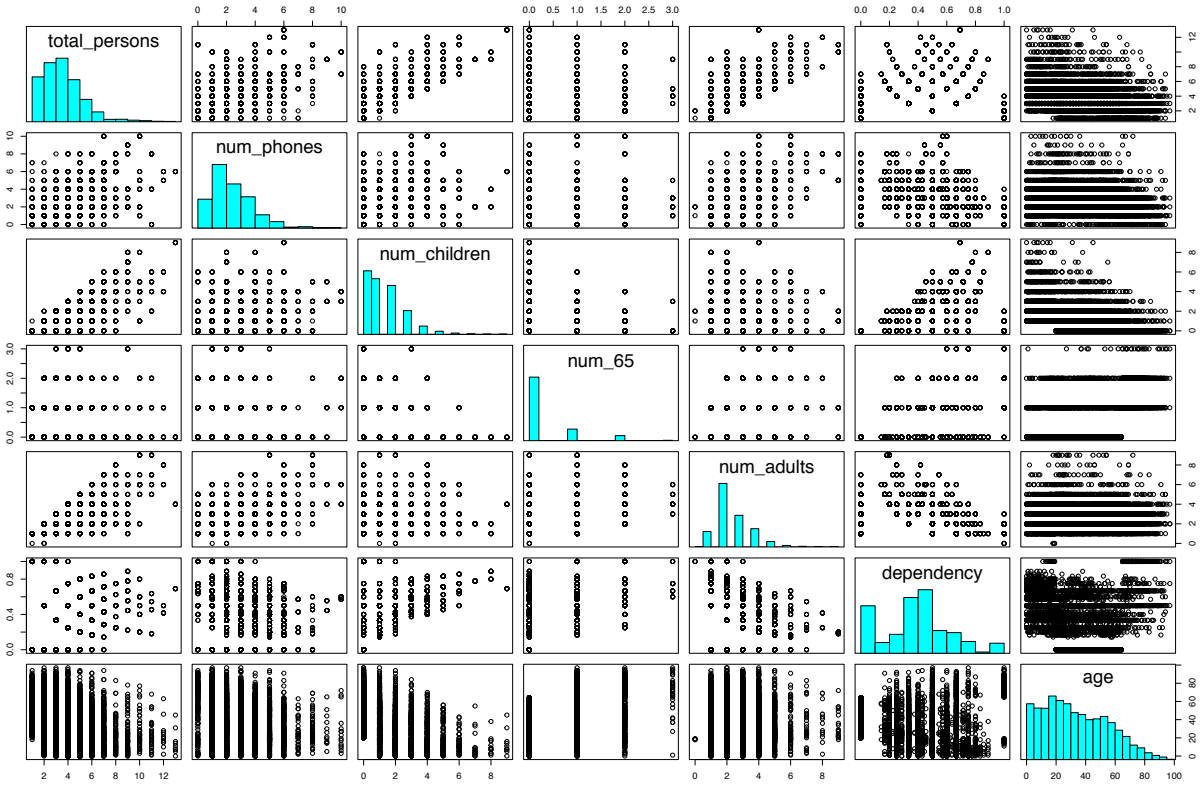
```
## [1] "Contingency Table for Target and trash_river"
##
##            0 total
##   0     4204  4204
##   1     2486  2486
##   total 6690  6690
## [1] "Proportions Contingency Table for Target and trash_river"
##
##              0 total
##   0      62.8  62.8
##   1      37.2  37.2
##   total 100.0 100.0
```

Parse non-binary numerical features into sub df's for EDA

```
# Create sub df for numerical features
x01_num_lst <- c("total_persons",
                 "num_phones",
                 "num_children",
                 "num_65",
                 "num_adults",
                 "dependency",
                 "age"
                 )

train_x_num_df01_s1 <- subset(x = train_df01, select = x01_num_lst)
train_y_df01 <- subset(x = train_df01, select = c("Target"))
```

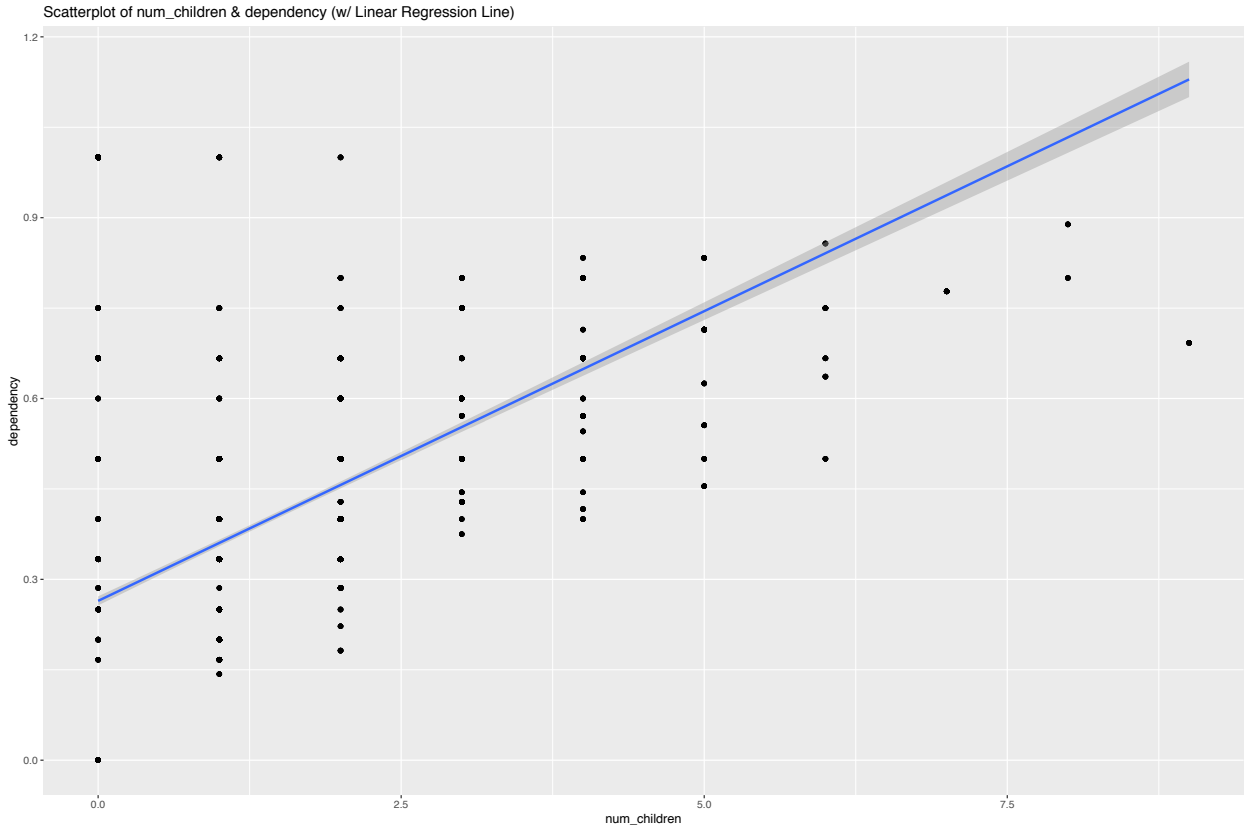
Create scatterplots for continuous features

```
# Scatterplot matrix
pairs(x = train_x_num_df01_s1, diag.panel = panel.hist)
```



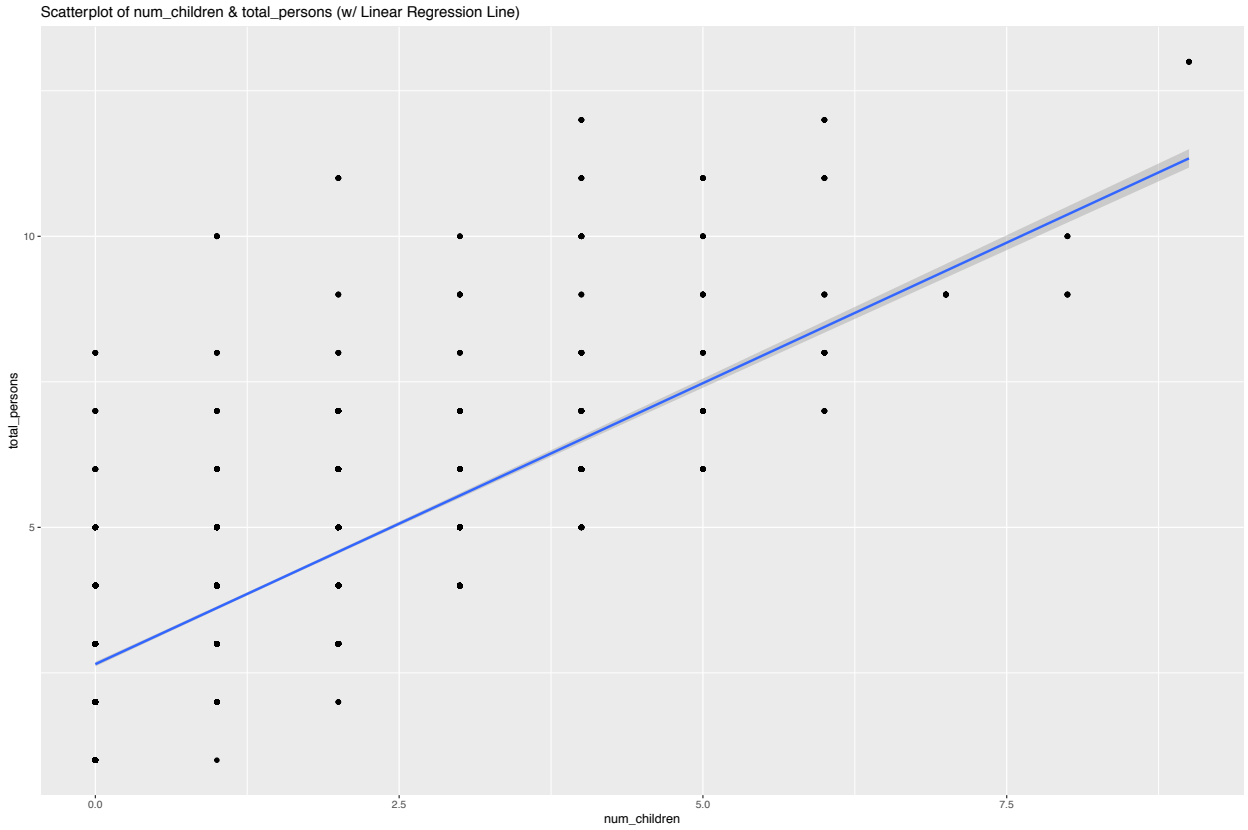

```
# Individual scatterplots of numerical features included in modeling phase
spg1 <- ggplot(train_x_num_df01_s1, aes(num_children, dependency)) +
  geom_point() +
  stat_smooth(method = "lm") +
  labs(title = "Scatterplot of num_children & dependency (w/ Linear Regression Line)")

spg2 <- ggplot(train_x_num_df01_s1, aes(num_children, total_persons)) +
  geom_point() +
  stat_smooth(method = "lm") +
  labs(title = "Scatterplot of num_children & total_persons (w/ Linear Regression Line)")

spg3 <- ggplot(train_x_num_df01_s1, aes(dependency, total_persons)) +
  geom_point() +
  stat_smooth(method = "lm") +
  labs(title = "Scatterplot of dependency & total_persons (w/ Linear Regression Line)")

spg1
```

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



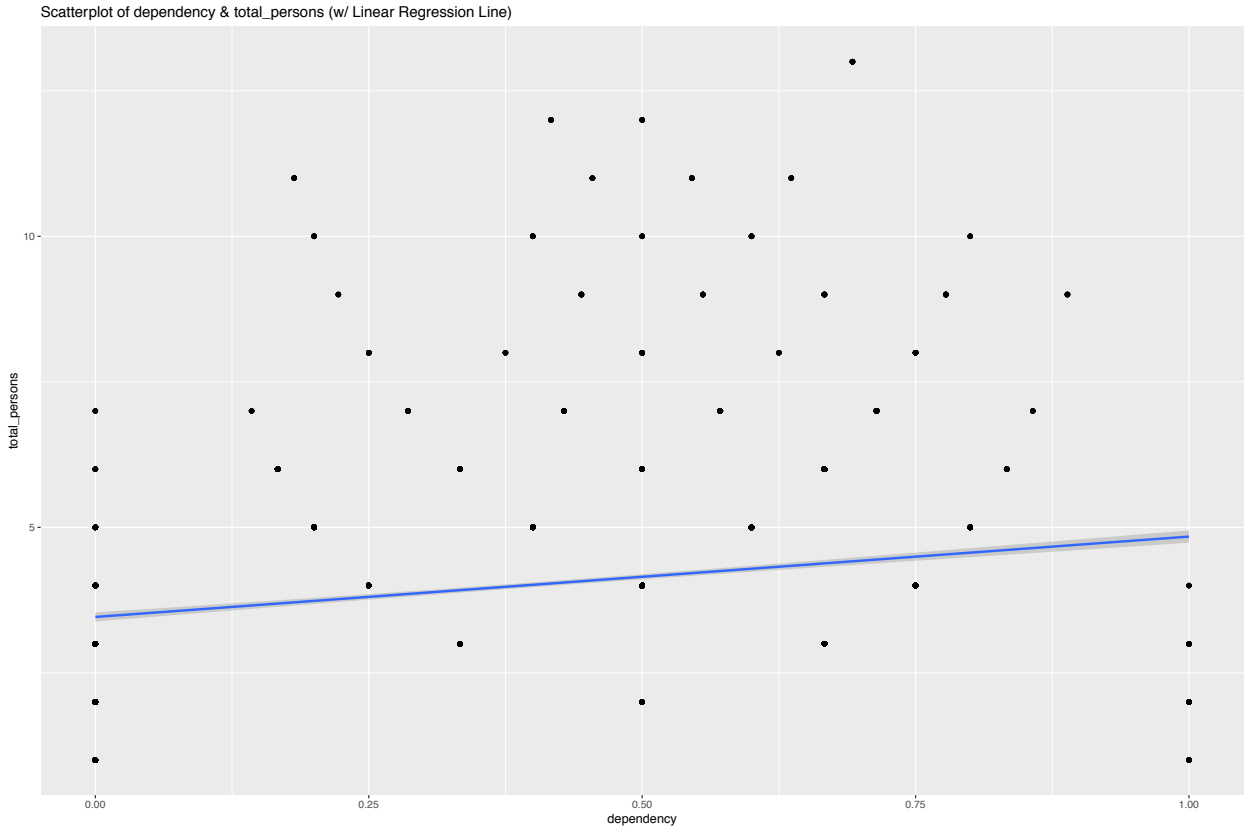

```
spg2
```

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




```
spg3
```

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

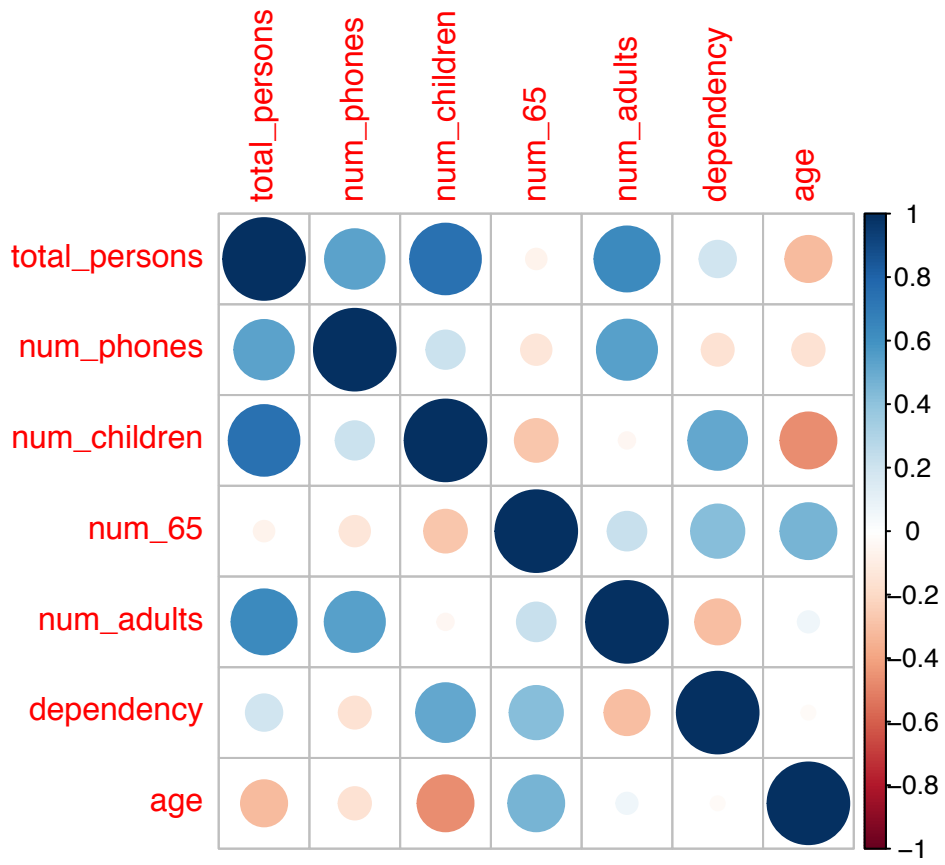


Correlation matrix for numerical features

```
train_cormx_df01 <- cor(train_x_num_df01_s1)
print(train_cormx_df01)
```

```
##                total_persons num_phones num_children      num_65  num_adults
## total_persons     1.00000000  0.5301462   0.74791846 -0.06328136  0.63308536
## num_phones        0.53014617  1.0000000   0.21993688 -0.13868083  0.54153466
## num_children      0.74791846  0.2199369   1.00000000 -0.27388480 -0.04033214
## num_65           -0.06328136 -0.1386808  -0.27388480  1.00000000  0.22413608
## num_adults        0.63308536  0.5415347  -0.04033214  0.22413608  1.00000000
## dependency        0.19873415 -0.1535990   0.51728356  0.42521492 -0.30408300
## age              -0.31858335 -0.1566772  -0.46929973  0.46841614  0.06772067
##                 dependency         age
## total_persons   0.19873415 -0.31858335
## num_phones     -0.15359903 -0.15667719
## num_children    0.51728356 -0.46929973
## num_65          0.42521492  0.46841614
## num_adults     -0.30408300  0.06772067
## dependency      1.00000000 -0.02933986
## age            -0.02933986  1.00000000
```

```
corrplot(train_cormx_df01)
```




```
# Send correlation matrix df to file for converting to table
write.csv(train_cormx_df01, "corr_mat.csv")
```

ADS502_Final_Classification_Modeling

```
crp <- read.csv("~/Desktop/502_Final_Train1.csv")
```

```
# splitting the data using a stratified random sampling approach
library(ggplot2)
library(lattice)
library(caret)
set.seed(333)
crp1 <- crp
indextrain <- createDataPartition(crp1$Target,
                                  p = .7,
                                  list = FALSE,
                                  times = 1)

crp1_train <- crp1[indextrain,]
crp1_test <- crp1[-indextrain,]
```

```
# Validation of train and test splits
# dim(crp1_train)[1]
# length(which(crp1_train$Target == 0))

train_proportion = length(which(crp1_train$Target == 0)) / dim(crp1_train)[1]
test_proportion = length(which(crp1_test$Target == 0)) / dim(crp1_test)[1]
train_proportion
```

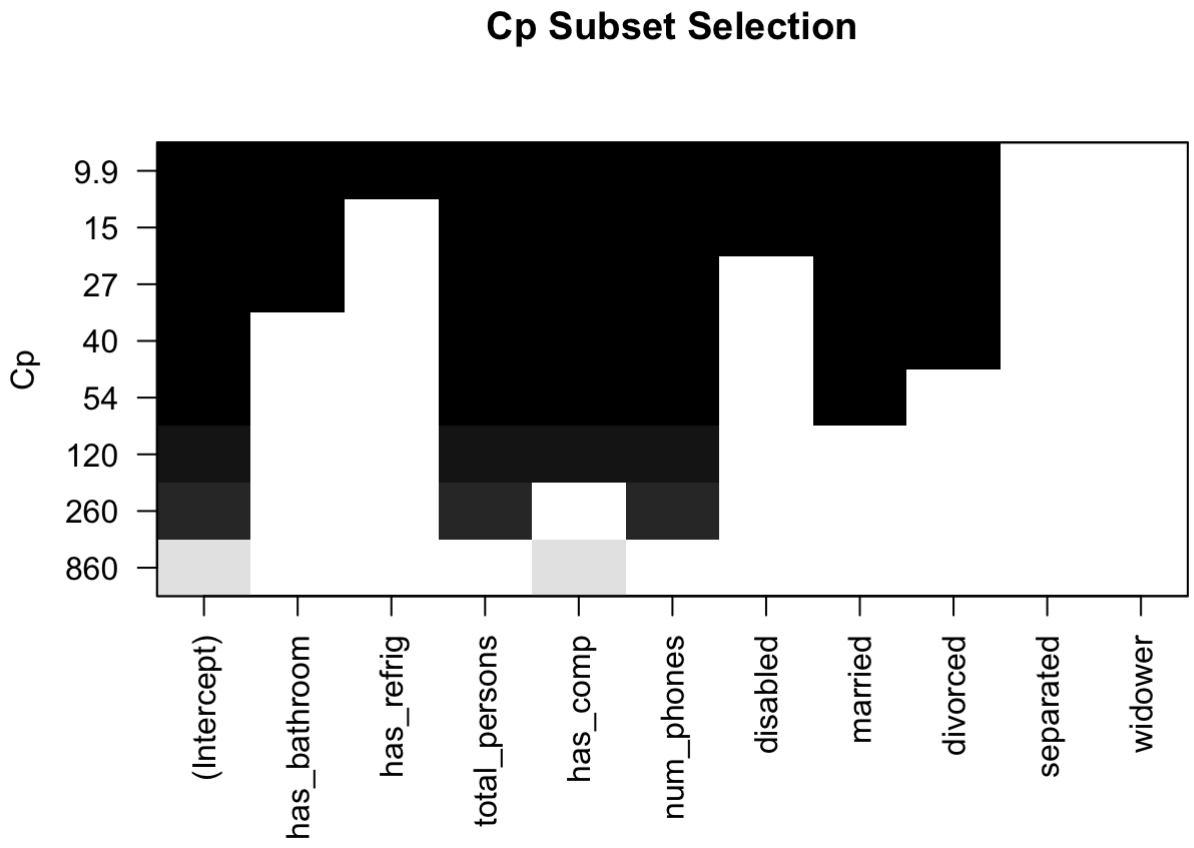
```
## [1] 0.6284006
```

```
test_proportion
```

```
## [1] 0.6250436
```

```
# splitting the predictor variables into groups w/target so that it is easier
# to process through Mallow's CP plots

group1 <- crp1_train[c(3:12)]
group1$Target <- crp1_train$Target
group2 <- crp1_train[c(13:22)]
group2$Target <- crp1_train$Target
group3 <- crp1_train[c(23:32)]
group3$Target <- crp1_train$Target
group4 <- crp1_train[c(33:41)]
```

```
# Mallow's CP Plot: G1

# install.packages("leaps")
library(leaps)
cp_gp1 = regsubsets(Target~.,
                        data = group1)
plot(cp_gp1,
     scale = "Cp",
     main = "Cp Subset Selection")
```

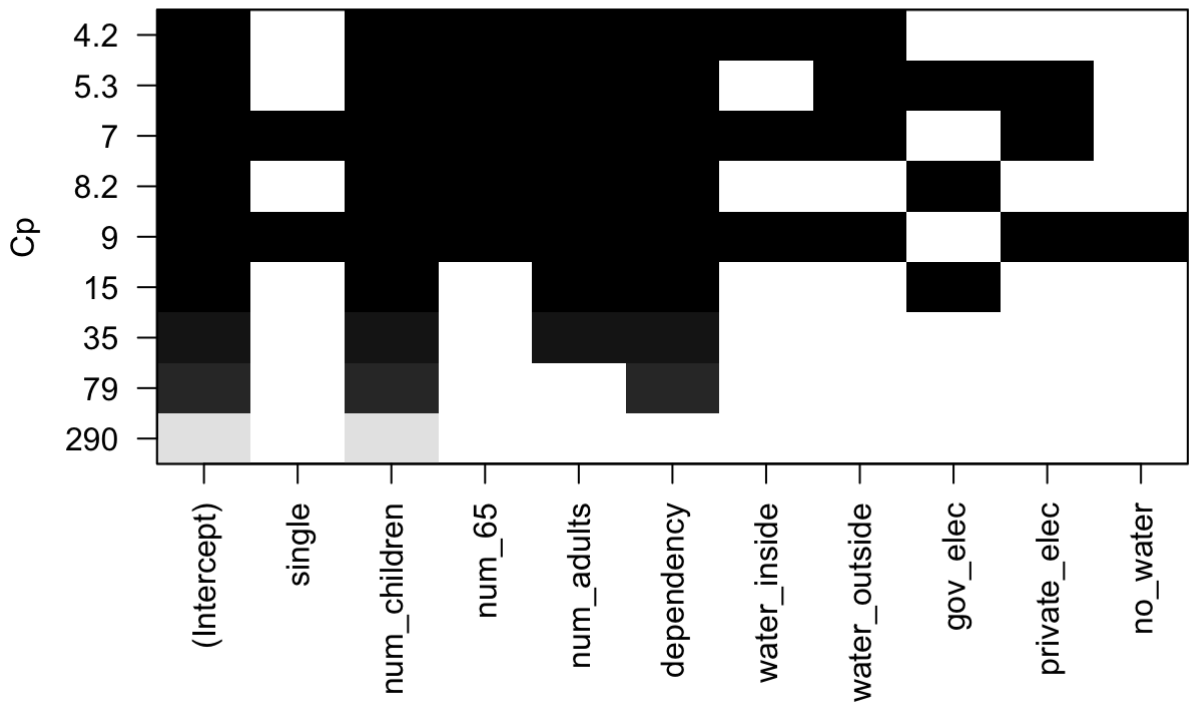



```
# Mallow's CP Plot: G2
cp_gp2 = regsubsets(Target~.,
                        data = group2)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 1 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
plot(cp_gp2,
     scale = "Cp",
     main = "Cp Subset Selection")
```

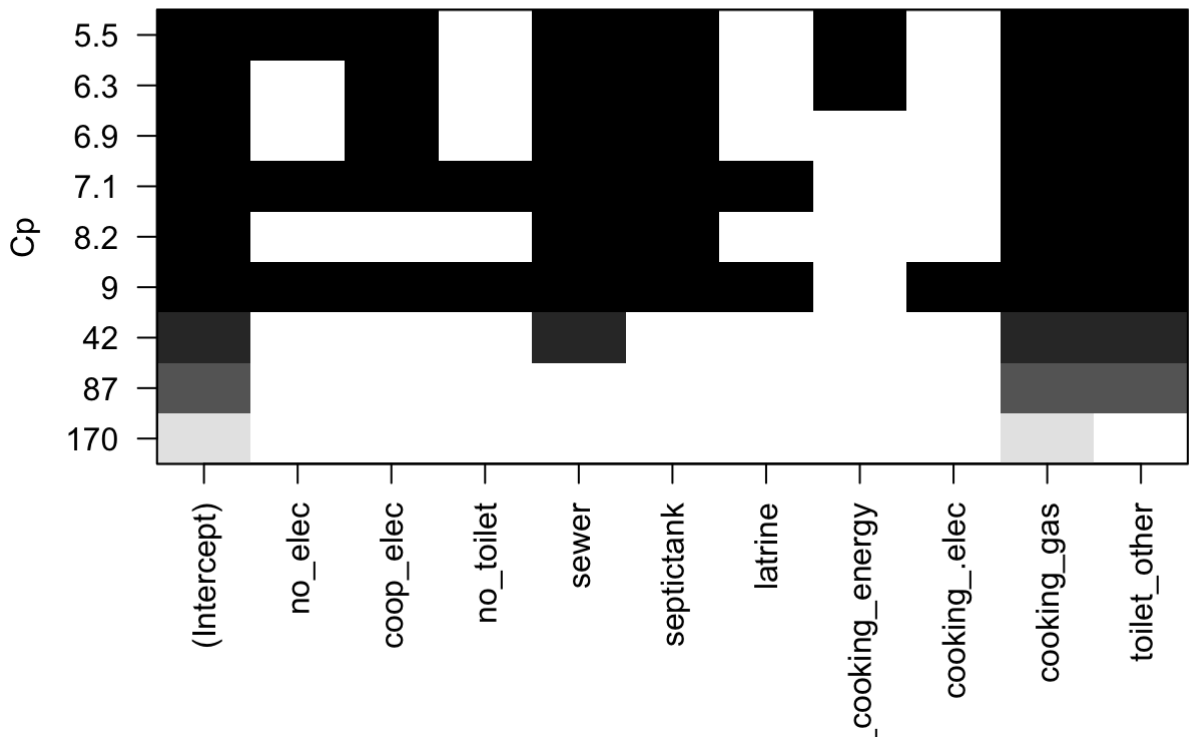



```
# Mallow's CP Plot: G3
cp_gp3 = regsubsets(Target~.,
                        data = group3)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 1 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
plot(cp_gp3,
     scale = "Cp",
     main = "Cp Subset Selection")
```

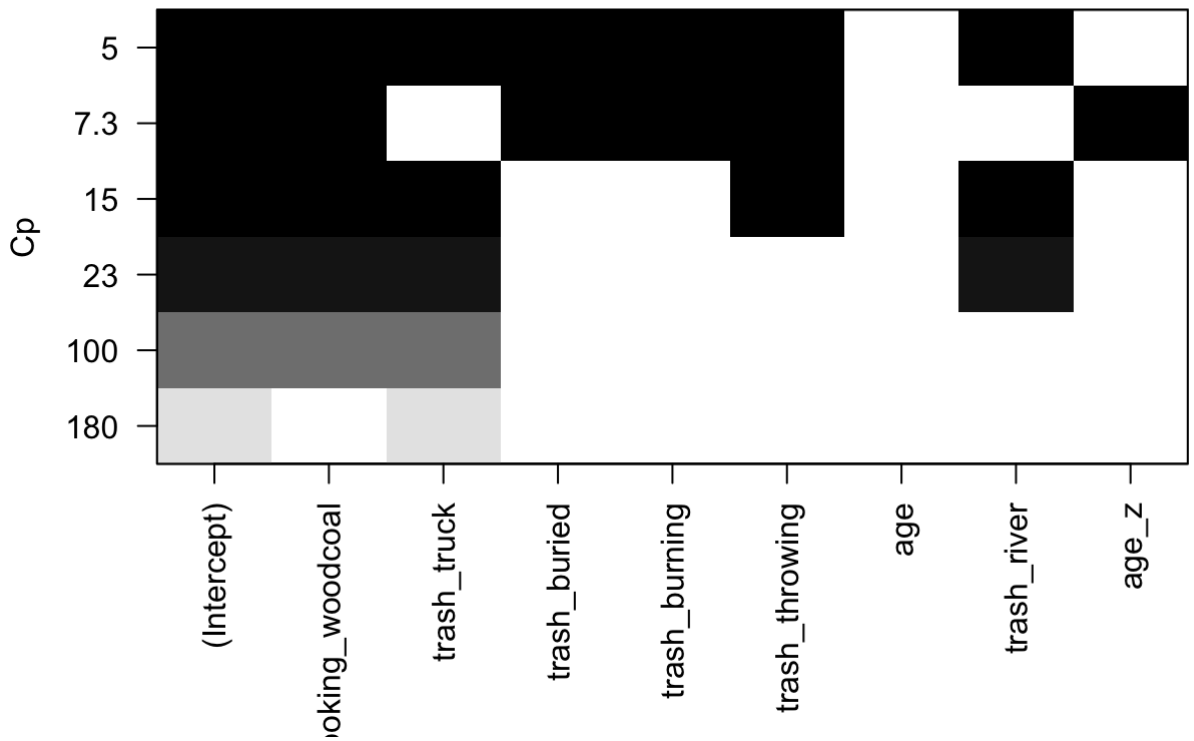



```
# Mallow's CP Plot: G4
cp_gp4 = regsubsets(Target~.,
                        data = group4)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 2 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
plot(cp_gp4,
     scale = "Cp",
     main = "Cp Subset Selection")
```

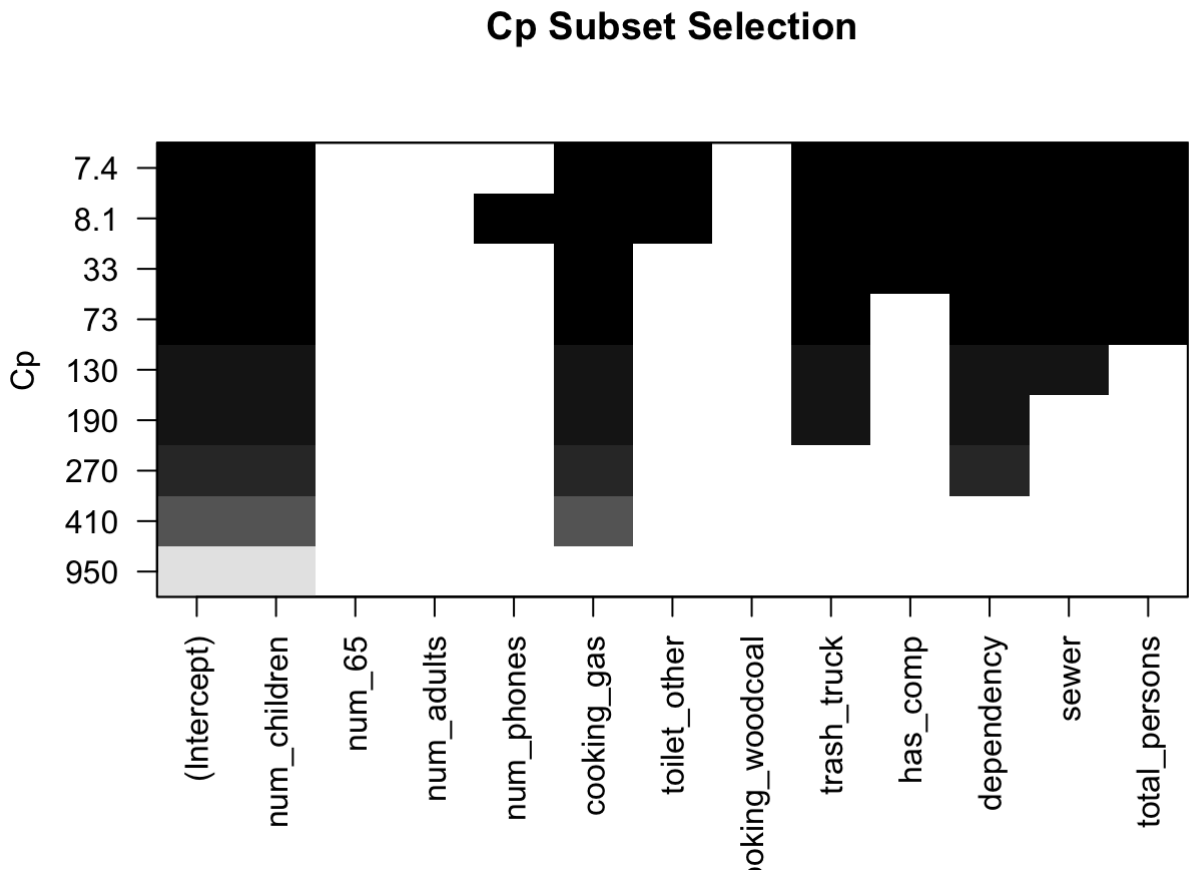



```
# Mallow's CP Plot:G5 (containing all the best variables from previous CP Plots)
group5 <- crp1_train[c(14:16)]
group5$Target <- crp1_train$Target
group5$total_persons <- crp1_train$total_persons
group5$num_phones <- crp1_train$num_phones
group5$cooking_gas <- crp1_train$cooking_gas
group5$toilet_other <- crp1_train$toilet_other
group5$cooking_woodcoal <- crp1_train$cooking_woodcoal
group5$trash_truck <- crp1_train$trash_truck
group5$has_comp <- crp1_train$has_comp
group5$num_children <- crp1_train$num_children
group5$dependency <- crp1_train$dependency
group5$sewer <- crp1_train$sewer

cp_gp5 = regsubsets(Target~.,
                        data = group5)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 1 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
plot(cp_gp5,
     scale = "Cp",
     main = "Cp Subset Selection")
```



```
# separating train and test
train <- crp1_train[c(14)]
train$Target <- crp1_train$Target
train$cooking_gas <- crp1_train$cooking_gas
train$trash_truck <- crp1_train$trash_truck
train$dependency <- crp1_train$dependency
train$sewer <- crp1_train$sewer
train$total_persons <- crp1_train$total_persons
train$Target <- as.factor(train$Target)

test <- crp1_test[c(14)]
test$Target <- crp1_test$Target
test$cooking_gas <- crp1_test$cooking_gas
test$trash_truck <- crp1_test$trash_truck
test$dependency <- crp1_test$dependency
test$sewer <- crp1_test$sewer
test$total_persons <- crp1_test$total_persons
test$Target <- as.factor(test$Target)

test.x <- crp1_test[c(14)]
test.x$cooking_gas <- crp1_test$cooking_gas
test.x$trash_truck <- crp1_test$trash_truck
test.x$dependency <- crp1_test$dependency
test.x$sewer <- crp1_test$sewer
test.x$total_persons <- crp1_test$total_persons
```

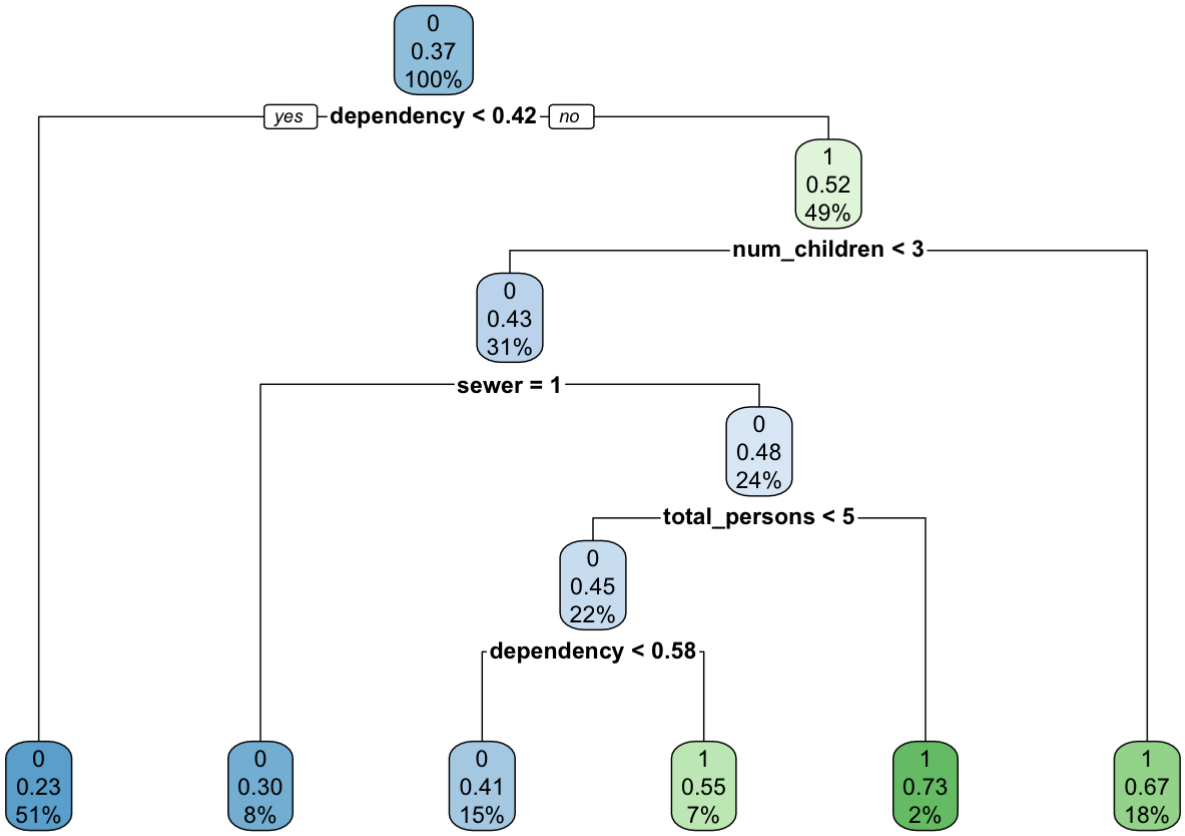
```
# Model 1: Random Forest Model
library(randomForest)
model1 <- randomForest(Target~., data = train)
```

```
# Evaluation of Model 1: Random Forest Model

# Confusion Matrix
train$rmpredict <- predict(object = model1, newdata = train)
test.ypredrf <- predict(object = model1, newdata = test.x)

t1 <- table(test$Target, test.ypredrf)
row.names(t1) <- c("Actual: No", "Actual: Poverty")
colnames(t1) <- c("Predicted: No", "Predicted: Poverty")
t1 <- addmargins(A = t1, FUN = list(Total = sum), quiet = TRUE)
t1
```

```
##                 test.ypredrf
##                  Predicted: No Predicted: Poverty Total
##   Actual: No              1618               174  1792
##   Actual: Poverty          575               500  1075
##   Total                   2193               674  2867
```

```
# separating train and test
train <- crp1_train[c(14)]
train$Target <- crp1_train$Target
train$cooking_gas <- crp1_train$cooking_gas
train$trash_truck <- crp1_train$trash_truck
train$dependency <- crp1_train$dependency
train$sewer <- crp1_train$sewer
train$total_persons <- crp1_train$total_persons
train$Target <- as.factor(train$Target)

test <- crp1_test[c(14)]
test$Target <- crp1_test$Target
test$cooking_gas <- crp1_test$cooking_gas
test$trash_truck <- crp1_test$trash_truck
test$dependency <- crp1_test$dependency
test$sewer <- crp1_test$sewer
test$total_persons <- crp1_test$total_persons
test$Target <- as.factor(test$Target)

test.x <- crp1_test[c(14)]
test.x$cooking_gas <- crp1_test$cooking_gas
test.x$trash_truck <- crp1_test$trash_truck
test.x$dependency <- crp1_test$dependency
test.x$sewer <- crp1_test$sewer
test.x$total_persons <- crp1_test$total_persons
```

```
# Model 2: CART Decision Tree
library(rpart)
model2 <- rpart(Target~., data = train)
library(rpart.plot)
rpart.plot(model2)
```

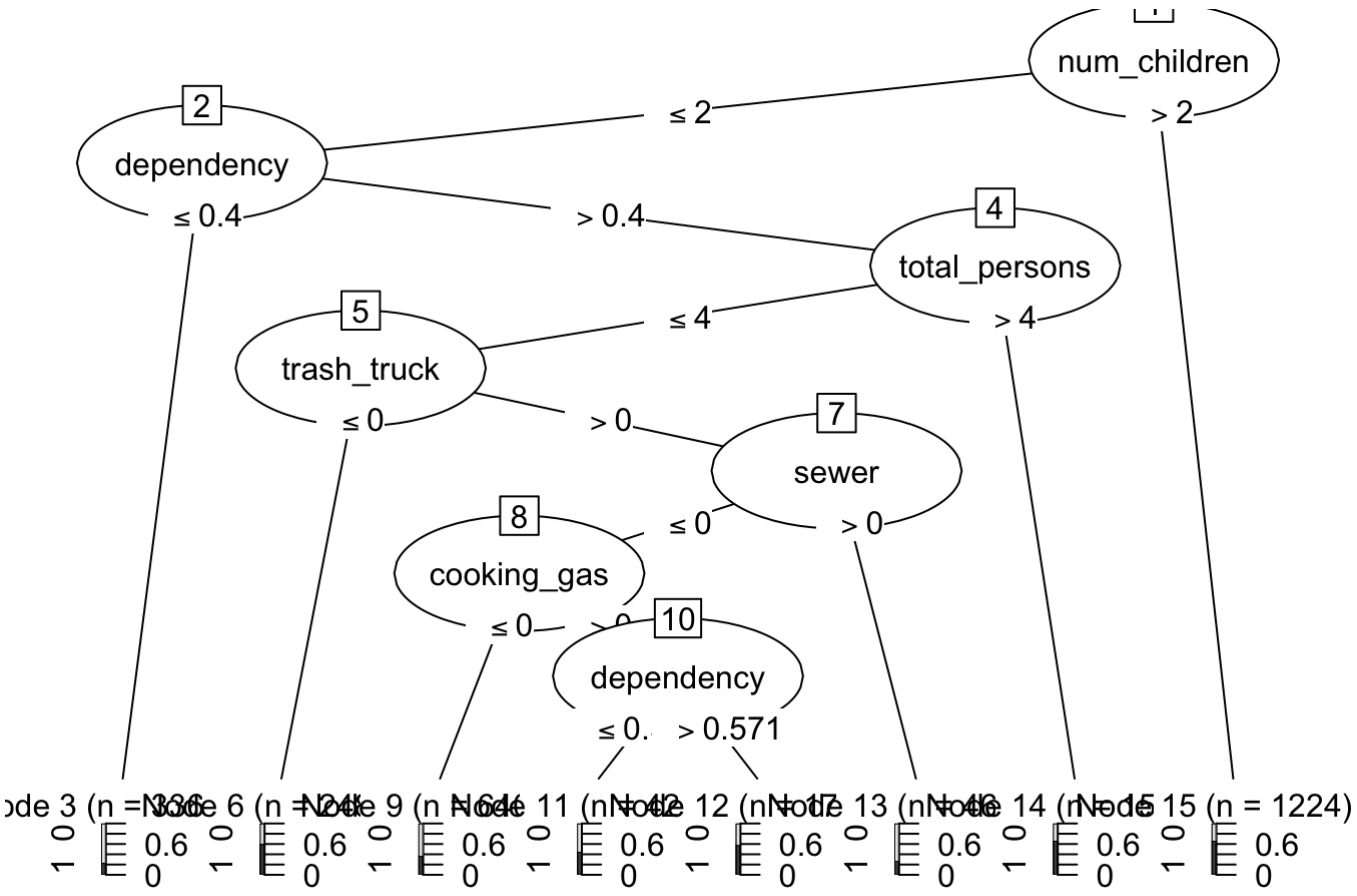



```
# Evaluation of Model 2: CART Decision Tree

# Confusion Matrix
train$cartpredict <- predict(object = model2, newdata = train)
test.ypredcart <- predict(object = model2,
                          newdata = test.x,
                          type = "class")

t2 <- table(test$Target, test.ypredcart)
row.names(t2) <- c("Actual: No", "Actual: Poverty")
colnames(t2) <- c("Predicted: No", "Predicted: Poverty")
t2 <- addmargins(A = t2, FUN = list(Total = sum), quiet = TRUE)
t2
```

```
##                  test.ypredcart
##                   Predicted: No Predicted: Poverty Total
##   Actual: No               1529                263  1792
##   Actual: Poverty           576                499  1075
##   Total                    2105                762  2867
```

```
# separating train and test
train <- crp1_train[c(14)]
train$Target <- crp1_train$Target
train$cooking_gas <- crp1_train$cooking_gas
train$trash_truck <- crp1_train$trash_truck
train$dependency <- crp1_train$dependency
train$sewer <- crp1_train$sewer
train$total_persons <- crp1_train$total_persons
train$Target <- as.factor(train$Target)

test <- crp1_test[c(14)]
test$Target <- crp1_test$Target
test$cooking_gas <- crp1_test$cooking_gas
test$trash_truck <- crp1_test$trash_truck
test$dependency <- crp1_test$dependency
test$sewer <- crp1_test$sewer
test$total_persons <- crp1_test$total_persons
test$Target <- as.factor(test$Target)

test.x <- crp1_test[c(14)]
test.x$cooking_gas <- crp1_test$cooking_gas
test.x$trash_truck <- crp1_test$trash_truck
test.x$dependency <- crp1_test$dependency
test.x$sewer <- crp1_test$sewer
test.x$total_persons <- crp1_test$total_persons
```

```
# Model 3: C5.0 Decision Tree
library (C50)
model3 <- C5.0(Target~., data = train, control = C5.0Control(minCases=75))

plot(model3, cex = 0.5)
```




```
# Evaluation of Model 3: C5.0 Decision Tree

# Confusion Matrix
train$c5predict <- predict(object = model3, newdata = train)

test.preddt <- predict(object = model3, newdata = test.x)

t3 <- table(test$Target, test.preddt)
row.names(t3) <- c("Actual: No", "Actual: Poverty")
colnames(t3) <- c("Predicted: No", "Predicted: Poverty")
t3 <- addmargins(A = t3, FUN = list(Total = sum), quiet = TRUE)
t3
```

```
##                  test.preddt
##                   Predicted: No Predicted: Poverty Total
##   Actual: No               1537                255  1792
##   Actual: Poverty           541                534  1075
##   Total                    2078                789  2867
```

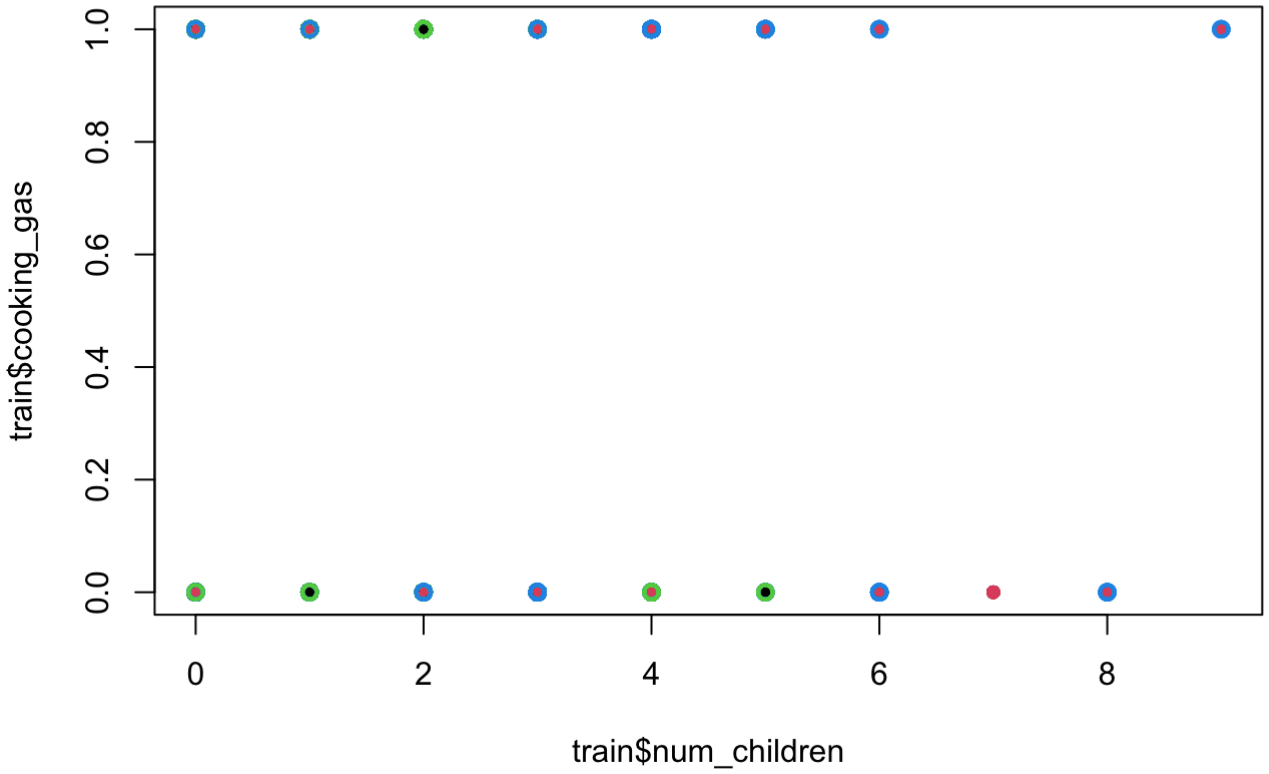
```
# separating train and test
train <- crp1_train[c(14)]
train$Target <- crp1_train$Target
train$cooking_gas <- crp1_train$cooking_gas
train$trash_truck <- crp1_train$trash_truck
train$dependency <- crp1_train$dependency
train$sewer <- crp1_train$sewer
train$total_persons <- crp1_train$total_persons
train$Target <- as.factor(train$Target)

test <- crp1_test[c(14)]
test$Target <- crp1_test$Target
test$cooking_gas <- crp1_test$cooking_gas
test$trash_truck <- crp1_test$trash_truck
test$dependency <- crp1_test$dependency
test$sewer <- crp1_test$sewer
test$total_persons <- crp1_test$total_persons
test$Target <- as.factor(test$Target)

test.x <- crp1_test[c(14)]
test.x$cooking_gas <- crp1_test$cooking_gas
test.x$trash_truck <- crp1_test$trash_truck
test.x$dependency <- crp1_test$dependency
test.x$sewer <- crp1_test$sewer
test.x$total_persons <- crp1_test$total_persons
```

```
# Model 4: K-Nearest Neighbor

library(class)
NROW(train)
```

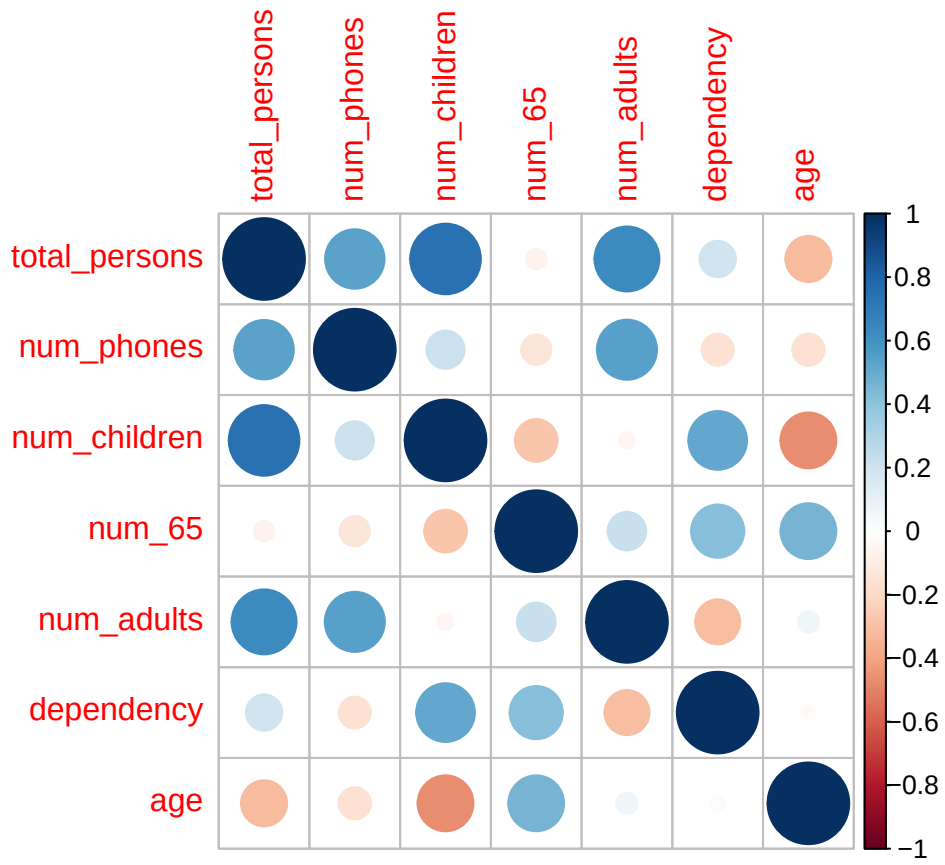
```
## [1] 6690
```

```
cl = test.x
knn.81 <- knn(train, test, cl= train$Target, k=81)
knn.3 <- knn(train, test, cl = train$Target, k=3)
```

```
plot(train$num_children, train$cooking_gas, col=as.numeric(train$Target),
     pch=19, cex = .8)
points(test$num_children, test$cooking_gas, col=as.numeric(knn.81)+2, lwd =2)
```




```
# Evaluation of Model 4: KNN 81
library(caret)
# confusionMatrix(table(knn.81 , test$Target))
```




```
# Send correlation matrix df to file for converting to table
write.csv(train_cormx_df01, "corr_mat.csv")
```

Model development using training set for evaluation on test data set

Train & evaluate multiple classification models

- **Poverty Status Index (PSI) = 0/-:** An individual who is not vulnerable to being impoverished.
- **PSI = 1/+:** An individual who is either vulnerable to being impoverished, moderately impoverished, or extremely impoverished.

Model 1 (M1): Random Forest Decision Trees (DTs)

```
# Train model
train_m1v1_rf <- randomForest(Target~., data = train_xy01_df01)

# Use trained model to predict y on test data
test_m1v1_rf_y_pred_df01 <- predict(object = train_m1v1_rf, newdata = test_x01_df01, type
↪  = "class")

# Create contingency table to review results
test_m1v1_rf_y_pred_tbl01 <- table(test_df01$Target, test_m1v1_rf_y_pred_df01)
row.names(test_m1v1_rf_y_pred_tbl01) <- c("Actual: PSI = 0 (-)",
```

```
                                          "Actual: PSI = 1 (+)")
colnames(test_m1v1_rf_y_pred_tbl01) <- c("Predicted: PSI = 0 (-)",
                                         "Predicted: PSI = 1 (+)")
test_m1v1_rf_y_pred_tbl01 <- addmargins(A = test_m1v1_rf_y_pred_tbl01,
                                        FUN = list(Total = sum), quiet = TRUE)
print(test_m1v1_rf_y_pred_tbl01)
```

```
##                      test_m1v1_rf_y_pred_df01
##                       Predicted: PSI = 0 (-) Predicted: PSI = 1 (+) Total
##   Actual: PSI = 0 (-)                   1617                     175  1792
##   Actual: PSI = 1 (+)                    574                     501  1075
##   Total                                 2191                     676  2867
```

```
# Pull values from confusion matrix table
test_m1v1_rf_y_pred_tp01 <- test_m1v1_rf_y_pred_tbl01[2, 2]
test_m1v1_rf_y_pred_fn01 <- test_m1v1_rf_y_pred_tbl01[2, 1]
test_m1v1_rf_y_pred_fp01 <- test_m1v1_rf_y_pred_tbl01[1, 2]
test_m1v1_rf_y_pred_tn01 <- test_m1v1_rf_y_pred_tbl01[1, 1]
print(test_m1v1_rf_y_pred_tp01)
```

```
## [1] 501
```

```
print(test_m1v1_rf_y_pred_fn01)
```

```
## [1] 574
```

```
print(test_m1v1_rf_y_pred_fp01)
```

```
## [1] 175
```

```
print(test_m1v1_rf_y_pred_tn01)
```

```
## [1] 1617
```

```
# ------------------------------------------------------------------------------
# Use trained model to predict y on train data for comparison
train_m1v1_rf_y_pred_df01 <- predict(object = train_m1v1_rf, newdata = train_x01_df01,
↪  type = "class")

# Create contingency table to review results
train_m1v1_rf_y_pred_tbl01 <- table(train_df01$Target, train_m1v1_rf_y_pred_df01)
row.names(train_m1v1_rf_y_pred_tbl01) <- c("Actual: PSI = 0 (-)",
                                           "Actual: PSI = 1 (+)")
colnames(train_m1v1_rf_y_pred_tbl01) <- c("Predicted: PSI = 0 (-)",
                                          "Predicted: PSI = 1 (+)")
train_m1v1_rf_y_pred_tbl01 <- addmargins(A = train_m1v1_rf_y_pred_tbl01,
                                         FUN = list(Total = sum), quiet = TRUE)
print(train_m1v1_rf_y_pred_tbl01)
```

```
##                      train_m1v1_rf_y_pred_df01
##                       Predicted: PSI = 0 (-) Predicted: PSI = 1 (+) Total
##   Actual: PSI = 0 (-)                   3811                     393  4204
##   Actual: PSI = 1 (+)                   1307                    1179  2486
##   Total                                 5118                    1572  6690
```

```
# Pull values from confusion matrix table
train_m1v1_rf_y_pred_tp01 <- train_m1v1_rf_y_pred_tbl01[2, 2]
train_m1v1_rf_y_pred_fn01 <- train_m1v1_rf_y_pred_tbl01[2, 1]
train_m1v1_rf_y_pred_fp01 <- train_m1v1_rf_y_pred_tbl01[1, 2]
train_m1v1_rf_y_pred_tn01 <- train_m1v1_rf_y_pred_tbl01[1, 1]
print(train_m1v1_rf_y_pred_tp01)
```

```
## [1] 1179
```

```
print(train_m1v1_rf_y_pred_fn01)
```

```
## [1] 1307
```

```
print(train_m1v1_rf_y_pred_fp01)
```

```
## [1] 393
```

```
print(train_m1v1_rf_y_pred_tn01)
```

```
## [1] 3811
```

```
# Generate classification evaluation measures
test_m1v1_rf_acc01 <- (test_m1v1_rf_y_pred_tp01 + test_m1v1_rf_y_pred_tn01) /
  (test_m1v1_rf_y_pred_tp01 + test_m1v1_rf_y_pred_fn01 + test_m1v1_rf_y_pred_fp01 +
↪  test_m1v1_rf_y_pred_tn01)

test_m1v1_rf_err_rate01 <- 1 - test_m1v1_rf_acc01
test_m1v1_rf_recall01 <- test_m1v1_rf_y_pred_tp01 / (test_m1v1_rf_y_pred_tp01 +
↪  test_m1v1_rf_y_pred_fn01)
test_m1v1_rf_tnr01 <- test_m1v1_rf_y_pred_tn01 / (test_m1v1_rf_y_pred_tn01 +
↪  test_m1v1_rf_y_pred_fp01)
test_m1v1_rf_prec01 <- test_m1v1_rf_y_pred_tp01 / (test_m1v1_rf_y_pred_tp01 +
↪  test_m1v1_rf_y_pred_fp01)
test_m1v1_rf_f101 <- (1 + 1^2) * ((test_m1v1_rf_prec01 * test_m1v1_rf_recall01) /
                                  ((1^2 * test_m1v1_rf_prec01) +
↪  test_m1v1_rf_recall01))
test_m1v1_rf_plr01 <- test_m1v1_rf_recall01 / (1 - test_m1v1_rf_tnr01)


train_m1v1_rf_acc01 <- (train_m1v1_rf_y_pred_tp01 + train_m1v1_rf_y_pred_tn01) /
  (train_m1v1_rf_y_pred_tp01 + train_m1v1_rf_y_pred_fn01 + train_m1v1_rf_y_pred_fp01 +
↪  train_m1v1_rf_y_pred_tn01)

train_m1v1_rf_err_rate01 <- 1 - train_m1v1_rf_acc01
train_m1v1_rf_recall01 <- train_m1v1_rf_y_pred_tp01 / (train_m1v1_rf_y_pred_tp01 +
↪  train_m1v1_rf_y_pred_fn01)
train_m1v1_rf_tnr01 <- train_m1v1_rf_y_pred_tn01 / (train_m1v1_rf_y_pred_tn01 +
↪  train_m1v1_rf_y_pred_fp01)
train_m1v1_rf_prec01 <- train_m1v1_rf_y_pred_tp01 / (train_m1v1_rf_y_pred_tp01 +
↪  train_m1v1_rf_y_pred_fp01)
train_m1v1_rf_f101 <- (1 + 1^2) * ((train_m1v1_rf_prec01 * train_m1v1_rf_recall01) /
                                   ((1^2 * train_m1v1_rf_prec01) +
↪  train_m1v1_rf_recall01))
train_m1v1_rf_plr01 <- train_m1v1_rf_recall01 / (1 - train_m1v1_rf_tnr01)
```

```r
# ============================================================================
# Generate evaluation measures table
measure_m1v1_rf_name <- c("M1: Random Forest DT",
                          "Baseline Accuracy",
                          "Accuracy",
                          "Error Rate",
                          "Sensitivity/TPR/Recall",
                          "Specificity/TNR",
                          "Precision",
                          "F1",
                          "Positive Liklihood Ratio"
                          )

measure_m1v1_rf_val01 <- c("Traing Data set",
                           paste0(round((4204 / 6690) * 100, 1), "%"),
                           paste0(round(train_m1v1_rf_acc01 * 100, 1), "%"),
                           paste0(round(train_m1v1_rf_err_rate01 * 100, 1), "%"),
                           paste0(round(train_m1v1_rf_recall01 * 100, 1), "%"),
                           paste0(round(train_m1v1_rf_tnr01 * 100, 1), "%"),
                           paste0(round(train_m1v1_rf_prec01 * 100, 1), "%"),
                           paste0(round(train_m1v1_rf_f101 * 100, 1), "%"),
                           paste0(round(train_m1v1_rf_plr01 * 1, 2), "")
                           )

measure_m1v1_rf_val02 <- c("Test Data set",
                           paste0(round((1792 / 2867) * 100, 1), "%"),
                           paste0(round(test_m1v1_rf_acc01 * 100, 1), "%"),
                           paste0(round(test_m1v1_rf_err_rate01 * 100, 1), "%"),
                           paste0(round(test_m1v1_rf_recall01 * 100, 1), "%"),
                           paste0(round(test_m1v1_rf_tnr01 * 100, 1), "%"),
                           paste0(round(test_m1v1_rf_prec01 * 100, 1), "%"),
                           paste0(round(test_m1v1_rf_f101 * 100, 1), "%"),
                           paste0(round(test_m1v1_rf_plr01 * 1, 2), "")
                           )

measure_m1v1_rf_tbl <- data.frame(measure_m1v1_rf_name, measure_m1v1_rf_val01,
↪  measure_m1v1_rf_val02)
print(measure_m1v1_rf_tbl)
```

```
##       measure_m1v1_rf_name measure_m1v1_rf_val01 measure_m1v1_rf_val02
## 1     M1: Random Forest DT       Traing Data set         Test Data set
## 2        Baseline Accuracy                 62.8%                 62.5%
## 3                 Accuracy                 74.6%                 73.9%
## 4               Error Rate                 25.4%                 26.1%
## 5   Sensitivity/TPR/Recall                 47.4%                 46.6%
## 6          Specificity/TNR                 90.7%                 90.2%
## 7                Precision                   75%                 74.1%
## 8                       F1                 58.1%                 57.2%
## 9 Positive Liklihood Ratio                  5.07                  4.77
```

```r
# Confirm confusion matrix results
confusionMatrix(table(test_m1v1_rf_y_pred_df01, test_df01$Target), positive = "1")
```

```
## Confusion Matrix and Statistics
```

```
##
##
## test_m1v1_rf_y_pred_df01    0    1
##                        0 1617  574
##                        1  175  501
##
##                Accuracy : 0.7388
##                  95% CI : (0.7223, 0.7548)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3979
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4660
##             Specificity : 0.9023
##          Pos Pred Value : 0.7411
##          Neg Pred Value : 0.7380
##              Prevalence : 0.3750
##          Detection Rate : 0.1747
##    Detection Prevalence : 0.2358
##       Balanced Accuracy : 0.6842
##
##        'Positive' Class : 1
##
```

```
print(paste0("Number of 0s in Test Set Original Target = ", length(which(test_df01$Target
↪  == 0))))
```

```
## [1] "Number of 0s in Test Set Original Target = 1792"
```

```
print(paste0("Number of 0s in Test Set Predicted = ",
↪  length(which(test_m1v1_rf_y_pred_df01 == 0))))
```

```
## [1] "Number of 0s in Test Set Predicted = 2191"
```

```
print(paste0("Number of 1s in Test Set Original Target = ", length(which(test_df01$Target
↪  == 1))))
```

```
## [1] "Number of 1s in Test Set Original Target = 1075"
```

```
print(paste0("Number of 1s in Test Set Predicted = ",
↪  length(which(test_m1v1_rf_y_pred_df01 == 1))))
```

```
## [1] "Number of 1s in Test Set Predicted = 676"
```

```
test_m1v1_rf_measure_df01 <- data.frame(test_df01$Target, test_m1v1_rf_y_pred_df01)
print(head(test_m1v1_rf_measure_df01))
```

```
##   test_df01.Target test_m1v1_rf_y_pred_df01
## 1                0                        0
## 2                0                        0
## 3                0                        0
## 4                0                        0
## 5                0                        0
```

```
## 6                0                          0
```

```
print(paste0("Number of True 1s (TP) in Test Set = ",
             length(which(test_m1v1_rf_measure_df01$test_df01.Target == 1 &
                            test_m1v1_rf_measure_df01$test_m1v1_rf_y_pred_df01 == 1))))
```

```
## [1] "Number of True 1s (TP) in Test Set = 501"
```

```
print(paste0("Number of False 0s (FN) in Test Set = ",
             length(which(test_m1v1_rf_measure_df01$test_df01.Target == 1 &
                            test_m1v1_rf_measure_df01$test_m1v1_rf_y_pred_df01 == 0))))
```

```
## [1] "Number of False 0s (FN) in Test Set = 574"
```

```
print(paste0("Number of False 1s (FP) in Test Set = ",
             length(which(test_m1v1_rf_measure_df01$test_df01.Target == 0 &
                            test_m1v1_rf_measure_df01$test_m1v1_rf_y_pred_df01 == 1))))
```

```
## [1] "Number of False 1s (FP) in Test Set = 175"
```

```
print(paste0("Number of True 0s (TN) in Test Set = ",
             length(which(test_m1v1_rf_measure_df01$test_df01.Target == 0 &
                            test_m1v1_rf_measure_df01$test_m1v1_rf_y_pred_df01 == 0))))
```

```
## [1] "Number of True 0s (TN) in Test Set = 1617"
```

Model 2 (M2): CART DT

```
# Train model
train_m2v1_cart <- rpart(Target~., data = train_xy01_df01)

# Use trained model to predict y on test data
test_m2v1_cart_y_pred_df01 <- predict(object = train_m2v1_cart, newdata = test_x01_df01,
↪  type = "class")

# Create contingency table to review results
test_m2v1_cart_y_pred_tbl01 <- table(test_df01$Target, test_m2v1_cart_y_pred_df01)
row.names(test_m2v1_cart_y_pred_tbl01) <- c("Actual: PSI = 0 (-)",
                                            "Actual: PSI = 1 (+)")
colnames(test_m2v1_cart_y_pred_tbl01) <- c("Predicted: PSI = 0 (-)",
                                           "Predicted: PSI = 1 (+)")
test_m2v1_cart_y_pred_tbl01 <- addmargins(A = test_m2v1_cart_y_pred_tbl01,
                                          FUN = list(Total = sum), quiet = TRUE)
print(test_m2v1_cart_y_pred_tbl01)
```

```
##                       test_m2v1_cart_y_pred_df01
##                        Predicted: PSI = 0 (-) Predicted: PSI = 1 (+) Total
##   Actual: PSI = 0 (-)                   1529                     263  1792
##   Actual: PSI = 1 (+)                    576                     499  1075
##   Total                                 2105                     762  2867
```

```
## Pull values from confusion matrix table
test_m2v1_cart_y_pred_tp01 <- test_m2v1_cart_y_pred_tbl01[2, 2]
test_m2v1_cart_y_pred_fn01 <- test_m2v1_cart_y_pred_tbl01[2, 1]
test_m2v1_cart_y_pred_fp01 <- test_m2v1_cart_y_pred_tbl01[1, 2]
test_m2v1_cart_y_pred_tn01 <- test_m2v1_cart_y_pred_tbl01[1, 1]
```

```
print(test_m2v1_cart_y_pred_tp01)
```

```
## [1] 499
```

```
print(test_m2v1_cart_y_pred_fn01)
```

```
## [1] 576
```

```
print(test_m2v1_cart_y_pred_fp01)
```

```
## [1] 263
```

```
print(test_m2v1_cart_y_pred_tn01)
```

```
## [1] 1529
```

```
# ------------------------------------------------------------------------------
# Use trained model to predict y on train data
train_m2v1_cart_y_pred_df01 <- predict(object = train_m2v1_cart, newdata =
↪  train_x01_df01, type = "class")

# Create contingency table to review results
train_m2v1_cart_y_pred_tbl01 <- table(train_df01$Target, train_m2v1_cart_y_pred_df01)
row.names(train_m2v1_cart_y_pred_tbl01) <- c("Actual: PSI = 0 (-)",
                                             "Actual: PSI = 1 (+)")
colnames(train_m2v1_cart_y_pred_tbl01) <- c("Predicted: PSI = 0 (-)",
                                            "Predicted: PSI = 1 (+)")
train_m2v1_cart_y_pred_tbl01 <- addmargins(A = train_m2v1_cart_y_pred_tbl01,
                                           FUN = list(Total = sum), quiet = TRUE)
print(train_m2v1_cart_y_pred_tbl01)
```

```
##                       train_m2v1_cart_y_pred_df01
##                        Predicted: PSI = 0 (-) Predicted: PSI = 1 (+) Total
##   Actual: PSI = 0 (-)                   3565                    639  4204
##   Actual: PSI = 1 (+)                   1346                   1140  2486
##   Total                                 4911                   1779  6690
```

```
# Pull values from confusion matrix table
train_m2v1_cart_y_pred_tp01 <- train_m2v1_cart_y_pred_tbl01[2, 2]
train_m2v1_cart_y_pred_fn01 <- train_m2v1_cart_y_pred_tbl01[2, 1]
train_m2v1_cart_y_pred_fp01 <- train_m2v1_cart_y_pred_tbl01[1, 2]
train_m2v1_cart_y_pred_tn01 <- train_m2v1_cart_y_pred_tbl01[1, 1]
print(train_m2v1_cart_y_pred_tp01)
```

```
## [1] 1140
```

```
print(train_m2v1_cart_y_pred_fn01)
```

```
## [1] 1346
```

```
print(train_m2v1_cart_y_pred_fp01)
```

```
## [1] 639
```

```
print(train_m2v1_cart_y_pred_tn01)
```

```
## [1] 3565
```

```
# Generate classification evaluation measures
test_m2v1_cart_acc01 <- (test_m2v1_cart_y_pred_tp01 + test_m2v1_cart_y_pred_tn01) /
  (test_m2v1_cart_y_pred_tp01 + test_m2v1_cart_y_pred_fn01 + test_m2v1_cart_y_pred_fp01 +
↪  test_m2v1_cart_y_pred_tn01)

test_m2v1_cart_err_rate01 <- 1 - test_m2v1_cart_acc01
test_m2v1_cart_recall01 <- test_m2v1_cart_y_pred_tp01 / (test_m2v1_cart_y_pred_tp01 +
↪  test_m2v1_cart_y_pred_fn01)
test_m2v1_cart_tnr01 <- test_m2v1_cart_y_pred_tn01 / (test_m2v1_cart_y_pred_tn01 +
↪  test_m2v1_cart_y_pred_fp01)
test_m2v1_cart_prec01 <- test_m2v1_cart_y_pred_tp01 / (test_m2v1_cart_y_pred_tp01 +
↪  test_m2v1_cart_y_pred_fp01)
test_m2v1_cart_f101 <- (1 + 1^2) * ((test_m2v1_cart_prec01 * test_m2v1_cart_recall01) /
                                    ((1^2 * test_m2v1_cart_prec01) +
↪  test_m2v1_cart_recall01))
test_m2v1_cart_plr01 <- test_m2v1_cart_recall01 / (1 - test_m2v1_cart_tnr01)


train_m2v1_cart_acc01 <- (train_m2v1_cart_y_pred_tp01 + train_m2v1_cart_y_pred_tn01) /
  (train_m2v1_cart_y_pred_tp01 + train_m2v1_cart_y_pred_fn01 +
↪  train_m2v1_cart_y_pred_fp01 + train_m2v1_cart_y_pred_tn01)

train_m2v1_cart_err_rate01 <- 1 - train_m2v1_cart_acc01
train_m2v1_cart_recall01 <- train_m2v1_cart_y_pred_tp01 / (train_m2v1_cart_y_pred_tp01 +
↪  train_m2v1_cart_y_pred_fn01)
train_m2v1_cart_tnr01 <- train_m2v1_cart_y_pred_tn01 / (train_m2v1_cart_y_pred_tn01 +
↪  train_m2v1_cart_y_pred_fp01)
train_m2v1_cart_prec01 <- train_m2v1_cart_y_pred_tp01 / (train_m2v1_cart_y_pred_tp01 +
↪  train_m2v1_cart_y_pred_fp01)
train_m2v1_cart_f101 <- (1 + 1^2) * ((train_m2v1_cart_prec01 * train_m2v1_cart_recall01)
↪  /
                                     ((1^2 * train_m2v1_cart_prec01) +
↪  train_m2v1_cart_recall01))
train_m2v1_cart_plr01 <- train_m2v1_cart_recall01 / (1 - train_m2v1_cart_tnr01)

# ============================================================================
# Generate evaluation measures table
measure_m2v1_cart_name <- c("M2: CART DT",
                            "Baseline Accuracy",
                            "Accuracy",
                            "Error Rate",
                            "Sensitivity/TPR/Recall",
                            "Specificity/TNR",
                            "Precision",
                            "F1",
                            "Positive Liklihood Ratio"
                            )

measure_m2v1_cart_val01 <- c("Traing Data set",
                             paste0(round((4204 / 6690) * 100, 1), "%"),
                             paste0(round(train_m2v1_cart_acc01 * 100, 1), "%"),
                             paste0(round(train_m2v1_cart_err_rate01 * 100, 1), "%"),
                             paste0(round(train_m2v1_cart_recall01 * 100, 1), "%"),
```

```
                          paste0(round(train_m2v1_cart_tnr01 * 100, 1), "%"),
                          paste0(round(train_m2v1_cart_prec01 * 100, 1), "%"),
                          paste0(round(train_m2v1_cart_f101 * 100, 1), "%"),
                          paste0(round(train_m2v1_cart_plr01 * 1, 2), "")
                          )

measure_m2v1_cart_val02 <- c("Test Data set",
                          paste0(round((1792 / 2867) * 100, 1), "%"),
                          paste0(round(test_m2v1_cart_acc01 * 100, 1), "%"),
                          paste0(round(test_m2v1_cart_err_rate01 * 100, 1), "%"),
                          paste0(round(test_m2v1_cart_recall01 * 100, 1), "%"),
                          paste0(round(test_m2v1_cart_tnr01 * 100, 1), "%"),
                          paste0(round(test_m2v1_cart_prec01 * 100, 1), "%"),
                          paste0(round(test_m2v1_cart_f101 * 100, 1), "%"),
                          paste0(round(test_m2v1_cart_plr01 * 1, 2), "")
                          )

measure_m2v1_cart_tbl <- data.frame(measure_m2v1_cart_name, measure_m2v1_cart_val01,
↪  measure_m2v1_cart_val02)
print(measure_m2v1_cart_tbl)
```

```
##     measure_m2v1_cart_name measure_m2v1_cart_val01 measure_m2v1_cart_val02
## 1              M2: CART DT         Traing Data set           Test Data set
## 2        Baseline Accuracy                   62.8%                   62.5%
## 3                 Accuracy                   70.3%                   70.7%
## 4               Error Rate                   29.7%                   29.3%
## 5   Sensitivity/TPR/Recall                   45.9%                   46.4%
## 6          Specificity/TNR                   84.8%                   85.3%
## 7                Precision                   64.1%                   65.5%
## 8                       F1                   53.5%                   54.3%
## 9 Positive Liklihood Ratio                    3.02                    3.16
```

```
# Confirm confusion matrix results
confusionMatrix(table(test_m2v1_cart_y_pred_df01, test_df01$Target), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##
## test_m2v1_cart_y_pred_df01    0    1
##                          0 1529  576
##                          1  263  499
##
##                Accuracy : 0.7074
##                  95% CI : (0.6903, 0.724)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3371
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4642
##             Specificity : 0.8532
##          Pos Pred Value : 0.6549
```

```
##          Neg Pred Value : 0.7264
##              Prevalence : 0.3750
##          Detection Rate : 0.1740
##    Detection Prevalence : 0.2658
##       Balanced Accuracy : 0.6587
##
##        'Positive' Class : 1
##
```

```
print(paste0("Number of 0s in Test Set Original Target = ", length(which(test_df01$Target
↪  == 0))))
```

```
## [1] "Number of 0s in Test Set Original Target = 1792"
```

```
print(paste0("Number of 0s in Test Set Predicted = ",
↪  length(which(test_m2v1_cart_y_pred_df01 == 0))))
```

```
## [1] "Number of 0s in Test Set Predicted = 2105"
```

```
print(paste0("Number of 1s in Test Set Original Target = ", length(which(test_df01$Target
↪  == 1))))
```

```
## [1] "Number of 1s in Test Set Original Target = 1075"
```

```
print(paste0("Number of 1s in Test Set Predicted = ",
↪  length(which(test_m2v1_cart_y_pred_df01 == 1))))
```

```
## [1] "Number of 1s in Test Set Predicted = 762"
```

```
test_m2v1_cart_measure_df01 <- data.frame(test_df01$Target, test_m2v1_cart_y_pred_df01)
print(head(test_m2v1_cart_measure_df01))
```

```
##   test_df01.Target test_m2v1_cart_y_pred_df01
## 1                0                          0
## 2                0                          0
## 3                0                          0
## 4                0                          0
## 5                0                          0
## 6                0                          0
```

```
print(paste0("Number of True 1s (TP) in Test Set = ",
             length(which(test_m2v1_cart_measure_df01$test_df01.Target == 1 &
                          test_m2v1_cart_measure_df01$test_m2v1_cart_y_pred_df01 ==
↪  1))))
```

```
## [1] "Number of True 1s (TP) in Test Set = 499"
```

```
print(paste0("Number of False 0s (FN) in Test Set = ",
             length(which(test_m2v1_cart_measure_df01$test_df01.Target == 1 &
                          test_m2v1_cart_measure_df01$test_m2v1_cart_y_pred_df01 ==
↪  0))))
```

```
## [1] "Number of False 0s (FN) in Test Set = 576"
```

```
print(paste0("Number of False 1s (FP) in Test Set = ",
             length(which(test_m2v1_cart_measure_df01$test_df01.Target == 0 &
                          test_m2v1_cart_measure_df01$test_m2v1_cart_y_pred_df01 ==
↪  1))))
```

```
## [1] "Number of False 1s (FP) in Test Set = 263"
```

```
print(paste0("Number of True 0s (TN) in Test Set = ",
             length(which(test_m2v1_cart_measure_df01$test_df01.Target == 0 &
                            test_m2v1_cart_measure_df01$test_m2v1_cart_y_pred_df01 ==
↪  0))))
```

```
## [1] "Number of True 0s (TN) in Test Set = 1529"
```

Model 3 (M3): C5.0 Decision Tree

```
# Train model
train_m3v1_c50 <- C5.0(Target~., data = train_xy01_df01, control =
↪  C5.0Control(minCases=75))

# Use trained model to predict y on test data
test_m3v1_c50_y_pred_df01 <- predict(object = train_m3v1_c50, newdata = test_x01_df01,
↪  type = "class")

# Create contingency table to review results
test_m3v1_c50_y_pred_tbl01 <- table(test_df01$Target, test_m3v1_c50_y_pred_df01)
row.names(test_m3v1_c50_y_pred_tbl01) <- c("Actual: PSI = 0 (-)",
                                           "Actual: PSI = 1 (+)")
colnames(test_m3v1_c50_y_pred_tbl01) <- c("Predicted: PSI = 0 (-)",
                                          "Predicted: PSI = 1 (+)")
test_m3v1_c50_y_pred_tbl01 <- addmargins(A = test_m3v1_c50_y_pred_tbl01,
                                         FUN = list(Total = sum), quiet = TRUE)
print(test_m3v1_c50_y_pred_tbl01)
```

```
##                       test_m3v1_c50_y_pred_df01
##                        Predicted: PSI = 0 (-) Predicted: PSI = 1 (+) Total
##   Actual: PSI = 0 (-)                   1537                    255  1792
##   Actual: PSI = 1 (+)                    541                    534  1075
##   Total                                 2078                    789  2867
```

```
# Pull values from confusion matrix table
test_m3v1_c50_y_pred_tp01 <- test_m3v1_c50_y_pred_tbl01[2, 2]
test_m3v1_c50_y_pred_fn01 <- test_m3v1_c50_y_pred_tbl01[2, 1]
test_m3v1_c50_y_pred_fp01 <- test_m3v1_c50_y_pred_tbl01[1, 2]
test_m3v1_c50_y_pred_tn01 <- test_m3v1_c50_y_pred_tbl01[1, 1]
print(test_m3v1_c50_y_pred_tp01)
```

```
## [1] 534
```

```
print(test_m3v1_c50_y_pred_fn01)
```

```
## [1] 541
```

```
print(test_m3v1_c50_y_pred_fp01)
```

```
## [1] 255
```

```
print(test_m3v1_c50_y_pred_tn01)
```

```
## [1] 1537
```

```
# ------------------------------------------------------------------------------
# Use trained model to predict y on train data
train_m3v1_c50_y_pred_df01 <- predict(object = train_m3v1_c50, newdata = train_x01_df01,
↪  type = "class")

# Create contingency table to review results
train_m3v1_c50_y_pred_tbl01 <- table(train_df01$Target, train_m3v1_c50_y_pred_df01)
row.names(train_m3v1_c50_y_pred_tbl01) <- c("Actual: PSI = 0 (-)",
                                            "Actual: PSI = 1 (+)")
colnames(train_m3v1_c50_y_pred_tbl01) <- c("Predicted: PSI = 0 (-)",
                                           "Predicted: PSI = 1 (+)")
train_m3v1_c50_y_pred_tbl01 <- addmargins(A = train_m3v1_c50_y_pred_tbl01,
                                          FUN = list(Total = sum), quiet = TRUE)
print(train_m3v1_c50_y_pred_tbl01)
```

```
##                       train_m3v1_c50_y_pred_df01
##                        Predicted: PSI = 0 (-) Predicted: PSI = 1 (+) Total
##   Actual: PSI = 0 (-)                   3569                    635  4204
##   Actual: PSI = 1 (+)                   1320                   1166  2486
##   Total                                 4889                   1801  6690
```

```
# Pull values from confusion matrix table
train_m3v1_c50_y_pred_tp01 <- train_m3v1_c50_y_pred_tbl01[2, 2]
train_m3v1_c50_y_pred_fn01 <- train_m3v1_c50_y_pred_tbl01[2, 1]
train_m3v1_c50_y_pred_fp01 <- train_m3v1_c50_y_pred_tbl01[1, 2]
train_m3v1_c50_y_pred_tn01 <- train_m3v1_c50_y_pred_tbl01[1, 1]
print(train_m3v1_c50_y_pred_tp01)
```

```
## [1] 1166
```

```
print(train_m3v1_c50_y_pred_fn01)
```

```
## [1] 1320
```

```
print(train_m3v1_c50_y_pred_fp01)
```

```
## [1] 635
```

```
print(train_m3v1_c50_y_pred_tn01)
```

```
## [1] 3569
```

```
# Generate classification evaluation measures
test_m3v1_c50_acc01 <- (test_m3v1_c50_y_pred_tp01 + test_m3v1_c50_y_pred_tn01) /
  (test_m3v1_c50_y_pred_tp01 + test_m3v1_c50_y_pred_fn01 + test_m3v1_c50_y_pred_fp01 +
↪  test_m3v1_c50_y_pred_tn01)

test_m3v1_c50_err_rate01 <- 1 - test_m3v1_c50_acc01
test_m3v1_c50_recall01 <- test_m3v1_c50_y_pred_tp01 / (test_m3v1_c50_y_pred_tp01 +
↪  test_m3v1_c50_y_pred_fn01)
test_m3v1_c50_tnr01 <- test_m3v1_c50_y_pred_tn01 / (test_m3v1_c50_y_pred_tn01 +
↪  test_m3v1_c50_y_pred_fp01)
test_m3v1_c50_prec01 <- test_m3v1_c50_y_pred_tp01 / (test_m3v1_c50_y_pred_tp01 +
↪  test_m3v1_c50_y_pred_fp01)
test_m3v1_c50_f101 <- (1 + 1^2) * ((test_m3v1_c50_prec01 * test_m3v1_c50_recall01) /
                                   ((1^2 * test_m3v1_c50_prec01) +
↪  test_m3v1_c50_recall01))
```

```
test_m3v1_c50_plr01 <- test_m3v1_c50_recall01 / (1 - test_m3v1_c50_tnr01)


train_m3v1_c50_acc01 <- (train_m3v1_c50_y_pred_tp01 + train_m3v1_c50_y_pred_tn01) /
  (train_m3v1_c50_y_pred_tp01 + train_m3v1_c50_y_pred_fn01 + train_m3v1_c50_y_pred_fp01 +
↪  train_m3v1_c50_y_pred_tn01)

train_m3v1_c50_err_rate01 <- 1 - train_m3v1_c50_acc01
train_m3v1_c50_recall01 <- train_m3v1_c50_y_pred_tp01 / (train_m3v1_c50_y_pred_tp01 +
↪  train_m3v1_c50_y_pred_fn01)
train_m3v1_c50_tnr01 <- train_m3v1_c50_y_pred_tn01 / (train_m3v1_c50_y_pred_tn01 +
↪  train_m3v1_c50_y_pred_fp01)
train_m3v1_c50_prec01 <- train_m3v1_c50_y_pred_tp01 / (train_m3v1_c50_y_pred_tp01 +
↪  train_m3v1_c50_y_pred_fp01)
train_m3v1_c50_f101 <- (1 + 1^2) * ((train_m3v1_c50_prec01 * train_m3v1_c50_recall01) /
                                    ((1^2 * train_m3v1_c50_prec01) +
↪  train_m3v1_c50_recall01))
train_m3v1_c50_plr01 <- train_m3v1_c50_recall01 / (1 - train_m3v1_c50_tnr01)

# ============================================================================
# Generate evaluation measures table
measure_m3v1_c50_name <- c("M3: C5.0 DT",
                           "Baseline Accuracy",
                           "Accuracy", "Error Rate",
                           "Sensitivity/TPR/Recall",
                           "Specificity/TNR",
                           "Precision",
                           "F1",
                           "Positive Liklihood Ratio"
                           )

measure_m3v1_c50_val01 <- c("Traing Data set",
                            paste0(round((4204 / 6690) * 100, 1), "%"),
                            paste0(round(train_m3v1_c50_acc01 * 100, 1), "%"),
                            paste0(round(train_m3v1_c50_err_rate01 * 100, 1), "%"),
                            paste0(round(train_m3v1_c50_recall01 * 100, 1), "%"),
                            paste0(round(train_m3v1_c50_tnr01 * 100, 1), "%"),
                            paste0(round(train_m3v1_c50_prec01 * 100, 1), "%"),
                            paste0(round(train_m3v1_c50_f101 * 100, 1), "%"),
                            paste0(round(train_m3v1_c50_plr01 * 1, 2), "")
                            )

measure_m3v1_c50_val02 <- c("Test Data set",
                            paste0(round((1792 / 2867) * 100, 1), "%"),
                            paste0(round(test_m3v1_c50_acc01 * 100, 1), "%"),
                            paste0(round(test_m3v1_c50_err_rate01 * 100, 1), "%"),
                            paste0(round(test_m3v1_c50_recall01 * 100, 1), "%"),
                            paste0(round(test_m3v1_c50_tnr01 * 100, 1), "%"),
                            paste0(round(test_m3v1_c50_prec01 * 100, 1), "%"),
                            paste0(round(test_m3v1_c50_f101 * 100, 1), "%"),
                            paste0(round(test_m3v1_c50_plr01 * 1, 2), "")
                            )
```

```
measure_m3v1_c50_tbl <- data.frame(measure_m3v1_c50_name, measure_m3v1_c50_val01,
↪  measure_m3v1_c50_val02)
print(measure_m3v1_c50_tbl)
```

```
##      measure_m3v1_c50_name measure_m3v1_c50_val01 measure_m3v1_c50_val02
## 1              M3: C5.0 DT        Traing Data set          Test Data set
## 2        Baseline Accuracy                  62.8%                  62.5%
## 3                 Accuracy                  70.8%                  72.2%
## 4               Error Rate                  29.2%                  27.8%
## 5   Sensitivity/TPR/Recall                  46.9%                  49.7%
## 6          Specificity/TNR                  84.9%                  85.8%
## 7                Precision                  64.7%                  67.7%
## 8                       F1                  54.4%                  57.3%
## 9 Positive Liklihood Ratio                   3.11                   3.49
```

```
# Confirm confusion matrix results
confusionMatrix(table(test_m3v1_c50_y_pred_df01, test_df01$Target), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##
## test_m3v1_c50_y_pred_df01    0    1
##                         0 1537  541
##                         1  255  534
##
##                Accuracy : 0.7224
##                  95% CI : (0.7056, 0.7387)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3744
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4967
##             Specificity : 0.8577
##          Pos Pred Value : 0.6768
##          Neg Pred Value : 0.7397
##              Prevalence : 0.3750
##          Detection Rate : 0.1863
##    Detection Prevalence : 0.2752
##       Balanced Accuracy : 0.6772
##
##        'Positive' Class : 1
##
```

```
print(paste0("Number of 0s in Test Set Original Target = ", length(which(test_df01$Target
↪  == 0))))
```

```
## [1] "Number of 0s in Test Set Original Target = 1792"
```

```
print(paste0("Number of 0s in Test Set Predicted = ",
↪  length(which(test_m3v1_c50_y_pred_df01 == 0))))
```

```
## [1] "Number of 0s in Test Set Predicted = 2078"
```

```
print(paste0("Number of 1s in Test Set Original Target = ", length(which(test_df01$Target
↪  == 1))))
```

```
## [1] "Number of 1s in Test Set Original Target = 1075"
```

```
print(paste0("Number of 1s in Test Set Predicted = ",
↪  length(which(test_m3v1_c50_y_pred_df01 == 1))))
```

```
## [1] "Number of 1s in Test Set Predicted = 789"
```

```
test_m3v1_c50_measure_df01 <- data.frame(test_df01$Target, test_m3v1_c50_y_pred_df01)
print(head(test_m3v1_c50_measure_df01))
```

```
##   test_df01.Target test_m3v1_c50_y_pred_df01
## 1                0                         0
## 2                0                         0
## 3                0                         0
## 4                0                         0
## 5                0                         0
## 6                0                         0
```

```
print(paste0("Number of True 1s (TP) in Test Set = ",
             length(which(test_m3v1_c50_measure_df01$test_df01.Target == 1 &
                          test_m3v1_c50_measure_df01$test_m3v1_c50_y_pred_df01 == 1))))
```

```
## [1] "Number of True 1s (TP) in Test Set = 534"
```

```
print(paste0("Number of False 0s (FN) in Test Set = ",
             length(which(test_m3v1_c50_measure_df01$test_df01.Target == 1 &
                          test_m3v1_c50_measure_df01$test_m3v1_c50_y_pred_df01 == 0))))
```

```
## [1] "Number of False 0s (FN) in Test Set = 541"
```

```
print(paste0("Number of False 1s (FP) in Test Set = ",
             length(which(test_m3v1_c50_measure_df01$test_df01.Target == 0 &
                          test_m3v1_c50_measure_df01$test_m3v1_c50_y_pred_df01 == 1))))
```

```
## [1] "Number of False 1s (FP) in Test Set = 255"
```

```
print(paste0("Number of True 0s (TN) in Test Set = ",
             length(which(test_m3v1_c50_measure_df01$test_df01.Target == 0 &
                          test_m3v1_c50_measure_df01$test_m3v1_c50_y_pred_df01 == 0))))
```

```
## [1] "Number of True 0s (TN) in Test Set = 1537"
```

Model 4 (M4): K-Nearest Neighbor (KNN), $k = 81$

```
# Train model
test_m4v1_knn81 <- knn(train_x01_df01, test_x01_df01, cl = train_xy01_df01$Target, k =
↪  81, prob = TRUE)

# Create contingency table to review results
test_m4v1_knn81_y_pred_tbl01 <- table(test_df01$Target, test_m4v1_knn81)

row.names(test_m4v1_knn81_y_pred_tbl01) <- c("Actual: PSI = 0 (-)",
                                             "Actual: PSI = 1 (+)")
```

```
colnames(test_m4v1_knn81_y_pred_tbl01) <- c("Predicted: PSI = 0 (-)",
                                            "Predicted: PSI = 1 (+)")
test_m4v1_knn81_y_pred_tbl01 <- addmargins(A = test_m4v1_knn81_y_pred_tbl01,
                                           FUN = list(Total = sum), quiet = TRUE)
print(test_m4v1_knn81_y_pred_tbl01)
```

```
##                       test_m4v1_knn81
##                        Predicted: PSI = 0 (-) Predicted: PSI = 1 (+) Total
##   Actual: PSI = 0 (-)                   1629                    163  1792
##   Actual: PSI = 1 (+)                    672                    403  1075
##   Total                                 2301                    566  2867
```

```
# Pull values from confusion matrix table
test_m4v1_knn81_y_pred_tp01 <- test_m4v1_knn81_y_pred_tbl01[2, 2]
test_m4v1_knn81_y_pred_fn01 <- test_m4v1_knn81_y_pred_tbl01[2, 1]
test_m4v1_knn81_y_pred_fp01 <- test_m4v1_knn81_y_pred_tbl01[1, 2]
test_m4v1_knn81_y_pred_tn01 <- test_m4v1_knn81_y_pred_tbl01[1, 1]
print(test_m4v1_knn81_y_pred_tp01)
```

```
## [1] 403
```

```
print(test_m4v1_knn81_y_pred_fn01)
```

```
## [1] 672
```

```
print(test_m4v1_knn81_y_pred_fp01)
```

```
## [1] 163
```

```
print(test_m4v1_knn81_y_pred_tn01)
```

```
## [1] 1629
```

```
# Generate classification evaluation measures
test_m4v1_knn81_acc01 <- (test_m4v1_knn81_y_pred_tp01 + test_m4v1_knn81_y_pred_tn01) /
  (test_m4v1_knn81_y_pred_tp01 + test_m4v1_knn81_y_pred_fn01 +
↪  test_m4v1_knn81_y_pred_fp01 + test_m4v1_knn81_y_pred_tn01)
test_m4v1_knn81_err_rate01 <- 1 - test_m4v1_knn81_acc01
test_m4v1_knn81_recall01 <- test_m4v1_knn81_y_pred_tp01 /
  (test_m4v1_knn81_y_pred_tp01 + test_m4v1_knn81_y_pred_fn01)
test_m4v1_knn81_tnr01 <- test_m4v1_knn81_y_pred_tn01 /
  (test_m4v1_knn81_y_pred_tn01 + test_m4v1_knn81_y_pred_fp01)
test_m4v1_knn81_prec01 <- test_m4v1_knn81_y_pred_tp01 /
  (test_m4v1_knn81_y_pred_tp01 + test_m4v1_knn81_y_pred_fp01)
test_m4v1_knn81_f101 <- (1 + 1^2) * ((test_m4v1_knn81_prec01 * test_m4v1_knn81_recall01)
↪  /
                                       ((1^2 * test_m4v1_knn81_prec01) +
↪  test_m4v1_knn81_recall01))
test_m4v1_knn81_plr01 <- test_m4v1_knn81_recall01 /
  (1 - test_m4v1_knn81_tnr01)

# ============================================================================
# Generate evaluation measures table
measure_test_m4v1_knn81_name <- c("M4: KNN-81",
                                  "Accuracy",
                                  "Error Rate",
```

```
                                "Sensitivity/TPR/Recall",
                                "Specificity/TNR",
                                "Precision",
                                "F1",
                                "Positive Liklihood Ratio"
                                )

measure_test_m4v1_knn81_val02 <- c("Eval. Measure Values",
                                paste0(round(test_m4v1_knn81_acc01 * 100, 1), "%"),
                                paste0(round(test_m4v1_knn81_err_rate01 * 100, 1), "%"),
                                paste0(round(test_m4v1_knn81_recall01 * 100, 1), "%"),
                                paste0(round(test_m4v1_knn81_tnr01 * 100, 1), "%"),
                                paste0(round(test_m4v1_knn81_prec01 * 100, 1), "%"),
                                paste0(round(test_m4v1_knn81_f101 * 100, 1), "%"),
                                paste0(round(test_m4v1_knn81_plr01 * 1, 2), "")
                                )

measure_test_m4v1_knn81_tbl <- data.frame(measure_test_m4v1_knn81_name,
↪  measure_test_m4v1_knn81_val02)
print(measure_test_m4v1_knn81_tbl)
```

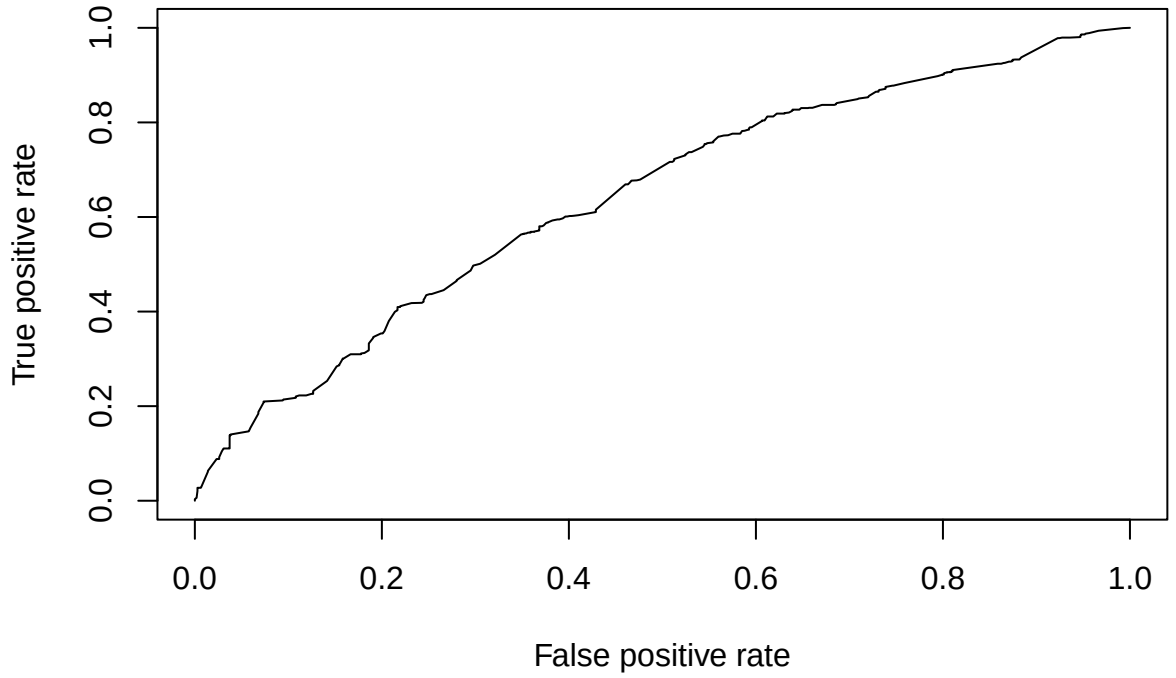
```
##   measure_test_m4v1_knn81_name measure_test_m4v1_knn81_val02
## 1                  M4: KNN-81          Eval. Measure Values
## 2                    Accuracy                         70.9%
## 3                  Error Rate                         29.1%
## 4       Sensitivity/TPR/Recall                         37.5%
## 5             Specificity/TNR                         90.9%
## 6                   Precision                         71.2%
## 7                          F1                         49.1%
## 8     Positive Liklihood Ratio                          4.12
```

```
# Confirm confusion matrix results
confusionMatrix(table(test_m4v1_knn81, test_df01$Target), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##
## test_m4v1_knn81    0    1
##               0 1629  672
##               1  163  403
##
##                Accuracy : 0.7088
##                  95% CI : (0.6917, 0.7253)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3136
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3749
##             Specificity : 0.9090
##          Pos Pred Value : 0.7120
##          Neg Pred Value : 0.7080
```

```
##              Prevalence : 0.3750
##          Detection Rate : 0.1406
##    Detection Prevalence : 0.1974
##       Balanced Accuracy : 0.6420
##
##        'Positive' Class : 1
##
```

```
# Plot receiver operator characteristic (ROC) curve
test_m4v1_knn81_df <- attr(test_m4v1_knn81, which = "prob")
test_m4v1_knn81_pred <- prediction(test_m4v1_knn81_df, test_df01$Target, label.ordering =
↪  c("1", "0"))
test_m4v1_knn81_roc <- performance(test_m4v1_knn81_pred, measure = "tpr", x.measure =
↪  "fpr")
plot(test_m4v1_knn81_roc)
```




```
# Calculate area under curve (AUC)
test_m4v1_knn81_auc <- performance(test_m4v1_knn81_pred, measure = "auc")
test_m4v1_knn81_auc <- test_m4v1_knn81_auc@y.values[[1]]
print(test_m4v1_knn81_auc)
```

```
## [1] 0.6417239
```

Model 5 (M5): KNN, $k = 3$

```
# Train model
test_m5v1_knn03 <- knn(train_x01_df01, test_x01_df01, cl = train_xy01_df01$Target, k = 3,
↪  prob = TRUE)

# Create contingency table to review results
test_m5v1_knn03_y_pred_tbl01 <- table(test_df01$Target, test_m5v1_knn03)

row.names(test_m5v1_knn03_y_pred_tbl01) <- c("Actual: PSI = 0 (-)",
                                             "Actual: PSI = 1 (+)")
colnames(test_m5v1_knn03_y_pred_tbl01) <- c("Predicted: PSI = 0 (-)",
                                            "Predicted: PSI = 1 (+)")
test_m5v1_knn03_y_pred_tbl01 <- addmargins(A = test_m5v1_knn03_y_pred_tbl01,
                                           FUN = list(Total = sum), quiet = TRUE)
print(test_m5v1_knn03_y_pred_tbl01)
```

```
##                       test_m5v1_knn03
##                        Predicted: PSI = 0 (-) Predicted: PSI = 1 (+) Total
##   Actual: PSI = 0 (-)                   1570                    222  1792
##   Actual: PSI = 1 (+)                    552                    523  1075
##   Total                                 2122                    745  2867
```

```
# Pull values from confusion matrix table
test_m5v1_knn03_y_pred_tp01 <- test_m5v1_knn03_y_pred_tbl01[2, 2]
test_m5v1_knn03_y_pred_fn01 <- test_m5v1_knn03_y_pred_tbl01[2, 1]
test_m5v1_knn03_y_pred_fp01 <- test_m5v1_knn03_y_pred_tbl01[1, 2]
test_m5v1_knn03_y_pred_tn01 <- test_m5v1_knn03_y_pred_tbl01[1, 1]
print(test_m5v1_knn03_y_pred_tp01)
```

```
## [1] 523
```

```
print(test_m5v1_knn03_y_pred_fn01)
```

```
## [1] 552
```

```
print(test_m5v1_knn03_y_pred_fp01)
```

```
## [1] 222
```

```
print(test_m5v1_knn03_y_pred_tn01)
```

```
## [1] 1570
```

```
# Generate classification evaluation measures
test_m5v1_knn03_acc01 <- (test_m5v1_knn03_y_pred_tp01 + test_m5v1_knn03_y_pred_tn01) /
  (test_m5v1_knn03_y_pred_tp01 + test_m5v1_knn03_y_pred_fn01 +
↪  test_m5v1_knn03_y_pred_fp01 + test_m5v1_knn03_y_pred_tn01)
test_m5v1_knn03_err_rate01 <- 1 - test_m5v1_knn03_acc01
test_m5v1_knn03_recall01 <- test_m5v1_knn03_y_pred_tp01 /
  (test_m5v1_knn03_y_pred_tp01 + test_m5v1_knn03_y_pred_fn01)
test_m5v1_knn03_tnr01 <- test_m5v1_knn03_y_pred_tn01 /
  (test_m5v1_knn03_y_pred_tn01 + test_m5v1_knn03_y_pred_fp01)
test_m5v1_knn03_prec01 <- test_m5v1_knn03_y_pred_tp01 /
  (test_m5v1_knn03_y_pred_tp01 + test_m5v1_knn03_y_pred_fp01)
test_m5v1_knn03_f101 <- (1 + 1^2) * ((test_m5v1_knn03_prec01 * test_m5v1_knn03_recall01)
↪  /
                                       ((1^2 * test_m5v1_knn03_prec01) +
↪  test_m5v1_knn03_recall01))
```

```
test_m5v1_knn03_plr01 <- test_m5v1_knn03_recall01 /
  (1 - test_m5v1_knn03_tnr01)

# ============================================================================
# Generate evaluation measures table
measure_test_m5v1_knn03_name <- c("M5: KNN-3",
                                  "Accuracy",
                                  "Error Rate",
                                  "Sensitivity/TPR/Recall",
                                  "Specificity/TNR",
                                  "Precision",
                                  "F1",
                                  "Positive Liklihood Ratio"
                                  )

measure_test_m5v1_knn03_val02 <- c("Eval. Measure Values",
                                   paste0(round(test_m5v1_knn03_acc01 * 100, 1), "%"),
                                   paste0(round(test_m5v1_knn03_err_rate01 * 100, 1), "%"),
                                   paste0(round(test_m5v1_knn03_recall01 * 100, 1), "%"),
                                   paste0(round(test_m5v1_knn03_tnr01 * 100, 1), "%"),
                                   paste0(round(test_m5v1_knn03_prec01 * 100, 1), "%"),
                                   paste0(round(test_m5v1_knn03_f101 * 100, 1), "%"),
                                   paste0(round(test_m5v1_knn03_plr01 * 1, 2), "")
                                   )

measure_test_m5v1_knn03_tbl <- data.frame(measure_test_m5v1_knn03_name,
↪  measure_test_m5v1_knn03_val02)
print(measure_test_m5v1_knn03_tbl)
```

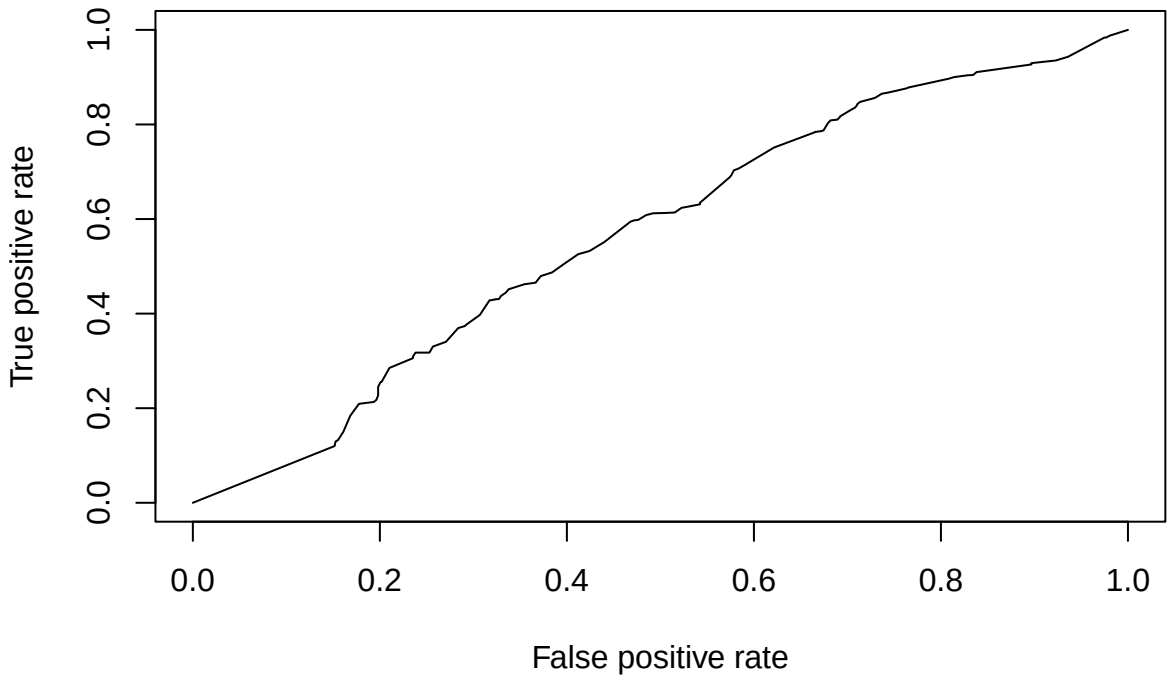
```
##   measure_test_m5v1_knn03_name measure_test_m5v1_knn03_val02
## 1                    M5: KNN-3          Eval. Measure Values
## 2                     Accuracy                           73%
## 3                   Error Rate                           27%
## 4       Sensitivity/TPR/Recall                         48.7%
## 5              Specificity/TNR                         87.6%
## 6                    Precision                         70.2%
## 7                           F1                         57.5%
## 8     Positive Liklihood Ratio                          3.93
```

```
# Confirm confusion matrix results
confusionMatrix(table(test_m5v1_knn03, test_df01$Target), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##
## test_m5v1_knn03    0    1
##               0 1570  552
##               1  222  523
##
##                Accuracy : 0.73
##                  95% CI : (0.7134, 0.7462)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                  Kappa : 0.3864
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4865
##             Specificity : 0.8761
##          Pos Pred Value : 0.7020
##          Neg Pred Value : 0.7399
##              Prevalence : 0.3750
##          Detection Rate : 0.1824
##    Detection Prevalence : 0.2599
##       Balanced Accuracy : 0.6813
##
##        'Positive' Class : 1
##
```

```r
# Plot receiver operator characteristic (ROC) curve
test_m5v1_knn03_df <- attr(test_m5v1_knn03, which = "prob")
test_m5v1_knn03_pred <- prediction(test_m5v1_knn03_df, test_df01$Target, label.ordering =
↪  c("1", "0"))
test_m5v1_knn03_roc <- performance(test_m5v1_knn03_pred, measure = "tpr", x.measure =
↪  "fpr")
plot(test_m5v1_knn03_roc)
```




```r
# Calculate area under curve (AUC)
test_m5v1_knn03_auc <- performance(test_m5v1_knn03_pred, measure = "auc")
```

```r
test_m5v1_knn03_auc <- test_m5v1_knn03_auc@y.values[[1]]
print(test_m5v1_knn03_auc)
```

```
## [1] 0.5698868
```