# 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 of the Random Forest Decision Trees, CART Decision Tree, C5.0 Decision Tree, or K-Nearest Neighbor being utilized will result in a model that predicts an individual's poverty status index based on demographic characteristics. More specifically, a secondary hypothesis is that the best predictor will be the Random Forest Decision Tree, based on sensitivity being greater than eighty percent. 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 was the Random Forest model, which had a sensitivity value of 0.742 and an accuracy value of 0.739. Further research regarding the accuracy and sensitivity of the model could be evaluated using additional predictor variables such as education level of individuals within a given household or location of household.

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

## 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). Individuals 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 circumstances that put individuals at risk for being impoverished. As poverty affects more than just the direct victims, addressing and overcoming it is of 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 incomes 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 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 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 of an individual based on several features related to their

living environment and family demographics. The general hypothesis of the study is that at least one of the four classification techniques being utilized will result in a model that predicts an individual's poverty status index based on demographic characteristics. More specifically, a secondary hypothesis is that the best predictor will be the random forest decision tree technique, focusing on sensitivity and precision being above 80%.

## Method

### 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 (Sarangam, 2020). The raw data was initially obtained as an unzipped SSV file that was adjusted in R-Studio for further use. The main system used for this analysis was R-Studio. 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 the empirical rule which specifies that z-score values above 3 or less than -3 are outliers.

**Sample Characteristics**

The population addressed in the hypothesis is in reference to all households within various countries around the world. For the purpose of this study however, 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 A, 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 does not have access to a gas stove (0) to cook their food. 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 what is the rate of individuals in a household who are below the age of 19 and over the age of 65. This is an important variable as it indicates the people that are dependent on the heads of the household, as they are most likely not bringing any income being that they are either too young to earn or retired. Sewer is a binary variable that indicates whether or not a household has a sewer available for their waste as indicated by a 1 or not as indicated by a 0. The variable of total_persons is a discrete variable that indicates the number of people within a household. The target variable which is the poverty status index is a binary variable where zero indicates a household that is not vulnerable to poverty and one indicates a household that is vulnerable to moderate or extreme poverty. 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, in order to provide important glimpses into patterns within the data, and thereby inform consequent predictive analytics efforts. Of the 40 features that remained, only seven were quantitative in nature. The descriptive statistics for each is included in Table 1, which includes general measurements of location, such as mean ($\bar{x}$) and median, and dispersion, 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 |
| Mode | | | | | | | |
| 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 |

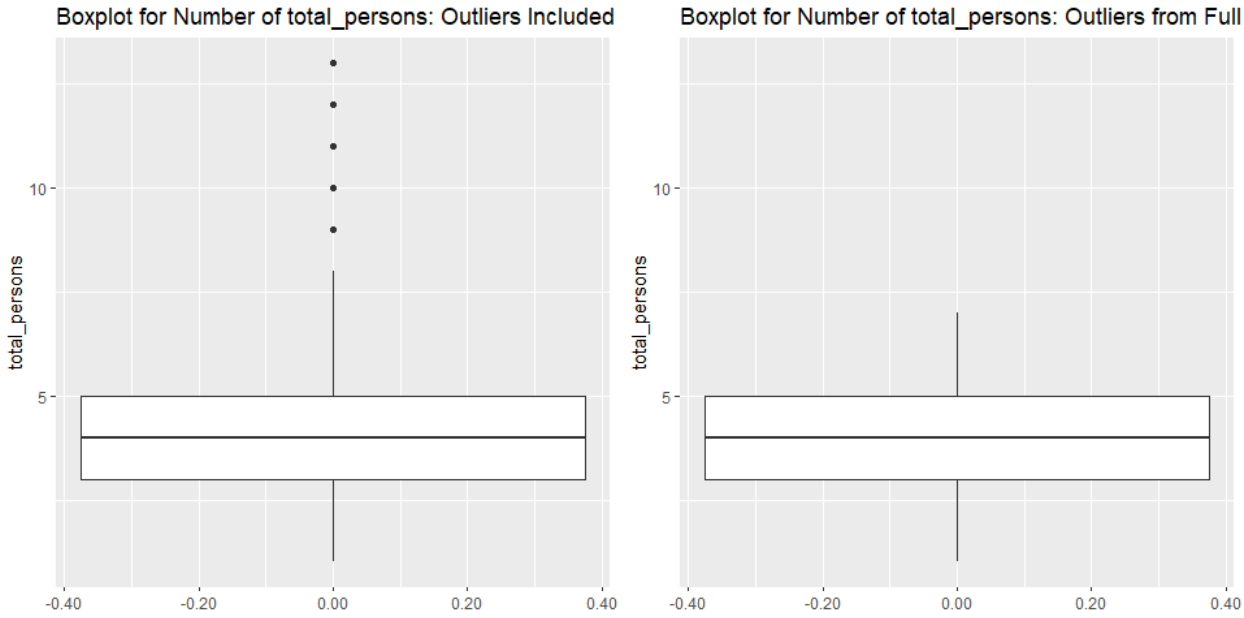
*Note*. $n$ = 6,690

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) was determined to be outliers, though $\bar{x}$ did not

decrease very dramatically (3.823) and the median remained the same. In fact, it should be noted that for every one of the seven quantitative variables, excluding the outliers did not change median value, which is indicative of its more robust nature as compared to mean. As another example the mean and median values for dependency were 0.400, with a standard deviation of 0.254. Figure 1 is the side-by-side comparison of boxplots for total_persons.

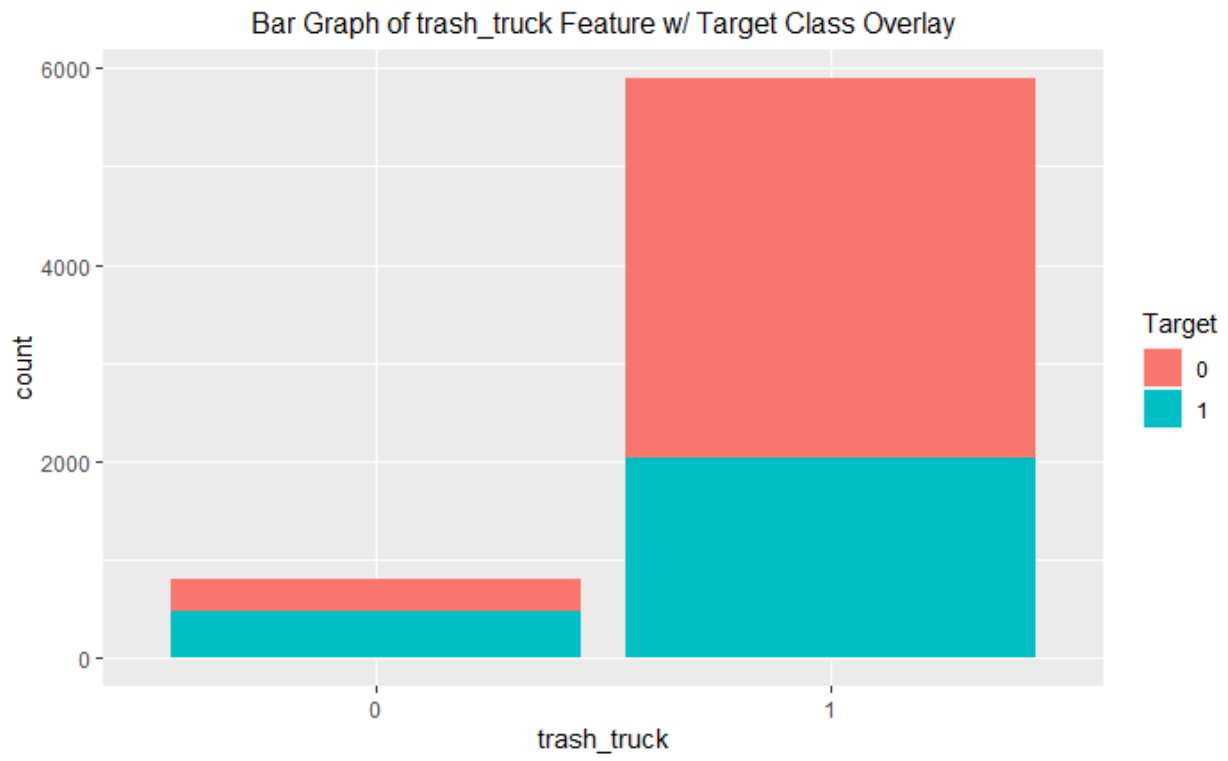
**Figure 1**.

*Boxplot comparison for total_persons*




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_truck, shows that the majority of households do have their trash hauled away via truck, but there is a fair mix of both target poverty status index classes (0 = non-vulnerable, 1 = vulnerable, moderate, extreme) with each trash_truck value.

**Figure 2**




## Modeling

A total of four different models were constructed after splitting the data into a train set and a test set. A stratified 70 to 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 A).

### Model 1: Random Forest Decision Tree

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.

**Table 2**

*Random Forest Evaluation Metrics*

| | |
|---|---|
| Accuracy | 0.739 |
| Sensitivity | 0.742 |
| Precision | 0.465 |
| F1 Score | 0.572 |

**Model 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 3.

**Table 3**

*CART Decision Tree Evaluation Metrics*

| | |
|---|---|
| Accuracy | 0.707 |
| Sensitivity | 0.655 |
| Precision | 0.464 |
| F1 Score | 0.543 |

**Model 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%. The evaluation metrics for this model can be seen in Table 4.

**Table 4**

*C5.0 Decision Tree Evaluation Metrics*

| | |
|---|---|
| Accuracy | 0.722 |
| Sensitivity | 0.677 |
| Precision | 0.497 |
| F1 Score | 0.573 |

**Model 4:  K-Nearest Neighbor 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 the K-Nearest Neighbor model was found to be 67.1%. The evaluation metrics for this model can be seen in Table 5.

**Table 5**

*KNN 81 Evaluation Metrics*

| | |
|---|---|
| Accuracy | 0.671 |
| Sensitivity | 0.693 |
| Precision | 0.687 |
| F1 Score | 0.688 |

**Model 5:  K-Nearest Neighbor 82**

A KNN-82 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 K-Nearest Neighbor model was found to be 67.2%. The evaluation metrics for this model can be seen in Table 6.

**Table 6**

*KNN 82 Evaluation Metrics*

| | |
|---|---|
| Accuracy | 0.672 |
| Sensitivity | 0.695 |
| Precision | 0.691 |
| F1 Score | 0.692 |

## 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 F1 score. The most accurate model out of the five models is Model 1 which uses the Random Forest Decision Tree algorithm. It had an accuracy value of 73.9%. The least accurate model was Model 5, the K-Nearest Neighbor 81 model, which had an accuracy score of 67.1%. The order of the models with the highest level of sensitivity to the least sensitive model is as follows: Random Forest (0.742), KNN-82 (0.695), KNN-81 (0.693), C5.0 (0.677), and CART (0.655). The most precise model was found to be the KNN-82 model with a precision value of 0.691. The least precise model was found to be the CART model with a precision value of 0.464. The model with the highest F1 value was found to be the KNN-82 model with a value of 0.692, while the model with the lowest F1 value was found to be the CART model with a value of 0.543. The results are summarized in Table 7.

**Table 7**

*Summary of the evaluation metrics for each of the models.*

| Model | Accuracy | Sensitivity | Precision | F1 |
|---|---|---|---|---|
| RandomForest | 0.739 | 0.742 | 0.465 | 0.572 |
| CART | 0.707 | 0.655 | 0.464 | 0.543 |
| C5.0 | 0.722 | 0.677 | 0.497 | 0.573 |
| KNN-81 | 0.671 | 0.693 | 0.687 | 0.688 |
| KNN-82 | 0.672 | 0.695 | 0.691 | 0.692 |

## Discussion

The purpose of this study was to use classification techniques to predict an individual's poverty status index based on several characteristics that are broadly available regardless of where the data is being collected. The metric of sensitivity was used as one of the main evaluation metrics as the overarching purpose of this study was to help determine which households are vulnerable to poverty in order to ensure that they receive sufficient funds necessary to survive. Therefore, the main evaluation metric that indicates how well the models do is dependent on how well they correctly predict households who are vulnerable in order to ensure that households that are vulnerable are not overlooked. Model 1 had a sensitivity value of 0.742 which indicates 74.2% of the households that are impoverished are correctly being marked as a vulnerable household. Model 2 has a sensitivity value of 0.707 which indicates that 70% of the households that are impoverished are correctly being marked as a vulnerable household. Model 3 has a sensitivity value of 0.677 which indicates that 67.7% of the households that are impoverished are correctly being marked as a vulnerable household. Model 4 has a sensitivity

value of 0.693 which indicates that 69.3% of the households that are impoverished are correctly being marked as a vulnerable household. Model 5 has a sensitivity value of 0.695 which indicates that 69.5% of the households that are impoverished are correctly being marked as a vulnerable household. Based on the results obtained from the study it should be noted that the Random Forest model is the most effective model in determining whether or not a Costa Rican household is impoverished. 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 poverty status index. 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 poverty status index based on several characteristics that are broadly available regardless of where the data is being collected. The study, however, was not able to meet the further specified hypothesis of achieving over 80% accuracy with the final model as the final model had an accuracy of 74.2%. In the future, this study could look to refining the model in order to increase the overall accuracy and sensitivity of the model. 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

Sarangam, A. (2020, December 17). Top 10 data mining tools. *Jigsaw Academy*. https://www.jigsawacademy.com/blogs/data-science/data-mining-tools/

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

# 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<br><chr> | v14a<br><int> | refrig<br><int> | hogar_total<br><int> | computer<br><int> | qmobilephone<br><int> | dis<br><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<br><chr> | has_bathroom<br><int> | has_refrig<br><int> | total_persons<br><int> | has_comp<br><int> | num_phones<br><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
```

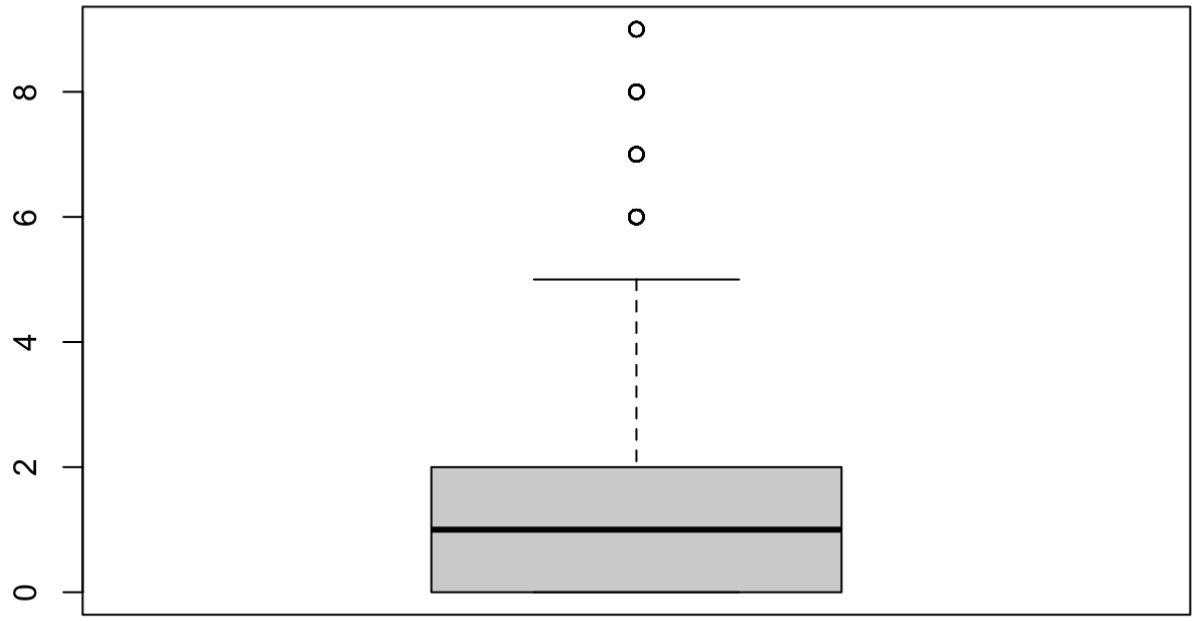
```
# checking that the max values for number of children and adults over
# 65 makes sense or if there are any outliers
max(crp_train1$num_children)
```

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

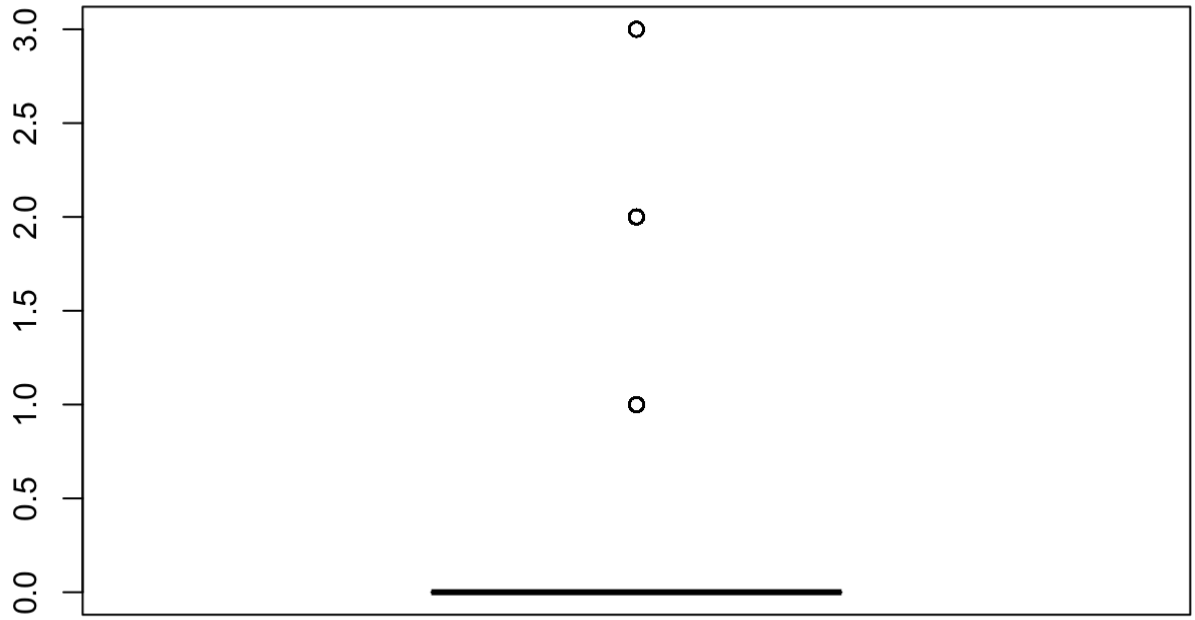
```
max(crp_train1$num_65)
```

```
## [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.
```

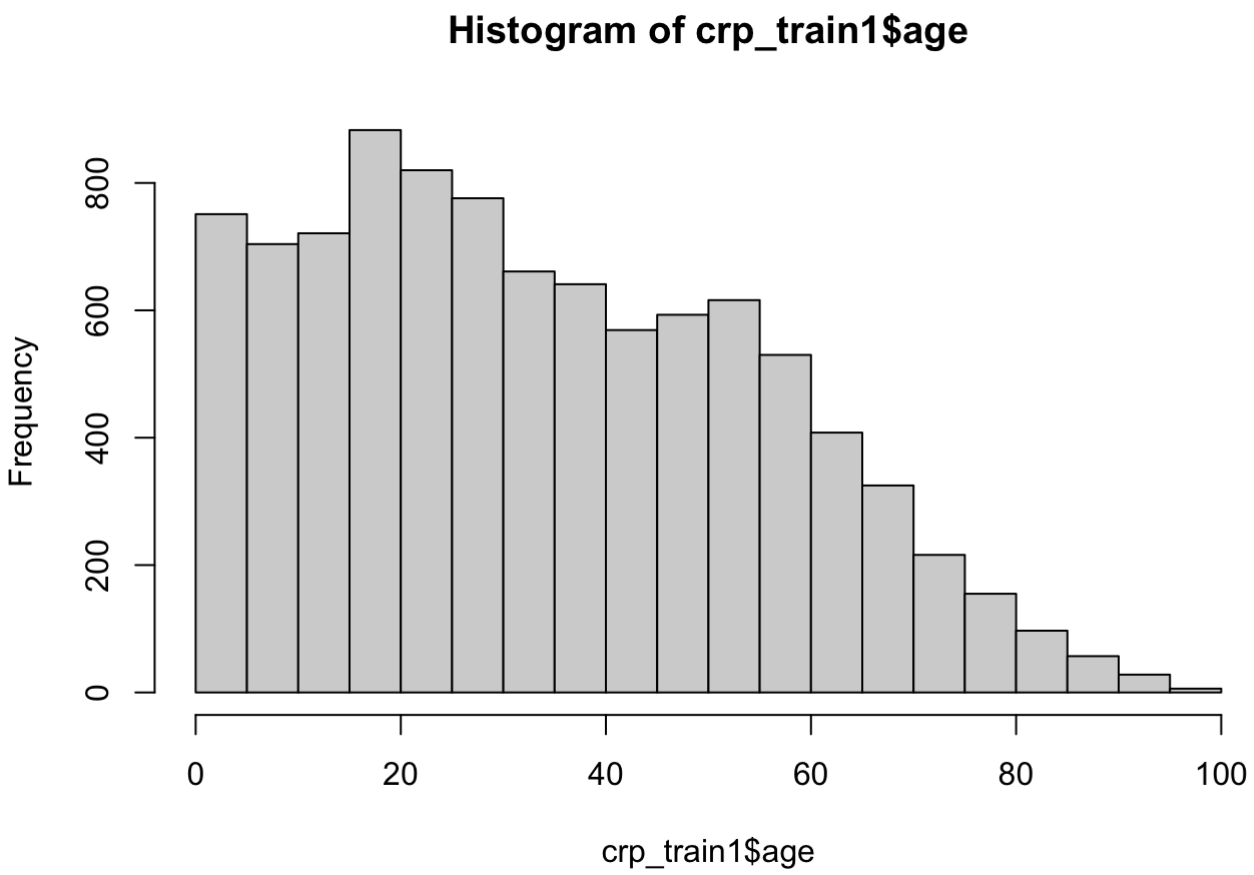
```
# fixing the yes/no responses in dependency rate
# dependency rate = (num_childre + num_65)/ total_persons
crp_train1$dependency <- (crp_train1$num_children +
                            crp_train1$num_65)/crp_train1$total_persons

head(crp_train1)
```

| id<br><chr> | has_bathroom<br><int> | has_refrig<br><int> | total_persons<br><int> | has_comp<br><int> | num_phones<br><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<br><chr> | v14a<br><int> | refrig<br><int> | hogar_total<br><int> | computer<br><int> | qmobilephone<br><int> | dis<br><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_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
```

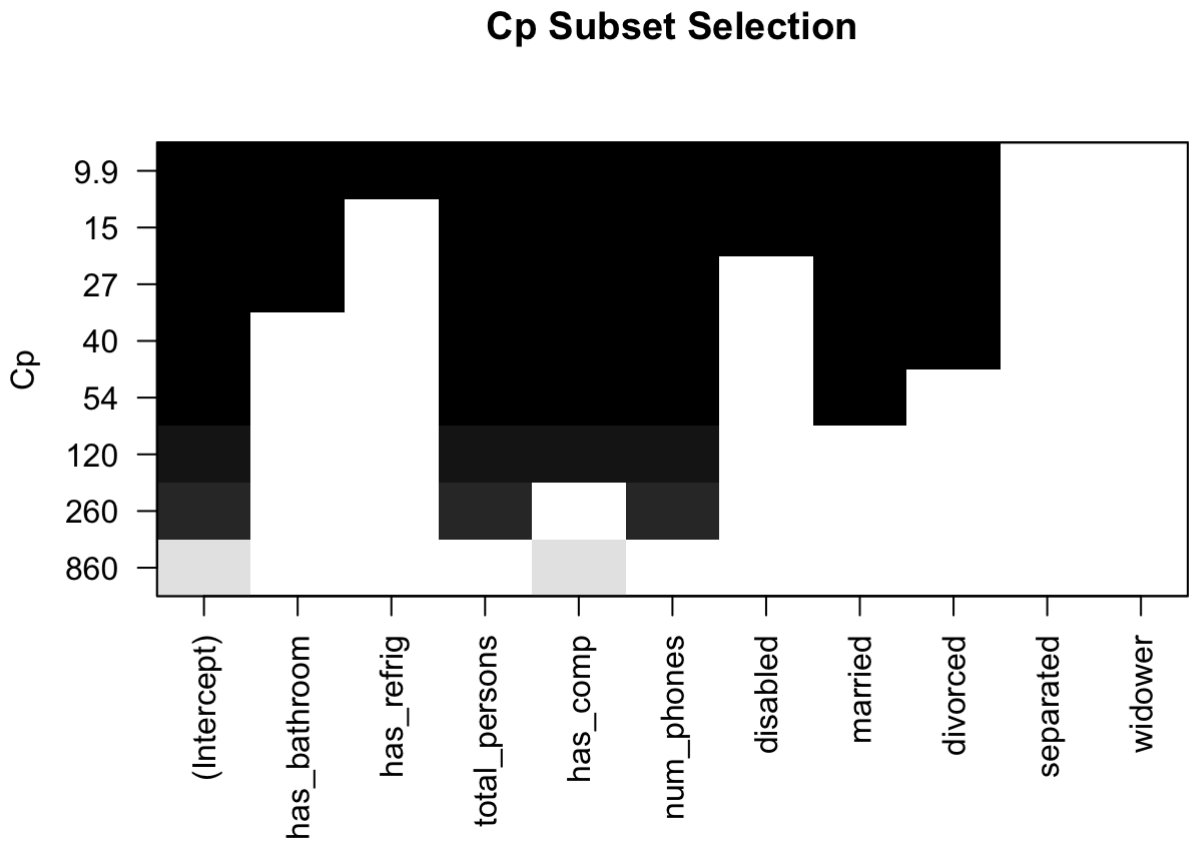
```
## [1] 0.6284006
```

```
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")
```

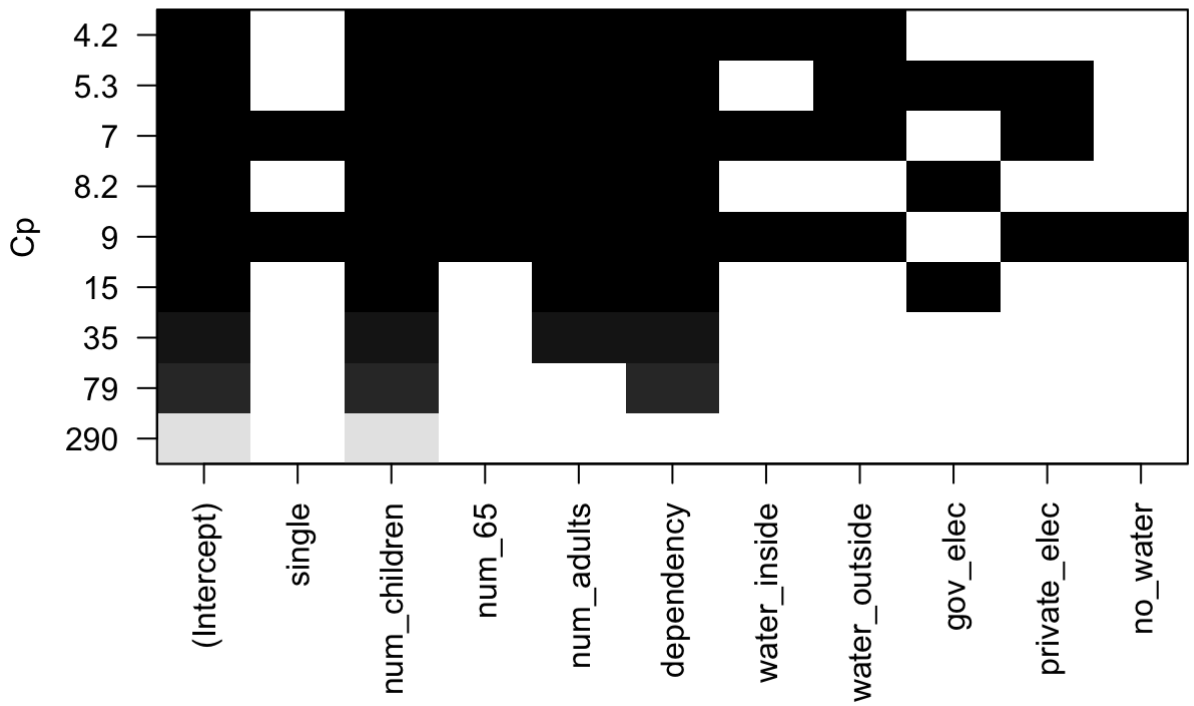



```
# Mallow's CP Plot: G2
cp_gp2 = regsubsets(Target~.,
                        data = group2)
```

```
## 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")
```

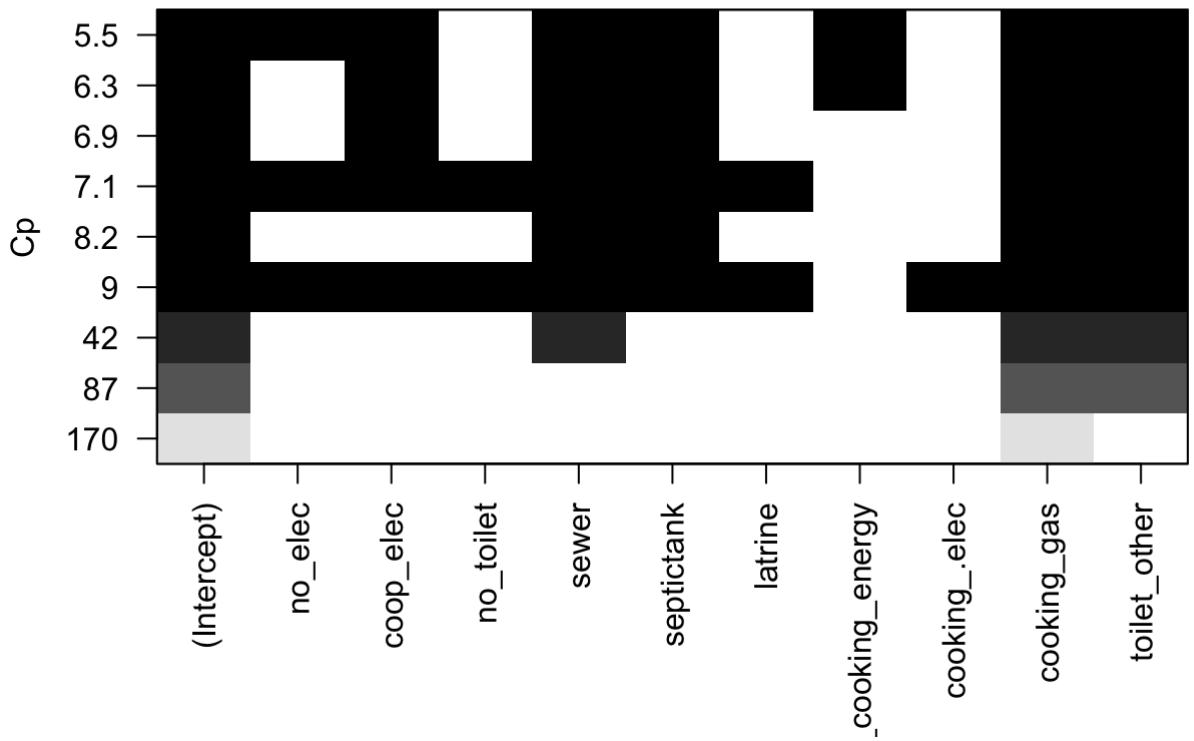



```
# Mallow's CP Plot: G3
cp_gp3 = regsubsets(Target~.,
                        data = group3)
```

```
## 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")
```

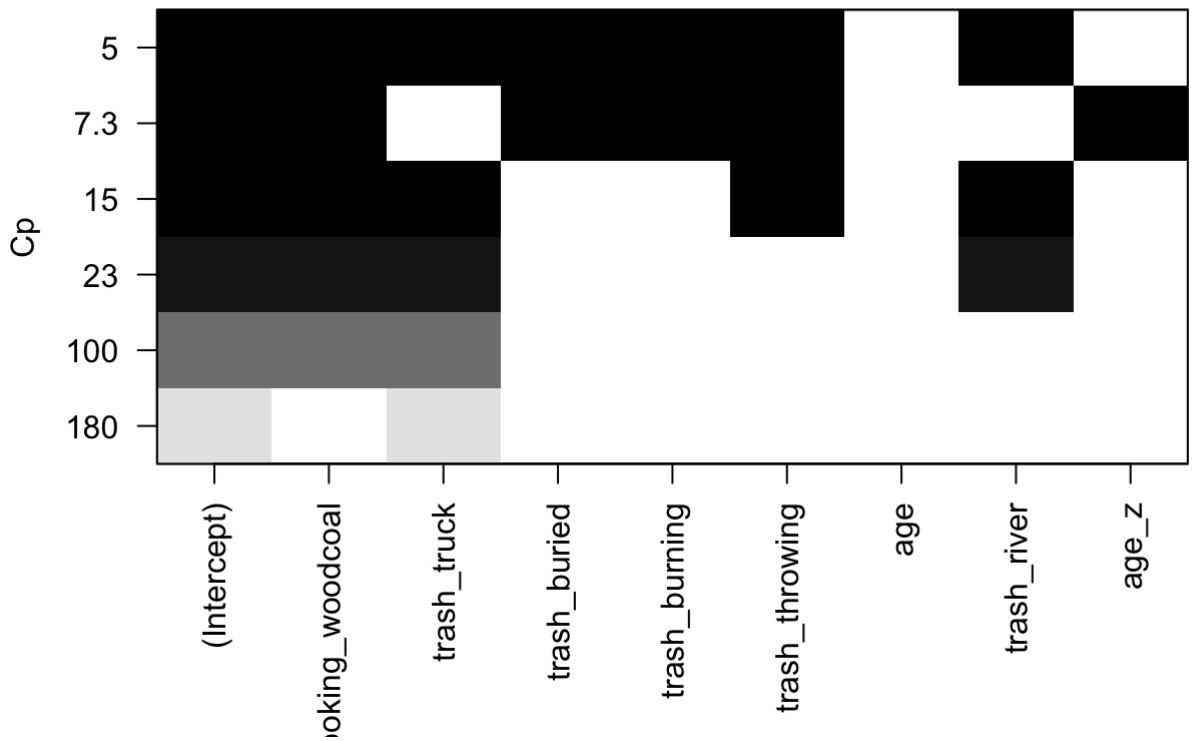



```
# Mallow's CP Plot: G4
cp_gp4 = regsubsets(Target~.,
                        data = group4)
```

```
## 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")
```

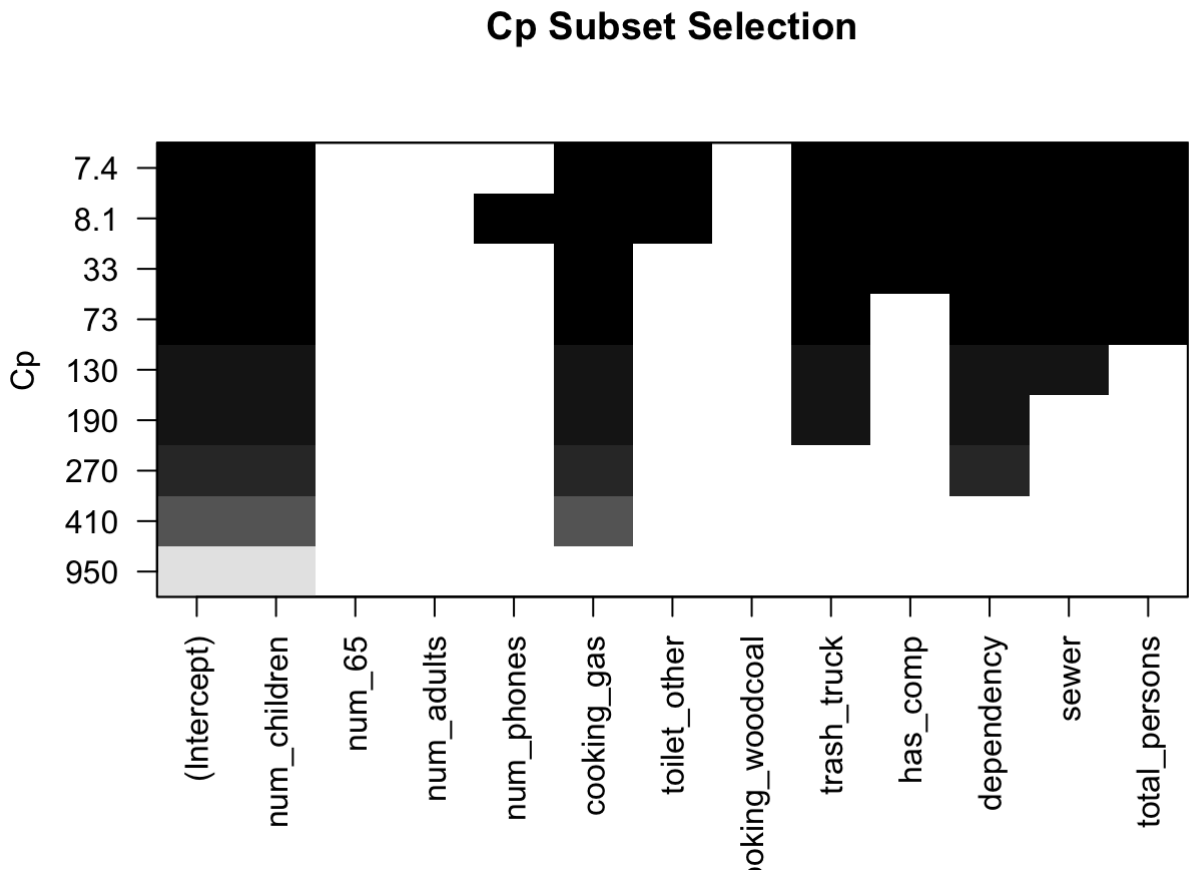



```
# Mallow's CP Plot:G5 (containing all the best variables from previous CP Plots)
group5 <- crp1_train[c(14:16)]
group5$Target <- crp1_train$Target
group5$total_persons <- crp1_train$total_persons
group5$num_phones <- crp1_train$num_phones
group5$cooking_gas <- crp1_train$cooking_gas
group5$toilet_other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

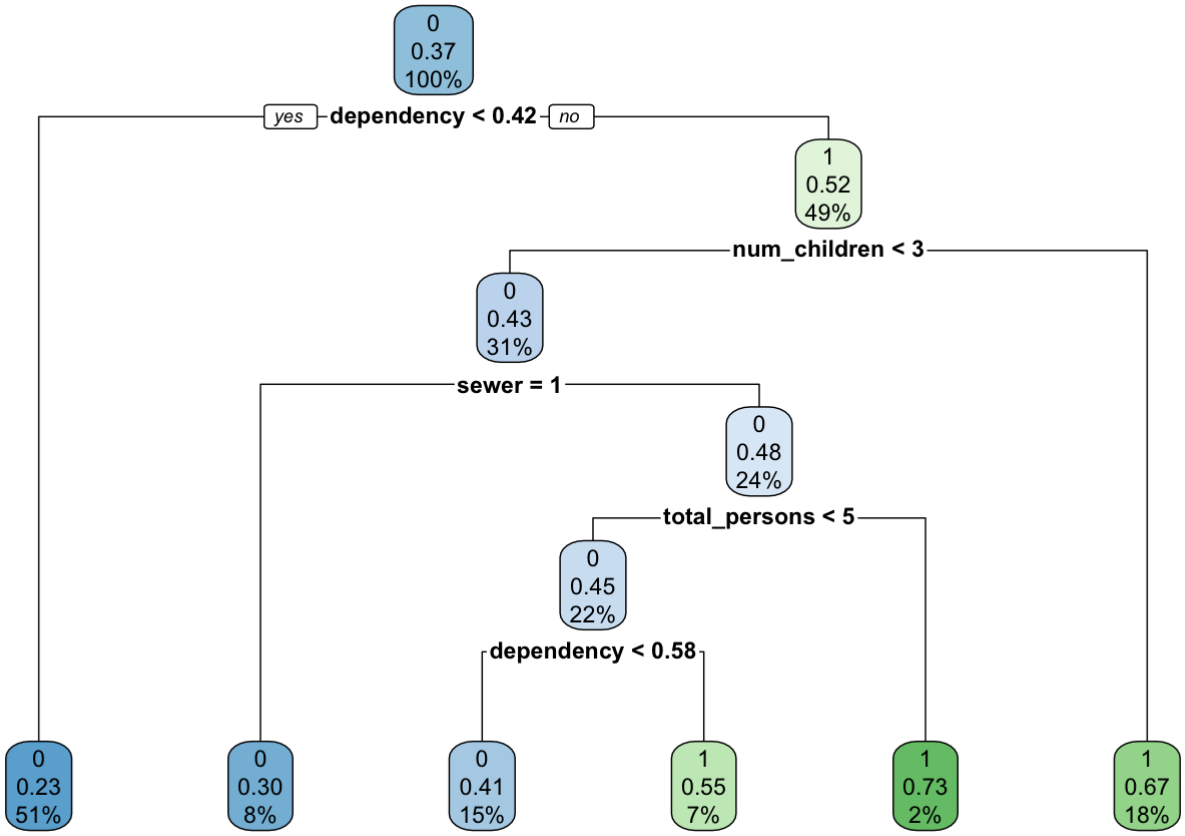
```
# Model 1: Random Forest Model
library(randomForest)
model1 <- randomForest(Target~., data = train)
```

```
# 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)
```

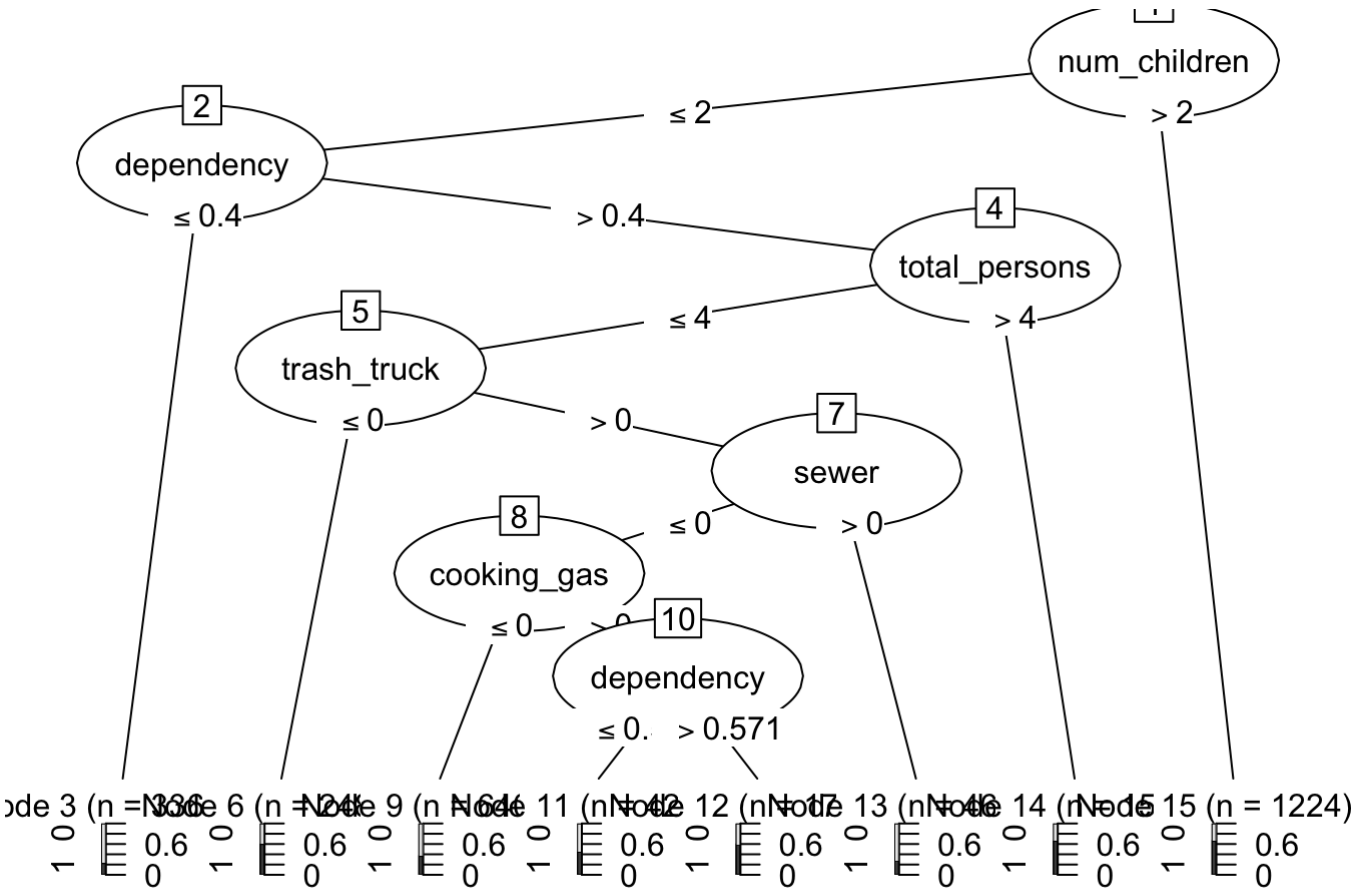



```
# Evaluation of Model 2: CART Decision Tree

# 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
```

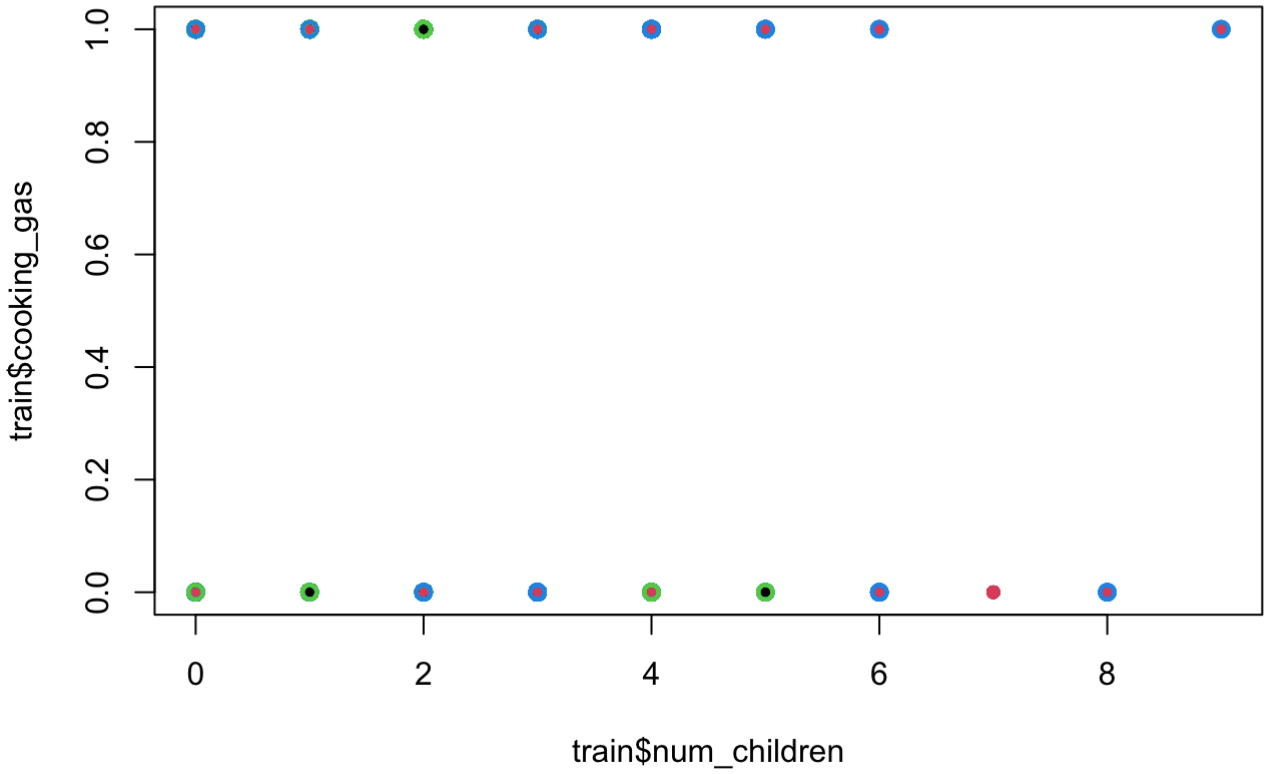
```
# separating train and test
train <- crp1_train[c(14)]
train$Target <- crp1_train$Target
train$cooking_gas <- crp1_train$cooking_gas
train$trash_truck <- crp1_train$trash_truck
train$dependency <- crp1_train$dependency
train$sewer <- crp1_train$sewer
train$total_persons <- crp1_train$total_persons
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$Target, k=81)
knn.82 <- knn(train, test, cl = train$Target, k=82)
```

```
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))
```

```
## Confusion Matrix and Statistics
##
##
## knn.81    0    1
##      0 1779   70
##      1   13 1005
##
##                Accuracy : 0.971
##                  95% CI : (0.9642, 0.9769)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9376
##
##  Mcnemar's Test P-Value : 7.906e-10
##
##             Sensitivity : 0.9927
##             Specificity : 0.9349
##          Pos Pred Value : 0.9621
##          Neg Pred Value : 0.9872
##              Prevalence : 0.6250
##          Detection Rate : 0.6205
##    Detection Prevalence : 0.6449
##       Balanced Accuracy : 0.9638
##
##        'Positive' Class : 0
##
```

```
# Evaluation of Model 4: KNN 82
library(caret)
confusionMatrix(table(knn.82, test$Target))
```

```
## Confusion Matrix and Statistics
##
##
## knn.82    0    1
##      0 1783   70
##      1    9 1005
##
##                Accuracy : 0.9724
##                  95% CI : (0.9658, 0.9781)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9405
##
##  Mcnemar's Test P-Value : 1.473e-11
##
##             Sensitivity : 0.9950
##             Specificity : 0.9349
##          Pos Pred Value : 0.9622
##          Neg Pred Value : 0.9911
##              Prevalence : 0.6250
##          Detection Rate : 0.6219
##    Detection Prevalence : 0.6463
##       Balanced Accuracy : 0.9649
##
##        'Positive' Class : 0
##
```

# ADS502-02-SP22 - Final Project

Carr_Aaron

04/18/2022

## Exploratory Data Analysis

## Final Project: Puerto Rico Poverty Index Prediction (Using R)

### 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)

set.seed(333)
```

### Load files into R dataframes

```
# Read training data file into R df
pr_train_df01 <- read.table(file = "502_Final_Train.csv", header = TRUE, sep = ",")

# Review first few rows
print(head(pr_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(pr_train_df01)[1]
```

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

```
# Review column names
#pr_train_cols_df01 <- colnames(pr_train_df01)
#print(pr_train_cols_df01)

# Check training data set features for null values
sapply(pr_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
pr_test_df01 <- read.table(file = "502_Final_Test.csv", header = TRUE, sep = ",")

# Review first few rows
print(head(pr_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(pr_test_df01)[1]
```

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

```
# Check test data set features for null values
sapply(pr_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

```
pr_train_df01$Target <- as.factor(pr_train_df01$Target)
pr_train_df01$has_bathroom <- as.factor(pr_train_df01$has_bathroom)
pr_train_df01$has_refrig <- as.factor(pr_train_df01$has_refrig)
pr_train_df01$has_comp <- as.factor(pr_train_df01$has_comp)
pr_train_df01$disabled <- as.factor(pr_train_df01$disabled)
pr_train_df01$married <- as.factor(pr_train_df01$married)
pr_train_df01$divorced <- as.factor(pr_train_df01$divorced)
pr_train_df01$separated <- as.factor(pr_train_df01$separated)
pr_train_df01$widower <- as.factor(pr_train_df01$widower)
pr_train_df01$single <- as.factor(pr_train_df01$single)
pr_train_df01$water_inside <- as.factor(pr_train_df01$water_inside)
pr_train_df01$water_outside <- as.factor(pr_train_df01$water_outside)
pr_train_df01$no_water <- as.factor(pr_train_df01$no_water)
pr_train_df01$gov_elec <- as.factor(pr_train_df01$gov_elec)
pr_train_df01$private_elec <- as.factor(pr_train_df01$private_elec)
pr_train_df01$no_elec <- as.factor(pr_train_df01$no_elec)
pr_train_df01$coop_elec <- as.factor(pr_train_df01$coop_elec)
pr_train_df01$no_toilet <- as.factor(pr_train_df01$no_toilet)
pr_train_df01$sewer <- as.factor(pr_train_df01$sewer)
pr_train_df01$septictank <- as.factor(pr_train_df01$septictank)
pr_train_df01$latrine <- as.factor(pr_train_df01$latrine)
pr_train_df01$toilet_other <- as.factor(pr_train_df01$toilet_other)
pr_train_df01$no_cooking_energy <- as.factor(pr_train_df01$no_cooking_energy)
pr_train_df01$cooking_.elec <- as.factor(pr_train_df01$cooking_.elec)
pr_train_df01$cooking_gas <- as.factor(pr_train_df01$cooking_gas)
pr_train_df01$cooking_woodcoal <- as.factor(pr_train_df01$cooking_woodcoal)
pr_train_df01$trash_truck <- as.factor(pr_train_df01$trash_truck)
pr_train_df01$trash_buried <- as.factor(pr_train_df01$trash_buried)
pr_train_df01$trash_burning <- as.factor(pr_train_df01$trash_burning)
pr_train_df01$trash_throwing <- as.factor(pr_train_df01$trash_throwing)
```

```
pr_train_df01$trash_river <- as.factor(pr_train_df01$trash_river)
```

### Factorize test binary/categorical data

```
pr_test_df01$Target <- as.factor(pr_test_df01$Target)
pr_test_df01$has_bathroom <- as.factor(pr_test_df01$has_bathroom)
pr_test_df01$has_refrig <- as.factor(pr_test_df01$has_refrig)
pr_test_df01$has_comp <- as.factor(pr_test_df01$has_comp)
pr_test_df01$disabled <- as.factor(pr_test_df01$disabled)
pr_test_df01$married <- as.factor(pr_test_df01$married)
pr_test_df01$divorced <- as.factor(pr_test_df01$divorced)
pr_test_df01$separated <- as.factor(pr_test_df01$separated)
pr_test_df01$widower <- as.factor(pr_test_df01$widower)
pr_test_df01$single <- as.factor(pr_test_df01$single)
pr_test_df01$water_inside <- as.factor(pr_test_df01$water_inside)
pr_test_df01$water_outside <- as.factor(pr_test_df01$water_outside)
pr_test_df01$no_water <- as.factor(pr_test_df01$no_water)
pr_test_df01$gov_elec <- as.factor(pr_test_df01$gov_elec)
pr_test_df01$private_elec <- as.factor(pr_test_df01$private_elec)
pr_test_df01$no_elec <- as.factor(pr_test_df01$no_elec)
pr_test_df01$coop_elec <- as.factor(pr_test_df01$coop_elec)
pr_test_df01$no_toilet <- as.factor(pr_test_df01$no_toilet)
pr_test_df01$sewer <- as.factor(pr_test_df01$sewer)
pr_test_df01$septictank <- as.factor(pr_test_df01$septictank)
pr_test_df01$latrine <- as.factor(pr_test_df01$latrine)
pr_test_df01$toilet_other <- as.factor(pr_test_df01$toilet_other)
pr_test_df01$no_cooking_energy <- as.factor(pr_test_df01$no_cooking_energy)
pr_test_df01$cooking_.elec <- as.factor(pr_test_df01$cooking_.elec)
pr_test_df01$cooking_gas <- as.factor(pr_test_df01$cooking_gas)
pr_test_df01$cooking_woodcoal <- as.factor(pr_test_df01$cooking_woodcoal)
pr_test_df01$trash_truck <- as.factor(pr_test_df01$trash_truck)
pr_test_df01$trash_buried <- as.factor(pr_test_df01$trash_buried)
pr_test_df01$trash_burning <- as.factor(pr_test_df01$trash_burning)
pr_test_df01$trash_throwing <- as.factor(pr_test_df01$trash_throwing)
pr_test_df01$trash_river <- as.factor(pr_test_df01$trash_river)
```

### Parse columns into different sub df's for modeling

```
pr_y01_lst <- c("Target")
pr_x01_lst <- c("num_children", "cooking_gas", "trash_truck", "dependency", "sewer",
↪  "total_persons")
pr_xy01_lst <- c("Target", "num_children", "cooking_gas", "trash_truck", "dependency",
↪  "sewer", "total_persons")
pr_train_xy01_df01 <- subset(x = pr_train_df01, select = pr_xy01_lst)
#print(pr_train_xy01_df01)

pr_train_x01_df01 <- subset(x = pr_train_df01, select = pr_x01_lst)
#print(pr_train_x01_df01)

pr_test_xy01_df01 <- subset(x = pr_test_df01, select = pr_xy01_lst)
#print(pr_test_xy01_df01)
```

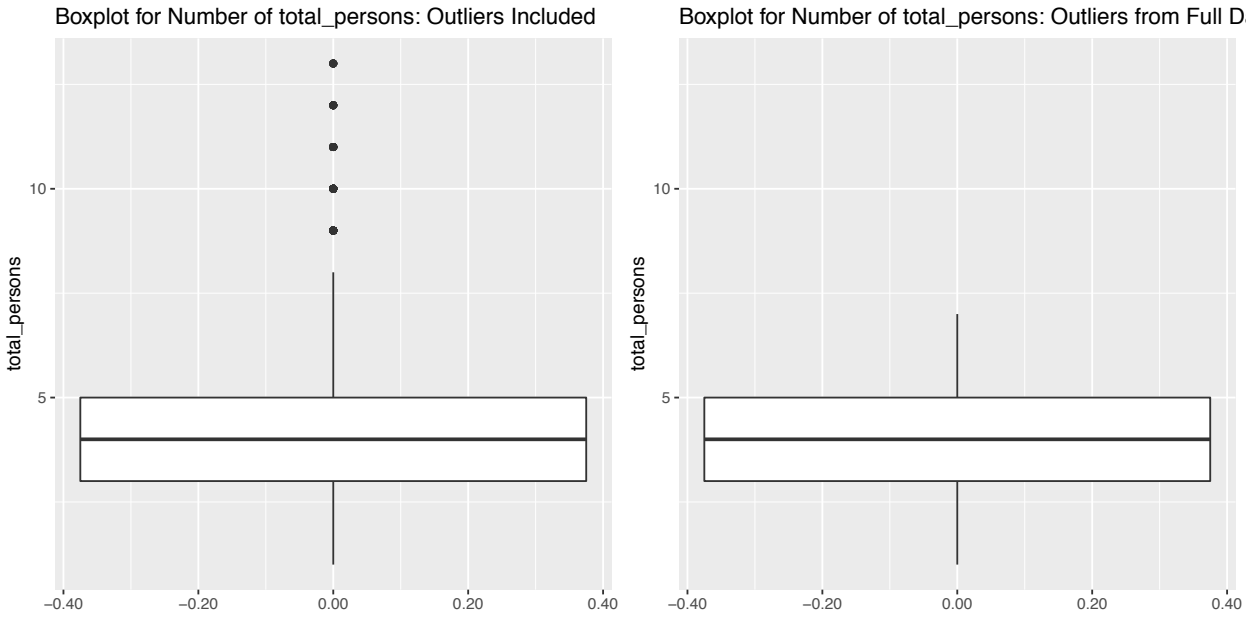
```
pr_test_x01_df01 <- subset(x = pr_test_df01, select = pr_x01_lst)
#print(pr_test_x01_df01)
```

### 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_stde <- round(sd(df_s1), 3)
xcol_iqr_df01_trunc_stde <- round(sd(df_s2), 3)

var01_min <- min(df[, xcol])
var01_max <- max(df[, xcol])
var02_min <- min(xcol_iqr_df01[, xcol])
var02_max <- max(xcol_iqr_df01[, xcol])

# 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, ": 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, ": Outliers from Full Dataset
  ↪  Excluded"))
```

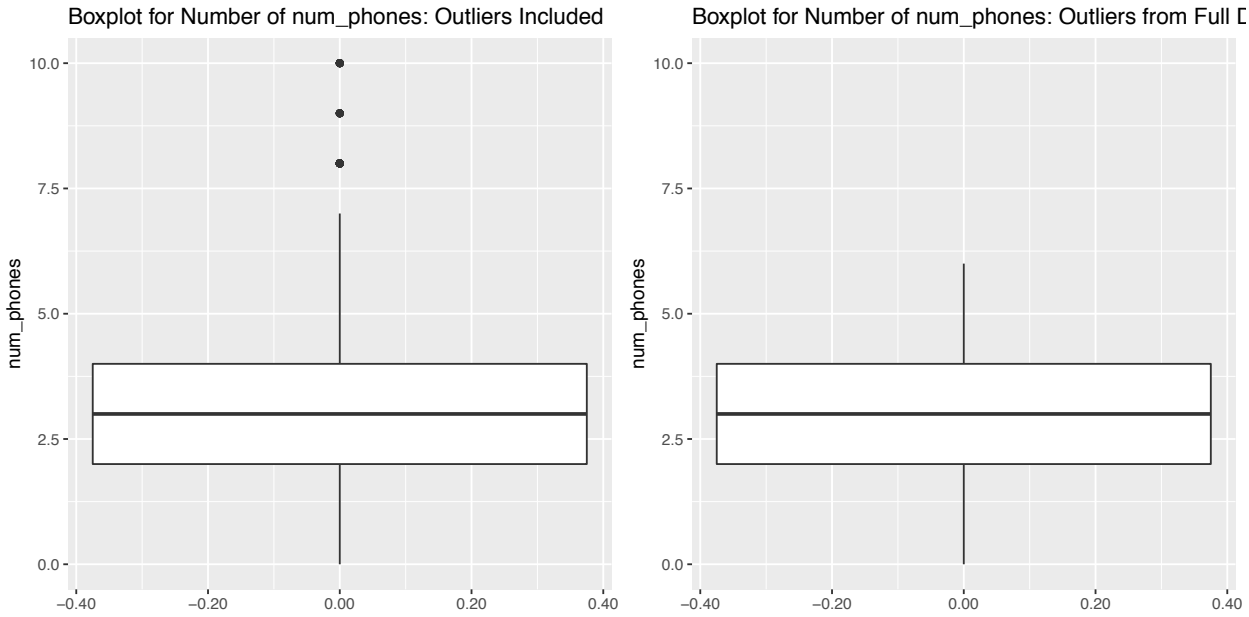
```r
# Map 2 plots to 1 grid
grid.arrange(ggp1, ggp2, ncol = 2)

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

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

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), "%)"),
                   "-",
                   "-",
                   "-",
                   "-",
                   "-",
                   "-",
                   xcol_iqr_df01_trunc_mean,
                   xcol_iqr_df01_trunc_med,
                   var02_min,
                   var02_max,
                   xcol_iqr_df01_trunc_stde
                   )

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 = pr_train_df01)
```



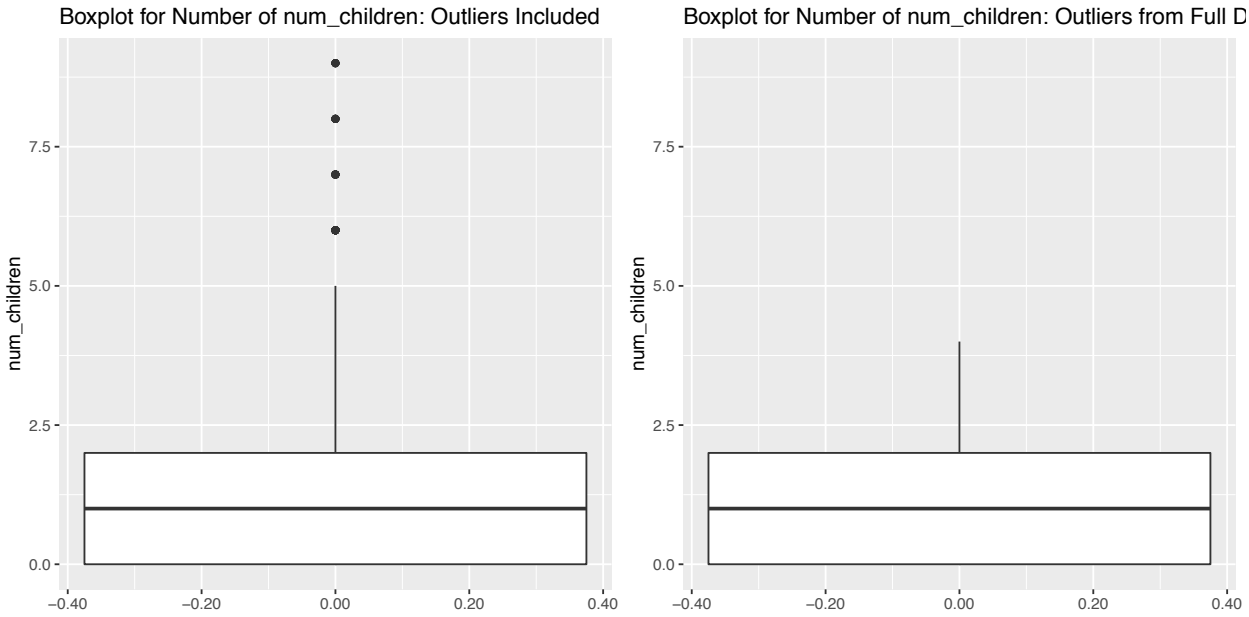

```
##               measure_name   measure_val01     measure_val02
## 1  Variable: total_persons All data points Outliers Excluded
## 2                    Count            6690      6466 (96.7%)
## 3                 NA Count               0                 -
## 4         IQR Outlier Limit            +/-3                 -
## 5           25th Percentile               3                 -
## 6           75th Percentile               5                 -
## 7  Lower Outlier Threshold               0                 -
## 8  Upper Outlier Threshold               8                 -
## 9                     Mean           4.013             3.823
## 10                  Median               4                 4
## 11                     Min               1                 1
## 12                     Max              13                 7
## 13      Standard Deviation           1.766             1.442
```

```
box_comp(xcol = "num_phones", df = pr_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                 -
## 4         IQR Outlier Limit            +/-3                 -
## 5           25th Percentile               2                 -
## 6           75th Percentile               4                 -
## 7  Lower Outlier Threshold              -1                 -
## 8  Upper Outlier Threshold               7                 -
## 9                      Mean           2.826             2.739
## 10                   Median               3                 3
## 11                      Min               0                 0
## 12                      Max              10                 6
## 13      Standard Deviation           1.486             1.329
```

```
box_comp(xcol = "num_children", df = pr_train_df01)
```

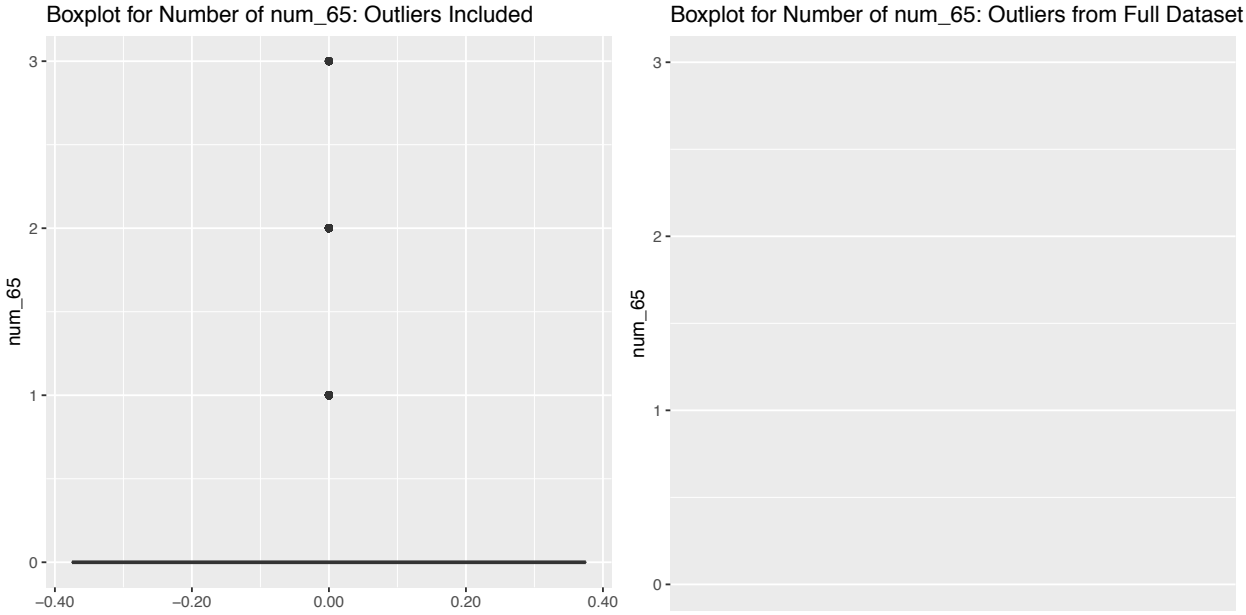



```
##               measure_name   measure_val01     measure_val02
## 1   Variable: num_children All data points Outliers Excluded
## 2                    Count            6690      6508 (97.3%)
## 3                 NA Count               0                 -
## 4         IQR Outlier Limit          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

```
box_comp(xcol = "num_65", df = pr_train_df01)
```

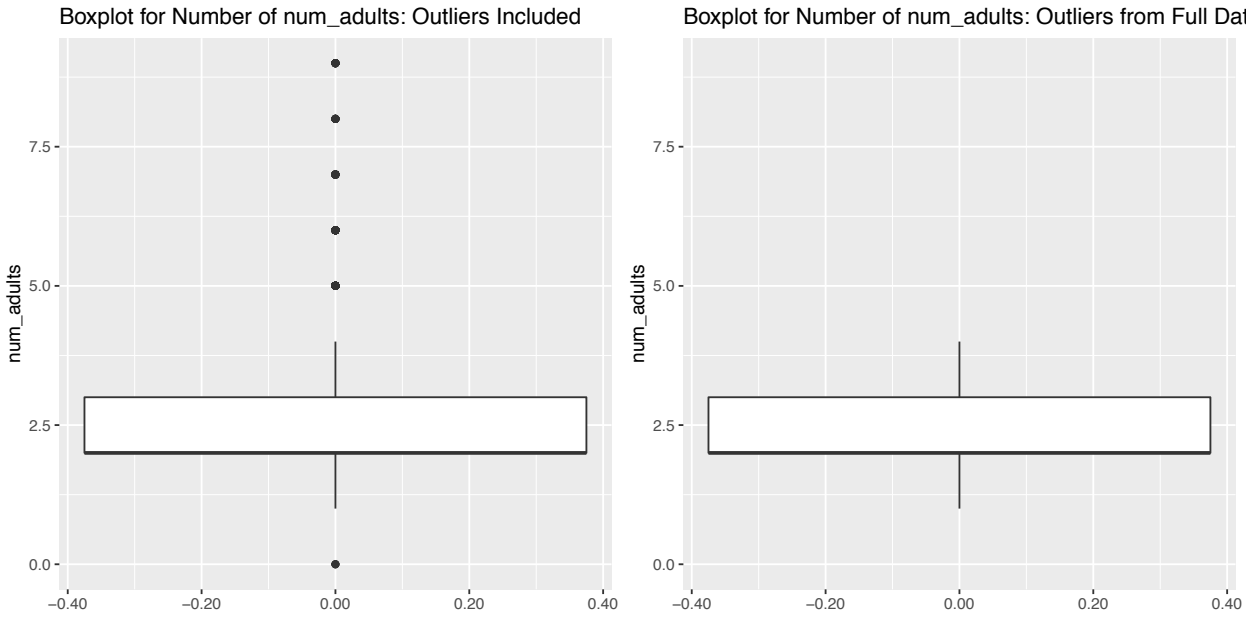
```
## Warning in min(xcol_iqr_df01[, xcol]): no non-missing arguments to min;
## returning Inf
```

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



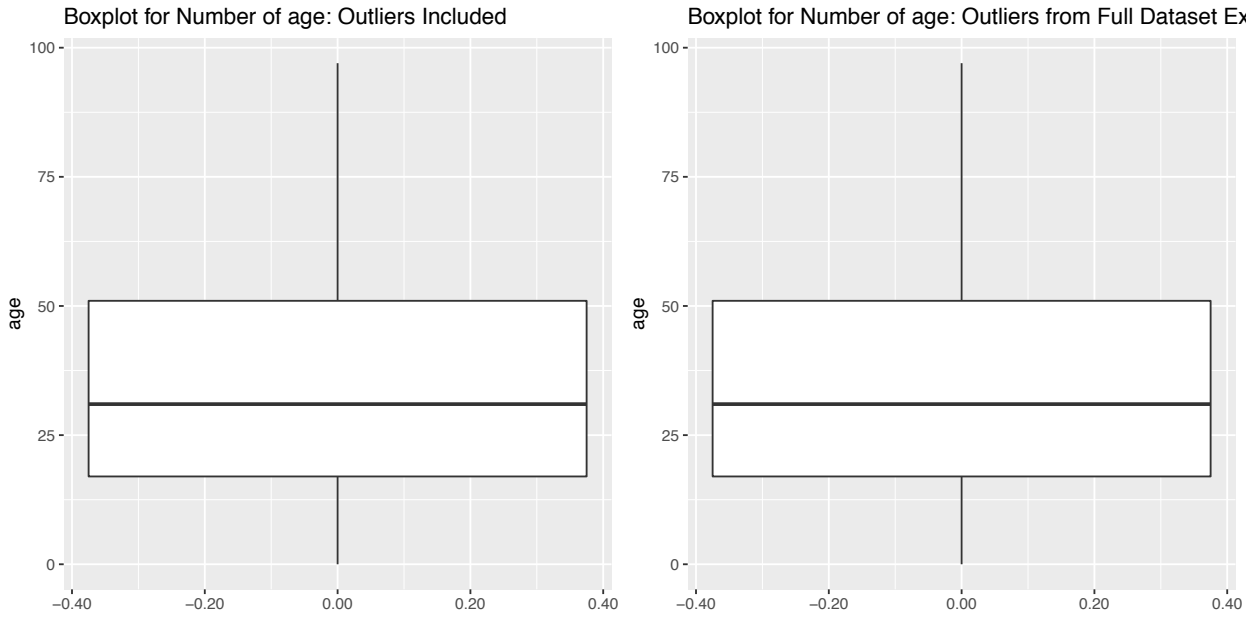

```
##              measure_name   measure_val01     measure_val02
## 1         Variable: num_65 All data points Outliers Excluded
## 2                    Count            6690            0 (0%)
## 3                 NA Count               0                -
## 4        IQR Outlier Limit            +/-0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

```
box_comp(xcol = "num_adults", df = pr_train_df01)
```



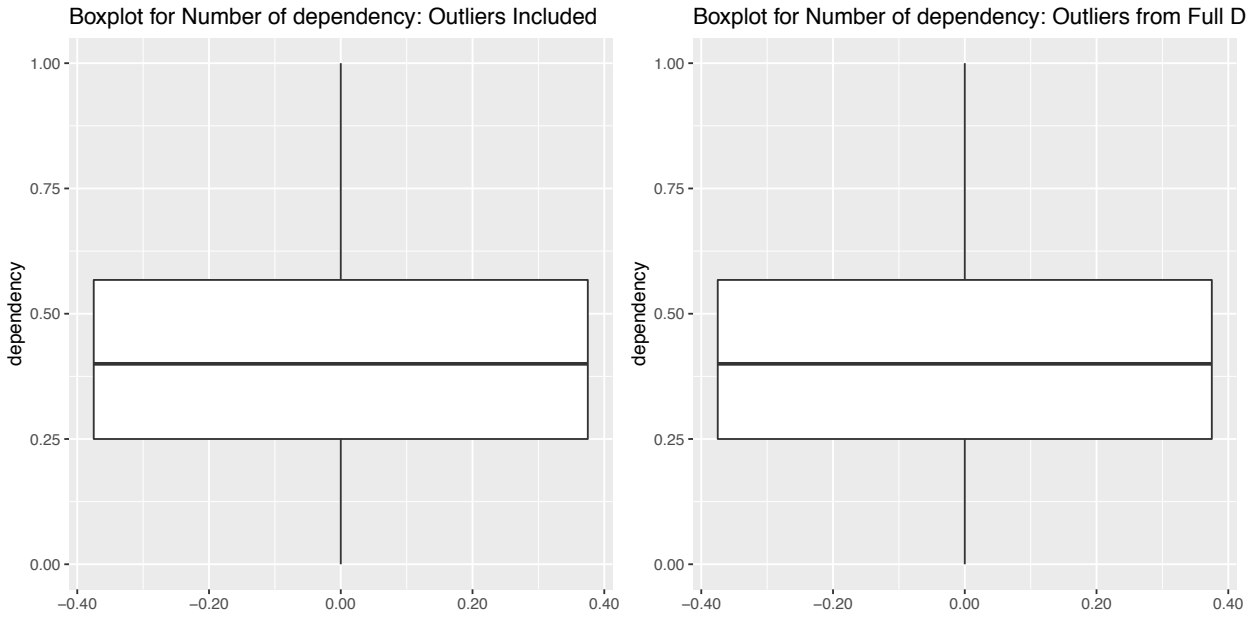

```
##              measure_name   measure_val01     measure_val02
## 1     Variable: num_adults All data points Outliers Excluded
## 2                    Count            6690      6284 (93.9%)
## 3                 NA Count               0                 -
## 4        IQR Outlier Limit          +/-1.5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

```
box_comp(xcol = "age", df = pr_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                 -
## 4       IQR Outlier Limit           +/-51                 -
## 5         25th Percentile              17                 -
## 6         75th Percentile              51                 -
## 7  Lower Outlier Threshold             -34                 -
## 8  Upper Outlier Threshold             102                 -
## 9                    Mean          34.092            34.092
## 10                 Median              31                31
## 11                    Min               0                 0
## 12                    Max              97                97
## 13     Standard Deviation          21.728            21.728
```

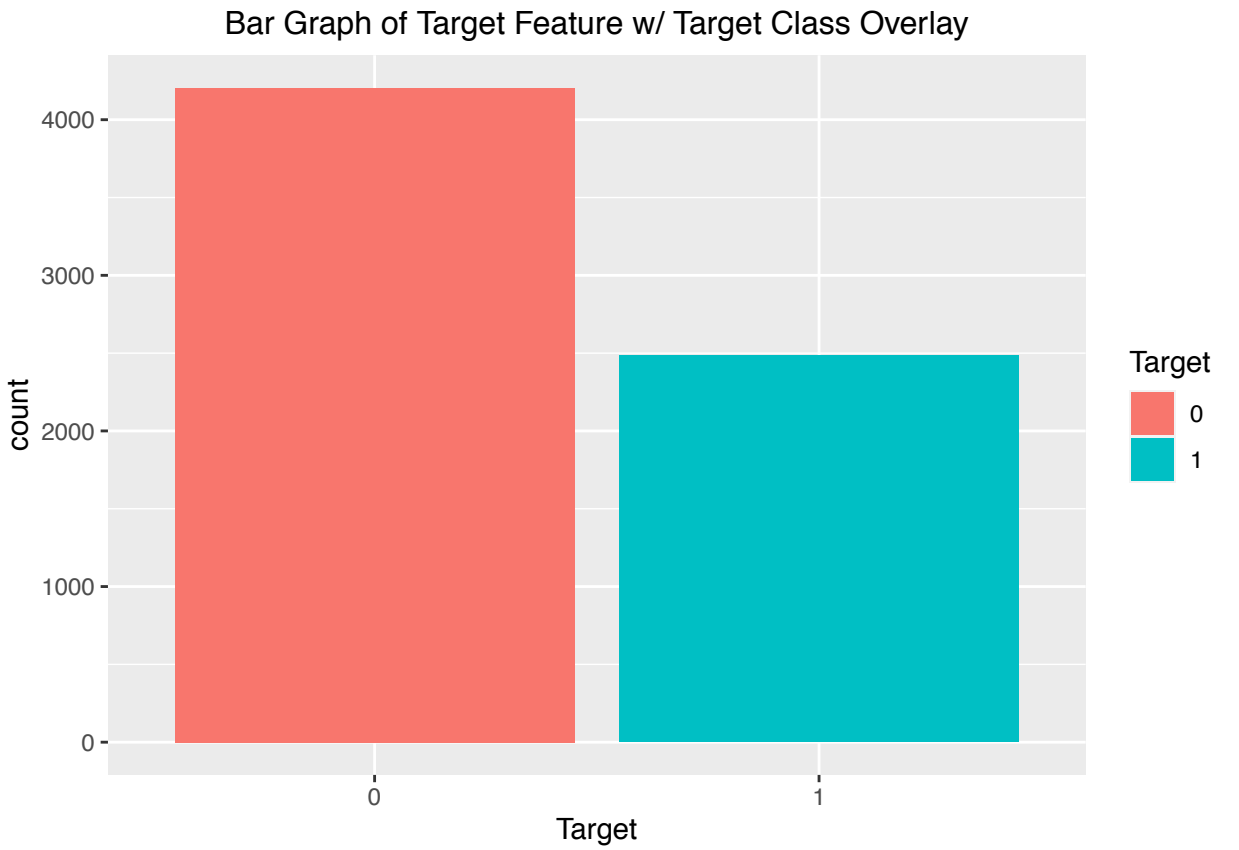
```
box_comp(xcol = "dependency", df = pr_train_df01)
```



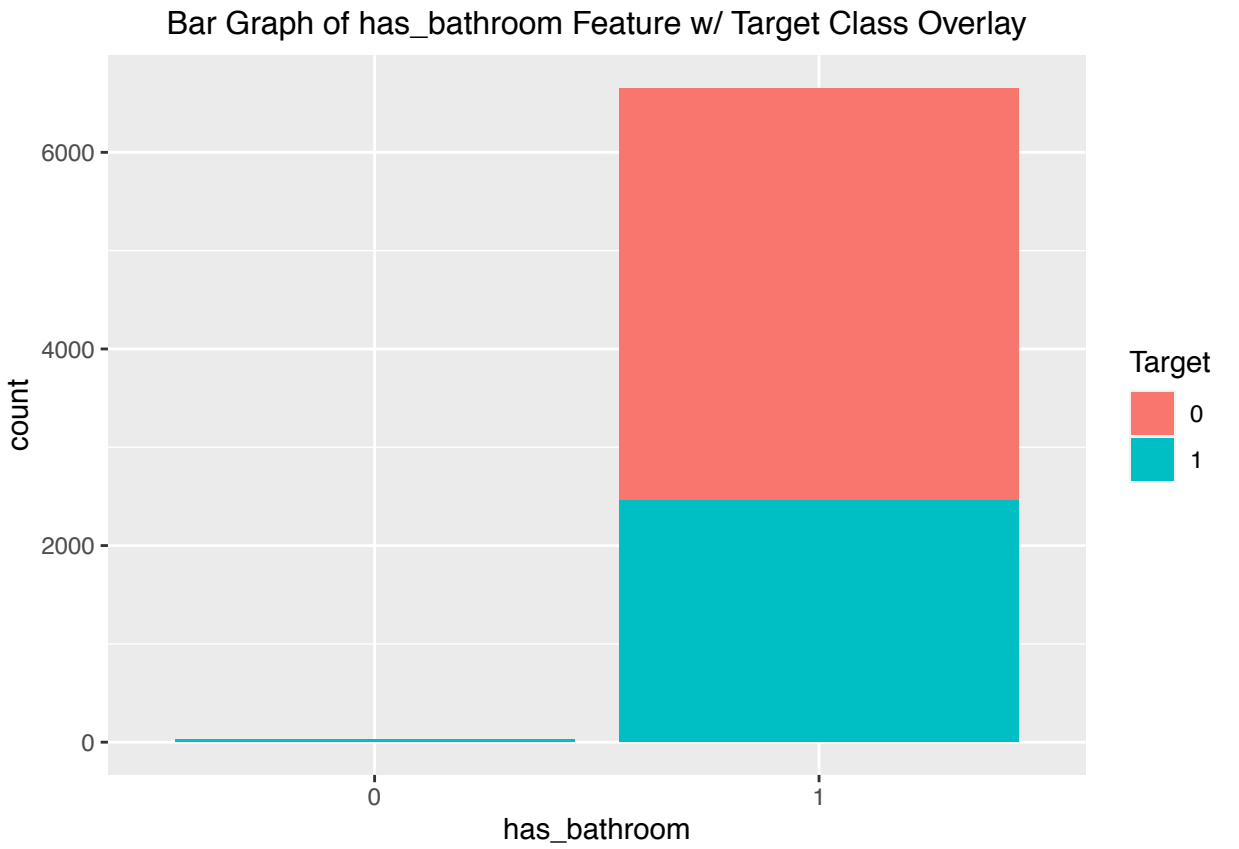

```
##              measure_name        measure_val01     measure_val02
## 1     Variable: dependency      All data points Outliers Excluded
## 2                    Count                 6690       6690 (100%)
## 3                 NA Count                    0                 -
## 4        IQR Outlier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

## Create bar charts for categorical variables

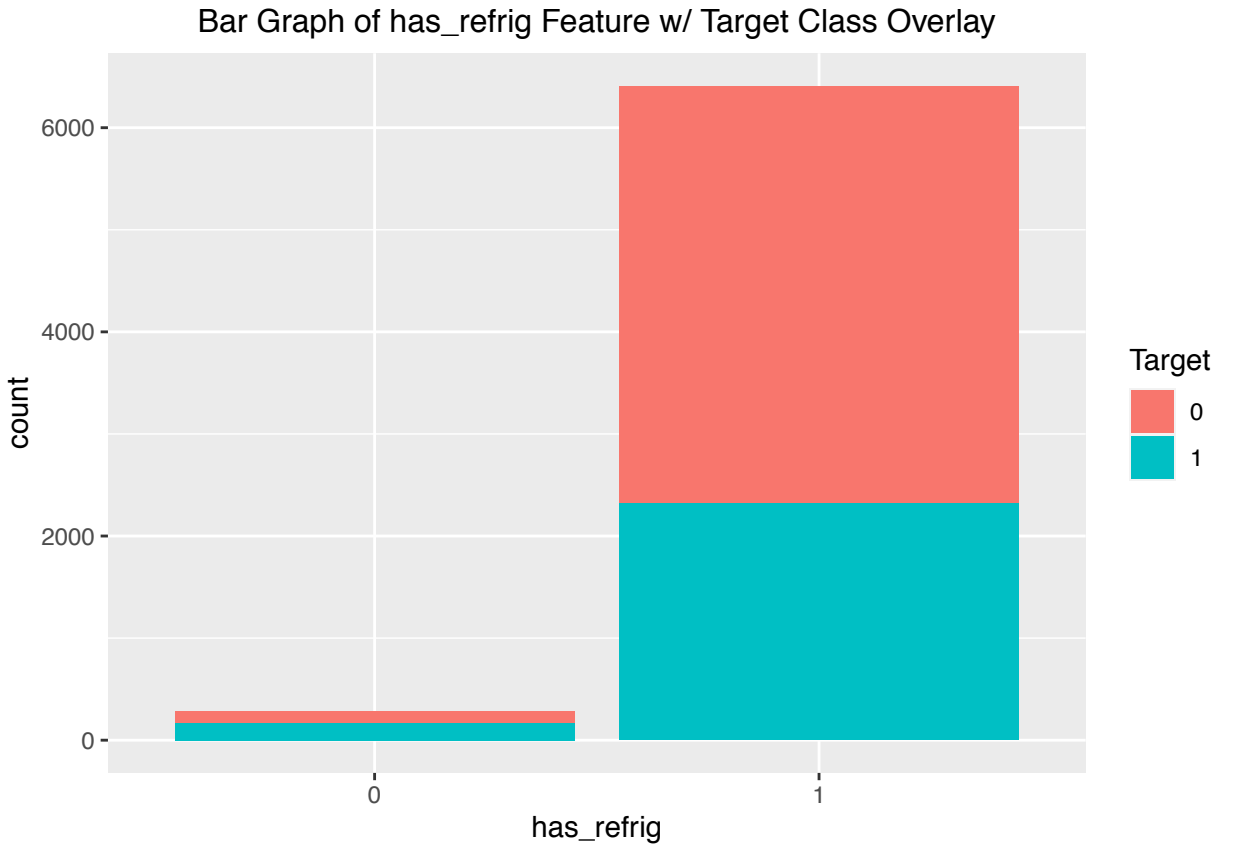
```
# Define function to produce formatted bar charts
plot_bar <- function(x = NA, y = NA, df = NA) {

# 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 = pr_train_df01)
```



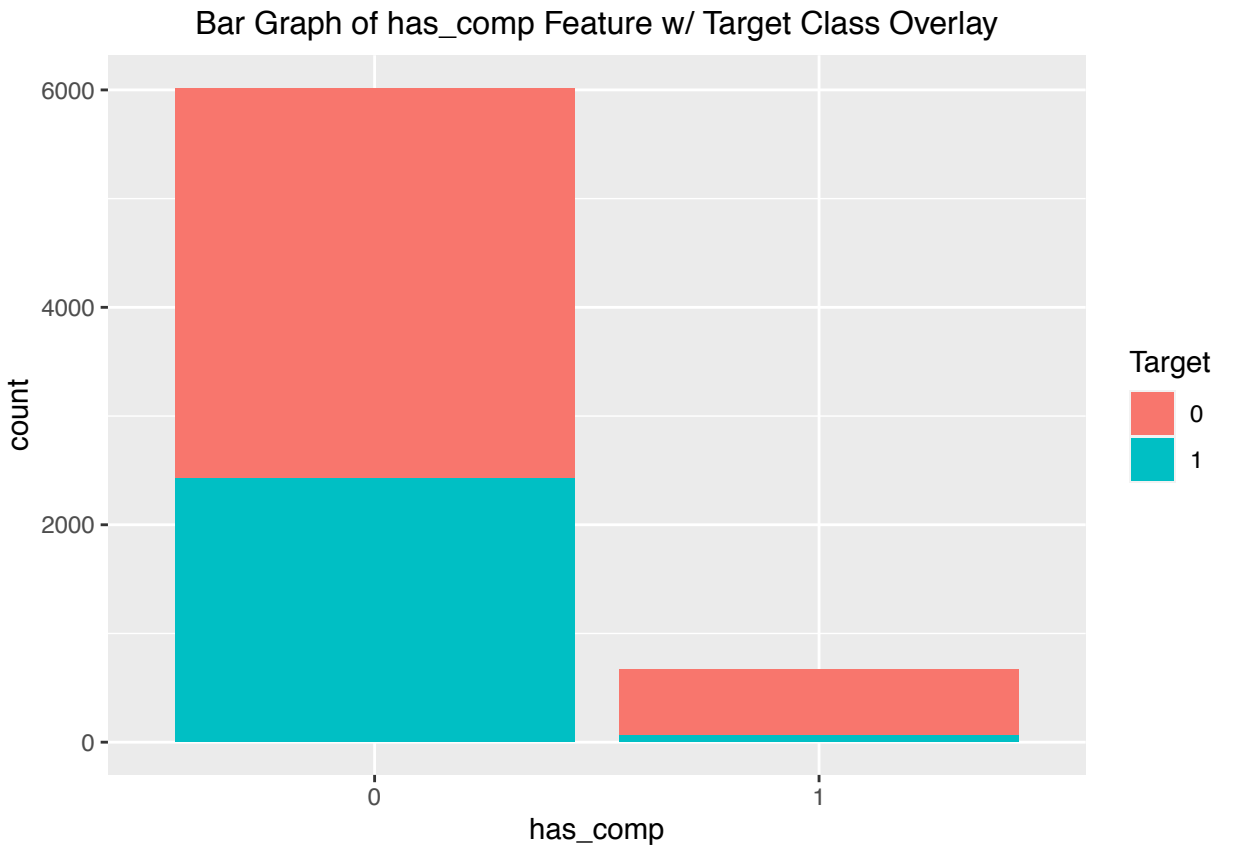

```
plot_bar(x = "has_bathroom", y = "Target", df = pr_train_df01)
```




```
plot_bar(x = "has_refrig", y = "Target", df = pr_train_df01)
```

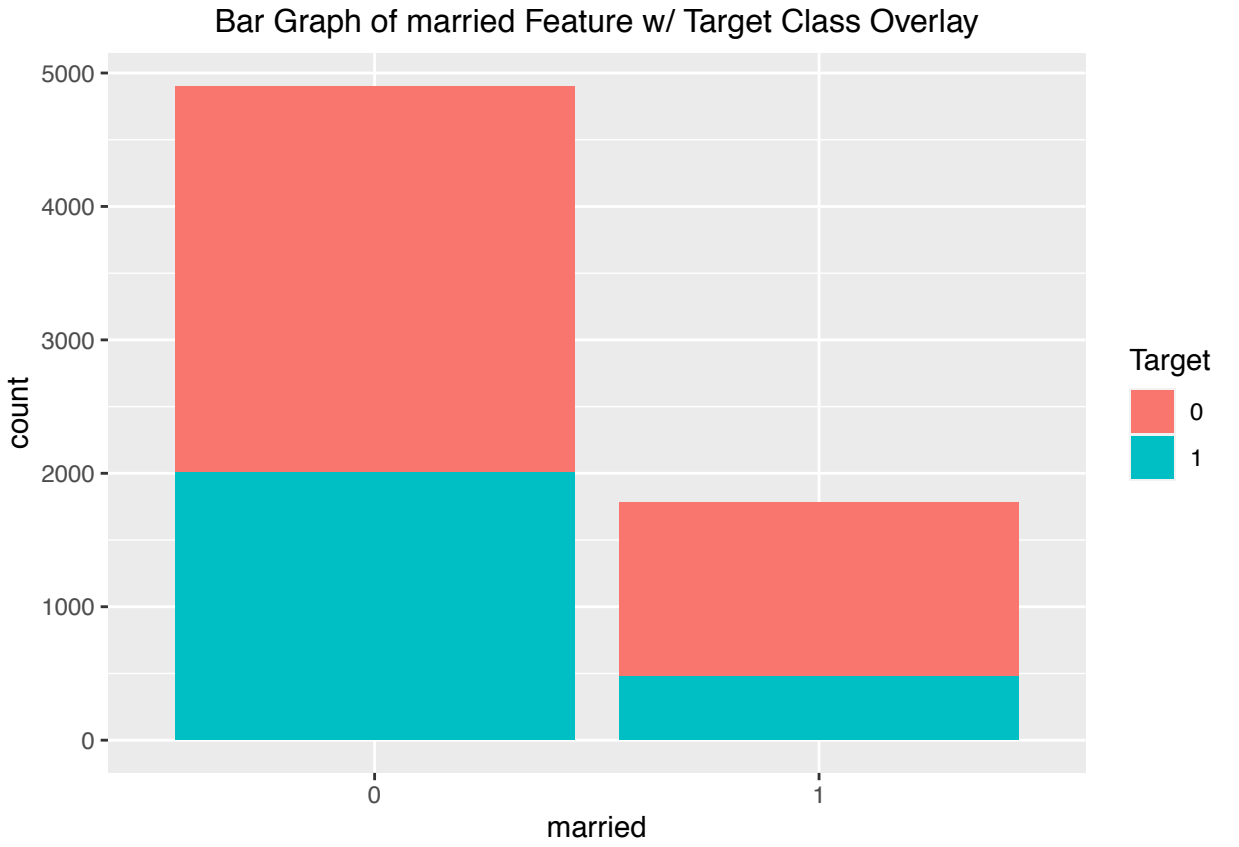



```
plot_bar(x = "has_comp", y = "Target", df = pr_train_df01)
```



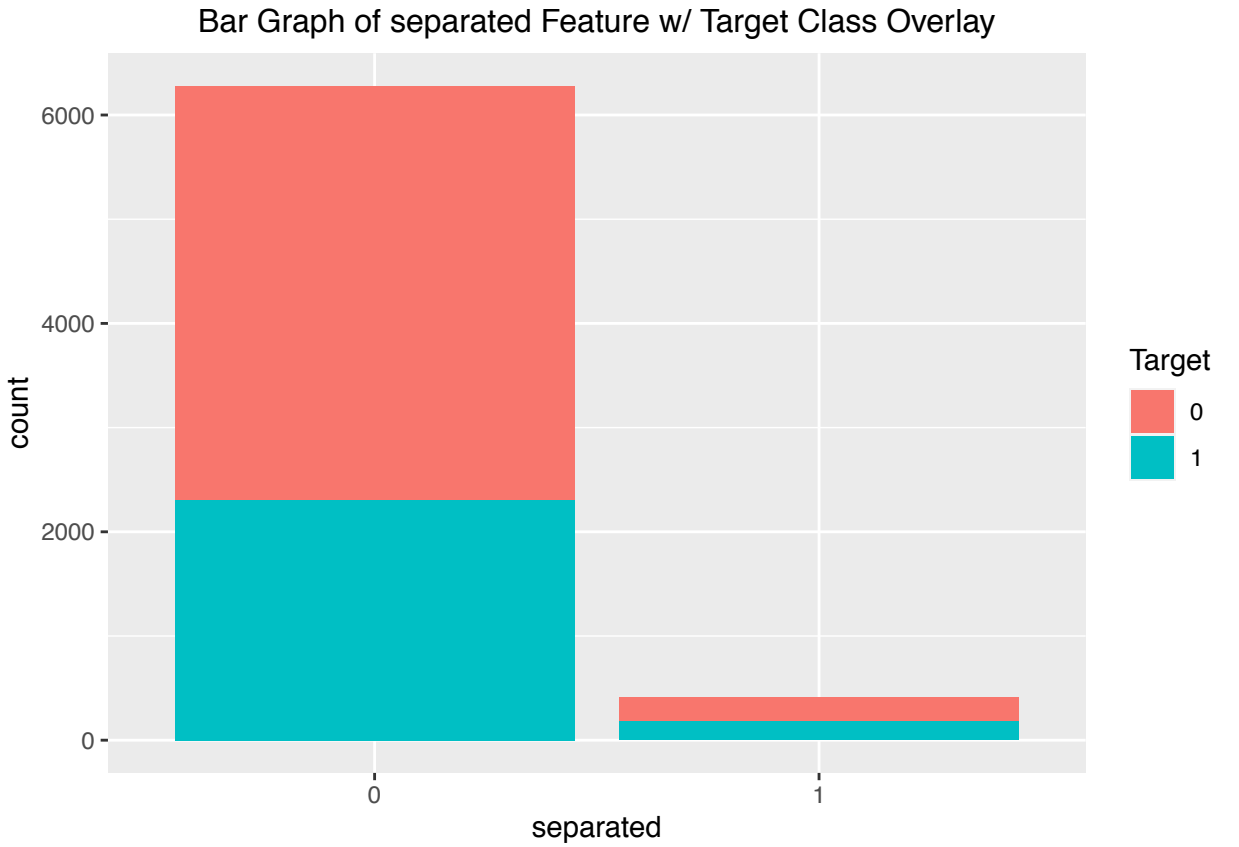

```
plot_bar(x = "disabled", y = "Target", df = pr_train_df01)
```


```
plot_bar(x = "married", y = "Target", df = pr_train_df01)
```




```
plot_bar(x = "divorced", y = "Target", df = pr_train_df01)
```

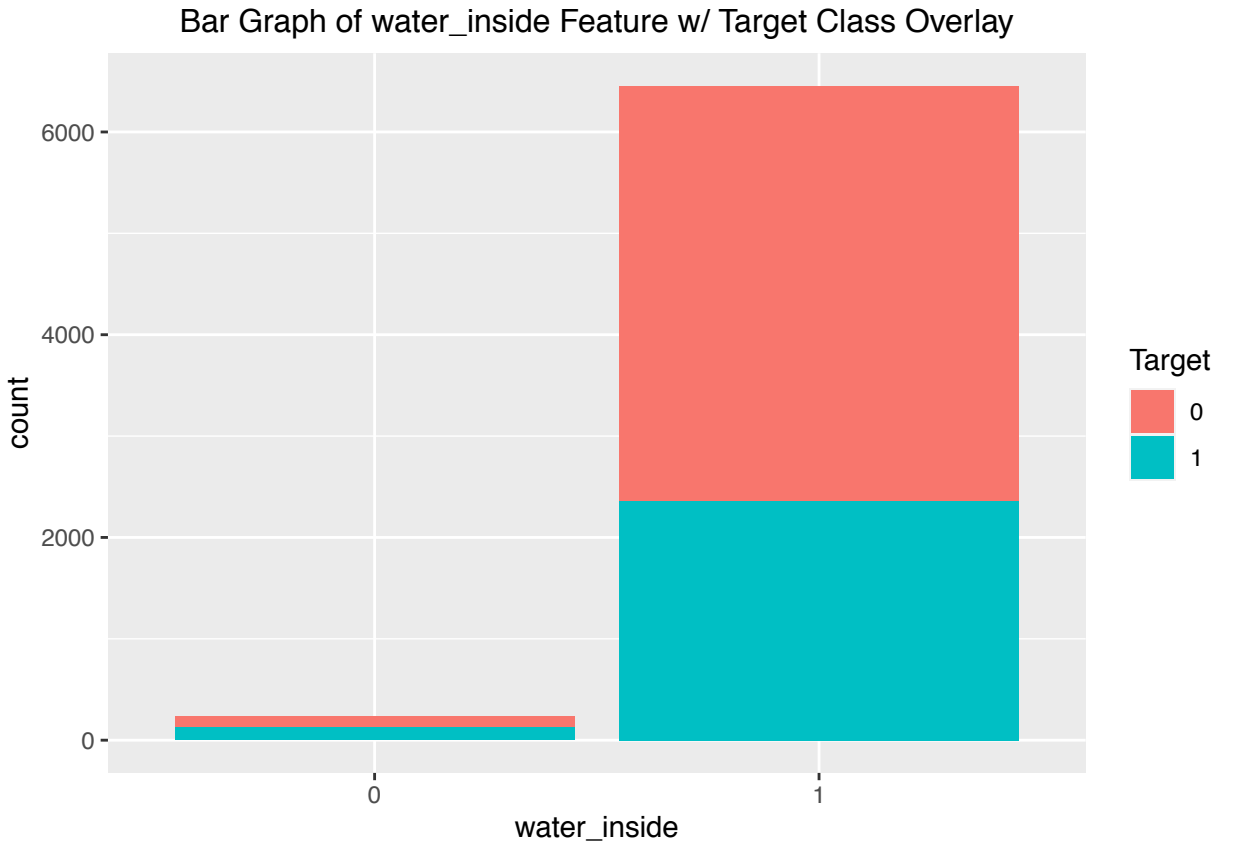

```
plot_bar(x = "separated", y = "Target", df = pr_train_df01)
```



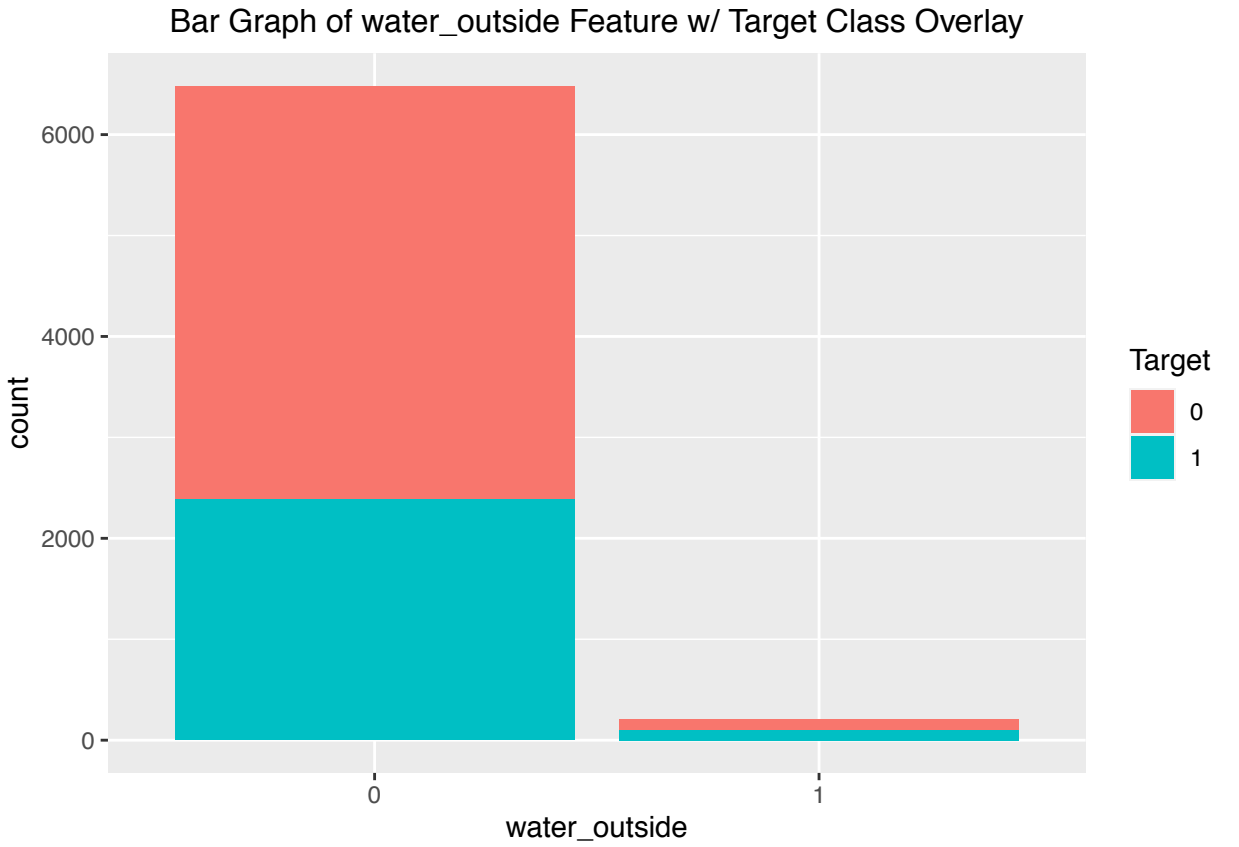

```
plot_bar(x = "widower", y = "Target", df = pr_train_df01)
```


```
plot_bar(x = "single", y = "Target", df = pr_train_df01)
```

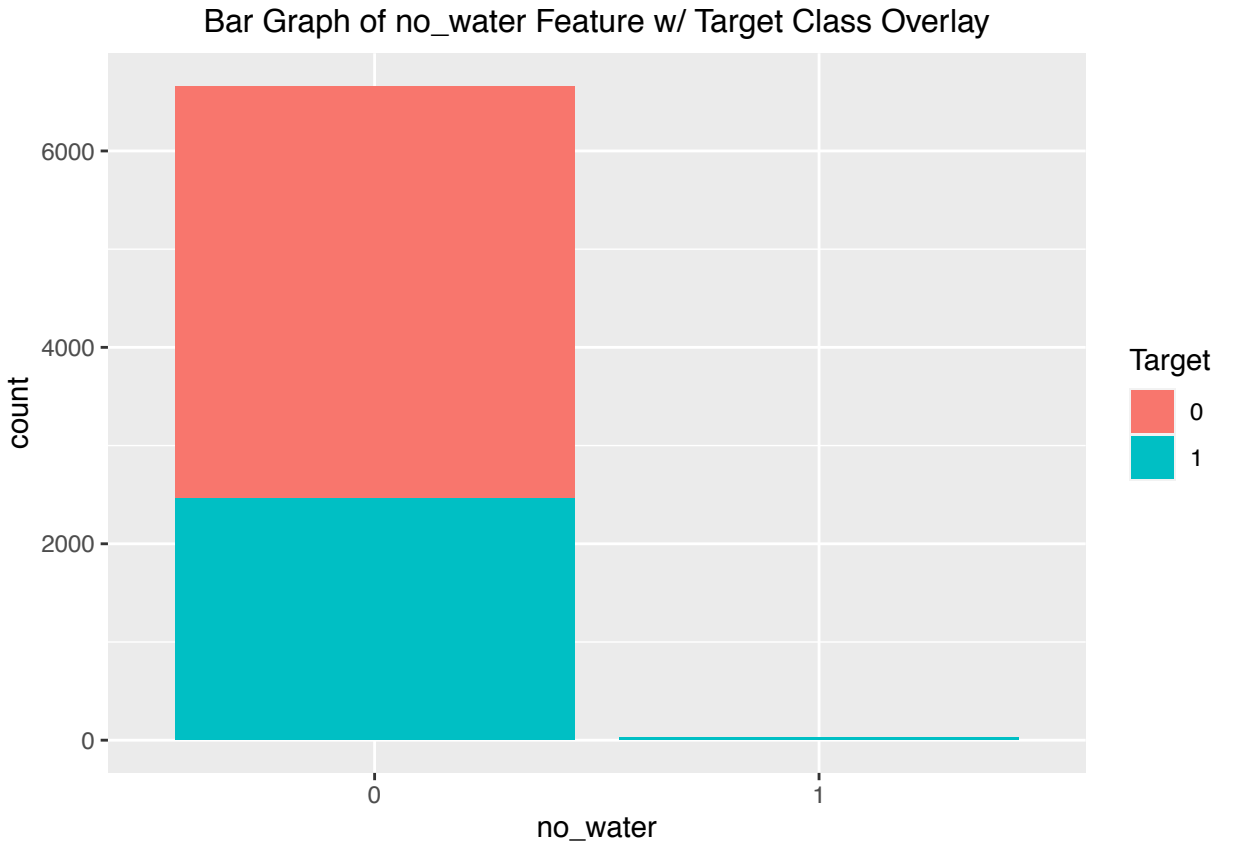

```
plot_bar(x = "water_inside", y = "Target", df = pr_train_df01)
```




```
plot_bar(x = "water_outside", y = "Target", df = pr_train_df01)
```

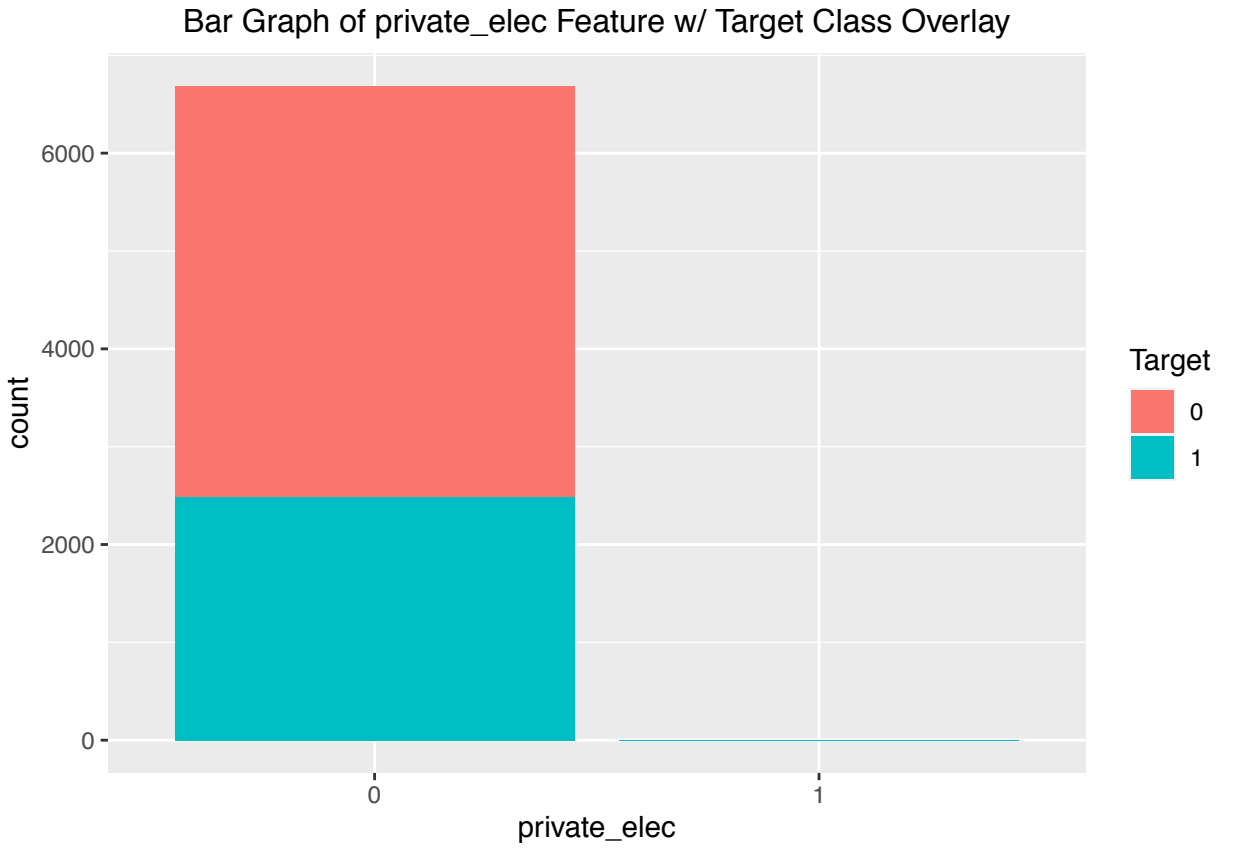



```
plot_bar(x = "no_water", y = "Target", df = pr_train_df01)
```

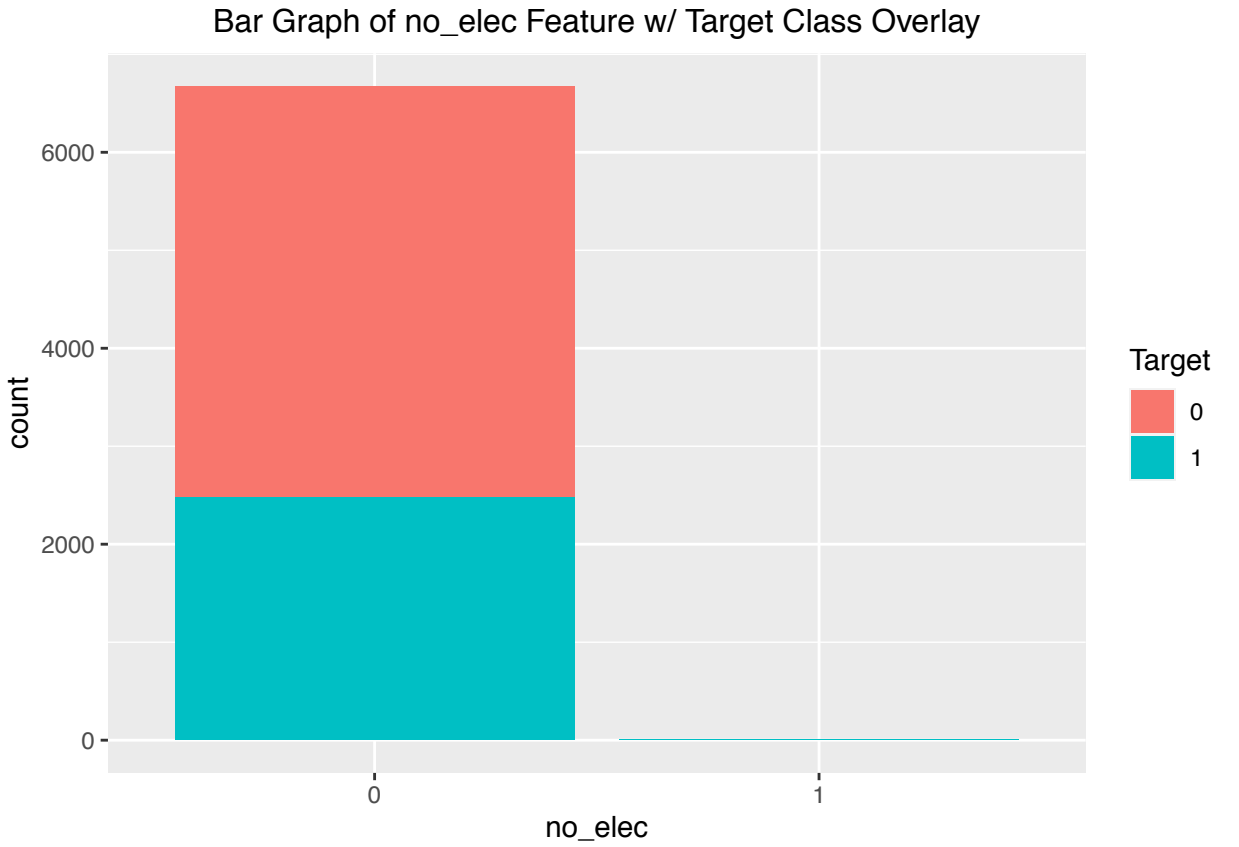



```
plot_bar(x = "gov_elec", y = "Target", df = pr_train_df01)
```


```
plot_bar(x = "private_elec", y = "Target", df = pr_train_df01)
```

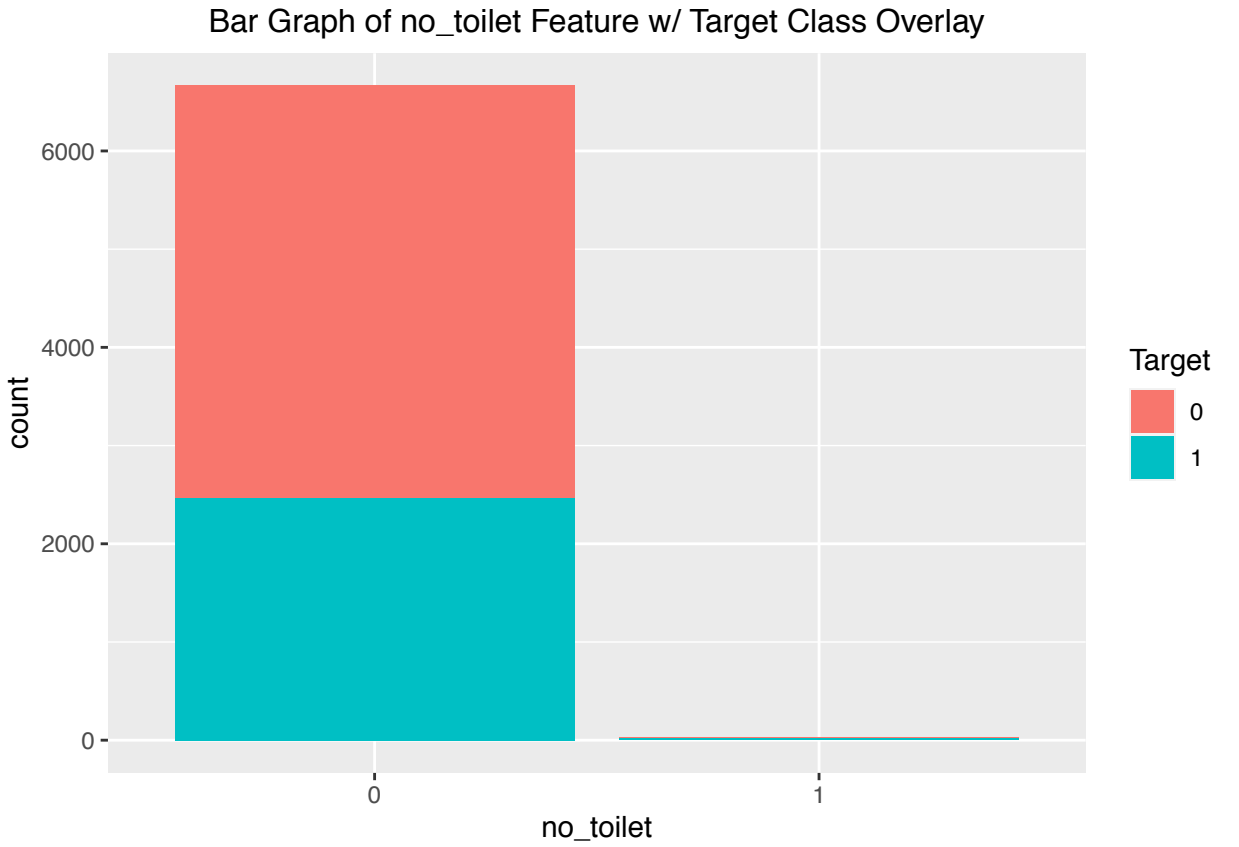



```
plot_bar(x = "no_elec", y = "Target", df = pr_train_df01)
```

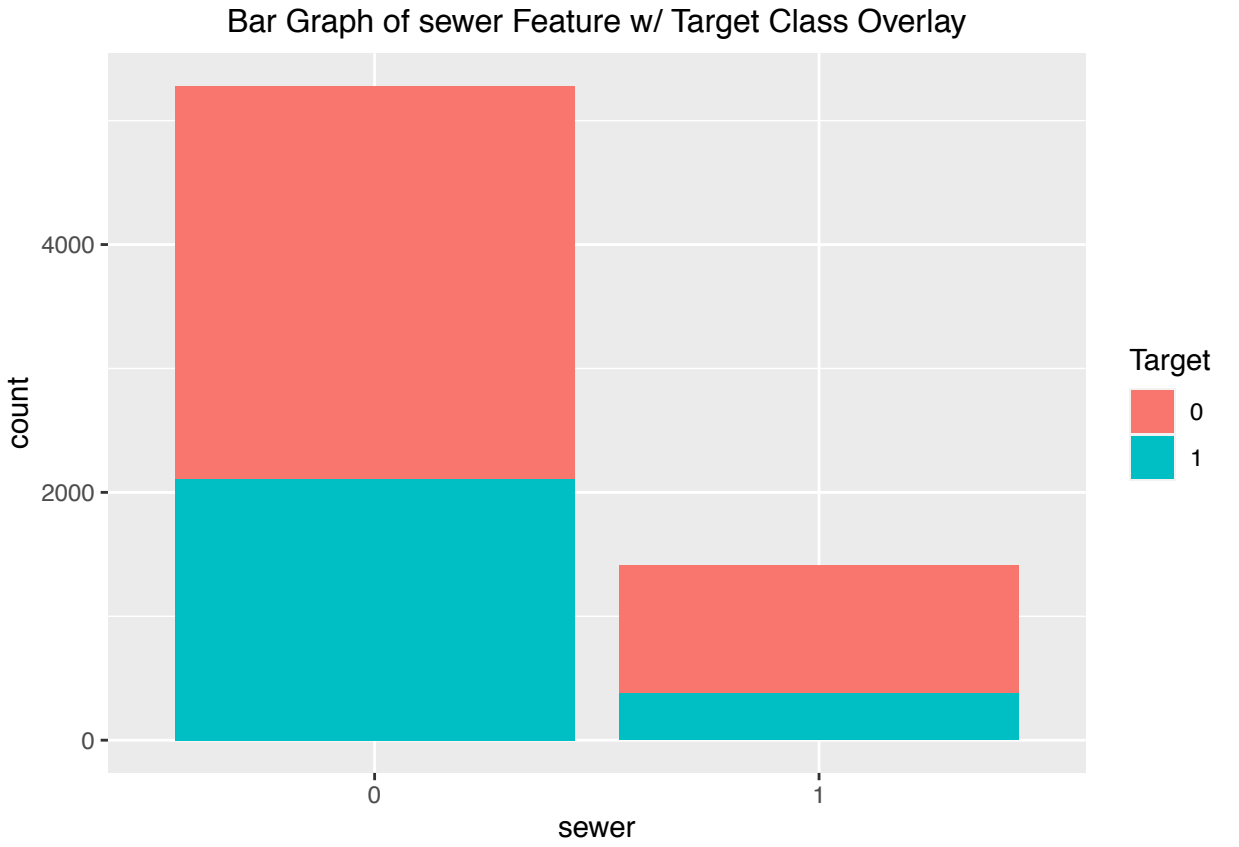



```
plot_bar(x = "coop_elec", y = "Target", df = pr_train_df01)
```


```
plot_bar(x = "no_toilet", y = "Target", df = pr_train_df01)
```



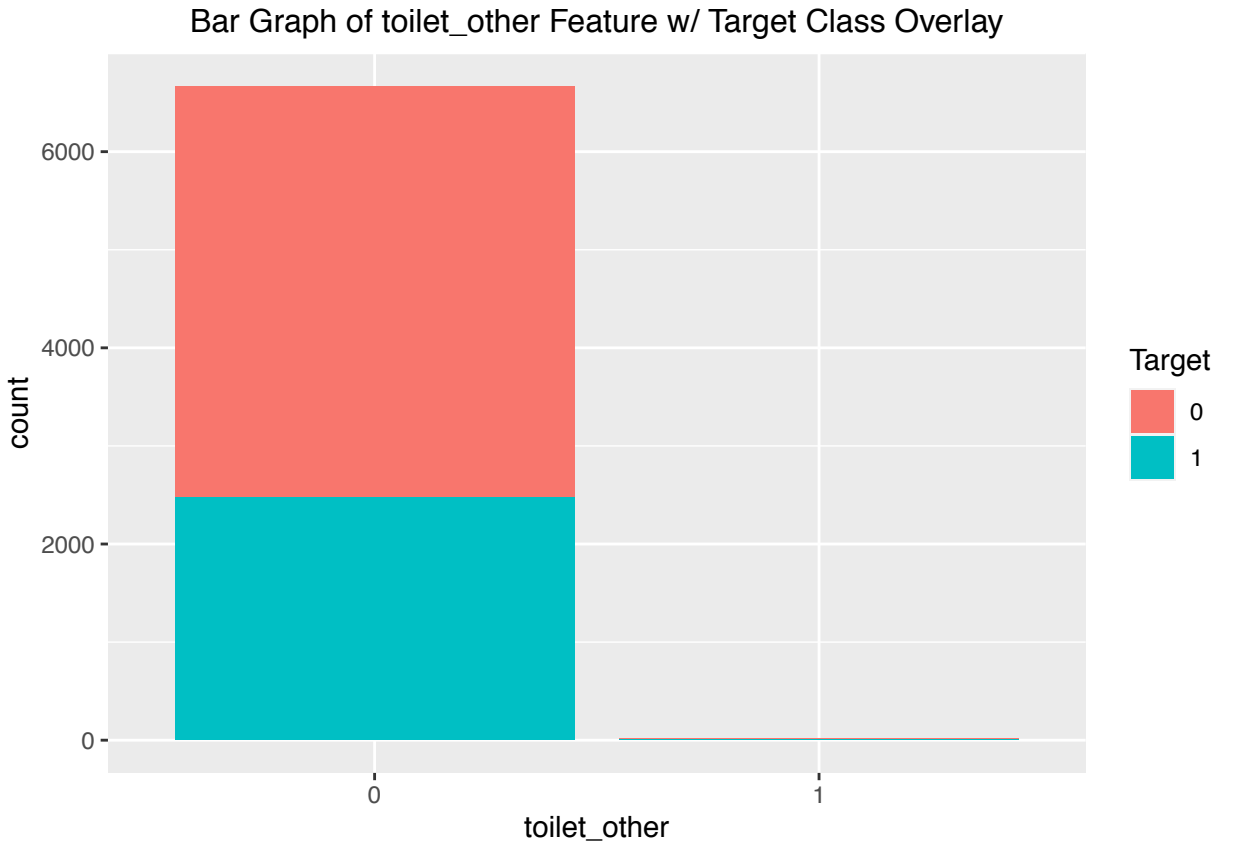

```
plot_bar(x = "sewer", y = "Target", df = pr_train_df01)
```



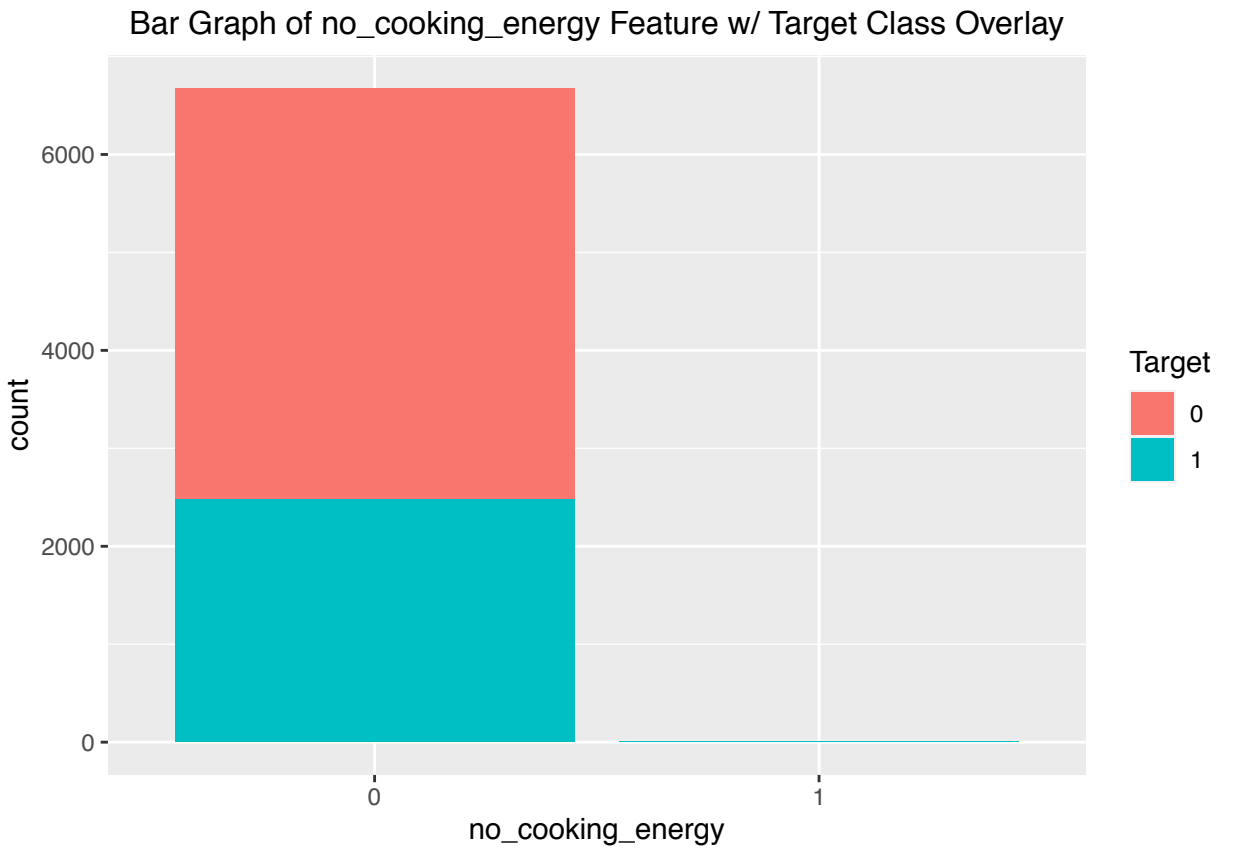

```
plot_bar(x = "septictank", y = "Target", df = pr_train_df01)
```


```
plot_bar(x = "latrine", y = "Target", df = pr_train_df01)
```


```
plot_bar(x = "toilet_other", y = "Target", df = pr_train_df01)
```




```
plot_bar(x = "no_cooking_energy", y = "Target", df = pr_train_df01)
```

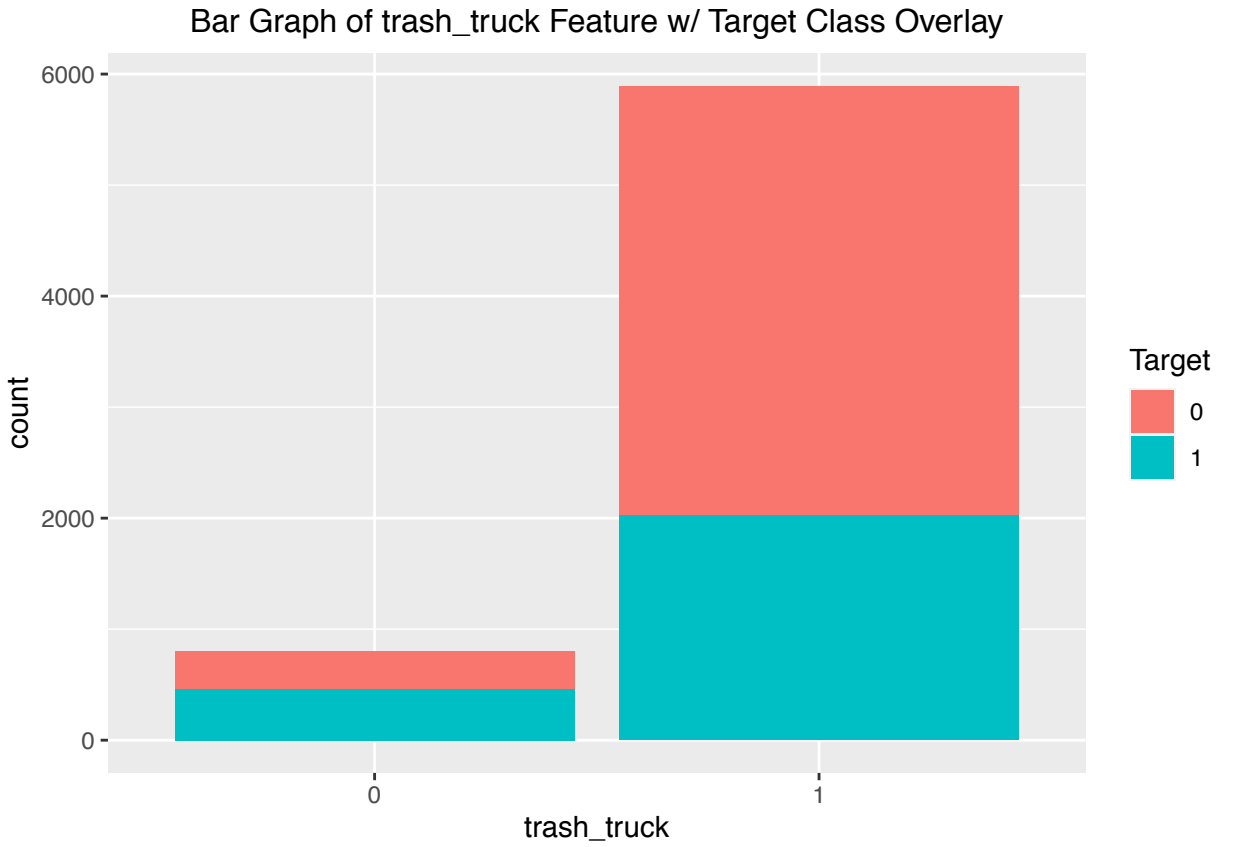



```
plot_bar(x = "cooking_.elec", y = "Target", df = pr_train_df01)
```


```
plot_bar(x = "cooking_gas", y = "Target", df = pr_train_df01)
```

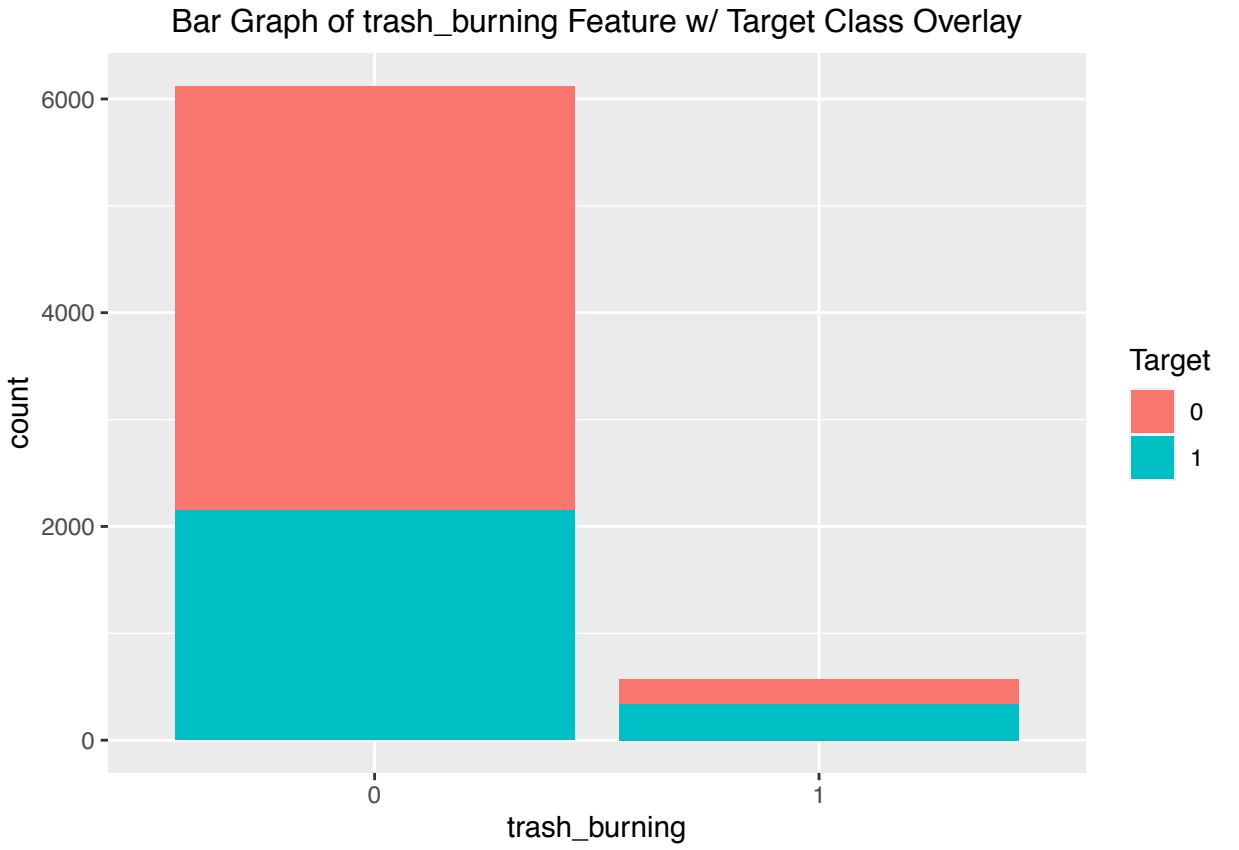

```
plot_bar(x = "cooking_woodcoal", y = "Target", df = pr_train_df01)
```


```
plot_bar(x = "trash_truck", y = "Target", df = pr_train_df01)
```

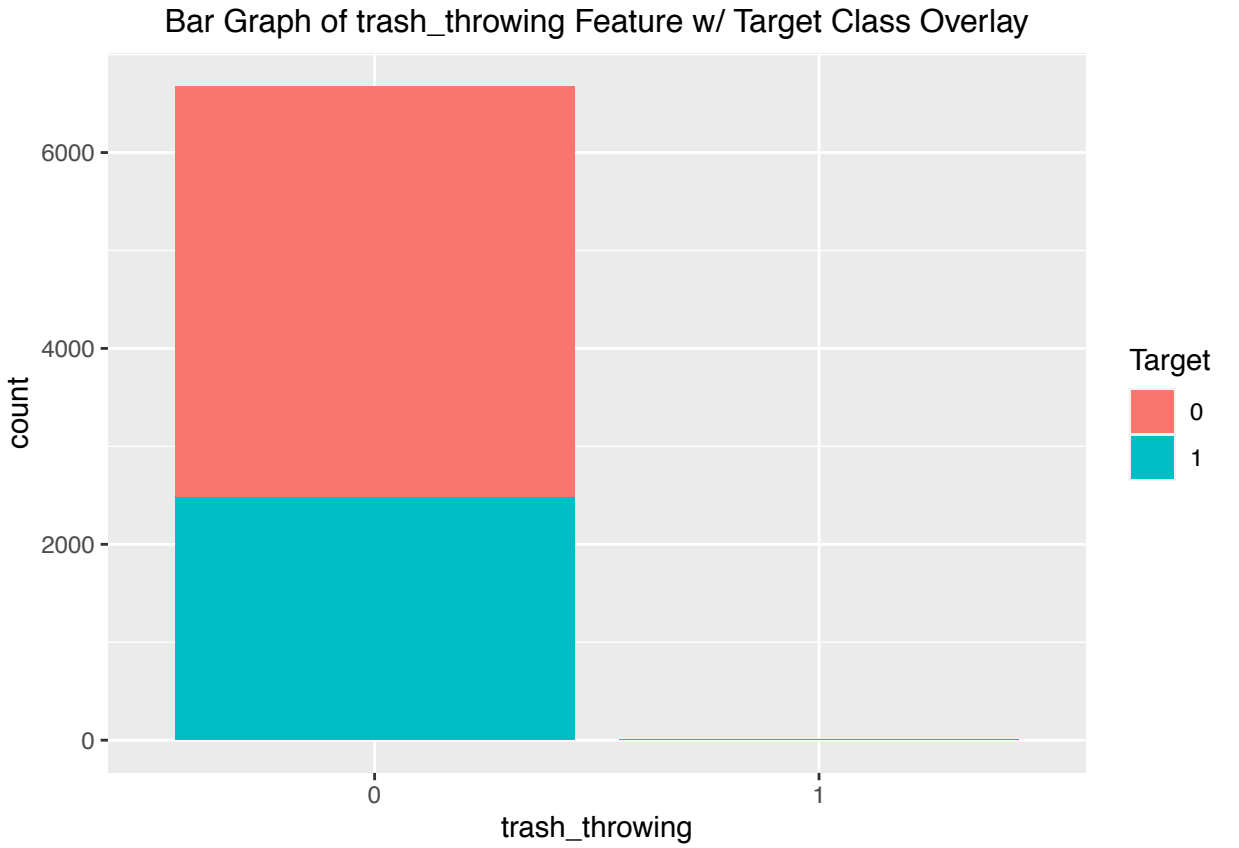



```
plot_bar(x = "trash_buried", y = "Target", df = pr_train_df01)
```


```
plot_bar(x = "trash_burning", y = "Target", df = pr_train_df01)
```




```
plot_bar(x = "trash_throwing", y = "Target", df = pr_train_df01)
```




```
plot_bar(x = "trash_river", y = "Target", df = pr_train_df01)
```


### Create contigency tables for categorical and dependent feature (`Target`)

```
# Contingency table for `marital` and `response`
r_cross_tab_df_01 <- table(pr_train_df01$Target, pr_train_df01$water_inside)
#r_cross_tab_df_01
r_cross_tab_df_02 <- addmargins(A = r_cross_tab_df_01, FUN = list(total = sum), quiet =
↪  TRUE)
r_cross_tab_df_02
```

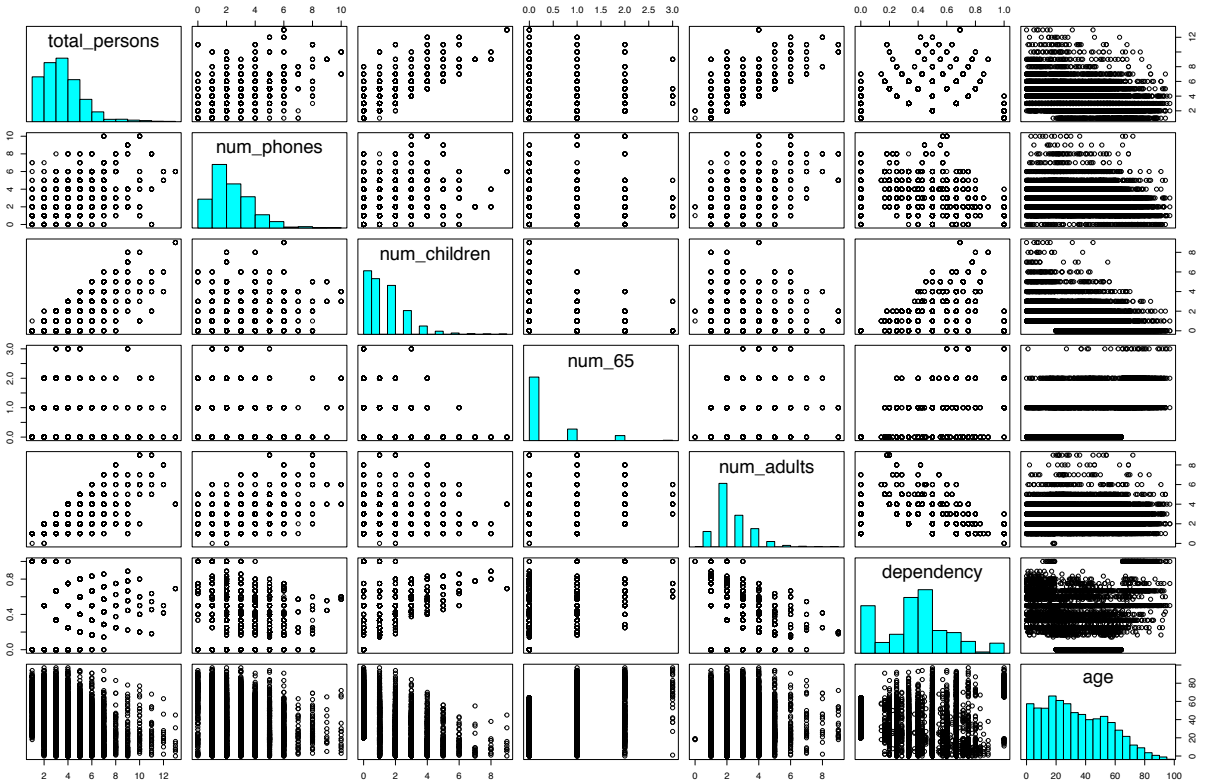
```
##
##             0    1 total
##   0       109 4095  4204
##   1       127 2359  2486
##   total   236 6454  6690
```

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

```
# Create sub df for numerical features
pr_x01_num_lst <- c("total_persons", "num_phones", "num_children", "num_65",
↪  "num_adults", "dependency", "age")
pr_train_x_num_df01_s1 <- subset(x = pr_train_df01, select = pr_x01_num_lst)
pr_train_y_df01 <- subset(x = pr_train_df01, select = c("Target"))
```

## Create scatter plot matrix for continuous features

```
#pairs(x = pr_train_df01[, c(5, 7, 14, 15, 16, 17, 39)], diag.panel = panel.hist)
pairs(x = pr_train_x_num_df01_s1, diag.panel = panel.hist)
```




## Histograms for continuous variables

See scatterplot matrix above

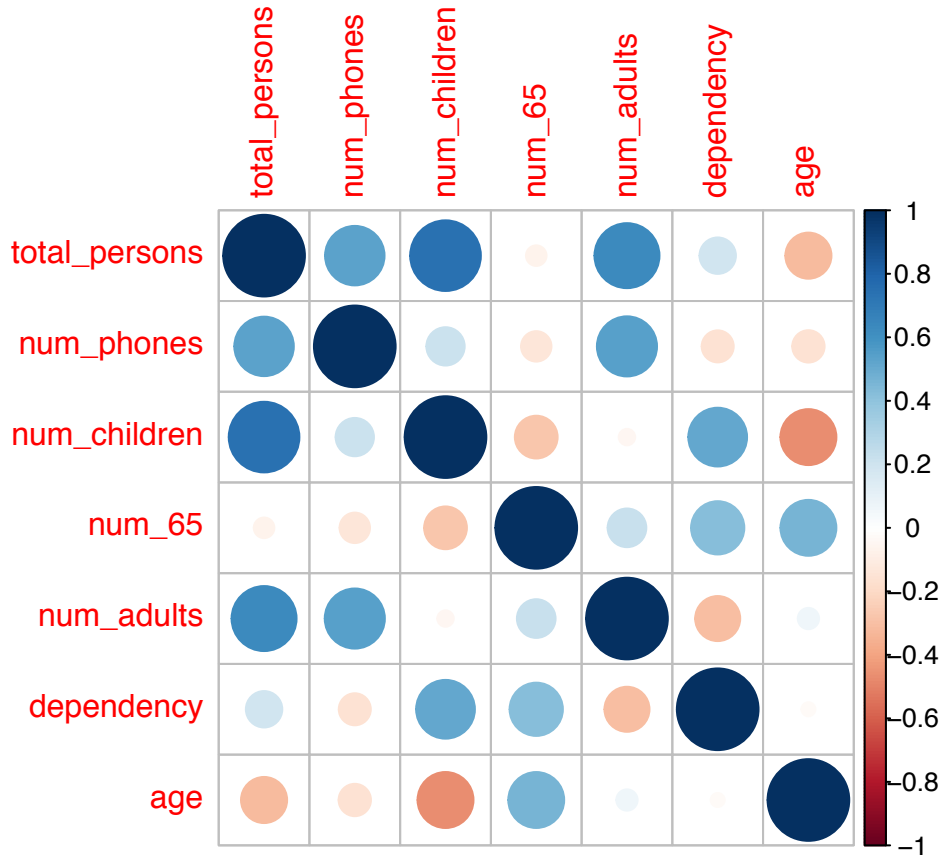
## Correlation matrix for numerical features

```
pr_train_cormx_df01 <- cor(pr_train_x_num_df01_s1)
print(pr_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(pr_train_cormx_df01)
```




## Principal Components Analysis (PCA)

```
#pr_pca_x_lst <- c("has_bathroom", "has_refrig", "has_comp")
#pr_train_pca_x_df01 <- subset(x = pr_train_df01, select = pr_pca_x_lst)
y = pr_train_df01$Target
pr_train_x_stand_pca_df01_s1 <- as.data.frame(scale(pr_train_x_num_df01_s1))
#print(pr_train_x_stand_pca_df01_s1)
#pr_train_lm_m1 <- lm(formula = y ~ total_persons + num_phones + num_children + num_65 +
↪  dependency + age, data = pr_train_x_stand_pca_df01_s1)
#print(pr_train_lm_m1)
#vif(pr_train_lm_m1)
pr_train_pca7 <- principal(r = pr_train_x_stand_pca_df01_s1, rotate = "varimax", nfactors
↪  = 7)
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

```
## Warning in principal(r = pr_train_x_stand_pca_df01_s1, rotate = "varimax", : The
## matrix is not positive semi-definite, scores found from Structure loadings
```

```
print(pr_train_pca7$loadings, cutoff = 0.49)
```

```
##
## Loadings:
##                RC1    RC2    RC5    RC4    RC3    RC6    RC7
## total_persons  0.743  0.591
## num_phones                   0.939
## num_children   0.907
## num_65                                  0.908
## num_adults             0.918
## dependency                                     0.855
## age                                  0.950
##
##                  RC1   RC2   RC5   RC4   RC3   RC6 RC7
## SS loadings    1.592 1.338 1.054 1.051 1.030 0.935   0
## Proportion Var 0.227 0.191 0.151 0.150 0.147 0.134   0
## Cumulative Var 0.227 0.419 0.569 0.719 0.866 1.000   1
```

```
pr_train_pca5 <- principal(r = pr_train_x_stand_pca_df01_s1, rotate = "varimax", nfactors
↪  = 5)
```

```
## Warning in cor.smooth(r): Matrix was not positive definite, smoothing was done
```

```
## Warning in principal(r = pr_train_x_stand_pca_df01_s1, rotate = "varimax", : The
## matrix is not positive semi-definite, scores found from Structure loadings
```

```
round(cor(pr_train_pca5$scores), 2)
```

```
##       RC1   RC2   RC3   RC4   RC5
## RC1  1.00  0.27  0.02 -0.68  0.48
## RC2  0.27  1.00 -0.09 -0.10  0.84
## RC3  0.02 -0.09  1.00  0.54 -0.28
## RC4 -0.68 -0.10  0.54  1.00 -0.38
## RC5  0.48  0.84 -0.28 -0.38  1.00
```

### Sub df's for modeling

pr_train_xy01_df01 pr_train_x01_df01 pr_test_xy01_df01 pr_test_x01_df01

### Train & evaluate multiple classification models

### Random Forest

```
# Train model
pr_train_rf_m1 <- randomForest(Target~., data = pr_train_xy01_df01)
```

```
# Use trained model to predict y on test data
pr_test_rf_m1_y_pred_df01 <- predict(object = pr_train_rf_m1, newdata = pr_test_x01_df01,
↪  type = "class")
#print(pr_test_rf_m1_y_pred_df01)

# Create contingency table to review results
pr_test_rf_m1_y_pred_tbl01 <- table(pr_test_df01$Target, pr_test_rf_m1_y_pred_df01)
row.names(pr_test_rf_m1_y_pred_tbl01) <- c("Actual: Vulnerable & Under (0/-)", "Actual:
↪  Mod to Extrm Poverty (1/+)")
colnames(pr_test_rf_m1_y_pred_tbl01) <- c("Predicted: Vulnerable & Under (0/-)",
↪  "Predicted: Mod to Extrm Poverty (1/+)")
```

```
pr_test_rf_m1_y_pred_tbl01 <- addmargins(A = pr_test_rf_m1_y_pred_tbl01, FUN = list(Total
↪  = sum), quiet = TRUE)
print(pr_test_rf_m1_y_pred_tbl01)
```

```
##                                    pr_test_rf_m1_y_pred_df01
##                                     Predicted: Vulnerable & Under (0/-)
##   Actual: Vulnerable & Under (0/-)                                 1617
##   Actual: Mod to Extrm Poverty (1/+)                                574
##   Total                                                            2191
##                                    pr_test_rf_m1_y_pred_df01
##                                     Predicted: Mod to Extrm Poverty (1/+)
##   Actual: Vulnerable & Under (0/-)                                   175
##   Actual: Mod to Extrm Poverty (1/+)                                 501
##   Total                                                              676
##                                    pr_test_rf_m1_y_pred_df01
##                                     Total
##   Actual: Vulnerable & Under (0/-)    1792
##   Actual: Mod to Extrm Poverty (1/+)  1075
##   Total                                2867
```

```
# Pull values from confusion matrix table
pr_test_rf_m1_y_pred_tp01 <- pr_test_rf_m1_y_pred_tbl01[2, 2]
pr_test_rf_m1_y_pred_fn01 <- pr_test_rf_m1_y_pred_tbl01[2, 1]
pr_test_rf_m1_y_pred_fp01 <- pr_test_rf_m1_y_pred_tbl01[1, 2]
pr_test_rf_m1_y_pred_tn01 <- pr_test_rf_m1_y_pred_tbl01[1, 1]
print(pr_test_rf_m1_y_pred_tp01)
```

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

```
print(pr_test_rf_m1_y_pred_fn01)
```

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

```
print(pr_test_rf_m1_y_pred_fp01)
```

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

```
print(pr_test_rf_m1_y_pred_tn01)
```

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

```
#pr_test_rf_m1_accur01 <- (pr_test_rf_m1_y_pred_tp01 + pr_test_rf_m1_y_pred_tn01) /
↪  (pr_test_rf_m1_y_pred_tp01 + pr_test_rf_m1_y_pred_fn01 + pr_test_rf_m1_y_pred_fp01 +
↪  pr_test_rf_m1_y_pred_tn01)
#print(paste0("Random Forest Model Accuracy on Test Data Set = ",
↪  round(pr_test_rf_m1_accur01 * 100, 1), "%"))
#pr_test_bsl_m1_accur01 <- round((1792 / 2867) * 100, 1)
#print(paste0("Baseline Model Accuracy on Test Data Set = ", pr_test_bsl_m1_accur01,
↪  "%"))

#================================================================================
# Use trained model to predict y on train data
pr_train_rf_m1_y_pred_df01 <- predict(object = pr_train_rf_m1, newdata =
↪  pr_train_x01_df01, type = "class")
#print(pr_train_rf_m1_y_pred_df01)
```

```
# Create contingency table to review results
pr_train_rf_m1_y_pred_tbl01 <- table(pr_train_df01$Target, pr_train_rf_m1_y_pred_df01)
row.names(pr_train_rf_m1_y_pred_tbl01) <- c("Actual: Vulnerable & Under (0/-)", "Actual:
↪  Mod to Extrm Poverty (1/+)")
colnames(pr_train_rf_m1_y_pred_tbl01) <- c("Predicted: Vulnerable & Under (0/-)",
↪  "Predicted: Mod to Extrm Poverty (1/+)")
pr_train_rf_m1_y_pred_tbl01 <- addmargins(A = pr_train_rf_m1_y_pred_tbl01, FUN =
↪  list(Total = sum), quiet = TRUE)
print(pr_train_rf_m1_y_pred_tbl01)
```

```
##                                   pr_train_rf_m1_y_pred_df01
##                                    Predicted: Vulnerable & Under (0/-)
##   Actual: Vulnerable & Under (0/-)                                3811
##   Actual: Mod to Extrm Poverty (1/+)                              1307
##   Total                                                           5118
##                                   pr_train_rf_m1_y_pred_df01
##                                    Predicted: Mod to Extrm Poverty (1/+)
##   Actual: Vulnerable & Under (0/-)                                 393
##   Actual: Mod to Extrm Poverty (1/+)                              1179
##   Total                                                           1572
##                                   pr_train_rf_m1_y_pred_df01
##                                    Total
##   Actual: Vulnerable & Under (0/-)    4204
##   Actual: Mod to Extrm Poverty (1/+)  2486
##   Total                               6690
```

```
# Pull values from confusion matrix table
pr_train_rf_m1_y_pred_tp01 <- pr_train_rf_m1_y_pred_tbl01[2, 2]
pr_train_rf_m1_y_pred_fn01 <- pr_train_rf_m1_y_pred_tbl01[2, 1]
pr_train_rf_m1_y_pred_fp01 <- pr_train_rf_m1_y_pred_tbl01[1, 2]
pr_train_rf_m1_y_pred_tn01 <- pr_train_rf_m1_y_pred_tbl01[1, 1]
print(pr_train_rf_m1_y_pred_tp01)
```

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

```
print(pr_train_rf_m1_y_pred_fn01)
```

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

```
print(pr_train_rf_m1_y_pred_fp01)
```

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

```
print(pr_train_rf_m1_y_pred_tn01)
```

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

```
#pr_train_rf_m1_accur01 <- (pr_train_rf_m1_y_pred_tp01 + pr_train_rf_m1_y_pred_tn01) /
↪  (pr_train_rf_m1_y_pred_tp01 + pr_train_rf_m1_y_pred_fn01 + pr_train_rf_m1_y_pred_fp01
↪  + pr_train_rf_m1_y_pred_tn01)
#print(paste0("Random Forest Model Accuracy on Training Data Set = ",
↪  round(pr_train_rf_m1_accur01 * 100, 1), "%"))
#pr_train_bsl_m1_accur01 <- round((4204 / 6690) * 100, 1)
#print(paste0("Baseline Model Accuracy on Training Data Set = ", pr_train_bsl_m1_accur01,
↪  "%"))
```

```r
# ============================================================================
pr_test_rf_m1_acc01 <- (pr_test_rf_m1_y_pred_tp01 + pr_test_rf_m1_y_pred_tn01) /
↪  (pr_test_rf_m1_y_pred_tp01 + pr_test_rf_m1_y_pred_fn01 + pr_test_rf_m1_y_pred_fp01 +
↪  pr_test_rf_m1_y_pred_tn01)
#print((pr_test_rf_m1_y_pred_tp01 + pr_test_rf_m1_y_pred_fn01 + pr_test_rf_m1_y_pred_fp01
↪  + pr_test_rf_m1_y_pred_tn01))
pr_test_rf_m1_err_rate01 <- 1 - pr_test_rf_m1_acc01
pr_test_rf_m1_recall01 <- pr_test_rf_m1_y_pred_tp01 / (pr_test_rf_m1_y_pred_tp01 +
↪  pr_test_rf_m1_y_pred_fn01)
pr_test_rf_m1_tnr01 <- pr_test_rf_m1_y_pred_tn01 / (pr_test_rf_m1_y_pred_tn01 +
↪  pr_test_rf_m1_y_pred_fp01)
pr_test_rf_m1_prec01 <- pr_test_rf_m1_y_pred_tp01 / (pr_test_rf_m1_y_pred_tp01 +
↪  pr_test_rf_m1_y_pred_fp01)
pr_test_rf_m1_f101 <- (1 + 1^2) * ((pr_test_rf_m1_prec01 * pr_test_rf_m1_recall01) /
↪  ((1^2 * pr_test_rf_m1_prec01) + pr_test_rf_m1_recall01))
pr_test_rf_m1_plr01 <- pr_test_rf_m1_recall01 / (1 - pr_test_rf_m1_tnr01)


pr_train_rf_m1_acc01 <- (pr_train_rf_m1_y_pred_tp01 + pr_train_rf_m1_y_pred_tn01) /
↪  (pr_train_rf_m1_y_pred_tp01 + pr_train_rf_m1_y_pred_fn01 + pr_train_rf_m1_y_pred_fp01
↪  + pr_train_rf_m1_y_pred_tn01)
#print((pr_train_rf_m1_y_pred_tp01 + pr_train_rf_m1_y_pred_fn01 +
↪  pr_train_rf_m1_y_pred_fp01 + pr_train_rf_m1_y_pred_tn01))
pr_train_rf_m1_err_rate01 <- 1 - pr_train_rf_m1_acc01
pr_train_rf_m1_recall01 <- pr_train_rf_m1_y_pred_tp01 / (pr_train_rf_m1_y_pred_tp01 +
↪  pr_train_rf_m1_y_pred_fn01)
pr_train_rf_m1_tnr01 <- pr_train_rf_m1_y_pred_tn01 / (pr_train_rf_m1_y_pred_tn01 +
↪  pr_train_rf_m1_y_pred_fp01)
pr_train_rf_m1_prec01 <- pr_train_rf_m1_y_pred_tp01 / (pr_train_rf_m1_y_pred_tp01 +
↪  pr_train_rf_m1_y_pred_fp01)
pr_train_rf_m1_f101 <- (1 + 1^2) * ((pr_train_rf_m1_prec01 * pr_train_rf_m1_recall01) /
↪  ((1^2 * pr_train_rf_m1_prec01) + pr_train_rf_m1_recall01))
pr_train_rf_m1_plr01 <- pr_train_rf_m1_recall01 / (1 - pr_train_rf_m1_tnr01)

measure_rf_name <- c("Model: Random Forest",
                     "Baseline Accuracy",
                     "Accuracy",
                     "Error Rate",
                     "Sensitivity/TPR/Recall",
                     "Specificity/TNR",
                     "Precision",
                     "F1",
                     "Positive Liklihood Ratio"
                     )

measure_rf_val01 <- c("Traing Data set",
                      paste0(round((4204 / 6690) * 100, 1), "%"),
                      paste0(round(pr_train_rf_m1_acc01 * 100, 1), "%"),
                      paste0(round(pr_train_rf_m1_err_rate01 * 100, 1), "%"),
                      paste0(round(pr_train_rf_m1_recall01 * 100, 1), "%"),
                      paste0(round(pr_train_rf_m1_tnr01 * 100, 1), "%"),
                      paste0(round(pr_train_rf_m1_prec01 * 100, 1), "%"),
                      paste0(round(pr_train_rf_m1_f101 * 100, 1), "%"),
```

```
                      paste0(round(pr_train_rf_m1_plr01 * 1, 2), "")
                      )

measure_rf_val02 <- c("Test Data set",
                      paste0(round((1792 / 2867) * 100, 1), "%"),
                      paste0(round(pr_test_rf_m1_acc01 * 100, 1), "%"),
                      paste0(round(pr_test_rf_m1_err_rate01 * 100, 1), "%"),
                      paste0(round(pr_test_rf_m1_recall01 * 100, 1), "%"),
                      paste0(round(pr_test_rf_m1_tnr01 * 100, 1), "%"),
                      paste0(round(pr_test_rf_m1_prec01 * 100, 1), "%"),
                      paste0(round(pr_test_rf_m1_f101 * 100, 1), "%"),
                      paste0(round(pr_test_rf_m1_plr01 * 1, 2), "")
                      )

measure_rf_tbl <- data.frame(measure_rf_name, measure_rf_val01, measure_rf_val02)
print(measure_rf_tbl)
```

```
##            measure_rf_name measure_rf_val01 measure_rf_val02
## 1     Model: Random Forest  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
```

```
confusionMatrix(table(pr_test_df01$Target, pr_test_rf_m1_y_pred_df01))
```

```
## Confusion Matrix and Statistics
##
##    pr_test_rf_m1_y_pred_df01
##        0    1
##   0 1617  175
##   1  574  501
##
##                Accuracy : 0.7388
##                  95% CI : (0.7223, 0.7548)
##     No Information Rate : 0.7642
##     P-Value [Acc > NIR] : 0.9993
##
##                   Kappa : 0.3979
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7380
##             Specificity : 0.7411
##          Pos Pred Value : 0.9023
##          Neg Pred Value : 0.4660
##              Prevalence : 0.7642
##          Detection Rate : 0.5640
##    Detection Prevalence : 0.6250
##       Balanced Accuracy : 0.7396
```

```
##
##        'Positive' Class : 0
##
```

```
print(sensitivity(pr_test_rf_m1_y_pred_df01, pr_test_df01$Target))
```

```
## [1] 0.9023438
```

```
print(paste0("Number of 0s in Test Set Original Target = ",
↪  length(which(pr_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(pr_test_rf_m1_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(pr_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(pr_test_rf_m1_y_pred_df01 == 1))))
```

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

```
pr_test_rf_m1_measure_df01 <- data.frame(pr_test_df01$Target, pr_test_rf_m1_y_pred_df01)
print(head(pr_test_rf_m1_measure_df01))
```

```
##   pr_test_df01.Target pr_test_rf_m1_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(pr_test_rf_m1_measure_df01$pr_test_df01.Target == 1 &
                          pr_test_rf_m1_measure_df01$pr_test_rf_m1_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(pr_test_rf_m1_measure_df01$pr_test_df01.Target == 1 &
                          pr_test_rf_m1_measure_df01$pr_test_rf_m1_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(pr_test_rf_m1_measure_df01$pr_test_df01.Target == 0 &
                          pr_test_rf_m1_measure_df01$pr_test_rf_m1_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(pr_test_rf_m1_measure_df01$pr_test_df01.Target == 0 &
```

```
                              pr_test_rf_m1_measure_df01$pr_test_rf_m1_y_pred_df01 == 0))))
```

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

### CART Decision Tree

```
# Train model
pr_train_cart_m1 <- rpart(Target~., data = pr_train_xy01_df01)
```

```
# Use trained model to predict y on test data
pr_test_cart_m1_y_pred_df01 <- predict(object = pr_train_cart_m1, newdata =
↪  pr_test_x01_df01, type = "class")
#print(pr_test_cart_m1_y_pred_df01)

# Create contingency table to review results
pr_test_cart_m1_y_pred_tbl01 <- table(pr_test_df01$Target, pr_test_cart_m1_y_pred_df01)
row.names(pr_test_cart_m1_y_pred_tbl01) <- c("Actual: Vulnerable & Under (0/-)", "Actual:
↪  Mod to Extrm Poverty (1/+)")
colnames(pr_test_cart_m1_y_pred_tbl01) <- c("Predicted: Vulnerable & Under (0/-)",
↪  "Predicted: Mod to Extrm Poverty (1/+)")
pr_test_cart_m1_y_pred_tbl01 <- addmargins(A = pr_test_cart_m1_y_pred_tbl01, FUN =
↪  list(Total = sum), quiet = TRUE)
print(pr_test_cart_m1_y_pred_tbl01)
```

```
##                                    pr_test_cart_m1_y_pred_df01
##                                     Predicted: Vulnerable & Under (0/-)
##   Actual: Vulnerable & Under (0/-)                                1529
##   Actual: Mod to Extrm Poverty (1/+)                               576
##   Total                                                           2105
##                                    pr_test_cart_m1_y_pred_df01
##                                     Predicted: Mod to Extrm Poverty (1/+)
##   Actual: Vulnerable & Under (0/-)                                  263
##   Actual: Mod to Extrm Poverty (1/+)                                499
##   Total                                                             762
##                                    pr_test_cart_m1_y_pred_df01
##                                     Total
##   Actual: Vulnerable & Under (0/-)    1792
##   Actual: Mod to Extrm Poverty (1/+)  1075
##   Total                               2867
```

```
# Pull values from confusion matrix table
pr_test_cart_m1_y_pred_tp01 <- pr_test_cart_m1_y_pred_tbl01[2, 2]
pr_test_cart_m1_y_pred_fn01 <- pr_test_cart_m1_y_pred_tbl01[2, 1]
pr_test_cart_m1_y_pred_fp01 <- pr_test_cart_m1_y_pred_tbl01[1, 2]
pr_test_cart_m1_y_pred_tn01 <- pr_test_cart_m1_y_pred_tbl01[1, 1]
print(pr_test_cart_m1_y_pred_tp01)
```

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

```
print(pr_test_cart_m1_y_pred_fn01)
```

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

```
print(pr_test_cart_m1_y_pred_fp01)
```

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

```
print(pr_test_cart_m1_y_pred_tn01)
```

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

```
#pr_test_cart_m1_accur01 <- (pr_test_cart_m1_y_pred_tp01 + pr_test_cart_m1_y_pred_tn01) /
↪  (pr_test_cart_m1_y_pred_tp01 + pr_test_cart_m1_y_pred_fn01 +
↪  pr_test_cart_m1_y_pred_fp01 + pr_test_cart_m1_y_pred_tn01)
#print(paste0("CART Model Accuracy on Test Data Set = ", round(pr_test_cart_m1_accur01 *
↪  100, 1), "%"))
#pr_test_bsl_m1_accur01 <- round((1792 / 2867) * 100, 1)
#print(paste0("Baseline Model Accuracy on Test Data Set = ", pr_test_bsl_m1_accur01,
↪  "%"))

#==============================================================================
# Use trained model to predict y on train data
pr_train_cart_m1_y_pred_df01 <- predict(object = pr_train_cart_m1, newdata =
↪  pr_train_x01_df01, type = "class")
#print(pr_train_cart_m1_y_pred_df01)

# Create contingency table to review results
pr_train_cart_m1_y_pred_tbl01 <- table(pr_train_df01$Target,
↪  pr_train_cart_m1_y_pred_df01)
row.names(pr_train_cart_m1_y_pred_tbl01) <- c("Actual: Vulnerable & Under (0/-)",
↪  "Actual: Mod to Extrm Poverty (1/+)")
colnames(pr_train_cart_m1_y_pred_tbl01) <- c("Predicted: Vulnerable & Under (0/-)",
↪  "Predicted: Mod to Extrm Poverty (1/+)")
pr_train_cart_m1_y_pred_tbl01 <- addmargins(A = pr_train_cart_m1_y_pred_tbl01, FUN =
↪  list(Total = sum), quiet = TRUE)
print(pr_train_cart_m1_y_pred_tbl01)
```

```
##                                    pr_train_cart_m1_y_pred_df01
##                                     Predicted: Vulnerable & Under (0/-)
##   Actual: Vulnerable & Under (0/-)                                3565
##   Actual: Mod to Extrm Poverty (1/+)                              1346
##   Total                                                           4911
##                                    pr_train_cart_m1_y_pred_df01
##                                     Predicted: Mod to Extrm Poverty (1/+)
##   Actual: Vulnerable & Under (0/-)                                 639
##   Actual: Mod to Extrm Poverty (1/+)                              1140
##   Total                                                           1779
##                                    pr_train_cart_m1_y_pred_df01
##                                     Total
##   Actual: Vulnerable & Under (0/-)    4204
##   Actual: Mod to Extrm Poverty (1/+)  2486
##   Total                               6690
```

```
# Pull values from confusion matrix table
pr_train_cart_m1_y_pred_tp01 <- pr_train_cart_m1_y_pred_tbl01[2, 2]
pr_train_cart_m1_y_pred_fn01 <- pr_train_cart_m1_y_pred_tbl01[2, 1]
pr_train_cart_m1_y_pred_fp01 <- pr_train_cart_m1_y_pred_tbl01[1, 2]
pr_train_cart_m1_y_pred_tn01 <- pr_train_cart_m1_y_pred_tbl01[1, 1]
```

```
print(pr_train_cart_m1_y_pred_tp01)
```

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

```
print(pr_train_cart_m1_y_pred_fn01)
```

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

```
print(pr_train_cart_m1_y_pred_fp01)
```

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

```
print(pr_train_cart_m1_y_pred_tn01)
```

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

```
#pr_train_cart_m1_accur01 <- (pr_train_cart_m1_y_pred_tp01 +
↪  pr_train_cart_m1_y_pred_tn01) / (pr_train_cart_m1_y_pred_tp01 +
↪  pr_train_cart_m1_y_pred_fn01 + pr_train_cart_m1_y_pred_fp01 +
↪  pr_train_cart_m1_y_pred_tn01)
#print(paste0("CART Model Accuracy on Training Data Set = ",
↪  round(pr_train_cart_m1_accur01 * 100, 1), "%"))
#pr_train_bsl_m1_accur01 <- round((4204 / 6690) * 100, 1)
#print(paste0("Baseline Model Accuracy on train Data Set = ", pr_train_bsl_m1_accur01,
↪  "%"))

# ============================================================================
pr_test_cart_m1_acc01 <- (pr_test_cart_m1_y_pred_tp01 + pr_test_cart_m1_y_pred_tn01) /
↪  (pr_test_cart_m1_y_pred_tp01 + pr_test_cart_m1_y_pred_fn01 +
↪  pr_test_cart_m1_y_pred_fp01 + pr_test_cart_m1_y_pred_tn01)
#print((pr_test_cart_m1_y_pred_tp01 + pr_test_cart_m1_y_pred_fn01 +
↪  pr_test_cart_m1_y_pred_fp01 + pr_test_cart_m1_y_pred_tn01))
pr_test_cart_m1_err_rate01 <- 1 - pr_test_cart_m1_acc01
pr_test_cart_m1_recall01 <- pr_test_cart_m1_y_pred_tp01 / (pr_test_cart_m1_y_pred_tp01 +
↪  pr_test_cart_m1_y_pred_fn01)
pr_test_cart_m1_tnr01 <- pr_test_cart_m1_y_pred_tn01 / (pr_test_cart_m1_y_pred_tn01 +
↪  pr_test_cart_m1_y_pred_fp01)
pr_test_cart_m1_prec01 <- pr_test_cart_m1_y_pred_tp01 / (pr_test_cart_m1_y_pred_tp01 +
↪  pr_test_cart_m1_y_pred_fp01)
pr_test_cart_m1_f101 <- (1 + 1^2) * ((pr_test_cart_m1_prec01 * pr_test_cart_m1_recall01)
↪  / ((1^2 * pr_test_cart_m1_prec01) + pr_test_cart_m1_recall01))
pr_test_cart_m1_plr01 <- pr_test_cart_m1_recall01 / (1 - pr_test_cart_m1_tnr01)


pr_train_cart_m1_acc01 <- (pr_train_cart_m1_y_pred_tp01 + pr_train_cart_m1_y_pred_tn01) /
↪  (pr_train_cart_m1_y_pred_tp01 + pr_train_cart_m1_y_pred_fn01 +
↪  pr_train_cart_m1_y_pred_fp01 + pr_train_cart_m1_y_pred_tn01)
#print((pr_train_cart_m1_y_pred_tp01 + pr_train_cart_m1_y_pred_fn01 +
↪  pr_train_cart_m1_y_pred_fp01 + pr_train_cart_m1_y_pred_tn01))
pr_train_cart_m1_err_rate01 <- 1 - pr_train_cart_m1_acc01
pr_train_cart_m1_recall01 <- pr_train_cart_m1_y_pred_tp01 / (pr_train_cart_m1_y_pred_tp01
↪  + pr_train_cart_m1_y_pred_fn01)
pr_train_cart_m1_tnr01 <- pr_train_cart_m1_y_pred_tn01 / (pr_train_cart_m1_y_pred_tn01 +
↪  pr_train_cart_m1_y_pred_fp01)
pr_train_cart_m1_prec01 <- pr_train_cart_m1_y_pred_tp01 / (pr_train_cart_m1_y_pred_tp01 +
↪  pr_train_cart_m1_y_pred_fp01)
```

```
pr_train_cart_m1_f101 <- (1 + 1^2) * ((pr_train_cart_m1_prec01 *
↪  pr_train_cart_m1_recall01) / ((1^2 * pr_train_cart_m1_prec01) +
↪  pr_train_cart_m1_recall01))
pr_train_cart_m1_plr01 <- pr_train_cart_m1_recall01 / (1 - pr_train_cart_m1_tnr01)

measure_cart_name <- c("Model: CART Decision Tree",
                       "Baseline Accuracy",
                       "Accuracy",
                       "Error Rate",
                       "Sensitivity/TPR/Recall",
                       "Specificity/TNR",
                       "Precision",
                       "F1",
                       "Positive Liklihood Ratio"
                       )

measure_cart_val01 <- c("Traing Data set",
                        paste0(round((4204 / 6690) * 100, 1), "%"),
                        paste0(round(pr_train_cart_m1_acc01 * 100, 1), "%"),
                        paste0(round(pr_train_cart_m1_err_rate01 * 100, 1), "%"),
                        paste0(round(pr_train_cart_m1_recall01 * 100, 1), "%"),
                        paste0(round(pr_train_cart_m1_tnr01 * 100, 1), "%"),
                        paste0(round(pr_train_cart_m1_prec01 * 100, 1), "%"),
                        paste0(round(pr_train_cart_m1_f101 * 100, 1), "%"),
                        paste0(round(pr_train_cart_m1_plr01 * 1, 2), "")
                        )

measure_cart_val02 <- c("Test Data set",
                        paste0(round((1792 / 2867) * 100, 1), "%"),
                        paste0(round(pr_test_cart_m1_acc01 * 100, 1), "%"),
                        paste0(round(pr_test_cart_m1_err_rate01 * 100, 1), "%"),
                        paste0(round(pr_test_cart_m1_recall01 * 100, 1), "%"),
                        paste0(round(pr_test_cart_m1_tnr01 * 100, 1), "%"),
                        paste0(round(pr_test_cart_m1_prec01 * 100, 1), "%"),
                        paste0(round(pr_test_cart_m1_f101 * 100, 1), "%"),
                        paste0(round(pr_test_cart_m1_plr01 * 1, 2), "")
                        )

measure_cart_tbl <- data.frame(measure_cart_name, measure_cart_val01, measure_cart_val02)
print(measure_cart_tbl)
```

```
##           measure_cart_name measure_cart_val01 measure_cart_val02
## 1 Model: CART Decision Tree    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
```

### C5.0 Decision Tree

```
# Train model
pr_train_c50_m1 <- C5.0(Target~., data = pr_train_xy01_df01, control =
↪  C5.0Control(minCases=75))
```

```
# Use trained model to predict y on test data
pr_test_c50_m1_y_pred_df01 <- predict(object = pr_train_c50_m1, newdata =
↪  pr_test_x01_df01, type = "class")
#print(pr_test_c50_m1_y_pred_df01)

# Create contingency table to review results
pr_test_c50_m1_y_pred_tbl01 <- table(pr_test_df01$Target, pr_test_c50_m1_y_pred_df01)
row.names(pr_test_c50_m1_y_pred_tbl01) <- c("Actual: Vulnerable & Under (0/-)", "Actual:
↪  Mod to Extrm Poverty (1/+)")
colnames(pr_test_c50_m1_y_pred_tbl01) <- c("Predicted: Vulnerable & Under (0/-)",
↪  "Predicted: Mod to Extrm Poverty (1/+)")
pr_test_c50_m1_y_pred_tbl01 <- addmargins(A = pr_test_c50_m1_y_pred_tbl01, FUN =
↪  list(Total = sum), quiet = TRUE)
print(pr_test_c50_m1_y_pred_tbl01)
```

```
##                                   pr_test_c50_m1_y_pred_df01
##                                    Predicted: Vulnerable & Under (0/-)
##   Actual: Vulnerable & Under (0/-)                                1537
##   Actual: Mod to Extrm Poverty (1/+)                               541
##   Total                                                           2078
##                                   pr_test_c50_m1_y_pred_df01
##                                    Predicted: Mod to Extrm Poverty (1/+)
##   Actual: Vulnerable & Under (0/-)                                  255
##   Actual: Mod to Extrm Poverty (1/+)                                534
##   Total                                                             789
##                                   pr_test_c50_m1_y_pred_df01
##                                    Total
##   Actual: Vulnerable & Under (0/-)    1792
##   Actual: Mod to Extrm Poverty (1/+)  1075
##   Total                               2867
```

```
# Pull values from confusion matrix table
pr_test_c50_m1_y_pred_tp01 <- pr_test_c50_m1_y_pred_tbl01[2, 2]
pr_test_c50_m1_y_pred_fn01 <- pr_test_c50_m1_y_pred_tbl01[2, 1]
pr_test_c50_m1_y_pred_fp01 <- pr_test_c50_m1_y_pred_tbl01[1, 2]
pr_test_c50_m1_y_pred_tn01 <- pr_test_c50_m1_y_pred_tbl01[1, 1]
print(pr_test_c50_m1_y_pred_tp01)
```

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

```
print(pr_test_c50_m1_y_pred_fn01)
```

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

```
print(pr_test_c50_m1_y_pred_fp01)
```

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

```
print(pr_test_c50_m1_y_pred_tn01)
```

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

```
#pr_test_c50_m1_accur01 <- (pr_test_c50_m1_y_pred_tp01 + pr_test_c50_m1_y_pred_tn01) /
↪  (pr_test_c50_m1_y_pred_tp01 + pr_test_c50_m1_y_pred_fn01 + pr_test_c50_m1_y_pred_fp01
↪  + pr_test_c50_m1_y_pred_tn01)
#print(paste0("C5.0 Model Accuracy on Test Data Set = ", round(pr_test_c50_m1_accur01 *
↪  100, 1), "%"))
#pr_test_bsl_m1_accur01 <- round((1792 / 2867) * 100, 1)
#print(paste0("Baseline Model Accuracy on Test Data Set = ", pr_test_bsl_m1_accur01,
↪  "%"))

#================================================================================
# Use trained model to predict y on train data
pr_train_c50_m1_y_pred_df01 <- predict(object = pr_train_c50_m1, newdata =
↪  pr_train_x01_df01, type = "class")
#print(pr_train_c50_m1_y_pred_df01)

# Create contingency table to review results
pr_train_c50_m1_y_pred_tbl01 <- table(pr_train_df01$Target, pr_train_c50_m1_y_pred_df01)
row.names(pr_train_c50_m1_y_pred_tbl01) <- c("Actual: Vulnerable & Under (0/-)", "Actual:
↪  Mod to Extrm Poverty (1/+)")
colnames(pr_train_c50_m1_y_pred_tbl01) <- c("Predicted: Vulnerable & Under (0/-)",
↪  "Predicted: Mod to Extrm Poverty (1/+)")
pr_train_c50_m1_y_pred_tbl01 <- addmargins(A = pr_train_c50_m1_y_pred_tbl01, FUN =
↪  list(Total = sum), quiet = TRUE)
print(pr_train_c50_m1_y_pred_tbl01)
```

```
##                                    pr_train_c50_m1_y_pred_df01
##                                     Predicted: Vulnerable & Under (0/-)
##   Actual: Vulnerable & Under (0/-)                                3569
##   Actual: Mod to Extrm Poverty (1/+)                              1320
##   Total                                                           4889
##                                    pr_train_c50_m1_y_pred_df01
##                                     Predicted: Mod to Extrm Poverty (1/+)
##   Actual: Vulnerable & Under (0/-)                                 635
##   Actual: Mod to Extrm Poverty (1/+)                              1166
##   Total                                                           1801
##                                    pr_train_c50_m1_y_pred_df01
##                                     Total
##   Actual: Vulnerable & Under (0/-)    4204
##   Actual: Mod to Extrm Poverty (1/+)  2486
##   Total                               6690
```

```
# Pull values from confusion matrix table
pr_train_c50_m1_y_pred_tp01 <- pr_train_c50_m1_y_pred_tbl01[2, 2]
pr_train_c50_m1_y_pred_fn01 <- pr_train_c50_m1_y_pred_tbl01[2, 1]
pr_train_c50_m1_y_pred_fp01 <- pr_train_c50_m1_y_pred_tbl01[1, 2]
pr_train_c50_m1_y_pred_tn01 <- pr_train_c50_m1_y_pred_tbl01[1, 1]
print(pr_train_c50_m1_y_pred_tp01)
```

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

```
print(pr_train_c50_m1_y_pred_fn01)
```

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

```
print(pr_train_c50_m1_y_pred_fp01)
```

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

```
print(pr_train_c50_m1_y_pred_tn01)
```

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

```
#pr_train_c50_m1_accur01 <- (pr_train_c50_m1_y_pred_tp01 + pr_train_c50_m1_y_pred_tn01) /
↪  (pr_train_c50_m1_y_pred_tp01 + pr_train_c50_m1_y_pred_fn01 +
↪  pr_train_c50_m1_y_pred_fp01 + pr_train_c50_m1_y_pred_tn01)
#print(paste0("C5.0 Model Accuracy on Training Data Set = ",
↪  round(pr_train_c50_m1_accur01 * 100, 1), "%"))
#pr_train_bsl_m1_accur01 <- round((4204 / 6690) * 100, 1)
#print(paste0("Baseline Model Accuracy on train Data Set = ", pr_train_bsl_m1_accur01,
↪  "%"))

# ============================================================================
pr_test_c50_m1_acc01 <- (pr_test_c50_m1_y_pred_tp01 + pr_test_c50_m1_y_pred_tn01) /
↪  (pr_test_c50_m1_y_pred_tp01 + pr_test_c50_m1_y_pred_fn01 + pr_test_c50_m1_y_pred_fp01
↪  + pr_test_c50_m1_y_pred_tn01)
#print((pr_test_c50_m1_y_pred_tp01 + pr_test_c50_m1_y_pred_fn01 +
↪  pr_test_c50_m1_y_pred_fp01 + pr_test_c50_m1_y_pred_tn01))
pr_test_c50_m1_err_rate01 <- 1 - pr_test_c50_m1_acc01
pr_test_c50_m1_recall01 <- pr_test_c50_m1_y_pred_tp01 / (pr_test_c50_m1_y_pred_tp01 +
↪  pr_test_c50_m1_y_pred_fn01)
pr_test_c50_m1_tnr01 <- pr_test_c50_m1_y_pred_tn01 / (pr_test_c50_m1_y_pred_tn01 +
↪  pr_test_c50_m1_y_pred_fp01)
pr_test_c50_m1_prec01 <- pr_test_c50_m1_y_pred_tp01 / (pr_test_c50_m1_y_pred_tp01 +
↪  pr_test_c50_m1_y_pred_fp01)
pr_test_c50_m1_f101 <- (1 + 1^2) * ((pr_test_c50_m1_prec01 * pr_test_c50_m1_recall01) /
↪  ((1^2 * pr_test_c50_m1_prec01) + pr_test_c50_m1_recall01))
pr_test_c50_m1_plr01 <- pr_test_c50_m1_recall01 / (1 - pr_test_c50_m1_tnr01)


pr_train_c50_m1_acc01 <- (pr_train_c50_m1_y_pred_tp01 + pr_train_c50_m1_y_pred_tn01) /
↪  (pr_train_c50_m1_y_pred_tp01 + pr_train_c50_m1_y_pred_fn01 +
↪  pr_train_c50_m1_y_pred_fp01 + pr_train_c50_m1_y_pred_tn01)
#print((pr_train_c50_m1_y_pred_tp01 + pr_train_c50_m1_y_pred_fn01 +
↪  pr_train_c50_m1_y_pred_fp01 + pr_train_c50_m1_y_pred_tn01))
pr_train_c50_m1_err_rate01 <- 1 - pr_train_c50_m1_acc01
pr_train_c50_m1_recall01 <- pr_train_c50_m1_y_pred_tp01 / (pr_train_c50_m1_y_pred_tp01 +
↪  pr_train_c50_m1_y_pred_fn01)
pr_train_c50_m1_tnr01 <- pr_train_c50_m1_y_pred_tn01 / (pr_train_c50_m1_y_pred_tn01 +
↪  pr_train_c50_m1_y_pred_fp01)
pr_train_c50_m1_prec01 <- pr_train_c50_m1_y_pred_tp01 / (pr_train_c50_m1_y_pred_tp01 +
↪  pr_train_c50_m1_y_pred_fp01)
pr_train_c50_m1_f101 <- (1 + 1^2) * ((pr_train_c50_m1_prec01 * pr_train_c50_m1_recall01)
↪  / ((1^2 * pr_train_c50_m1_prec01) + pr_train_c50_m1_recall01))
pr_train_c50_m1_plr01 <- pr_train_c50_m1_recall01 / (1 - pr_train_c50_m1_tnr01)

measure_c50_name <- c("Model: C5.0",
                      "Baseline Accuracy",
                      "Accuracy", "Error Rate",
```

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

measure_c50_val01 <- c("Traing Data set",
                       paste0(round((4204 / 6690) * 100, 1), "%"),
                       paste0(round(pr_train_c50_m1_acc01 * 100, 1), "%"),
                       paste0(round(pr_train_c50_m1_err_rate01 * 100, 1), "%"),
                       paste0(round(pr_train_c50_m1_recall01 * 100, 1), "%"),
                       paste0(round(pr_train_c50_m1_tnr01 * 100, 1), "%"),
                       paste0(round(pr_train_c50_m1_prec01 * 100, 1), "%"),
                       paste0(round(pr_train_c50_m1_f101 * 100, 1), "%"),
                       paste0(round(pr_train_c50_m1_plr01 * 1, 2), "")
                       )

measure_c50_val02 <- c("Test Data set",
                       paste0(round((1792 / 2867) * 100, 1), "%"),
                       paste0(round(pr_test_c50_m1_acc01 * 100, 1), "%"),
                       paste0(round(pr_test_c50_m1_err_rate01 * 100, 1), "%"),
                       paste0(round(pr_test_c50_m1_recall01 * 100, 1), "%"),
                       paste0(round(pr_test_c50_m1_tnr01 * 100, 1), "%"),
                       paste0(round(pr_test_c50_m1_prec01 * 100, 1), "%"),
                       paste0(round(pr_test_c50_m1_f101 * 100, 1), "%"),
                       paste0(round(pr_test_c50_m1_plr01 * 1, 2), "")
                       )

measure_c50_tbl <- data.frame(measure_c50_name, measure_c50_val01, measure_c50_val02)
print(measure_c50_tbl)
```

```
##           measure_c50_name measure_c50_val01 measure_c50_val02
## 1              Model: C5.0   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
```

## KNN

```
# Train model
#pr_train_knn_m1 <- C5.0(Target~., data = pr_train_xy01_df01, control =
↪  C5.0Control(minCases=75))
#cl = test.x
pr_train_knn_m1 <- knn(pr_train_xy01_df01, pr_test_xy01_df01, cl =
↪  pr_train_xy01_df01$Target, k = 81)
```

```
# Use trained model to predict y on test data
#pr_test_knn_m1_y_pred_df01 <- predict(object = pr_train_knn_m1, newdata =
↪  pr_test_x01_df01, type = "class")
#print(pr_test_knn_m1_y_pred_df01)

# Create contingency table to review results
pr_test_knn_m1_y_pred_tbl01 <- table(pr_test_df01$Target, pr_train_knn_m1)
row.names(pr_test_knn_m1_y_pred_tbl01) <- c("Actual: Vulnerable & Under (0/-)", "Actual:
↪  Mod to Extrm Poverty (1/+)")
colnames(pr_test_knn_m1_y_pred_tbl01) <- c("Predicted: Vulnerable & Under (0/-)",
↪  "Predicted: Mod to Extrm Poverty (1/+)")
pr_test_knn_m1_y_pred_tbl01 <- addmargins(A = pr_test_knn_m1_y_pred_tbl01, FUN =
↪  list(Total = sum), quiet = TRUE)
print(pr_test_knn_m1_y_pred_tbl01)
```

```
##                                    pr_train_knn_m1
##                                     Predicted: Vulnerable & Under (0/-)
##   Actual: Vulnerable & Under (0/-)                                  1783
##   Actual: Mod to Extrm Poverty (1/+)                                  70
##   Total                                                             1853
##                                    pr_train_knn_m1
##                                     Predicted: Mod to Extrm Poverty (1/+)
##   Actual: Vulnerable & Under (0/-)                                     9
##   Actual: Mod to Extrm Poverty (1/+)                                1005
##   Total                                                             1014
##                                    pr_train_knn_m1
##                                     Total
##   Actual: Vulnerable & Under (0/-)    1792
##   Actual: Mod to Extrm Poverty (1/+)  1075
##   Total                               2867
```

```
# Pull values from confusion matrix table
pr_test_knn_m1_y_pred_tp01 <- pr_test_knn_m1_y_pred_tbl01[2, 2]
pr_test_knn_m1_y_pred_fn01 <- pr_test_knn_m1_y_pred_tbl01[2, 1]
pr_test_knn_m1_y_pred_fp01 <- pr_test_knn_m1_y_pred_tbl01[1, 2]
pr_test_knn_m1_y_pred_tn01 <- pr_test_knn_m1_y_pred_tbl01[1, 1]
print(pr_test_knn_m1_y_pred_tp01)
```

```
## [1] 1005
```

```
print(pr_test_knn_m1_y_pred_fn01)
```

```
## [1] 70
```

```
print(pr_test_knn_m1_y_pred_fp01)
```

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

```
print(pr_test_knn_m1_y_pred_tn01)
```

```
## [1] 1783
```

```
#pr_test_knn_m1_accur01 <- (pr_test_knn_m1_y_pred_tp01 + pr_test_knn_m1_y_pred_tn01) /
↪  (pr_test_knn_m1_y_pred_tp01 + pr_test_knn_m1_y_pred_fn01 + pr_test_knn_m1_y_pred_fp01
↪  + pr_test_knn_m1_y_pred_tn01)
#print(paste0("C5.0 Model Accuracy on Test Data Set = ", round(pr_test_knn_m1_accur01 *
↪  100, 1), "%"))
```

```
#pr_test_bsl_m1_accur01 <- round((1792 / 2867) * 100, 1)
#print(paste0("Baseline Model Accuracy on Test Data Set = ", pr_test_bsl_m1_accur01,
↪  "%"))

#================================================================================
# Use trained model to predict y on train data
#pr_train_knn_m1_y_pred_df01 <- predict(object = pr_train_knn_m1, newdata =
↪  pr_train_x01_df01, type = "class")
#print(pr_train_knn_m1_y_pred_df01)

# Create contingency table to review results
#pr_train_knn_m1_y_pred_tbl01 <- table(pr_train_df01$Target, pr_train_knn_m1_y_pred_df01)
#row.names(pr_train_knn_m1_y_pred_tbl01) <- c("Actual: Vulnerable & Under (0/-)",
↪  "Actual: Mod to Extrm Poverty (1/+)")
#colnames(pr_train_knn_m1_y_pred_tbl01) <- c("Predicted: Vulnerable & Under (0/-)",
↪  "Predicted: Mod to Extrm Poverty (1/+)")
#pr_train_knn_m1_y_pred_tbl01 <- addmargins(A = pr_train_knn_m1_y_pred_tbl01, FUN =
↪  list(Total = sum), quiet = TRUE)
#print(pr_train_knn_m1_y_pred_tbl01)

# Pull values from confusion matrix table
#pr_train_knn_m1_y_pred_tp01 <- pr_train_knn_m1_y_pred_tbl01[2, 2]
#pr_train_knn_m1_y_pred_fn01 <- pr_train_knn_m1_y_pred_tbl01[2, 1]
#pr_train_knn_m1_y_pred_fp01 <- pr_train_knn_m1_y_pred_tbl01[1, 2]
#pr_train_knn_m1_y_pred_tn01 <- pr_train_knn_m1_y_pred_tbl01[1, 1]
#print(pr_train_knn_m1_y_pred_tp01)
#print(pr_train_knn_m1_y_pred_fn01)
#print(pr_train_knn_m1_y_pred_fp01)
#print(pr_train_knn_m1_y_pred_tn01)

#pr_train_knn_m1_accur01 <- (pr_train_knn_m1_y_pred_tp01 + pr_train_knn_m1_y_pred_tn01) /
↪  (pr_train_knn_m1_y_pred_tp01 + pr_train_knn_m1_y_pred_fn01 +
↪  pr_train_knn_m1_y_pred_fp01 + pr_train_knn_m1_y_pred_tn01)
#print(paste0("C5.0 Model Accuracy on Training Data Set = ",
↪  round(pr_train_knn_m1_accur01 * 100, 1), "%"))
#pr_train_bsl_m1_accur01 <- round((4204 / 6690) * 100, 1)
#print(paste0("Baseline Model Accuracy on train Data Set = ", pr_train_bsl_m1_accur01,
↪  "%"))

# ================================================================================
pr_test_knn_m1_acc01 <- (pr_test_knn_m1_y_pred_tp01 + pr_test_knn_m1_y_pred_tn01) /
↪  (pr_test_knn_m1_y_pred_tp01 + pr_test_knn_m1_y_pred_fn01 + pr_test_knn_m1_y_pred_fp01
↪  + pr_test_knn_m1_y_pred_tn01)
#print((pr_test_knn_m1_y_pred_tp01 + pr_test_knn_m1_y_pred_fn01 +
↪  pr_test_knn_m1_y_pred_fp01 + pr_test_knn_m1_y_pred_tn01))
pr_test_knn_m1_err_rate01 <- 1 - pr_test_knn_m1_acc01
pr_test_knn_m1_recall01 <- pr_test_knn_m1_y_pred_tp01 / (pr_test_knn_m1_y_pred_tp01 +
↪  pr_test_knn_m1_y_pred_fn01)
pr_test_knn_m1_tnr01 <- pr_test_knn_m1_y_pred_tn01 / (pr_test_knn_m1_y_pred_tn01 +
↪  pr_test_knn_m1_y_pred_fp01)
pr_test_knn_m1_prec01 <- pr_test_knn_m1_y_pred_tp01 / (pr_test_knn_m1_y_pred_tp01 +
↪  pr_test_knn_m1_y_pred_fp01)
pr_test_knn_m1_f101 <- (1 + 1^2) * ((pr_test_knn_m1_prec01 * pr_test_knn_m1_recall01) /
↪  ((1^2 * pr_test_knn_m1_prec01) + pr_test_knn_m1_recall01))
```

```
pr_test_knn_m1_plr01 <- pr_test_knn_m1_recall01 / (1 - pr_test_knn_m1_tnr01)


#pr_train_knn_m1_acc01 <- (pr_train_knn_m1_y_pred_tp01 + pr_train_knn_m1_y_pred_tn01) /
↪  (pr_train_knn_m1_y_pred_tp01 + pr_train_knn_m1_y_pred_fn01 +
↪  pr_train_knn_m1_y_pred_fp01 + pr_train_knn_m1_y_pred_tn01)
#print((pr_train_knn_m1_y_pred_tp01 + pr_train_knn_m1_y_pred_fn01 +
↪  pr_train_knn_m1_y_pred_fp01 + pr_train_knn_m1_y_pred_tn01))
#pr_train_knn_m1_err_rate01 <- 1 - pr_train_knn_m1_acc01
#pr_train_knn_m1_recall01 <- pr_train_knn_m1_y_pred_tp01 / (pr_train_knn_m1_y_pred_tp01 +
↪  pr_train_knn_m1_y_pred_fn01)
#pr_train_knn_m1_tnr01 <- pr_train_knn_m1_y_pred_tn01 / (pr_train_knn_m1_y_pred_tn01 +
↪  pr_train_knn_m1_y_pred_fp01)
#pr_train_knn_m1_prec01 <- pr_train_knn_m1_y_pred_tp01 / (pr_train_knn_m1_y_pred_tp01 +
↪  pr_train_knn_m1_y_pred_fp01)
#pr_train_knn_m1_f101 <- (1 + 1^2) * ((pr_train_knn_m1_prec01 * pr_train_knn_m1_recall01)
↪  / ((1^2 * pr_train_knn_m1_prec01) + pr_train_knn_m1_recall01))
#pr_train_knn_m1_plr01 <- pr_train_knn_m1_recall01 / (1 - pr_train_knn_m1_tnr01)

measure_knn_name <- c("Model: KNN",
                      "Accuracy",
                      "Error Rate",
                      "Sensitivity/TPR/Recall",
                      "Specificity/TNR",
                      "Precision",
                      "F1",
                      "Positive Liklihood Ratio"
                      )

#measure_knn_val01 <- c("Traing Data set", paste0(round((4204 / 6690) * 100, 1), "%"),
↪  paste0(round(pr_train_knn_m1_acc01 * 100, 1), "%"),
↪  paste0(round(pr_train_knn_m1_err_rate01 * 100, 1), "%"),
↪  paste0(round(pr_train_knn_m1_recall01 * 100, 1), "%"),
↪  paste0(round(pr_train_knn_m1_tnr01 * 100, 1), "%"),
↪  paste0(round(pr_train_knn_m1_prec01 * 100, 1), "%"),
↪  paste0(round(pr_train_knn_m1_f101 * 100, 1), "%"), paste0(round(pr_train_knn_m1_plr01
↪  * 1, 2), ""))

measure_knn_val02 <- c("Eval. Measure Values",
                       paste0(round(pr_test_knn_m1_acc01 * 100, 1), "%"),
                       paste0(round(pr_test_knn_m1_err_rate01 * 100, 1), "%"),
                       paste0(round(pr_test_knn_m1_recall01 * 100, 1), "%"),
                       paste0(round(pr_test_knn_m1_tnr01 * 100, 1), "%"),
                       paste0(round(pr_test_knn_m1_prec01 * 100, 1), "%"),
                       paste0(round(pr_test_knn_m1_f101 * 100, 1), "%"),
                       paste0(round(pr_test_knn_m1_plr01 * 1, 2), "")
                       )

measure_knn_tbl <- data.frame(measure_knn_name, measure_knn_val02)
print(measure_knn_tbl)
```

```
##           measure_knn_name    measure_knn_val02
## 1               Model: KNN Eval. Measure Values
```

```
## 2                 Accuracy                97.2%
## 3               Error Rate                 2.8%
## 4   Sensitivity/TPR/Recall                93.5%
## 5          Specificity/TNR                99.5%
## 6                Precision                99.1%
## 7                       F1                96.2%
## 8 Positive Liklihood Ratio               186.15
```

```
confusionMatrix(table(pr_test_df01$Target, pr_train_knn_m1))
```

```
## Confusion Matrix and Statistics
##
##    pr_train_knn_m1
##        0    1
##   0 1783    9
##   1   70 1005
##
##                Accuracy : 0.9724
##                  95% CI : (0.9658, 0.9781)
##     No Information Rate : 0.6463
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9405
##
##  Mcnemar's Test P-Value : 1.473e-11
##
##             Sensitivity : 0.9622
##             Specificity : 0.9911
##          Pos Pred Value : 0.9950
##          Neg Pred Value : 0.9349
##              Prevalence : 0.6463
##          Detection Rate : 0.6219
##    Detection Prevalence : 0.6250
##       Balanced Accuracy : 0.9767
##
##        'Positive' Class : 0
##
```

```
print(sensitivity(pr_train_knn_m1, pr_test_df01$Target))
```

```
## [1] 0.9949777
```