Aishwarya Saibewar
DATA 5322
April 27, 2023

Predicting Ownership of Dwellings Using Support Vector Machines

Abstract

In the current market conditions, both renting and owning real estate will have an impact in the long run. And understanding real estate needs and trends becomes critical for policymakers as well as consumers. Therefore, it is essential to investigate both short-term and long-term market trends in order to arrive at a well-informed decision. Understanding supply and demand plays a crucial role in determining housing needs, as the market is heading into a recession. Thus it becomes imperative to develop an algorithm that accurately predicts the ownership of a dwelling. This research uses data from Integrated Public Use Microdata Series (IPUMS USA) to perform analysis and examine the accuracy of Support Vector Machines in predicting whether a dwelling is occupied by an owner or renter. Three models were developed using linear, polynomial, and radial kernels of SVM to understand the influential factors and thereby make predictions. The first model using linear kernel was able to classify the ownership of a dwelling as owner or renter with a test error rate of 13.8%. The second model using a polynomial kernel was able to classify the dwelling's ownership as owner or renter with a test error rate of 12.3%. The third model using a radial kernel resulted in a test error rate of 13.4%.

Introduction

In today's world, where the economy is constantly wavering, it is important to make meaningful and informed decisions about housing needs. Policymakers and planners rely on statistical results to evaluate the housing needs of people at various income levels. These insights will assist researchers in determining trends in the housing market, such as the types of homes that are available, pricing by zip code, and the preferences of individuals for housing.

In this research, algorithms were trained to make predictions by using the data from the Integrated Public Use Microdata Series (IPUMS USA). The IPUMS USA database gives people access to information about the US population. It gives data from census and surveys and is publicly accessible. The American Community Surveys from 2000 to the present, the Puerto Rican Surveys from 2005 to the present, and the decennial censuses from 1790 to 2010 are all included in the IPUMS data. Users of IPUMS have access to microdata, which is useful for providing more in-depth information about specific families and individuals. In addition, as the data is integrated over time, it permits researchers to perform comparative research, investigate patterns and draw related insights.

For the purpose of this study, we are utilizing census data that contains individual and housing-related variables. Individual demographic information like age, income, education level, and marital status are included in the dataset. It additionally incorporates other housing variables

like year of construction, number of rooms, number of bedrooms, number of vehicles, cost of electricity, cost of water, and population density of the surrounding area. Using this data, we will predict whether a dwelling is occupied by an owner or renter using Support Vector Machines.

Theoretical Background

Support Vector Machine is an "out-of-the-box" and supervised machine learning technique that is primarily used for classification and regression problems. In SVM, the primary objective is to determine the optimal separation line, known as a hyperplane, that accurately separates the data into various classes. The margin of a hyperplane is defined as the distance between the plane and the closest points for each class. The best separation of classes is achieved by the maximal margin classifier, which selects a hyperplane that maximizes the distance between the plane and the points (margin). The data points closest to the hyperplane between at least two classes are called support vectors. These are the data points that are more likely to be misclassified. Therefore, the larger the margin or distance between the support vectors, the easier it is for the algorithm to accurately categorize. However, the maximal margin classifier can only be applied to linearly separable data therefore the application of this is severely limited. In addition, the maximal marginal classifier is very sensitive to support vectors i.e., the addition of a new observation can lead to a dramatic shift in the maximal margin hyperplane, and the margin will be very small around this hyperplane. Thus, such sensitivity can lead to overfitting. This concern can be prevented by Support Vector Classifier.

A support Vector Classifier is a soft margin classifier that allows some data points to be misclassified or be on the wrong side of the margin and some data points to be within the margin. The number of observations that are allowed to be misclassified is determined by the C. The observations within the margin and the observations lying on the margin are called support vectors. The support vector classifier is changed when any of the support vectors are changed. The larger value of C increases the size of the margin, and the smaller value of C decreases the margin size. Cost is the inverse of C. When the cost is small, the size of the margin increases, and it allows many observations to be misclassified. Likewise, when the cost is large, the size of the margin decreases, and only fewer observations are misclassified which may result in overfitting as the model becomes more sensitive to individual observations. Therefore, the optimal value of cost that improves the model accuracy is determined using cross-validation. However, the drawback of Support Vector Classifiers is that they can only be applied to linearly separable data. It cannot classify non-linearly separable data. Therefore, to overcome this limitation of the Support Vector Classifier the Kernel method of Support vector machines (SVM) is used.

Support Vector Machines can be used to create non-linear decision boundaries. The kernel is a functional relationship between two observations. The change in the function changes the shape of the decision boundary. It allows the transformation of data to high-dimensional space where the concept of support vector machines can be applied to separate classes. A linear Kernel uses the dot product between two observations to evaluate the similarity, resulting in an SVM with a straight-line boundary between classes. Despite its computational efficiency and lower

risk of overfitting, a linear kernel cannot be applied to non-linearly separable data. The linear Kernel has the form

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

A polynomial kernel can fit a higher-dimensional space with polynomials of degree d, resulting in a more flexible decision boundary. This Kernel uses the power function to create non-linear decision boundaries. However, they are computationally expensive with the increase in the degree of the polynomial Kernel. The polynomial Kernel has the form

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

A radial Kernel uses the Euclidean distance between two observations to create radial boundaries. The radial distance is multiplied by a positive constant gamma. Gamma is a tuning parameter that governs the distance of influence of a single observation. The larger value of gamma suggests that the observations need to be close to each other in order to be considered as a group. And the smaller value of gamma suggests that more observations can be grouped together as the similarity radius is high. Therefore, the optimal value of gamma that improves the model accuracy is determined using cross-validation. The radial Kernel has the form

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

Methodology

Data Preparation

23 potential variables from the IPUMS USA dataset were considered for predicting the ownership of dwellings. The oldest married member of the family was considered the owner of the household in order to subset the data because the size of the data was too large. To prepare the data for the model, the no-charge values for the cost of electricity and the cost of water were set to zero and the missing values were imputed using the median of the non-missing values. Since the hyperplane should be computed on continuous numerical data, reasonable values were chosen from a range of values for the year of construction and the categorical variables such as marital status were encoded. As we are solving a classification problem, the response variable was converted to a factor.

Models

First, the dataset was divided into training and testing sections, with 50% serving as a training dataset for the models and the remainder for testing. The training dataset with imputed missing values was used to train the models and the predictions are made on the test data.

The first SVM model used a linear Kernel to predict the ownership of a dwelling. The predictors used in the model are population density of the surrounding area, cost of electricity, number of rooms, household income, year of construction, number of bedrooms, owner's age, and number of vehicles. The model was tuned to find the best value of cost on the training data through cross-validation. This helped in determining the cost value that minimizes the error rate on the test data.

The second SVM model used a polynomial Kernel to predict the ownership of a dwelling. The predictors used in the model are the cost of electricity, cost of water, number of vehicles, number of rooms, household income, year of construction, number of bedrooms, and owner's age. The model was tuned to find the best parameter values of degree and cost on the training data. The optimal values are then used in predicting the test data as they minimize the test error rate.

The third SVM model used a radial Kernel to predict the same classification problem. The predictors used in the model are the cost of water, number of rooms, household income, number of couples, and population density of the surrounding area. Cross-validation of the training data at various cost and gamma values was done to find the optimal value of the hyperparameters that minimize the error rate on the test data.

Computational Results

Linear Kernel

First, a linear kernel was used in the SVM model to predict the ownership of a dwelling. The model was tuned for different values of cost on the training data and the optimal value of cost was selected through cross-validation. The best model has a cost of 0.75 and an error rate of 13.8% on the test data. This model was developed using 8 predictors and the strongest pair of predictors were selected to display the relationship between the variables. An SVM plot is shown in Figure 1. The household income and the number of rooms are plotted on the x and y axes, respectively. In the below plots, the yellow highlighted area reflects the region of feature space of the 'Owner' class and the area highlighted in red represents the 'Renter' class. In addition, the support vectors are represented by 'x' and the non-support vectors are represented by 'o'. Since the kernel is linear, the resulting plot also includes a linear decision boundary separating the two classes. Despite the decision boundary clearly separating the two classes, a few observations have been misclassified on either side of the decision boundary.

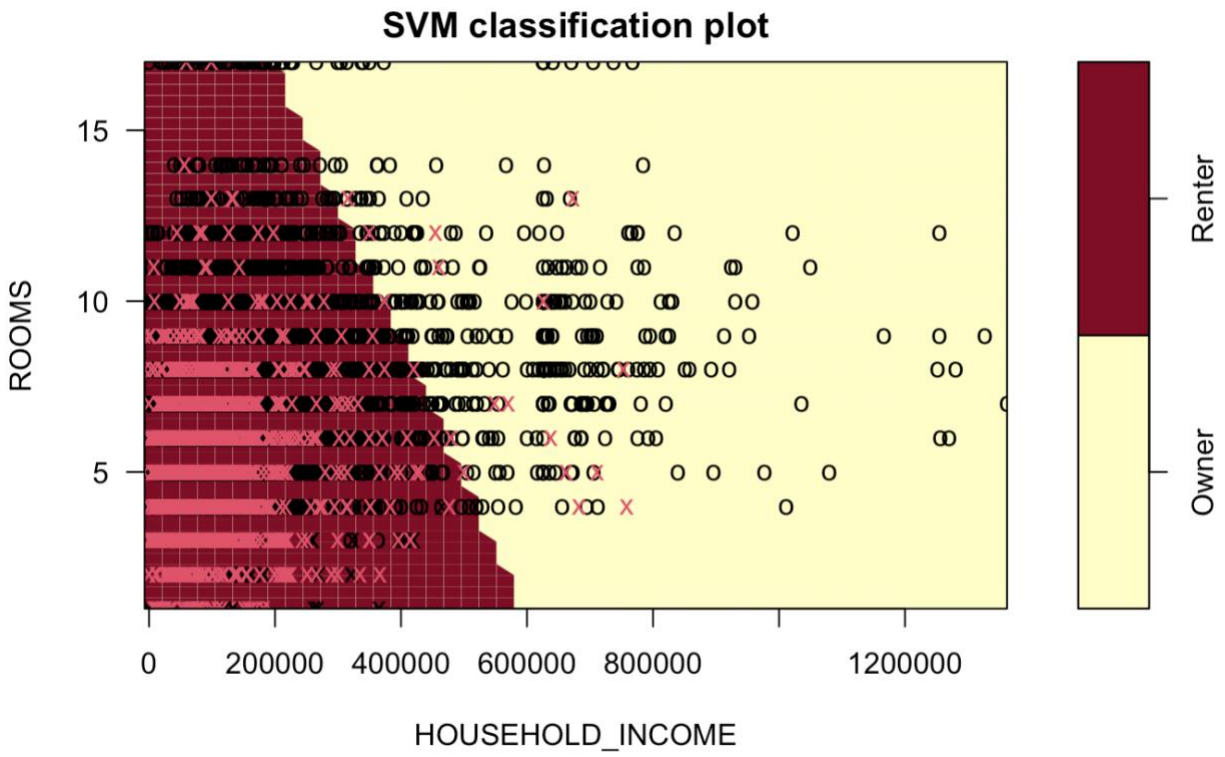


Figure 1. SVM plot of Household Income vs Number of Rooms using a linear kernel

Polynomial Kernel

Next, a polynomial kernel was used in the SVM model to predict the ownership of a dwelling. The model was tuned for different values of cost, degree, and coef0 on the training data. Through cross-validation, the optimal values of the three hyperparameters were selected. The best model had a cost of 0.05, degree 4, coef0 of 1, and an error rate of 12.3% on the test data. This model was also developed using 8 predictors and the strongest pair of predictor variables were selected to display the relationship between the variables. Figure 2 shows the SVM plot with variables number of vehicles and household income plotted on the x and y axes, respectively. The observations are separated by the curved decision boundary (since the kernel = polynomial). Additionally, it can be observed that a few observations have been incorrectly classified on either side of the decision boundary.

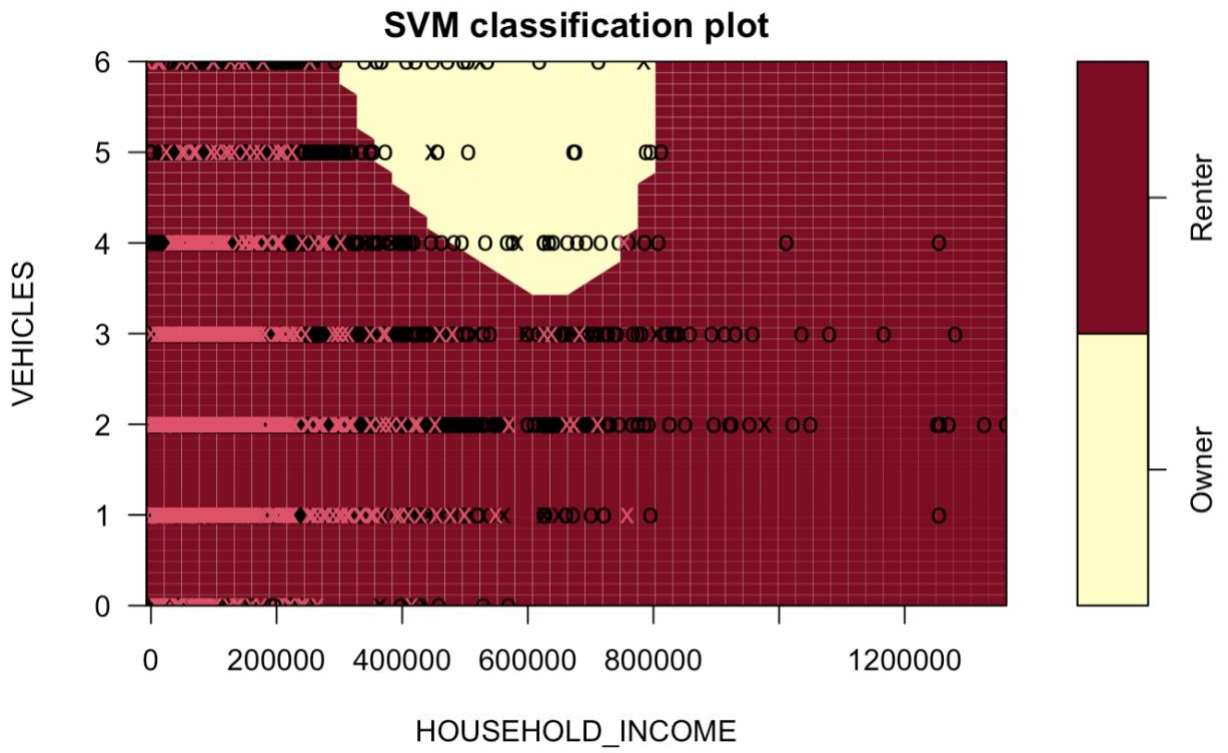


Figure 2. SVM plot of Household income vs Number of Vehicles using a polynomial kernel

As the SVM model with a polynomial kernel performs better than all the other models the confusion matrix of this model is shown in Figure 3. The majority of the observations in the dataset could be attributed to owners, and the model correctly predicts 6772 out of 6894 observations. This can be justifiable as just a subset of data was considered for analysis purposes i.e., only the people who are married. Married individuals are more likely to settle down in a location that they call home and ultimately invest in owning a house as opposed to renting it.

	Predicted	
Actual	Owner	Renter
Owner	6772	122
Renter	891	428

Figure 3. Confusion Matrix of SVM model using Polynomial Kernel

Radial Kernel

Lastly, a radial kernel was used in the SVM model to predict the ownership of a dwelling. On the training data, the model was tuned for various cost and gamma values. The best value for each of the three hyperparameters cost and gamma was determined through cross-validation. The best model had a cost of 1, a gamma of 0.5, and an error rate of 13.4% on the test data. This model was also developed using 5 predictors and the strongest pair of predictor variables were selected to display the relationship between the variables. Figure 4 shows the SVM plot with the population density and the number of rooms plotted on the x and y axes, respectively. The decision boundary is a radial curve (since the kernel is radial) separating the two classes. Likewise, it may be very well seen that a few observations have been misclassified on either side of the decision boundary.

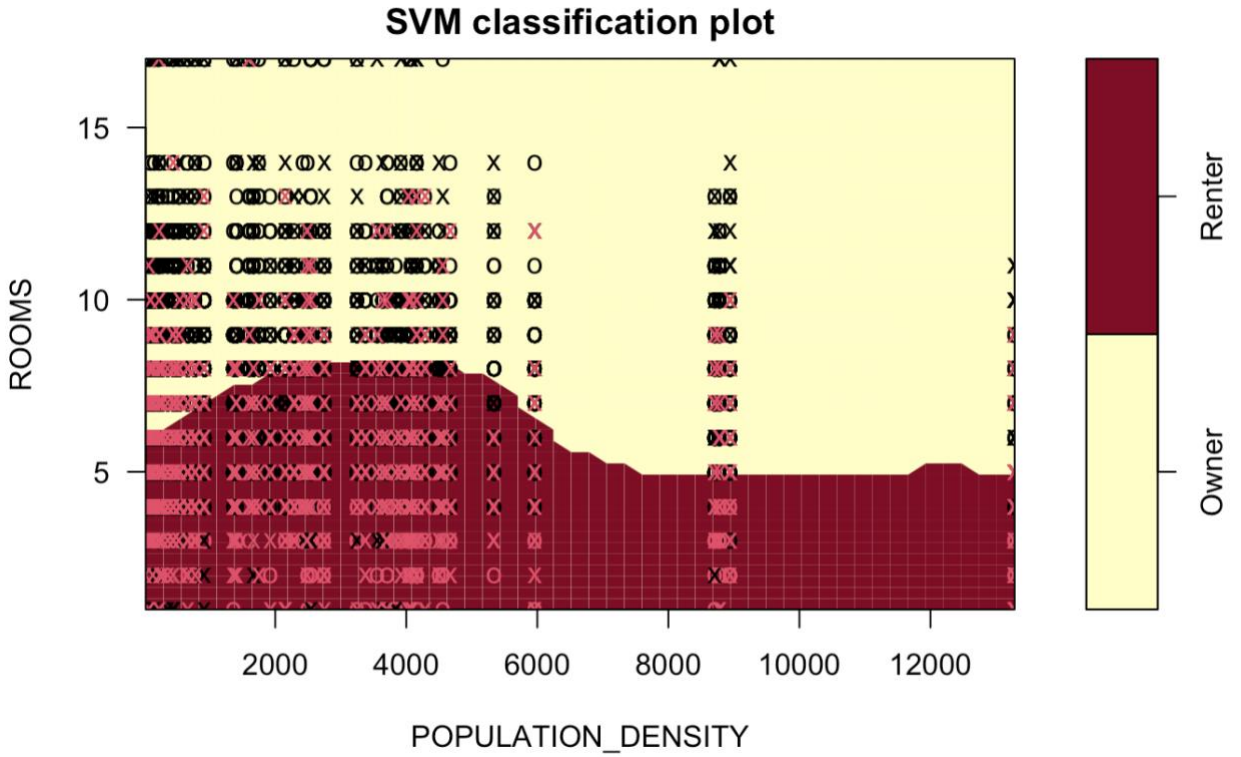


Figure 4. SVM plot of Population Density vs Number of Rooms using a radial kernel

Discussion

According to the findings of this research, Support Vector Machines can be used to predict the ownership of dwellings. IPUMS USA dataset was utilized for the study and the subset of the data was utilized by considering the oldest married member of the family as the owner of the household. As part of this research, three SVM models were developed, the first of which used a

linear Kernel for predicting the ownership of a dwelling. This model performed with a test error rate of 13.8% and a training time of 1.28 sec. The second model with a polynomial kernel was able to predict the ownership of a dwelling with a test error rate of 12.3% and a training time of 0.95 sec. The third model with the radial Kernel was able to classify the dwelling's ownership as owner or renter with a test error rate of 13.4% and a training time of 1.31 sec. Although the training time of these models at a specific hyperparameter value is low, the time taken to tune the models at different values of cost, gamma, and degree is high. The tuning time for linear, polynomial, and radial kernels is 65.59, 602.5, and 650.2 sec respectively. It can be observed that the tuning time is significantly more for the polynomial and radial kernels. Of all the developed models, the SVM model using a polynomial kernel performed the best at predicting the ownership of a dwelling. In addition, this suggests that the polynomial model's predictors, such as household income, electricity cost, water cost, number of vehicles, number of rooms, year of construction, number of bedrooms, and owner's age are more effective in addressing this classification problem.

As per the results of the analysis, the majority of observations in the dataset were classified as owners. This makes sense because most married couples would rather have a life that is more stable than if they were single. They also tend to own homes because their combined income is higher, which allows them to afford one. Likewise, wedded couples like to remain in a decent area that additionally upholds their kid's schooling and way of life.

Although the models were able to successfully use the decision boundary to separate the classes, a few observations were misclassified on either side of the boundary. This could be because the model was unable to effectively generalize the data, necessitating the use of additional predictors to classify the observations into classes.

Conclusion

According to the findings of this study, the Support Vector Machine with a polynomial Kernel performed the best in predicting the dwelling's ownership. The housing variables that are most influential in the prediction are household income, number of bedrooms, electricity cost, water cost, number of vehicles, number of rooms, year of construction, and owner's age.

In order to improve the model's accuracy, additional variables that have an effect on the nature of the dwelling should be incorporated into future research because the model was not able to accurately classify all of the observations. Additionally, the polynomial and radial kernels are not ideal when working with larger datasets because they require a lot of tuning time to run. As a result, additional machine algorithms like neural networks may be taken into consideration in future studies. The researchers and the policymakers will be able to make decisions with greater confidence as a result of this.

References

"U.S. Census data for social, economic, and Health Research." IPUMS USA. Available at: https://usa.ipums.org/usa/ (Accessed: April 30, 2023).

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani - An Introduction to Statistical Learning

Code Implementation Written Homework 2

Aishwarya Saibewar

4/20/2023

```
#Importing the required libraries
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.1.2
```

```
library(dplyr)
library(tictoc)
library(e1071)
```

Loading the youth_data.Rdata file

```
load("/Users/aishwaryasaibewar/Documents/SeattleUniversity-MSDS/Courses/SU Course Work/SPRING_2023/Stati
```

```
#number of variables in the data
length(data)
```

```
## [1] 23
```

```
#Column names of the data
colnames(data)
```

```
##  [1] "SERIAL"    "DENSITY"   "OWNERSHP"  "OWNERSHPD" "COSTELEC"  "COSTGAS"
##  [7] "COSTWATR"  "COSTFUEL"  "HHINCOME"  "ROOMS"     "BUILTYR2"  "BEDROOMS"
## [13] "VEHICLES"  "NFAMS"     "NCOUPLES"  "PERNUM"    "PERWT"     "AGE"
## [19] "MARST"     "BIRTHYR"   "EDUC"      "EDUCD"     "INCTOT"
```

DATA CLEANING

Understanding the data

```
#creating new dataframe with selected variables
selected_data <-data %>%
  dplyr::select(SERIAL,DENSITY,OWNERSHP,COSTELEC,COSTWATR,ROOMS,HHINCOME,BUILTYR2,BEDROOMS,VEHICLES,NFA
```

```
#As every family has multiple inhabitants and SERIAL is the same for everyone from the family. Consider

housing_data <- selected_data %>%
  group_by(SERIAL) %>% # Group the data based on SERIAL identification number of each family
  filter(AGE == max(AGE)) %>% #Filter the oldest members in the family and consider them as the owner o
  slice(1) %>% # If there are multiple old members with same age, choose the person in the first row
  ungroup()
```

```
#Cross-verify to check if there are any duplicate serial id's
# number of households
length(housing_data$SERIAL)
```

```
## [1] 30802
```

```
#number of Unique households
length(unique(housing_data$SERIAL))
```

```
## [1] 30802
```

#Find the count of missing values in each column

```
missingcolumns <- colSums(is.na(housing_data))
missingcolumns
```

```
##   SERIAL  DENSITY OWNERSHP COSTELEC COSTWATR    ROOMS HHINCOME BUILTYR2
##        0        0        0        0        0        0        0        0
## BEDROOMS VEHICLES    NFAMS NCOUPLES      AGE     EDUC    MARST
##        0        0        0        0        0        0        0
```

Below are the descriptions and needed manipulations for each variable

DENSITY

```
attr(housing_data$DENSITY, 'var_desc')
```

```
## [1] "DENSITY reports the average local population density among residents of each Public Use Microda
```

```r
# Unique values of DENSITY
unique(housing_data$DENSITY)
```

```
##  [1]   920.0  3640.9    22.5  3710.4   448.2   155.4  5962.1   230.4   362.4
## [10]    96.8  4031.9  4084.5   150.3   765.8  5339.8  3919.8   132.1  2156.8
## [19]  4291.1    66.2   473.8  3558.4  2754.9   665.1   237.8  4071.2 13284.6
## [28]  1758.5   222.9  8710.1   667.7  1927.8  8944.0   779.2  4501.4  1411.3
## [37]  2543.5  3376.5  3248.8  4560.6  1366.5  8784.4  2425.1   548.4  2746.3
## [46]  3843.3  2295.9   217.4  1667.9  4677.1  2493.3   529.2  1607.0  2288.1
## [55]  4161.6   280.3
```

OWNERSHP

```r
attr(housing_data$OWNERSHP, 'var_desc')
```

```
## [1] "OWNERSHP indicates whether the housing unit was rented or owned by its inhabitants. Housing uni
```

```r
# Unique values of OWNERSHP
unique(housing_data$OWNERSHP)
```

```
## <labelled<integer>[2]>: Ownership of dwelling (tenure) [general version]
## [1] 1 2
##
## Labels:
##  value                       label
##      0                         N/A
##      1 Owned or being bought (loan)
##      2                      Rented
```

```r
#Convert the values to meaningful categories
housing_data$OWNERSHP <- ifelse(housing_data$OWNERSHP == 1, 'Owner', 'Renter') # Owner if the dwelling
                                                                               # Renter if not
```

COSTELEC

```r
attr(housing_data$COSTELEC, 'var_desc')
```

```
## [1] "COSTELEC for 1970 reports each rented housing unit's annual electricity cost, excluding amounts
```

```r
# Unique values of COSTELEC
unique(housing_data$COSTELEC)
```

```
## <labelled<double>[59]>: Annual electricity cost
##  [1] 9990 1080  600 3600 1560 1800  840 9997 1440 3360 9993 1920 2040 2400  960
## [16] 1200 2160 4200  480 1320 3120 2280 3960  360 3840 3000  120 1680  240 5040
```

```
## [31]  720 2640 4560 3480 2520 2880 4680 4800 2760 5160 4320 5760 4920 3240 5640
## [46] 3720 4080 6000   48 5280 5400 4440 5520 6600 6360 6480 5880 6240 6120
##
## Labels:
##  value
##      0
##      2
##   9993
##   9994
##   9995
##   9996
##   9997
##   9998
##                                                                                label
##                                                                                  N/A
##                                                                       1 or $2 (2000)
##                           No charge or no electricity used (1990, 2000, 2003-onward ACS/PRCS)
##                                                          Electricity not used (1970, 1980)
##                                                 Electricity included in rent or no charge (1980)
##                                                          Electricity included in rent (1970)
##               Electricity included in rent or in condo fee (1990, 2000, 2003-onward ACS/PRCS)
##  No charge, no electricity used, or electricity included in rent or condo fee (2000-2002 ACS)
```

```
#Manipulating Annual Electricity Cost
#Since there is either no charge or the charge is included in the rent consider the value of COSTELEC f

housing_data$COSTELEC[housing_data$COSTELEC %in% c(9993, 9994,9995,9996,9997,9998)] <- 0
```

```
# verify the unique values
unique(housing_data$COSTELEC)
```

```
## <labelled<double>[58]>: Annual electricity cost
##  [1] 9990 1080  600 3600 1560 1800  840    0 1440 3360 1920 2040 2400  960 1200
## [16] 2160 4200  480 1320 3120 2280 3960  360 3840 3000  120 1680  240 5040  720
## [31] 2640 4560 3480 2520 2880 4680 4800 2760 5160 4320 5760 4920 3240 5640 3720
## [46] 4080 6000   48 5280 5400 4440 5520 6600 6360 6480 5880 6240 6120
##
## Labels:
##  value
##      0
##      2
##   9993
##   9994
##   9995
##   9996
##   9997
##   9998
##                                                                                label
##                                                                                  N/A
##                                                                       1 or $2 (2000)
##                           No charge or no electricity used (1990, 2000, 2003-onward ACS/PRCS)
##                                                          Electricity not used (1970, 1980)
##                                                 Electricity included in rent or no charge (1980)
```

```
##                                                    Electricity included in rent (1970)
##                Electricity included in rent or in condo fee (1990, 2000, 2003-onward ACS/PRCS)
##  No charge, no electricity used, or electricity included in rent or condo fee (2000-2002 ACS)
```

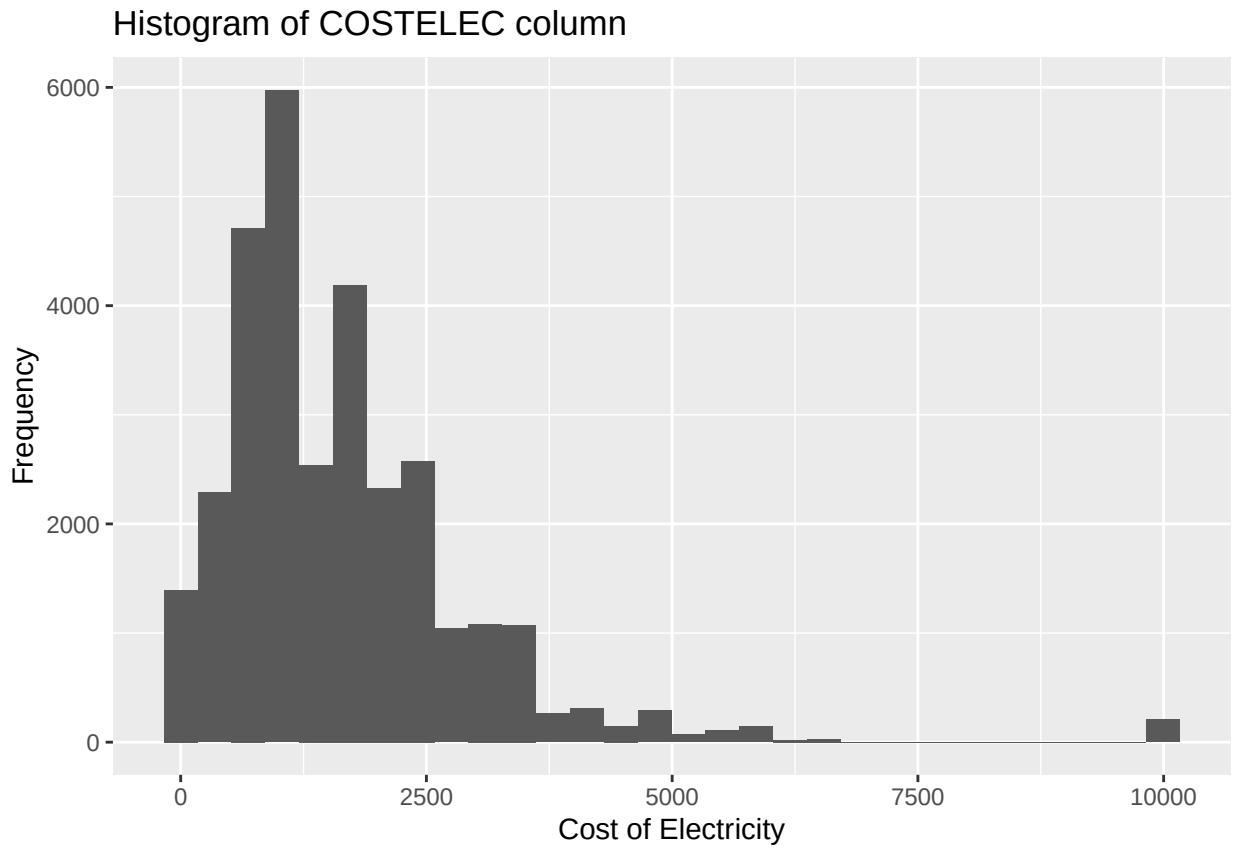
```
#Find the rows in the data where the COSTELEC is 0. This suggests that there are 1257 NA values of COST

sum(housing_data$COSTELEC == 0)
```

```
## [1] 1257
```

```
 # Histogram of COSTELEC
ggplot(housing_data, aes(x = COSTELEC)) + geom_histogram() +labs(title = "Histogram of COSTELEC column"
```

```
## Don't know how to automatically pick scale for object of type haven_labelled/vctrs_vctr/double. Defau
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#Find the median of the COSTELEC column
median_value <- median(housing_data$COSTELEC, na.rm = TRUE)
```

```
#Since median is less likely to be affected by outliers use median
# Imputing the missing values with the median
housing_data$COSTELEC[is.na(housing_data$COSTELEC)] <- median_value
```

#COSTWATR

```
attr(housing_data$COSTWATR, 'var_desc')
```

```
## [1] "COSTWATR for 1970 reports each rented housing unit's annual water cost, excluding amounts inclu
```

```
#Unique values of COSTWATR
unique(housing_data$COSTWATR)
```

```
## <labelled<double>[133]>: Annual water cost
##   [1]  360 1800 9993 9997 1500  120 1600   50 1900 1700 1400 5000  110  600 1100
##  [16]  780  190  100  210  720 1000   40  240 2000 2200  270 2300  180  540 2600
##  [31]  940   80  750 1200 3600  850 3900  150  960   90  200  900 2400  500   30
##  [46] 3000  870  400  660  380 3300  800  230 1300  480  830  570  700   70  260
##  [61]  770  320   60   20 2100  300  420  250  890 2700 2900  860  130  450  740
##  [76]  340  280  160  950  980  650  790  140  170  590  330  680  560  840  460
##  [91] 2500  970  440  620 3400  310  920  220  630  810 3700  350  610 3200  640
## [106]  550  290  490  580   10  910  880  390  990 3100  430 3500  710  760  930
## [121] 2800  470  690  530  510 3800  410  820  370  670    4  520  730
##
## Labels:
##  value
##      0
##      2
##   9993
##   9995
##   9997
##   9998
##                                                                          label
##                                                                            N/A
##                                                                 1 or $2 (2000)
##                       No charge or no used (1990, 2000, 2003-onward ACS/PRCS)
##                              Water included in rent or no charge (1970, 1980)
##    Water included in rent or in condo fee (1990, 2000, 2003-onward ACS/PRCS)
##  No charge, none used, or water included in rent or condo fee (2000-2002 ACS)
```

```
#Manipulating Annual Water Cost
#Since there is either no charge or the charge is included in the rent consider the value of COSTWATR f

housing_data$COSTWATR[housing_data$COSTWATR %in% c(9993,9995,9997,9998)] <- 0
```

```
# Verify the unique values
unique(housing_data$COSTWATR)
```

```
## <labelled<double>[132]>: Annual water cost
##   [1]  360 1800    0 1500  120 1600   50 1900 1700 1400 5000  110  600 1100  780
##  [16]  190  100  210  720 1000   40  240 2000 2200  270 2300  180  540 2600  940
##  [31]   80  750 1200 3600  850 3900  150  960   90  200  900 2400  500   30 3000
##  [46]  870  400  660  380 3300  800  230 1300  480  830  570  700   70  260  770
##  [61]  320   60   20 2100  300  420  250  890 2700 2900  860  130  450  740  340
##  [76]  280  160  950  980  650  790  140  170  590  330  680  560  840  460 2500
##  [91]  970  440  620 3400  310  920  220  630  810 3700  350  610 3200  640  550
```

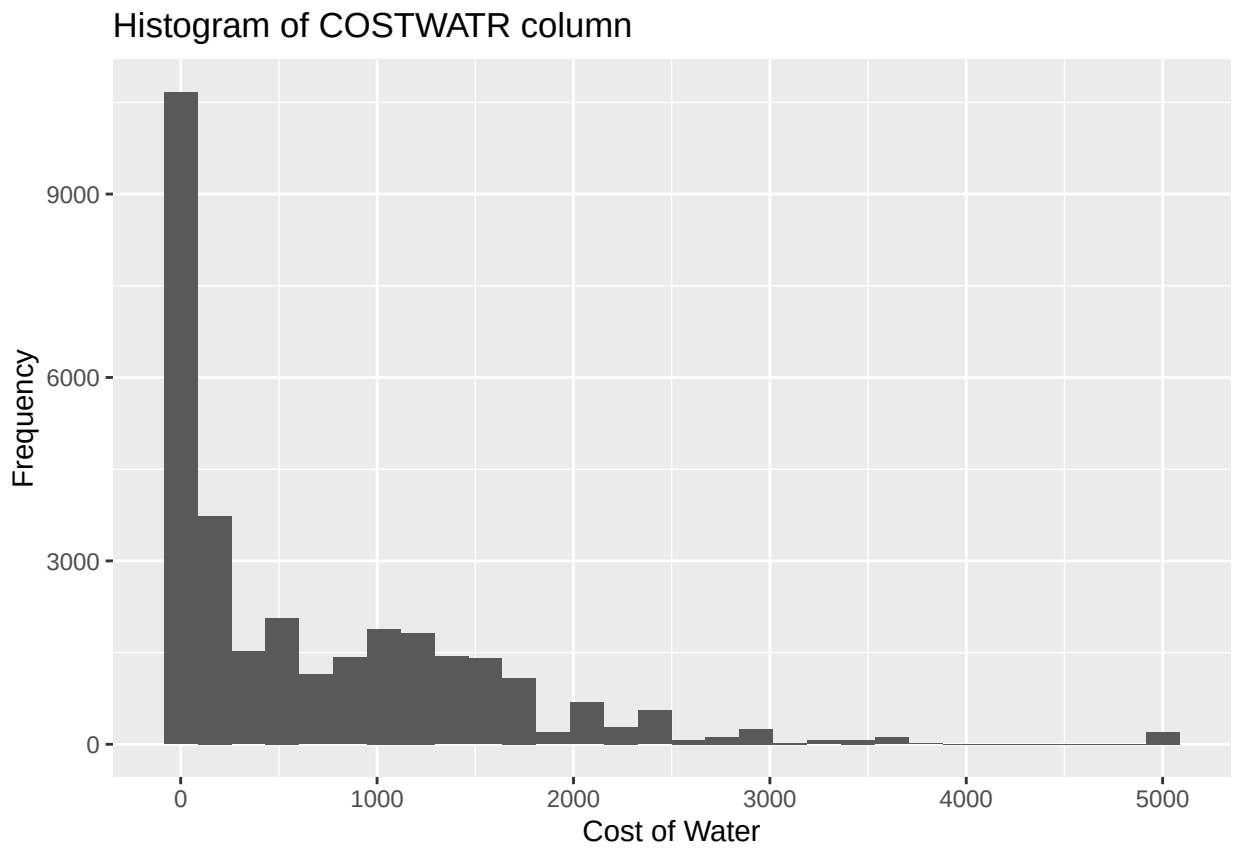
```
## [106]  290  490  580   10  910  880  390  990 3100  430 3500  710  760  930 2800
## [121]  470  690  530  510 3800  410  820  370  670    4  520  730
##
## Labels:
##  value
##      0
##      2
##   9993
##   9995
##   9997
##   9998
##                                                                        label
##                                                                         N/A
##                                                                1 or $2 (2000)
##                       No charge or no used (1990, 2000, 2003-onward ACS/PRCS)
##                              Water included in rent or no charge (1970, 1980)
##    Water included in rent or in condo fee (1990, 2000, 2003-onward ACS/PRCS)
##  No charge, none used, or water included in rent or condo fee (2000-2002 ACS)
```

```
#Find the rows in the data where the ROOMS is 0. This suggests that there are 8611 NA values of COSTWAT

sum(housing_data$COSTWATR == 0)
```

```
## [1] 8611
```

```
 #Histogram of COSTWATR
ggplot(housing_data, aes(x = COSTWATR)) +geom_histogram() +labs(title = "Histogram of COSTWATR column",
```

```
## Don't know how to automatically pick scale for object of type haven_labelled/vctrs_vctr/double. Defau

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Find the median of the COSTWATR column
median_value <- median(housing_data$COSTWATR, na.rm = TRUE)
```

```
#Since median is less likely to be affected by outliers use median
# Imputing the missing values with the median
housing_data$COSTWATR[is.na(housing_data$COSTWATR)] <- median_value
```

ROOMS

```
attr(housing_data$ROOMS, 'var_desc')
```

```
## [1] "ROOMS reports the number of whole rooms used for living purposes that are contained in the housi
```

```
#Unique values of ROOMS
unique(housing_data$ROOMS)
```

```
## <labelled<integer>[15]>: Number of rooms
##  [1]  7  6  5  4  1  3 10  8  9 12 17  2 11 13 14
##
## Labels:
##  value             label
```

```
##      0               N/A
##      1            1 room
##      9 9 (9+, 1960-2007)
```

```
#Find the rows in the data where the ROOMS is 0. This suggests that there are no NA values of ROOMS in

sum(housing_data$ROOMS == 0)
```

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

HHINCOME

```
attr(housing_data$HHINCOME, 'var_desc')
```

```
## [1] "HHINCOME reports the total money income of all household members age 15+ during the previous yea
```

```
#Unique values of HHINCOME
#unique(housing_data$HHINCOME)
```

```
#Find the rows in the data where the household income is 9999999.This suggests that there are no NA val

sum(housing_data$HHINCOME == 9999999)
```

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

BUILTYR2

```
attr(housing_data$BUILTYR2, 'var_desc')
```

```
## [1] "BUILTYR2 reports the decade in which the structure was built. \n\nThis variable is particularly
```

```
#Unique values of BUILTYR2
unique(housing_data$BUILTYR2)
```

```
## <labelled<integer>[11]>: Age of structure, decade
##  [1]  7  4  5  6  1 15  3  9  2 25 26
##
## Labels:
##  value
##      0
##      1
##      2
##      3
##      4
##      5
```

```
##      6
##      7
##      8
##      9
##     10
##     11
##     12
##     13
##     14
##     15
##     16
##     17
##     18
##     19
##     20
##     21
##     22
##     23
##     24
##     25
##     26
##                                                                                label
##                                                                                  N/A
##                                                                       1939 or earlier
##                                                                            1940-1949
##                                                                            1950-1959
##                                                                            1960-1969
##                                                                            1970-1979
##                                                                            1980-1989
##                                    1990-1994 (1990-1999 in the 2005-onward ACS and the PRCS)
##                                                 1995-1999 (1995-1998 in the 2000-2002 ACS)
##  2000-2004 (1999-2002 in the 2000-2002 ACS, 2000-2009 in the 2021-onward ACS and PRCS)
##          2005 (2005 or later in datasets containing 2005, 2006, or 2007 ACS/PRCS data)
##                                                                                    2006
##                                                                                    2007
##                                                                                    2008
##                                                                                    2009
##                                               2010 (2010-2019 in the 2021-onward ACS and PRCS)
##                                                                                    2011
##                                                                                    2012
##                                                                                    2013
##                                                                                    2014
##                                                                                    2015
##                                                                                    2016
##                                                                                    2017
##                                                                                    2018
##                                                                                    2019
##                                                                                    2020
##                                                                                    2021
```

```
#Find the rows in the data where the Built in year is 0. This suggests that there are no NA values of B

sum(housing_data$BUILTYR2 == 0)
```

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

```
#Convert it into a numerical data by choosing a particular value in the range of values.

housing_data$BUILTYR2<- ifelse(housing_data$BUILTYR2 == 1, 1939,
                        ifelse(housing_data$BUILTYR2 == 2, 1945,
                        ifelse(housing_data$BUILTYR2 == 3, 1955,
                        ifelse(housing_data$BUILTYR2 == 4, 1965,
                        ifelse(housing_data$BUILTYR2 == 5, 1975,
                        ifelse(housing_data$BUILTYR2 == 6, 1985,
                        ifelse(housing_data$BUILTYR2 == 7, 1992,
                        ifelse(housing_data$BUILTYR2 == 8, 1997,
                        ifelse(housing_data$BUILTYR2 == 9, 2003,housing_data$BUILTYR2)))))))))
```

BEDROOMS

```
attr(housing_data$BEDROOMS, 'var_desc')
```

```
## [1] "BEDROOMS reports the number of bedrooms within the housing unit.\n\nIn 1960, not all households
```

```
attr(housing_data$OWNERSHP, 'labels')
```

```
## NULL
```

```
#Unique values of BEDROOMS
unique(housing_data$BEDROOMS)
```

```
## <labelled<integer>[7]>: Number of bedrooms
## [1] 4 3 5 1 6 8 2
##
## Labels:
##  value                               label
##      0                                 N/A
##      1                         No bedrooms
##      2                                   1
##      3                                   2
##      4                                   3
##      5  4 (1970-2000, 2000-2007 ACS/PRCS)
##      6 5+ (1970-2000, 2000-2007 ACS/PRCS)
##      7                                   6
##      8                                   7
##      9                                   8
##     10                                   9
##     11                                  10
##     12                                  11
##     13                                  12
##     14                                  13
##     15                                  14
##     16                                  15
```

```
##     17                                  16
##     18                                  17
##     19                                  18
##     20                                  19
##     21                                  20
##     22                                  21
```

```
#According to labeling, number of bedrooms is equal to 'label number' minus 1

housing_data$BEDROOMS<-housing_data$BEDROOMS-1
head(housing_data)
```

```
## # A tibble: 6 x 15
##    SERIAL DENSITY OWNERSHP  COSTELEC   COSTWATR ROOMS HHINCOME BUILTYR2 BEDROOMS
##     <dbl>   <dbl> <chr>    <dbl+lbl>  <dbl+lbl> <int> <dbl+lb>    <dbl>    <dbl>
## 1 1371772   920   Owner         9990  360           7    75000     1992        3
## 2 1371773  3641.  Renter        1080 1800           6    13600     1965        3
## 3 1371774    22.5 Owner          600    0 [N/A~     5     7000     1975        3
## 4 1371775  3710.  Renter        3600    0 [N/A~     4    50500     1985        2
## 5 1371776   448.  Owner         1560    0 [N/A~     5   155300     1985        3
## 6 1371777   155.  Owner         1800 1500           7    75000     1992        4
## # ... with 6 more variables: VEHICLES <int+lbl>, NFAMS <int+lbl>,
## #   NCOUPLES <int+lbl>, AGE <int+lbl>, EDUC <int+lbl>, MARST <int+lbl>
```

#VEHICLES

```
attr(housing_data$VEHICLES, 'var_desc')
```

```
## [1] "VEHICLES reports the number of cars, vans, and trucks of one-ton capacity or less kept at home f
```

```
# Unique values of NFAMS
unique(housing_data$VEHICLES)
```

```
## <labelled<integer>[7]>: Vehicles available
## [1] 2 3 1 9 4 5 6
##
## Labels:
##  value                      label
##      0                        N/A
##      1                  1 available
##      6 6 (6+, 2000, ACS and PRCS)
##      7                         7+
##      9      No vehicles available
```

```
#Find the rows in the data where the VEHICLES is 0. This suggests that there are NA values of VEHICLES

sum(housing_data$VEHICLES == 0)
```

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

```r
# Set the "No vehicles available" to 0
housing_data$VEHICLES[housing_data$VEHICLES == 9] <- 0
```

NFAMS

```r
attr(housing_data$NFAMS, 'var_desc')
```

```
## [1] "NFAMS is a constructed variable that counts the number of families within each unit. A \"family
```

```r
# Unique values of NFAMS
unique(housing_data$NFAMS)
```

```
## <labelled<integer>[10]>: Number of families in household
##  [1]  1  2  6  3  4  5  8  7 13 10
##
## Labels:
##  value                    label
##      0 0 families (vacant unit)
##      1          1 family or N/A
##      2              2 families
```

NCOUPLES

```r
attr(housing_data$NCOUPLES, 'var_desc')
```

```
## [1] "NCOUPLES is a constructed variable (using SPLOC) that counts the number of married and cohabiti
```

```r
# Unique values of NCOUPLES
unique(housing_data$NCOUPLES)
```

```
## <labelled<integer>[4]>: Number of couples in household
## [1] 0 1 2 3
##
## Labels:
##  value            label
##      0 0 couples or N/A
```

AGE

```r
attr(housing_data$AGE, 'var_desc')
```

```
## [1] "AGE reports the person's age in years as of the last birthday.\n\nPlease see the Comparability
```

```
# Unique values of AGE
unique(housing_data$AGE)
```

```
## <labelled<integer>[73]>: Age
##  [1] 52 22 62 50 93 61 45 26 74 40 41 28 37 71 67 73 81 65 53 63 60 47 43 51 87
## [26] 80 35 76 79 23 29 56 59 42 78 27 75 36 66 68 55 70 49 88 54 46 83 31 24 84
## [51] 39 32 77 64 33 69 57 34 82 25 58 85 86 72 44 38 19 30 48 20 21 89 18
##
## Labels:
##  value                                label
##      0                Less than 1 year old
##     90            90 (90+ in 1980 and 1990)
##    100              100 (100+ in 1960-1970)
##    112 112 (112+ in the 1980 internal data)
##    115 115 (115+ in the 1990 internal data)
```

EDUC

```
attr(housing_data$EDUC, 'var_desc')
```

```
## [1] "EDUC indicates respondents' educational attainment, as measured by the highest year of school o
```

```
# Unique values of EDUC
unique(housing_data$EDUC)
```

```
## <labelled<integer>[11]>: Educational attainment [general version]
##  [1]  7 10  6  8  2 11  5  1  4  0  3
##
## Labels:
##  value                      label
##      0       N/A or no schooling
##      1 Nursery school to grade 4
##      2       Grade 5, 6, 7, or 8
##      3                  Grade 9
##      4                 Grade 10
##      5                 Grade 11
##      6                 Grade 12
##      7         1 year of college
##      8        2 years of college
##      9        3 years of college
##     10        4 years of college
##     11       5+ years of college
```

```
#Encoding the categorical variable (EDUC)
#housing_data <- housing_data %>%
#  mutate(HAS_COLLEGE_DEGREE = ifelse(EDUC >= 7, 1, 0),
#         HAS_SCHOOLING = ifelse(EDUC <=6, 1, 0))
```

MARST

```
attr(housing_data$MARST, 'var_desc')
```

```
## [1] "MARST gives each person's current marital status."
```

```
# Unique values of MARST
unique(housing_data$MARST)
```

```
## <labelled<integer>[6]>: Marital status
## [1] 6 4 3 1 5 2
##
## Labels:
##  value                    label
##      1 Married, spouse present
##      2  Married, spouse absent
##      3               Separated
##      4                Divorced
##      5                 Widowed
##      6    Never married/single
```

```
#Encoding the categorical variable (Marital Status)
housing_data <- housing_data %>%
  mutate(IS_MARRIED = ifelse(MARST == 1 | MARST == 2, 1, 0),
         IS_SEPARATED = ifelse(MARST == 3, 1, 0),
         IS_DIVORCED = ifelse(MARST == 4, 1, 0),
         IS_WIDOWED = ifelse(MARST == 5, 1, 0),
         IS_SINGLE = ifelse(MARST == 6, 1, 0))
```

```
#str(housing_data)
```

#Filter the owners who are single

```
#x <- housing_data %>%
#  filter(IS_SINGLE == 1)
#x
```

#Filter the household data with who are married

```
housing_data <- housing_data %>%
  filter(IS_MARRIED == 1)
```

```
#Converting the response variable to factors
housing_data$OWNERSHP<- as.factor(housing_data$OWNERSHP)
```

#Rename the column names of housing_data

```
colnames(housing_data) <- c(("SERIAL"),("POPULATION_DENSITY"),("HOMEOWNERSHIP"),("ELECTRICITY_COST"),("
```

PROBLEM-1

Linear SVM Model

#Fit the model one the data where the owner is the oldest married member of the family

```
#creating new dataframe with selected variables

#linear model---- HOMEOWNERSHIP,POPULATION_DENSITY,ELECTRICITY_COST,ROOMS,HOUSEHOLD_INCOME,BUILT_IN_YEA
#polynomial model---- HOMEOWNERSHIP,ELECTRICITY_COST,WATER_COST,VEHICLES,ROOMS,HOUSEHOLD_INCOME,BUILT_I
#radial model---- HOMEOWNERSHIP,WATER_COST,ROOMS,HOUSEHOLD_INCOME,VEHICLES,NCOUPLES,POPULATION_DENSITY

linear_data <-housing_data %>%
  dplyr::select(HOMEOWNERSHIP,POPULATION_DENSITY,ELECTRICITY_COST,ROOMS,HOUSEHOLD_INCOME,BUILT_IN_YEAR,
```

```
#Split the data into train and test by considering 50% of data as training data and reserving the remai

set.seed(1)
train <- sample(nrow(linear_data) * 0.5)
linear.train <- linear_data[train, ]
linear.test <- linear_data[-train, ]
```

```
#Tune the model at different values of cost
set.seed(1)
tic()
tune.out.linear <- tune(svm, HOMEOWNERSHIP ~ . , data = linear.train, kernel = "linear",
    ranges = list(cost = c(0.01, 0.1,0.5,0.75, 1, 10)))
toc()
```

```
## 68.805 sec elapsed
```

```
summary(tune.out.linear)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.75
##
## - best performance: 0.1424734
##
## - Detailed performance results:
##    cost     error dispersion
## 1  0.01 0.1570838 0.01564138
```

```
## 2  0.10 0.1435685 0.01128769
## 3  0.50 0.1427170 0.01002472
## 4  0.75 0.1424734 0.01050867
## 5  1.00 0.1424734 0.01050867
## 6 10.00 0.1424734 0.01050867
```

```
bestmodel_linear <- tune.out.linear$best.model
summary(bestmodel_linear)
```

```
##
## Call:
## best.tune(method = svm, train.x = HOMEOWNERSHIP ~ ., data = linear.train,
##     ranges = list(cost = c(0.01, 0.1, 0.5, 0.75, 1, 10)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.75
##
## Number of Support Vectors:  2688
##
##  ( 1347 1341 )
##
##
## Number of Classes:  2
##
## Levels:
##  Owner Renter
```

```
#Training error
ypred <- predict(bestmodel_linear, linear.train)
table(predict = ypred, truth = linear.train$HOMEOWNERSHIP)
```

```
##         truth
## predict  Owner Renter
##   Owner   6814   1116
##   Renter    45    237
```

```
training.err<- mean(ypred != linear.train$HOMEOWNERSHIP)
cat("Training  error rate for SVM model using linear kernel is  ", training.err)
```

```
## Training  error rate for SVM model using linear kernel is   0.1413785
```

```
#Test error
ypred <- predict(bestmodel_linear, linear.test)
table(predict = ypred, truth = linear.test$HOMEOWNERSHIP)
```

```
##         truth
## predict  Owner Renter
##   Owner   6846   1091
##   Renter    48    228
```

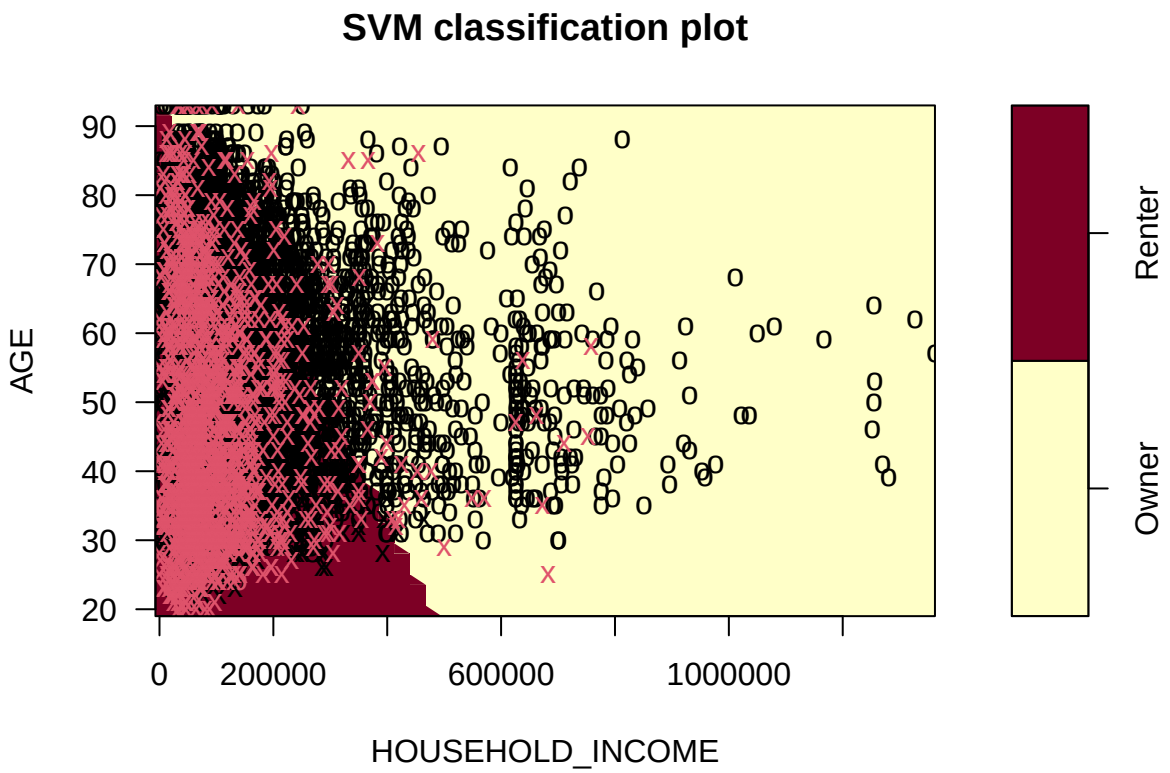
```
test.err <- mean(ypred != linear.test$HOMEOWNERSHIP)
cat("Test error rate for SVM model using linear kernel is  ", test.err)
```

```
## Test error rate for SVM model using linear kernel is   0.1386826
```

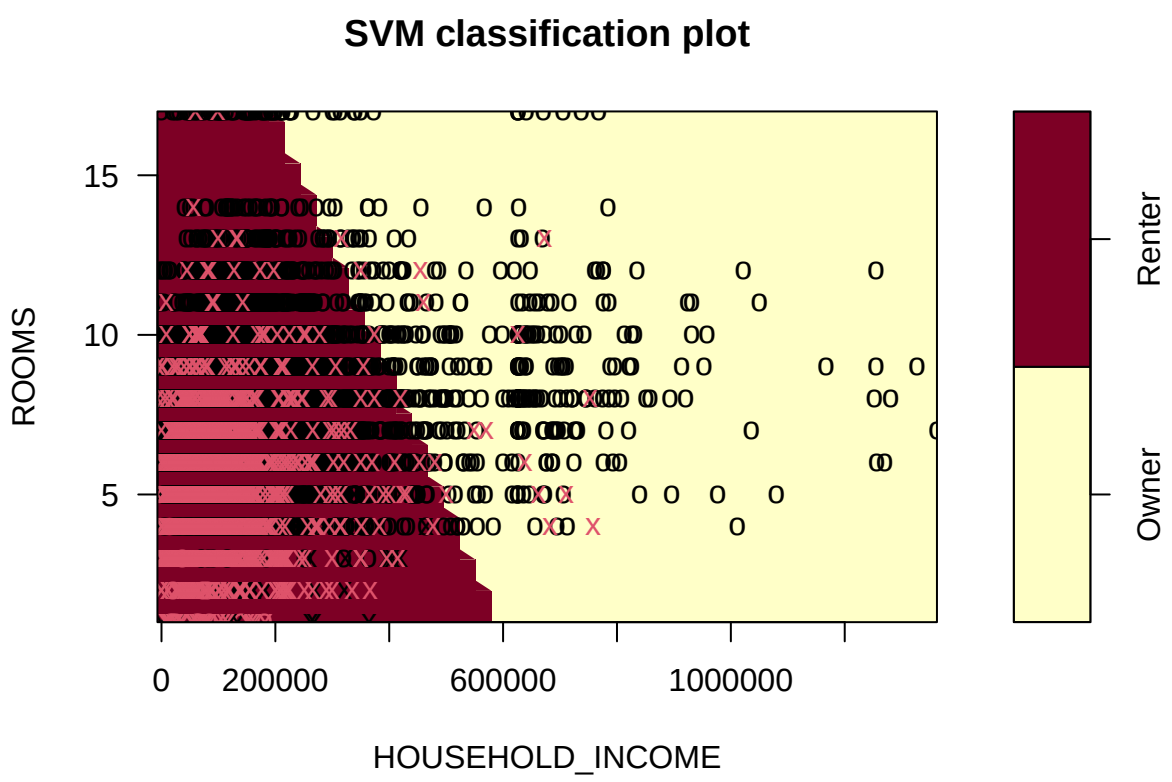
```
#Understand the strongest predictors in the model
w <- t(bestmodel_linear$coefs) %*% bestmodel_linear$SV
w
```

```
##      POPULATION_DENSITY ELECTRICITY_COST     ROOMS HOUSEHOLD_INCOME
## [1,]         -0.1316379        0.1099101 0.1685585        0.3635271
##      BUILT_IN_YEAR  BEDROOMS       AGE   VEHICLES
## [1,]   0.007937511 0.2997763 0.2935839 0.09942806
```

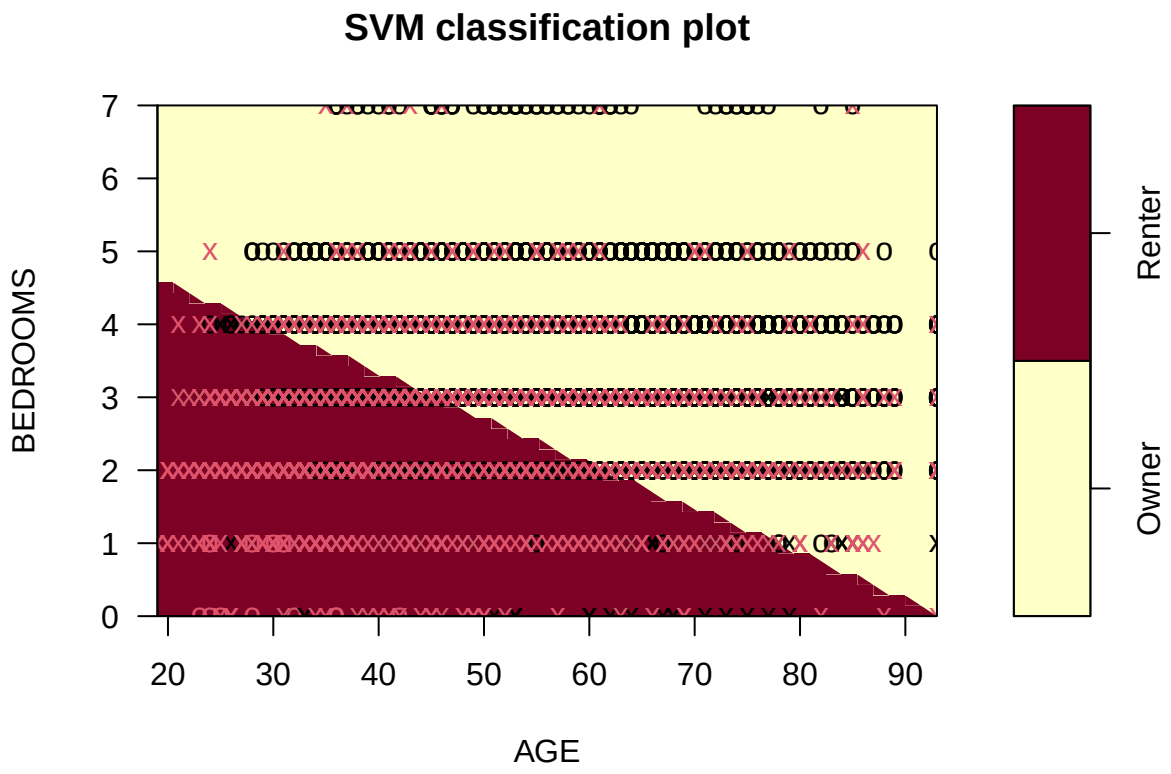
```
plot(bestmodel_linear, linear.train,AGE~HOUSEHOLD_INCOME)
```



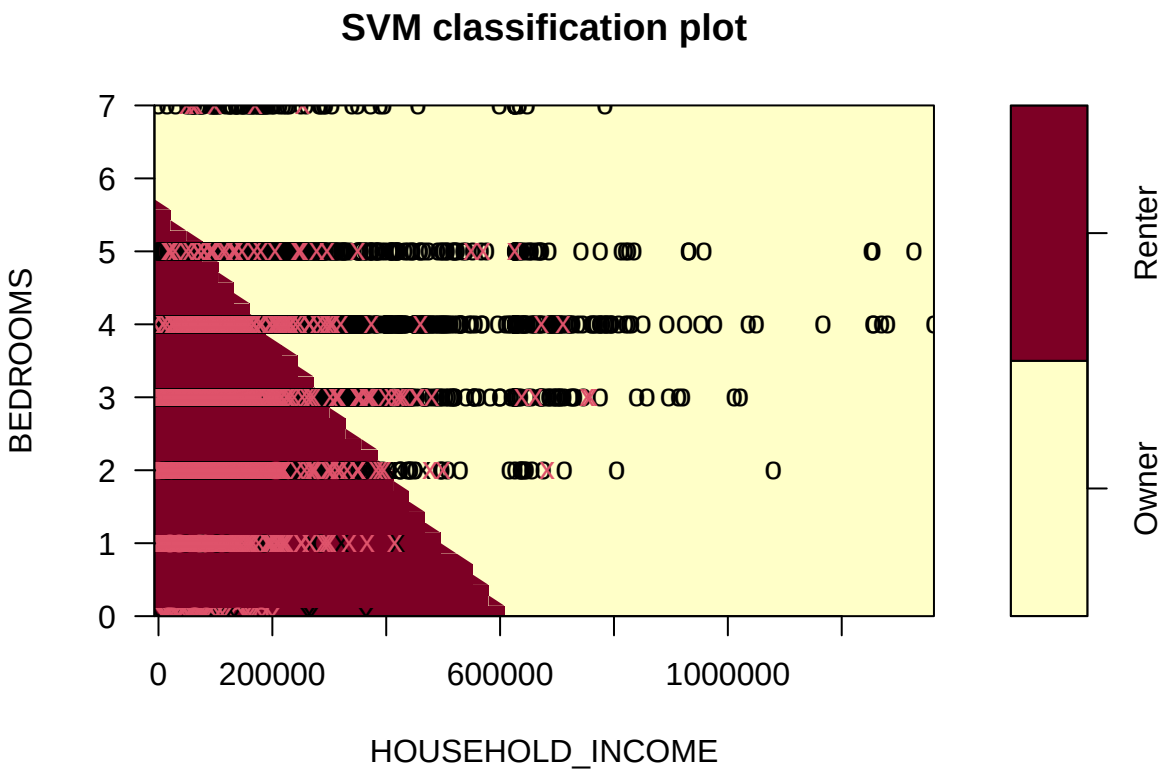

```
plot(bestmodel_linear, linear.train,ROOMS~HOUSEHOLD_INCOME)
```




```
plot(bestmodel_linear, linear.train,BEDROOMS~AGE)
```




```
plot(bestmodel_linear, linear.train,BEDROOMS~HOUSEHOLD_INCOME)
```



First, a linear kernel was used in the SVM model to predict the ownership of a dwelling. The model was tuned for different values of cost on the training data and the optimal value of cost was selected through cross-validation. The best model has a cost of 0.75 and an error rate of 13.8% on the test data. This model was developed using 8 predictors and the strongest pair of predictors were selected to display the relationship between the variables. An SVM plot is shown in Figure 1. The household income and the number of rooms are plotted on the x and y axes, respectively. In the below plot, the yellow highlighted area reflects the region of feature space of the 'Owner' class and the area highlighted in red represents class 'Renter'. In addition, the support vectors are represented by 'x' and the non-support vectors are represented by 'o'. Since the kernel is linear, the resulting plot also includes a linear decision boundary separating the two classes. Despite the decision boundary clearly separating the two classes, a few data points have been misclassified.

```
# Fit the best linear model and note the time taken to run the model
tic()
svmfit_linear <- svm(HOMEOWNERSHIP ~ . , data = linear.train, kernel = "linear", cost = 0.75)
toc()
```

```
## 1.289 sec elapsed
```

PROBLEM-2

Polynomial SVM Model

#Fit the model one the data where the owner is the oldest married member of the family

```
#creating new dataframe with selected variables

#linear model---- HOMEOWNERSHIP,POPULATION_DENSITY,ELECTRICITY_COST,ROOMS,HOUSEHOLD_INCOME,BUILT_IN_YEA
#polynomial model---- HOMEOWNERSHIP,ELECTRICITY_COST,WATER_COST,VEHICLES,ROOMS,HOUSEHOLD_INCOME,BUILT_I
#radial model---- HOMEOWNERSHIP,WATER_COST,ROOMS,HOUSEHOLD_INCOME,VEHICLES,NCOUPLES,POPULATION_DENSITY

polynomial_data <-housing_data %>%
  dplyr::select(HOMEOWNERSHIP,ELECTRICITY_COST,WATER_COST,VEHICLES,ROOMS,HOUSEHOLD_INCOME,BUILT_IN_YEAR
```

```
#Split the data into train and test by considering 70% of data as training data and reserving the remai

set.seed(1)
train <- sample(nrow(polynomial_data) * 0.5)
polynomial.train <- polynomial_data[train, ]
polynomial.test <- polynomial_data[-train, ]
```

```
#Tune the model at different values of cost,degree,coef0

tic()
set.seed(1)
tune.out.polynomial <- tune(svm, HOMEOWNERSHIP ~ ., data = polynomial.train,
    kernel = "polynomial",
    ranges = list(
      degree= c(2,3,4),
      cost = c(0.01,0.05,0.1,0.5,0.75, 1, 10),
      coef0 = c(0,1)
    )
  )
toc()
```

```
## 605.152 sec elapsed
```

Approx– 9-10 min

```
summary(tune.out.polynomial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  degree cost coef0
##       4 0.05     1
##
## - best performance: 0.1272536
##
## - Detailed performance results:
##    degree  cost coef0     error  dispersion
## 1       2  0.01     0 0.1647571 0.011582631
## 2       3  0.01     0 0.1429594 0.011045419
## 3       4  0.01     0 0.1561113 0.010092083
```

```
## 4       2  0.05     0 0.1647571 0.011582631
## 5       3  0.05     0 0.1326102 0.011891110
## 6       4  0.05     0 0.1450299 0.010566412
## 7       2  0.10     0 0.1647571 0.011582631
## 8       3  0.10     0 0.1311490 0.011803332
## 9       4  0.10     0 0.1433259 0.010557121
## 10      2  0.50     0 0.1647571 0.011582631
## 11      3  0.50     0 0.1294450 0.010761352
## 12      4  0.50     0 0.1424746 0.009980357
## 13      2  0.75     0 0.1647571 0.011582631
## 14      3  0.75     0 0.1295669 0.010151214
## 15      4  0.75     0 0.1418663 0.010269667
## 16      2  1.00     0 0.1647571 0.011582631
## 17      3  1.00     0 0.1301757 0.010314316
## 18      4  1.00     0 0.1419883 0.010909817
## 19      2 10.00     0 0.1647571 0.011582631
## 20      3 10.00     0 0.1298098 0.009961098
## 21      4 10.00     0 0.1423537 0.010180619
## 22      2  0.01     1 0.1384547 0.011834447
## 23      3  0.01     1 0.1310272 0.011860778
## 24      4  0.01     1 0.1285927 0.011774147
## 25      2  0.05     1 0.1302975 0.012970333
## 26      3  0.05     1 0.1285930 0.011025829
## 27      4  0.05     1 0.1272536 0.011638836
## 28      2  0.10     1 0.1295669 0.012691508
## 29      3  0.10     1 0.1279838 0.011055657
## 30      4  0.10     1 0.1278627 0.012952526
## 31      2  0.50     1 0.1292020 0.014332885
## 32      3  0.50     1 0.1279841 0.011795085
## 33      4  0.50     1 0.1277408 0.010401633
## 34      2  0.75     1 0.1292018 0.014111270
## 35      3  0.75     1 0.1277405 0.011752380
## 36      4  0.75     1 0.1279844 0.010882584
## 37      2  1.00     1 0.1293236 0.014289750
## 38      3  1.00     1 0.1277403 0.011248933
## 39      4  1.00     1 0.1284713 0.011381100
## 40      2 10.00     1 0.1295671 0.014247332
## 41      3 10.00     1 0.1274969 0.011968501
## 42      4 10.00     1 0.1330983 0.013135710
```

```
bestmodel_polynomial <- tune.out.polynomial$best.model
summary(bestmodel_polynomial)
```

```
##
## Call:
## best.tune(method = svm, train.x = HOMEOWNERSHIP ~ ., data = polynomial.train,
##     ranges = list(degree = c(2, 3, 4), cost = c(0.01, 0.05, 0.1,
##         0.5, 0.75, 1, 10), coef0 = c(0, 1)), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.05
```

```
##      degree:  4
##      coef.0:  1
##
## Number of Support Vectors:  2383
##
##  ( 1247 1136 )
##
##
## Number of Classes:  2
##
## Levels:
##  Owner Renter
```

```
#Training error
ypred <- predict(bestmodel_polynomial, polynomial.train)
table(predict = ypred, truth = polynomial.train$HOMEOWNERSHIP)
```

```
##         truth
## predict  Owner Renter
##   Owner   6750    900
##   Renter   109    453
```

```
training.err<- mean(ypred != polynomial.train$HOMEOWNERSHIP)
cat("Training error rate for SVM model using polynomial kernel is ", training.err)
```

```
## Training error rate for SVM model using polynomial kernel is  0.122869
```

```
#Test error
ypred <- predict(bestmodel_polynomial, polynomial.test)
table(predict = ypred, truth = polynomial.test$HOMEOWNERSHIP)
```

```
##         truth
## predict  Owner Renter
##   Owner   6772    891
##   Renter   122    428
```

```
test.err <- mean(ypred != polynomial.test$HOMEOWNERSHIP)
cat("Test error rate for SVM model using polynomial kernel is  ", test.err)
```

```
## Test error rate for SVM model using polynomial kernel is   0.123341
```

```
#Test error in a neat form as it has highest accuracy
ypred <- predict(bestmodel_polynomial, polynomial.test)
Predicted<-ypred
Actual<-polynomial.test$HOMEOWNERSHIP
table(Actual,Predicted)
```

```
##         Predicted
## Actual   Owner Renter
##   Owner   6772    122
##   Renter   891    428
```

```
errorrate<- mean(Actual!=Predicted)
cat("Test error rate for SVM model using polynomial kernel is ", errorrate)
```

```
## Test error rate for SVM model using polynomial kernel is  0.123341
```

```
accuracy<-mean(Actual==Predicted)
cat("Test accuracy for SVM model using polynomial kernel is ", accuracy)
```

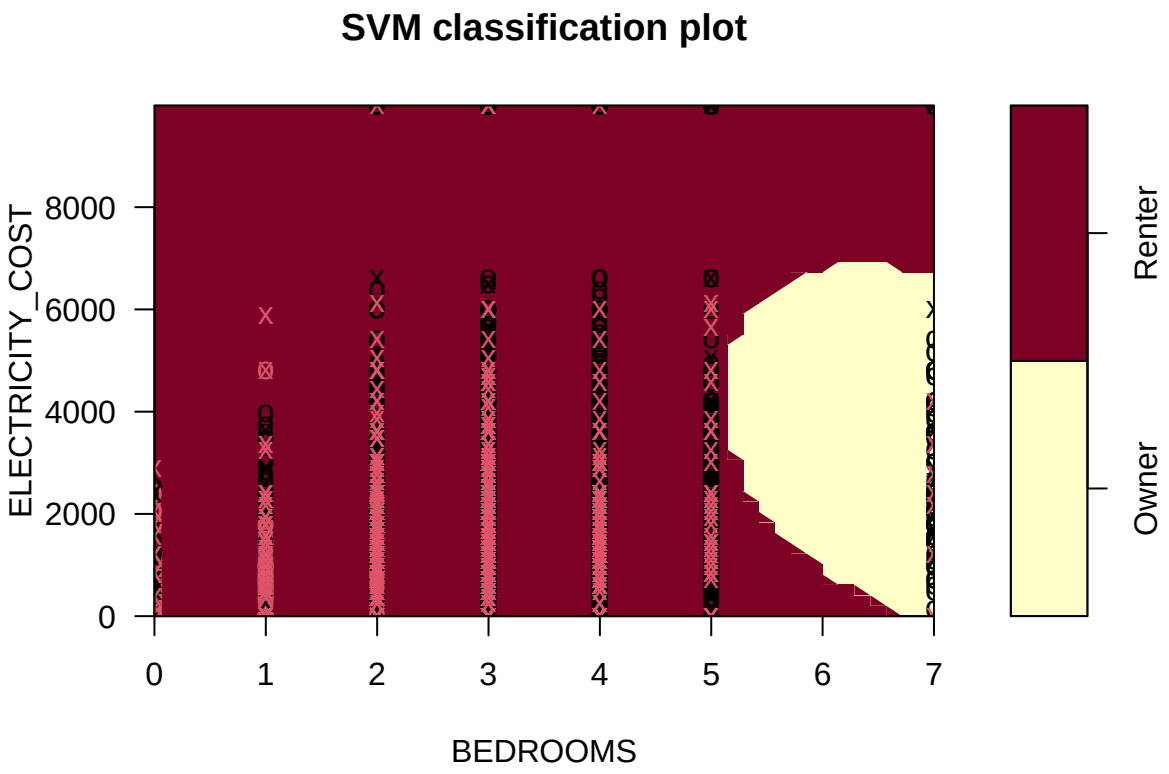
```
## Test accuracy for SVM model using polynomial kernel is  0.876659
```

As the SVM model with a polynomial kernel performs better than all the other models the confusion matrix of this model is shown in Figure 2. The majority of the observations in the dataset could be attributed to owners, and the model correctly predicts 6772 out of 6894 observations. This can be justifiable as just a subset of data was considered for analysis purposes i.e., only the people who are married. Married individuals are more likely to settle down in a location that they call home and ultimately invest in owning a house as opposed to renting it.

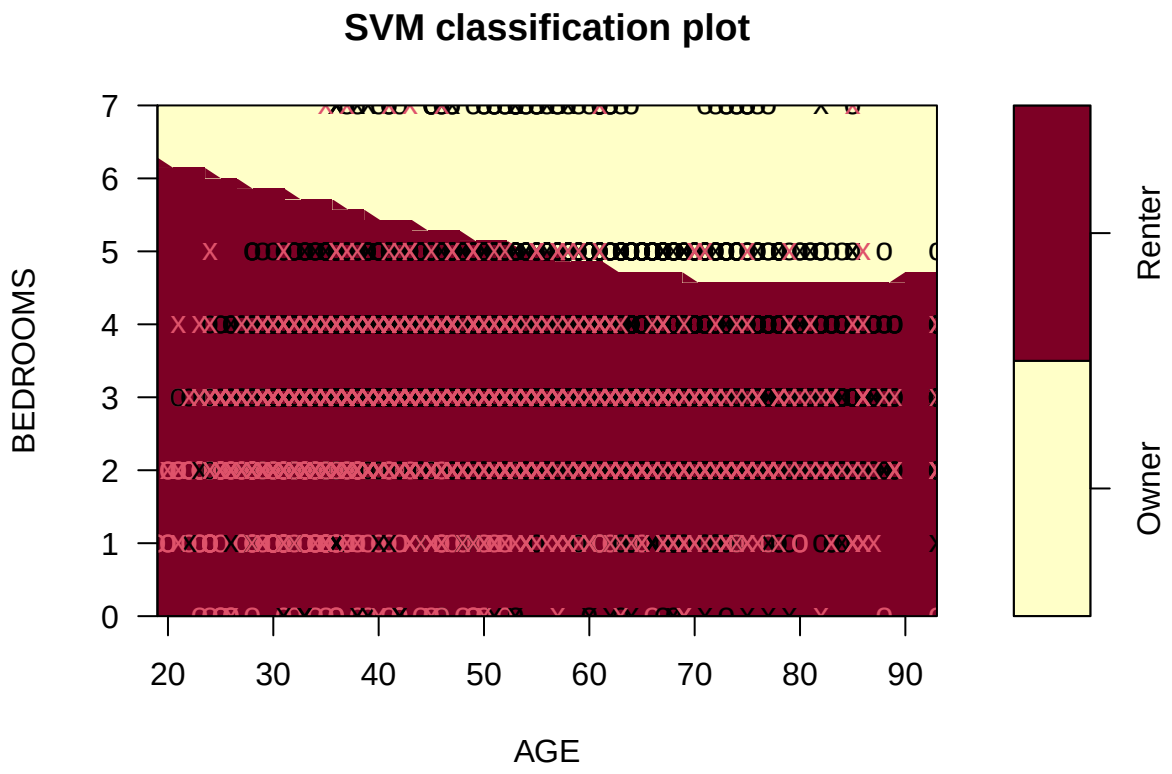
```
#Understand the strongest predictors in the model
w <- t(bestmodel_polynomial$coefs) %*% bestmodel_polynomial$SV
w
```

```
##      ELECTRICITY_COST WATER_COST   VEHICLES       ROOMS HOUSEHOLD_INCOME
## [1,]      -0.08805544 -0.1216872 -0.1389925 -0.03833782       0.06420289
##      BUILT_IN_YEAR  BEDROOMS         AGE
## [1,]   -0.02567793 0.0655641 -0.02980459
```

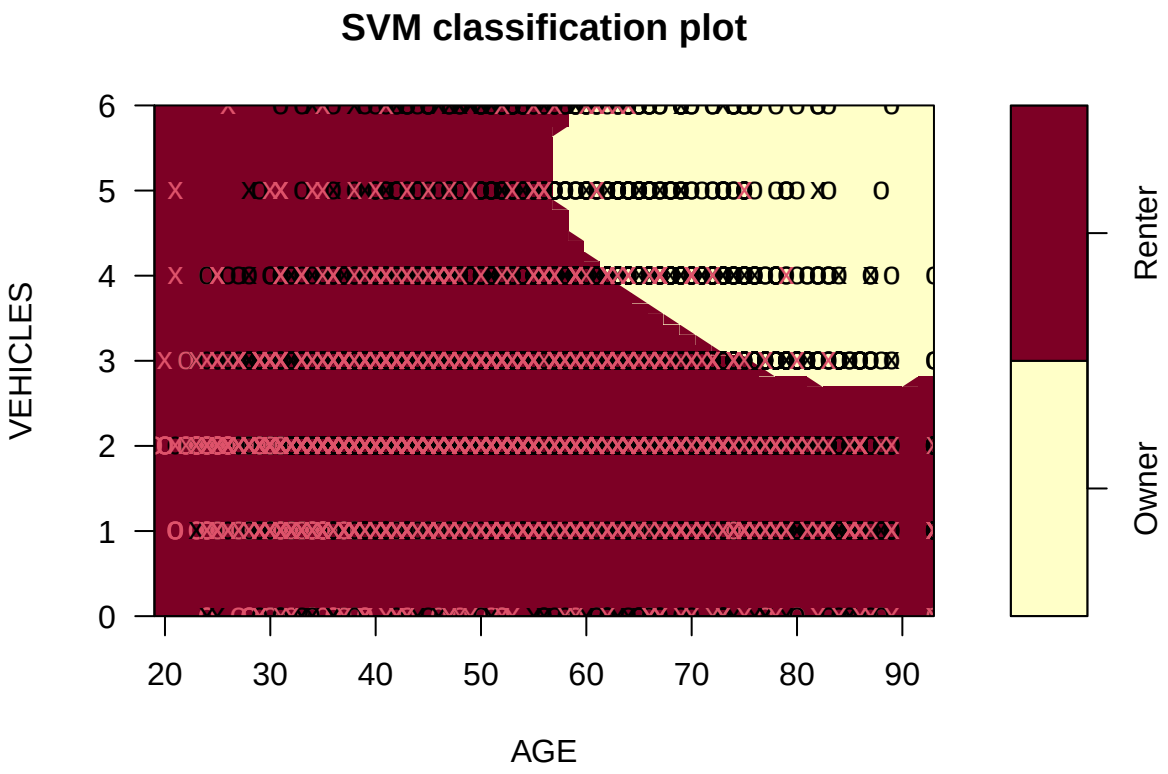
```
plot(bestmodel_polynomial, polynomial.train,ELECTRICITY_COST~BEDROOMS)
```



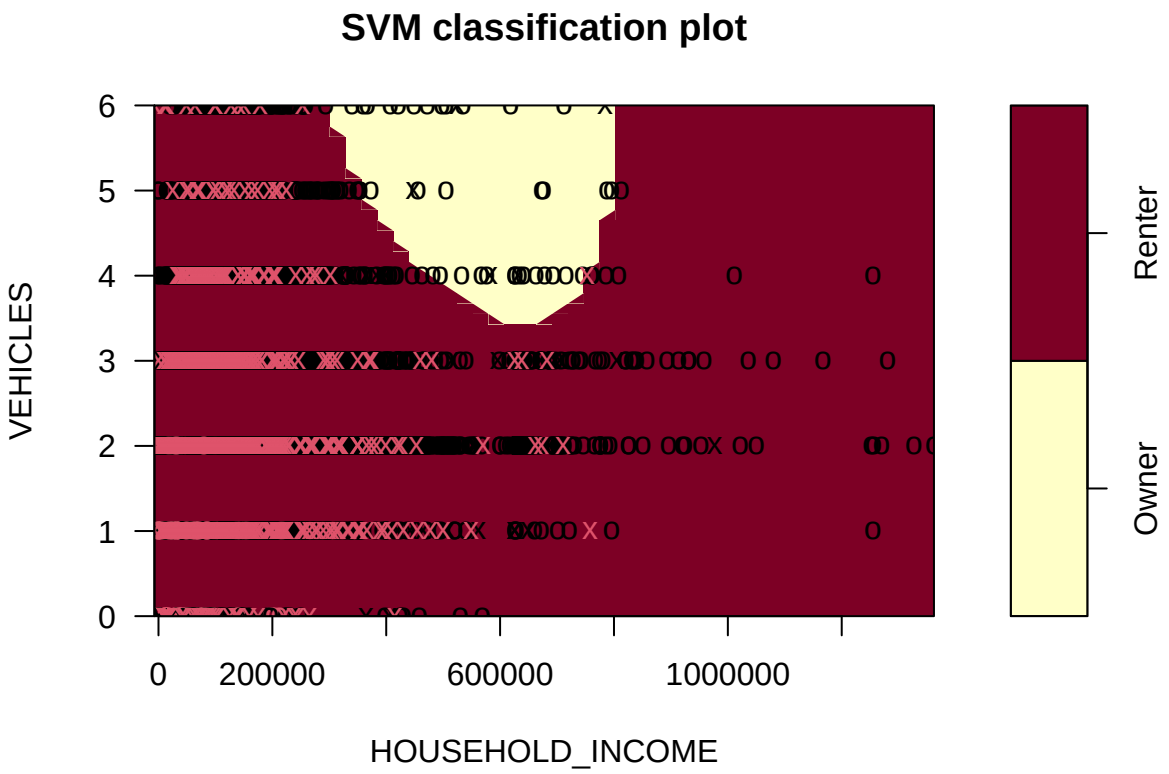

```
plot(bestmodel_polynomial, polynomial.train,BEDROOMS~AGE)
```




```
plot(bestmodel_polynomial, polynomial.train,VEHICLES~AGE)
```




```
plot(bestmodel_polynomial, polynomial.train,VEHICLES~HOUSEHOLD_INCOME,xlab="Number of Vehicles",ylab="H
```



Next, a polynomial kernel was used in the SVM model to predict the ownership of a dwelling. The model was tuned for different values of cost, degree, and coef0 on the training data. Through cross-validation, the optimal value of the three hyperparameters was selected. The best model had a cost of 0.05, degree 4, coef0 of 1, and an error rate of 12.3% on the test data. This model was also developed using 8 predictors and the strongest pair of predictor variables were selected to display the relationship between the variables. Figure 2 shows the SVM plot with variables number of bedrooms and cost of electricity plotted on the x and y axes, respectively. The observations are separated by the curved decision boundary (since the kernel = polynomial) and the area shaded in yellow represents class 'Owner' while the area in red represents class 'Renter'. The support vectors are represented by 'x' and the non-support vectors are represented by 'o'. Also, it can be observed that a few observations have been incorrectly classified on either side of the decision boundary.

```
#Note the time taken to fit the best polynomial model
tic()
svmfit_polynomial <- svm(HOMEOWNERSHIP ~ . , data = linear.train, kernel = "polynomial", degree=4, coef
toc()
```

```
## 0.953 sec elapsed
```

PROBLEM-3

Radial SVM Model

#Fit the model one the data where the owner is the oldest married member of the family

```
#creating new dataframe with selected variables

#linear model---- HOMEOWNERSHIP,POPULATION_DENSITY,ELECTRICITY_COST,ROOMS,HOUSEHOLD_INCOME,BUILT_IN_YEA
#polynomial model---- HOMEOWNERSHIP,ELECTRICITY_COST,WATER_COST,VEHICLES,ROOMS,HOUSEHOLD_INCOME,BUILT_I
#radial model---- HOMEOWNERSHIP,WATER_COST,ROOMS,HOUSEHOLD_INCOME,VEHICLES,NCOUPLES,POPULATION_DENSITY


radial_data <-housing_data %>%
  dplyr::select(HOMEOWNERSHIP,WATER_COST,ROOMS,HOUSEHOLD_INCOME,VEHICLES,NCOUPLES,POPULATION_DENSITY)
```

```
#Split the data into train and test by considering 50% of data as training data and reserving the remai

set.seed(1)
train <- sample(nrow(radial_data) * 0.5)
radial.train <- radial_data[train, ]
radial.test <- radial_data[-train, ]
```

```
#Tune the model at different values of cost,gamma through cross-validation

set.seed(1)
tic()
tune.out.radial <- tune(svm, HOMEOWNERSHIP ~ ., data = radial.train,
    kernel = "radial",
    ranges = list(
      cost = c(0.01,0.1, 0.5, 0.75, 1, 10, 50),
      gamma = c(0.5, 1, 2, 3, 4)
    )
  )
toc()
```

```
## 653.789 sec elapsed
```

```
summary(tune.out.radial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1   0.5
##
## - best performance: 0.1386986
##
## - Detailed performance results:
##     cost gamma     error  dispersion
## 1   0.01   0.5 0.1647571 0.011582631
## 2   0.10   0.5 0.1539204 0.012212230
## 3   0.50   0.5 0.1412558 0.010774306
## 4   0.75   0.5 0.1396727 0.010651018
## 5   1.00   0.5 0.1386986 0.009804285
```

```
## 6  10.00   0.5 0.1445439 0.014517901
## 7  50.00   0.5 0.1556254 0.015611400
## 8   0.01   1.0 0.1647571 0.011582631
## 9   0.10   1.0 0.1550160 0.013730109
## 10  0.50   1.0 0.1419873 0.011641395
## 11  0.75   1.0 0.1402828 0.011611245
## 12  1.00   1.0 0.1395521 0.012727211
## 13 10.00   1.0 0.1531898 0.016531880
## 14 50.00   1.0 0.1662184 0.014996194
## 15  0.01   2.0 0.1647571 0.011582631
## 16  0.10   2.0 0.1580605 0.013189953
## 17  0.50   2.0 0.1494154 0.013548551
## 18  0.75   2.0 0.1460056 0.013981648
## 19  1.00   2.0 0.1452744 0.012568959
## 20 10.00   2.0 0.1590338 0.016393511
## 21 50.00   2.0 0.1790040 0.014177995
## 22  0.01   3.0 0.1647571 0.011582631
## 23  0.10   3.0 0.1604958 0.013546007
## 24  0.50   3.0 0.1508765 0.013155404
## 25  0.75   3.0 0.1501468 0.013520784
## 26  1.00   3.0 0.1500246 0.014178473
## 27 10.00   3.0 0.1641485 0.017145904
## 28 50.00   3.0 0.1854595 0.013788891
## 29  0.01   4.0 0.1647571 0.011582631
## 30  0.10   4.0 0.1623221 0.012819660
## 31  0.50   4.0 0.1525813 0.011039054
## 32  0.75   4.0 0.1520945 0.012674796
## 33  1.00   4.0 0.1500243 0.013785900
## 34 10.00   4.0 0.1693859 0.016287722
## 35 50.00   4.0 0.1893569 0.013827372
```

```
bestmodel_radial <- tune.out.radial$best.model
summary(bestmodel_radial)
```

```
##
## Call:
## best.tune(method = svm, train.x = HOMEOWNERSHIP ~ ., data = radial.train,
##     ranges = list(cost = c(0.01, 0.1, 0.5, 0.75, 1, 10, 50), gamma = c(0.5,
##         1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  3013
##
##  ( 1777 1236 )
##
##
## Number of Classes:  2
##
## Levels:
```

```
##  Owner Renter
```

```
#gamma of best radial modle
bestmodel_radial$gamma
```

```
## [1] 0.5
```

```
#Training error
ypred <- predict(bestmodel_radial, radial.train)
table(predict = ypred, truth = radial.train$HOMEOWNERSHIP)
```

```
##         truth
## predict  Owner Renter
##   Owner   6714    944
##   Renter   145    409
```

```
training.err<- mean(ypred != radial.train$HOMEOWNERSHIP)
cat("Training error rate for SVM model using radial kernel is ", training.err)
```

```
## Training error rate for SVM model using radial kernel is  0.1326108
```

```
#Test error
ypred <- predict(bestmodel_radial, radial.test)
table(predict = ypred, truth = radial.test$HOMEOWNERSHIP)
```

```
##         truth
## predict  Owner Renter
##   Owner   6741    949
##   Renter   153    370
```

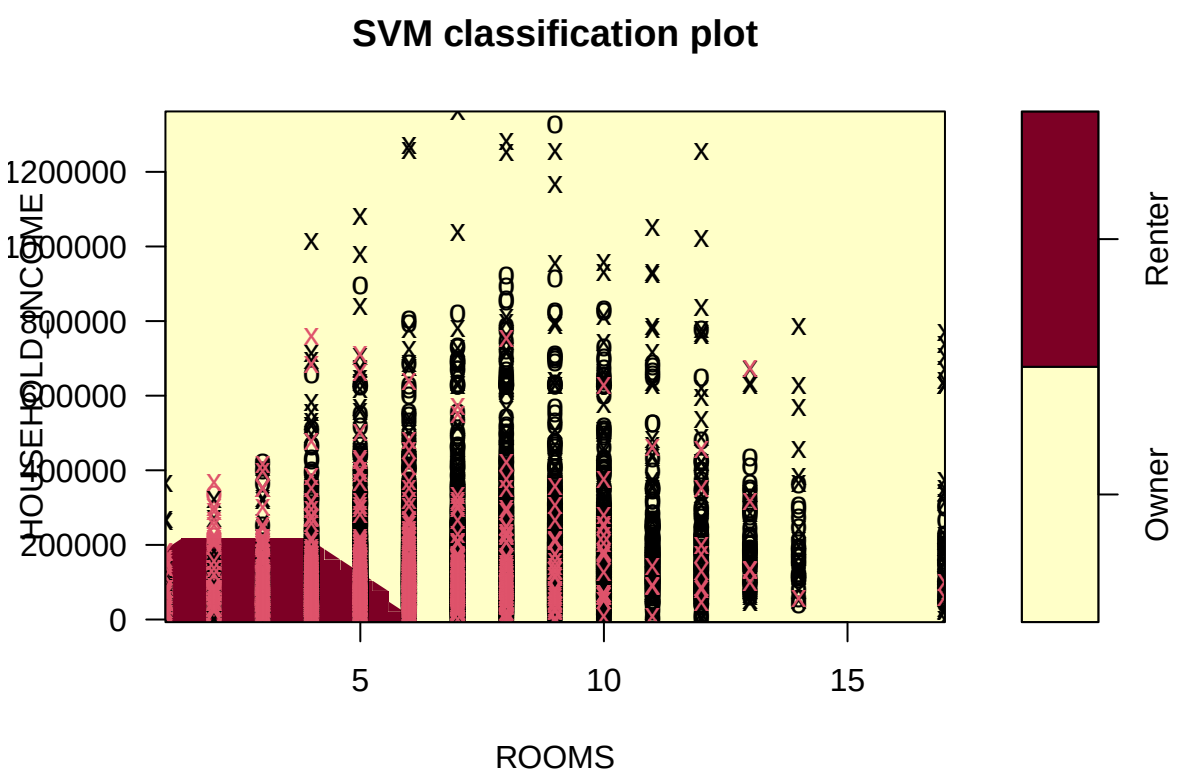
```
test.err <- mean(ypred != radial.test$HOMEOWNERSHIP)
cat("Test error rate for SVM model using radial kernel is", test.err)
```

```
## Test error rate for SVM model using radial kernel is 0.1341775
```

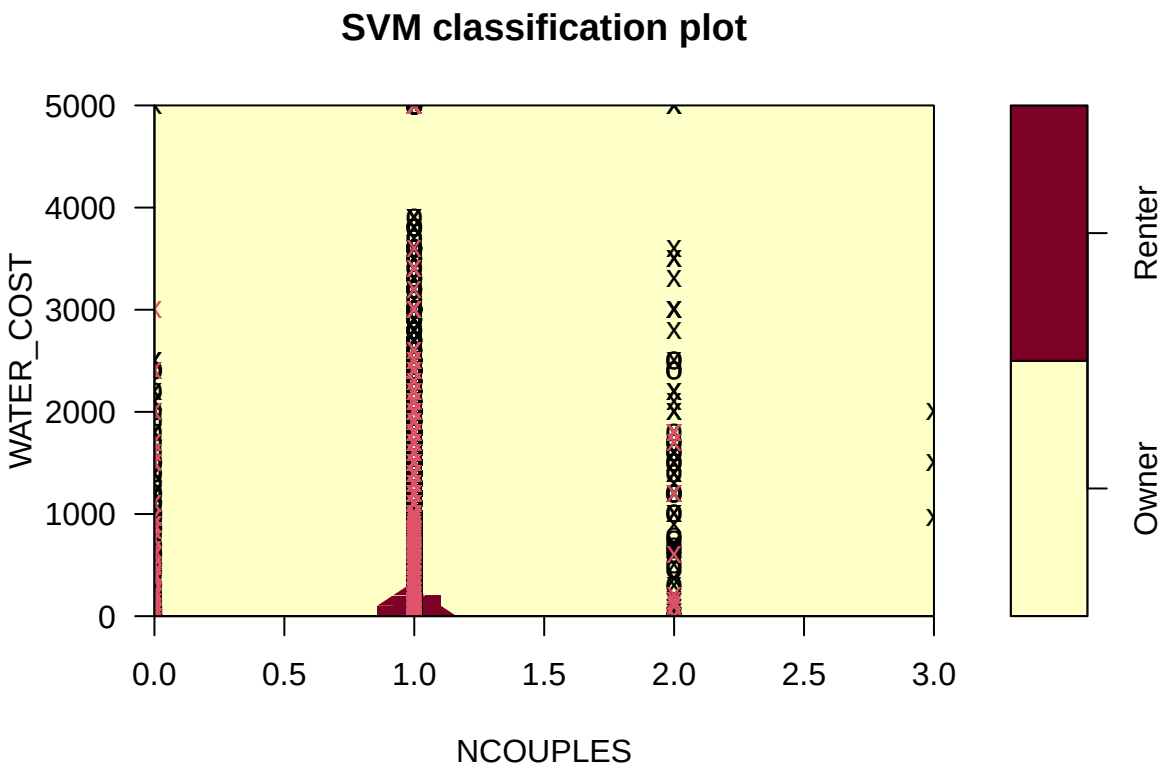
```
#Understand the strongest predictors in the model
w <- t(bestmodel_radial$coefs) %*% bestmodel_radial$SV
w
```

```
##      WATER_COST    ROOMS HOUSEHOLD_INCOME VEHICLES NCOUPLES POPULATION_DENSITY
## [1,]   50.33676 82.37668         57.62859 45.67071 58.66509           -41.7926
```

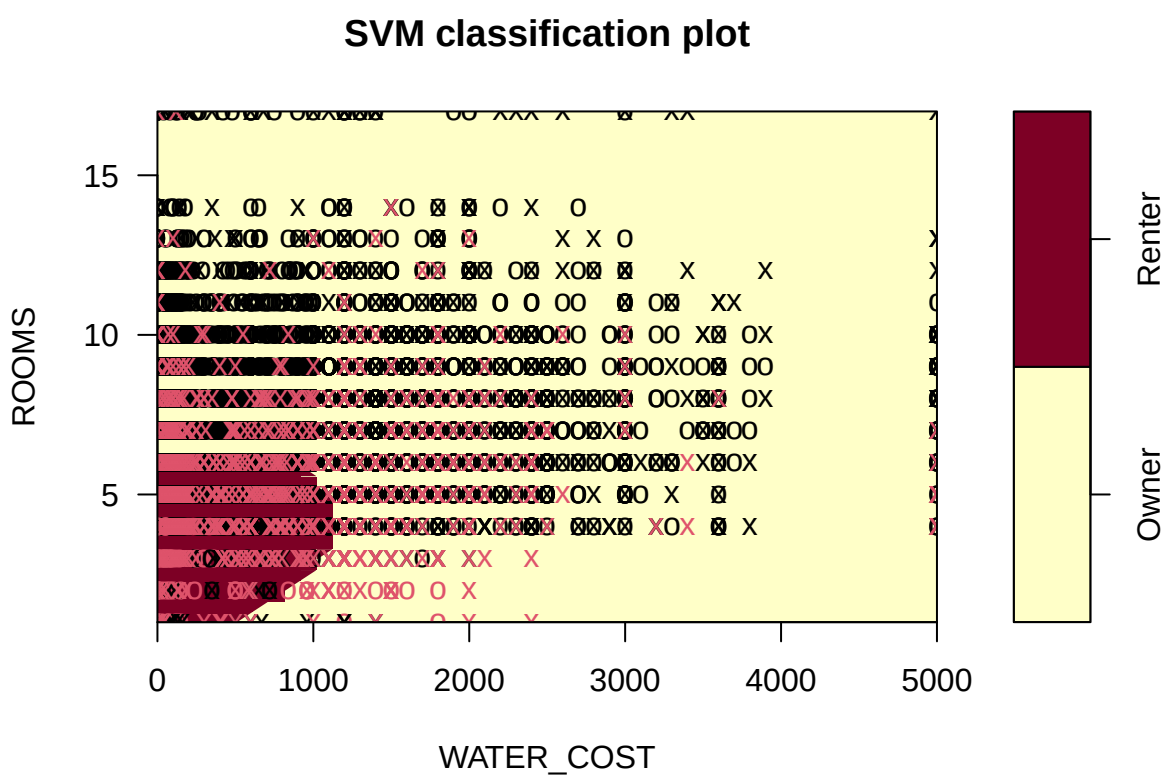
```
plot(bestmodel_radial, radial.train,HOUSEHOLD_INCOME~ROOMS)
```



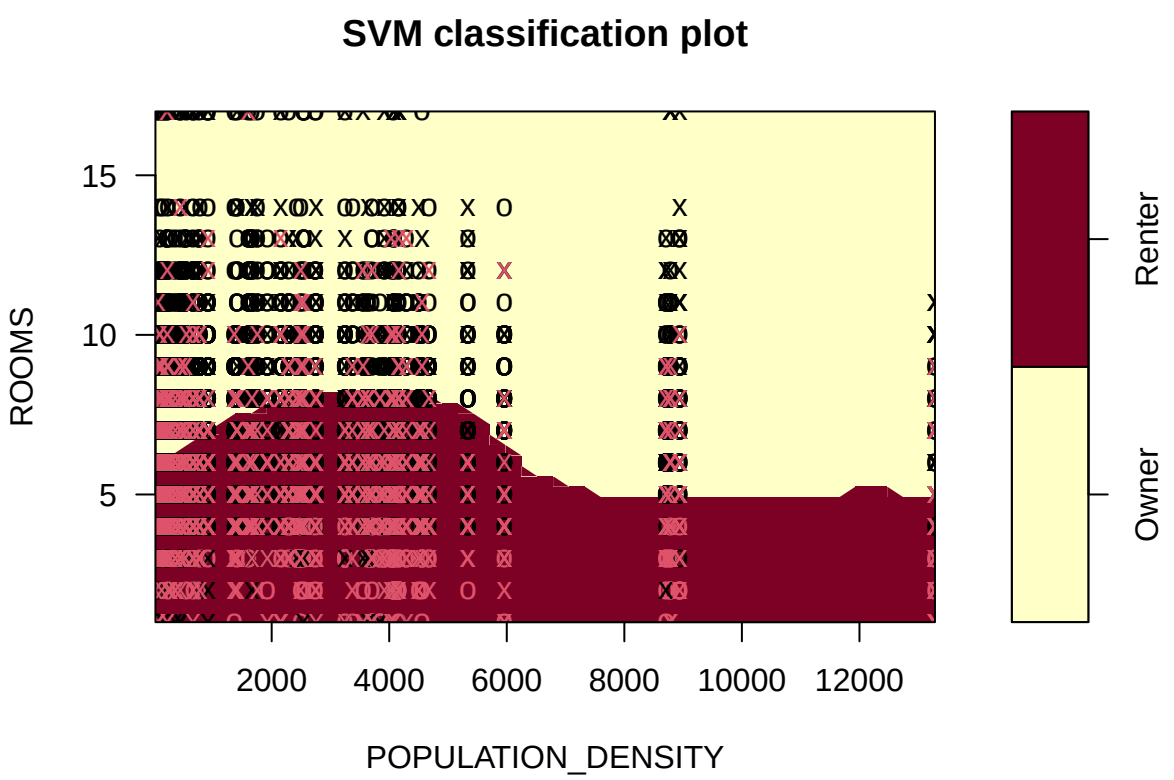

```
plot(bestmodel_radial, radial.train,WATER_COST~NCOUPLES)
```




```
plot(bestmodel_radial, radial.train,ROOMS~WATER_COST)
```




```
plot(bestmodel_radial, radial.train,ROOMS~POPULATION_DENSITY,xlab="Number of Rooms",ylab="Population Den
```



Lastly, a radial kernel was used in the SVM model to predict the ownership of a dwelling. On the training data, the model was tuned for various cost and gamma values. The best value for each of the three hyper parameters cost and gamma was determined through cross-validation. The best model had a cost of 1, a gamma of 0.5, and an error rate of 13.4% on the test data. This model was also developed using 5 predictors and the strongest pair of predictor variables were selected to display the relationship between the variables. Figure 2 shows the SVM plot with the population density and the number of rooms plotted on the x and y axes, respectively. The decision boundary is a radial curve (since the kernel is radial) separating the two classes with the area shaded in yellow representing class 'Owner' and the area in red representing class 'Renter'. The support vectors are represented by 'x' and the non-support vectors are represented by 'o'. Likewise, it very well may be seen that a few observations have been misclassified on either side of the decision boundary.

```
#Note the time taken to fit the radial model
tic()
svmfit_radial <- svm(HOMEOWNERSHIP ~ ., data = radial.train, kernel = "radial", gamma = 0.5, cost = 1)
toc()
```

```
## 1.336 sec elapsed
```