Predicting Ownership of Housing using Support Vector Machine

Karthika Selvaraj

April 26th, 2023

Abstract

This study utilizes Support Vector Machine to predict if a residence is occupied by owners or tenants based on information retrieved about the occupants and the property using the U.S census for social, economic and health research IPSUMS data. To identify individuals who may be at a higher risk of experiencing housing insecurity or financial instability. Model building involves various stages such as data pre-processing and feature selection to improve the classification model and predict the type of resident living in a housing unit. The theory of Support Vector Machines and their application in making predictions are then briefly discussed, along with different kernelization. This paper analyzes the different hyper parameters such as kernels type and regularization parameter that fits the predicting model better.

1 Introduction and Overview

The Integrated Public Use Microdata Series (IPUMS) is the database recorded from the surveys that are statistically validated for researchers conducting various types of analysis. The database collects information on the social and economic factors such as the information regarding the people living in a housing unit from 1790 to present. This paper uses the variables such as income, age, marital status of the occupants and the cost of living with respect to water, fuel usage annually, year the house was built and density of the population in that area. The goal of this study is to investigate the highly influential factors to predict the type of residents using the ML models. To explore the ability of the model to classify the occupants into two different classes such as homeowners and renters while optimizing the model efficiency.

2 Theoretical Background

A Supervised Machine Learning algorithm called Support Vector Machine (SVM) is utilized for classifying efficiently. This technique draws a hyper-plane that divides the data classes. The hyper plane has n-1 form in 'n' dimensional space, where 'n' is the number of features. The maximum margin classifier, which is a straightforward and understandable classifier, is extended to the support vector classifier. This then demonstrates the classifier, despite its elegance and simplicity, cannot be used with most data sets since it requires the classes to be well separated from each other. In place of the maximal margin classifier, the support vector classifier is provided, which is more versatile. The support vector machine, a non-linear boundary, is accommodated as an extended version of the support vector classifier.

2.1 Maximal Marginal Classifier

The hypothetical classifier that best explains how SVM functions is the Maximal-Margin Classifier. The maximal marginal classifier is crucial to SVM. Using the number of feature variables as input to the

n-dimensional plane, the hyperplane divides the variables into classes that are clearly separated linearly. When there are more than three dimensions, it can be challenging to visualize a hyperplane, but the idea of a (p 1) - dimensional hyperplane is still relevant. Below is a diagram illustrating the definition of a hyperplane in two dimensions.

$$\beta 0 + \beta 1X1 + \beta 2X2 = 0$$

On the hyperplane is where data point X is located. It is easy to determine which side of the hyperplane a point is on by computing the sign on the left-hand side. Our objective is to build a classifier from the training data that will correctly classify the test observation based on its feature measurements. On the basis of which side of the hyperplane they are on, the test observations are divided into categories. Suppose the observation belongs to a specific class (1,-1).

$$\beta 0 + \beta 1xi1 + \beta 2xi2 + \cdots + \beta pxip \quad > 0 \text{ belongs to class 1}$$

$$\beta 0 + \beta 1xi1 + \beta 2xi2 + \cdots + \beta pxip \quad < 0 \text{ belongs to class -1}$$

The hyperplane that is farthest distant from the training observations is called the marginal hyperplane. That is, we could calculate the (perpendicular) distance between each training observation and a specific separating hyperplane; the margin, or shortest distance between the observations and the hyperplane, is the least of these distances. The separating hyperplane with the largest margin is known as the maximal margin hyperplane. It is also known as the hyperplane with the closest proximity to the training observations. The side of the maximal margin hyperplane on which a test observation lies can therefore be used to categorize it. The Maximal Margin Classifier has been extended by the Support Vector Classifier. The position and direction of the hyperplane are affected by the data points that are closest to it. We raise the classifier's margin by using these support vectors. The position of the hyperplane will change if the support vectors are changed.

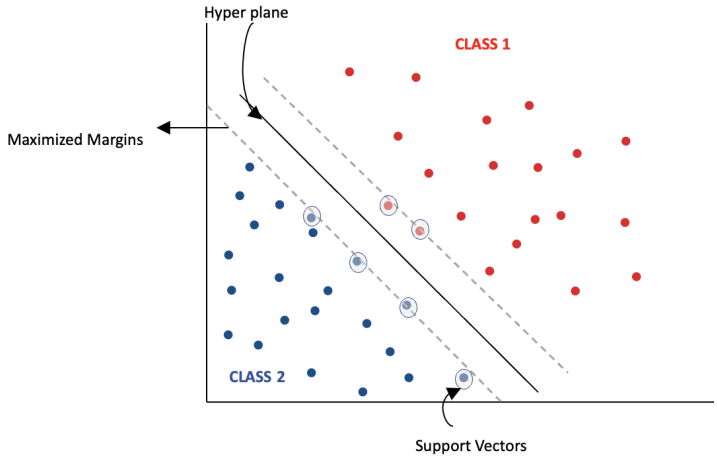


Fig 1: Maximal Marginal Classifier

2.2 Non-Separable Case

There are a number of situations in which it is hard to use a hyperplane to solidly divide a two-class dataset. As a result, it is necessary to modify the condition of maximizing the margin of the hyperplane separating the two classes. The maximal margin classifier for non-separable instances is generalized as a support vector classifier. The hyperplane draws a "soft margin" in such cases to almost separate the classes. Fitting a hyperplane to precisely separate the data isn't always the best course of action. The model may overfit the data if the Maximal Margin Classifier is allowed to be extremely sensitive to a small number of training examples. The objective is to maintain the soft margin as big as feasible while allowing a predetermined number of errors, so that new data points can still be identified correctly.

2.3 Support Vector Machine

The support vector machine (SVM) is a consequence of widening the feature space to allow a non-linear border between the classes using kernels. It is an extension of the support vector classifier. Based on the inner product of two observations, the kernel function determines how similar two data points are in the original feature space. The precise characteristics of the data and the issue being handled can influence the choice of the kernel function.

Equation with ‘p' as the number of predictors and xi,xi’ as the pairs of training observations represents the Linear Kernel.

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

The equation with d as the degree, a positive integer, represents the Polynomial Kernel. By adjusting the support vector classifier in a higher dimensional space known as the Support Vector Machine, it provides a considerably more flexible decision boundary.

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

Equation with a positive constant serves as the representation for the Radial Kernel.The effect of a single training example during transformation is controlled by this parameter, which affects how closely the decision borders encircle points in the input space. Points that are far apart are regarded as comparable when gamma is modest. As a result, there are more point clusters and smoother (perhaps less accurate) decision borders. Points are closer together when the gamma value is higher (this could lead to overfitting).

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

3 Methodology

The IPUMS dataset includes survey data from people all throughout the United States with a variety of socioeconomic and household factors. The aim is to look at variables that may be relevant to determining the kind of residence of the household. Pre-processing the data includes choosing the individual with the highest age within each family as the entry point, adding any missing variables, and converting categorical variables to factor data type. Before creating a binary classification model using SVM, a

subset of potential predictors are selected out due to the size of the dataset. To better understand their effect on prediction as a result of the feature selection, the following variables are highlighted.

- Population Density Weighted, Count of Rooms, Number of rooms, built-in year of the home Construction year of the home.
- Ownership of the property, cost of fuel and water annually, Annual Imputed Income, Age, Marital Status.

3.1 Predicting the type of residents using Social and Economical Factors

The initial goal of this study was to determine whether there was a pattern in the ownership of homes based on socioeconomic data collected from people and their households through the IPUMS survey. The e1070 package in R was used to implement the Support Vector Machines for this assignment. The train data had 3987 samples and the test data had 1709 samples since the data were arbitrarily split into train and test sets with a 70:30 ratio. Different kernels with cross-validation were explored while constructing the SVM model. A linear kernel model with multiple cost values ranging from 0.01 to 100 was assigned to fit the training data. Following this the radial kernel model was assigned to fit the data with the cost values ranging from 1 to 100 and gamma values from 0.5 to 4. Finally, the polynomial kernel model with tuned hyper parameters of cost values ranging from 0.001 to 100, degree values from 2 to 5 and coefficients ranging 0 and 1 were also used to fit the training data. By comparing the above three optimal models obtained from cross validation, the best suitable kernel for this data is determined.

4 Computational Results

4.1 Cross-Validation of Parameters

In the initial SVM modeling, the linear kernel is used with all 10 variables to predict who is holding the property. To find the ideal parameters a cross validation is performed to have the lowest error rate for the training dataset. The best accuracy reached among these were 77% for the cost value of 50.

The models were then trained for polynomial kernels with degrees 2 to 5 and cost ranges between 0.001 and 100. The most accurate model has a degree of 3, cost value of 10, and an accuracy rate of 79%.

For further exploration the model was also trained with the radial kennel with the different combination of cost values ranging from 1 to 100 and gamma values from 0.5 to 4. The optimal model has the cost value of 1 and gamma value of 0.5 with the overall accuracy of 78%.

4.2 SVM Model using Linear Kernel

A confusion matrix for linear kernel model demonstrated that there is a correlation between the variables. It can accurately predict renters class with an accuracy of 84% and owners class with an accuracy of 63%, which makes the model more trustworthy for the renters group. The simulation lasts for 42.646 seconds.

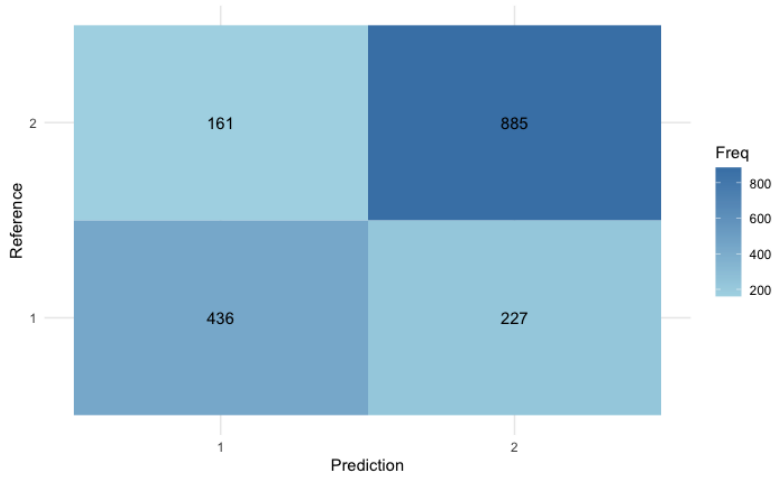


Fig 2: Confusion Matrix - Tuned SVM model with Linear Kernel

The strongly correlated variables such as the density of the local population and the annual income of the individuals are plotted. The below graph shows the hyperplane dividing feature space into two classes linearly even though the owner and renter class data points are not well separated from each other. From the plot we can infer that the people who annually earn lesser than 700,000 only rent a house for living. This scenario is not entirely true in reality as the data points are not well separable, the linear kernel model does not perform efficiently. The plus symbol represents the data points that are support vectors which influence the shift in the hyperplane.

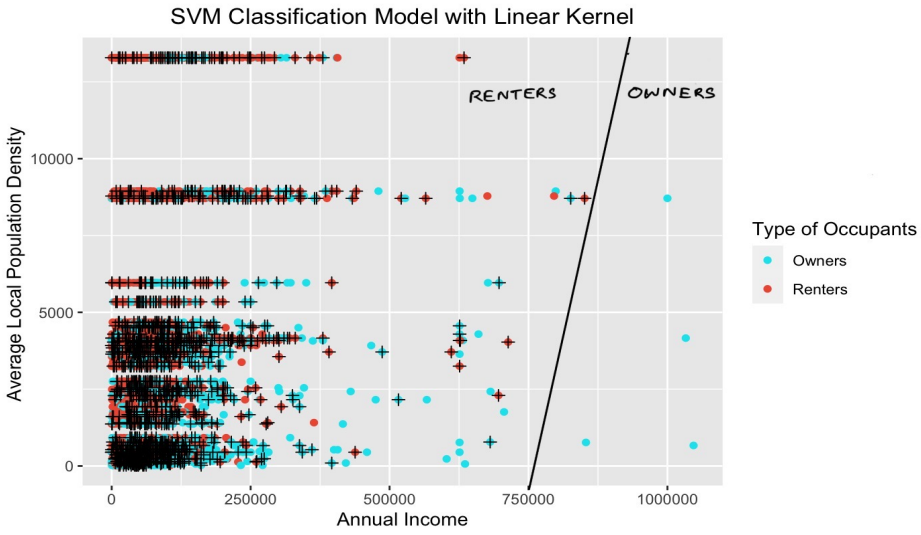


Fig 3: SVM Model with Linear Kernel - Hyper plane on Annual Income vs Population Density

4.3 SVM Model using Radial Kernel

The confusion matrix for the tuned hyper parameter model with radial kernel is shown below. The accuracy in predicting the “renter” class has decreased to 82% and “owner” class has increased to 69% . Thus the model tries to make it more reliable to both the classes. The computational time of the model is 1.364 seconds.

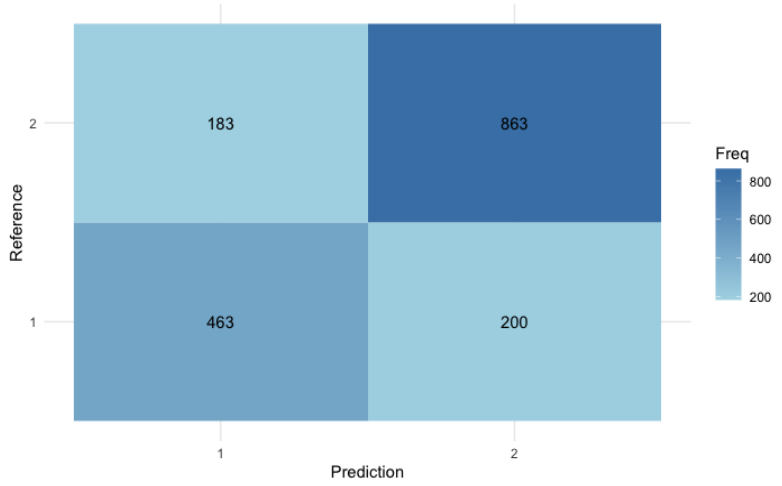


Fig 4: Confusion Matrix - Tuned SVM model with Radial Kernel

The other highly correlated predictors are annual income and number of rooms. The classes are not well separated for these variables as well. The hyperplane cuts the feature space indicating that the model is able to see the pattern and classify the type of occupants. From the plot it can be inferred that the individual with annual income less than $250,000 rents a home consisting of 5 rooms on an average.

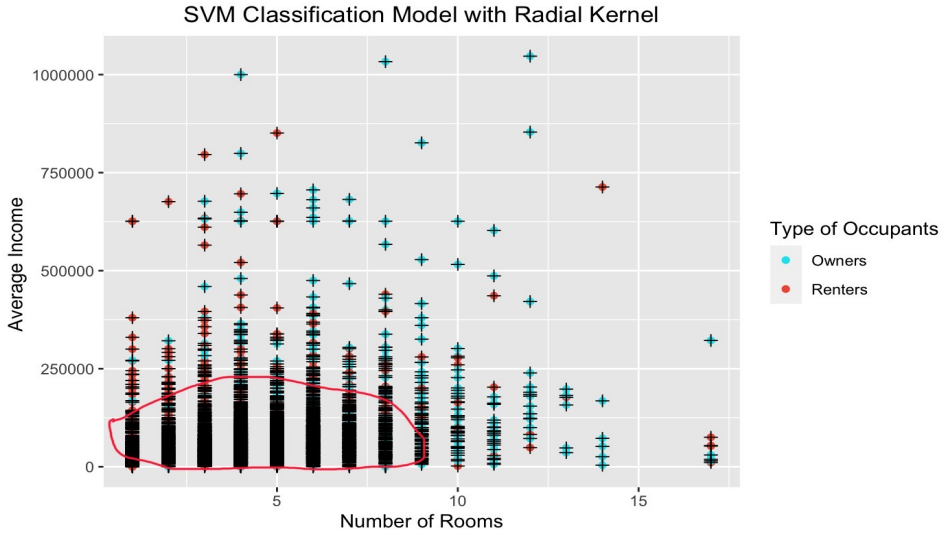


Fig 5: SVM Model with Radial Kernel - Hyper plane on Number of rooms vs Annual Income

4.4 SVM Model using Polynomial Kernel

The confusion matrix for the polynomial kernel is shown below and the overall model accuracy for the “owner” class has improved to 78%. Despite this, the model is unable to improve its performance in classifying the “owner” class, since the number of owner data points are less when compared to the renter data. Hence this model results in a bias towards the “owner” class. The computational time is 40.82 sec for the optimal model.

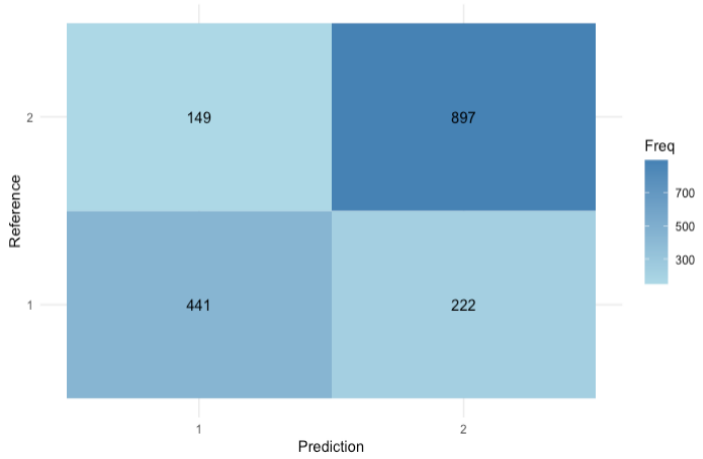


Fig 6: Confusion Matrix - Tuned SVM model with Polynomial Kernel

The age of the individual and the number of rooms in their house fall under highly correlated variables. Though the classes are not well separable the clustering of data points for each class is observed. The hyperplane gives a clear cut on these feature spaces. The graph shows that the people whose age is forty and above own a house with 5 to 13 rooms.

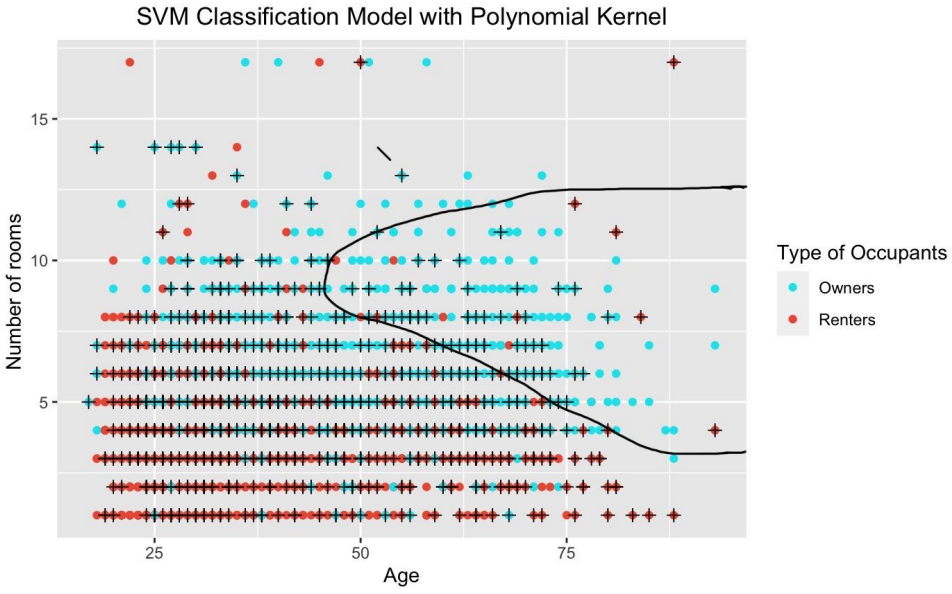


Fig 7: SVM Model with Polynomial Kernel - Hyper plane on Age vs Number of rooms

Discussion

The result of this study shows that the model has predicted the type of occupants with a moderate accuracy using SVM for different kernels. Of the algorithms tested, we found that the model with polynomial kernel with degree three has accuracy of 78% in predicting the type of occupants. The observations for the “renter” class is more than the "owner” class. Thus the model is not trained enough to learn more about the “owner” class. The annual income of the individuals is one of the strong predictors used in determining the type of occupants residing in the house. From the plots we could infer that the individuals with annual income more than $700,000 own a house with more than three rooms in any local area. This analysis reveals that certain socioeconomic groups are less likely to own homes which can be less stable and more expensive. Policymakers could use this information to educate and start counseling programs related to homeownership. This is also done to target affordable housing programs, such as increasing funding for down payment assistance, providing tax incentives for developers to build affordable housing.

Conclusion

This study uses Support Vector Machine to predict if a residence is occupied by owners or tenants based on information retrieved from the U.S census for social, economic and health research IPSUMS data about the occupants and the property. It is identified from the plots that the annual income, density of local population and number of rooms are the strongly correlated variables to classify the type of occupants. The hyperplanes separated the feature space into two classes even though the data points were not well separated. Of the three models used, the tuned hyper parameter model using a polynomial kernel with a degree 3 and cost of 5 was able to predict the “renter” class at 85%, “owner” class at 69% and overall accuracy of 78%. The computational time for the model to train using the polynomial kernel was 40.8 sec. As a future proposition to obtain better results, the model should be trained on the equal number of observations for each class. This analysis could provide policymakers with valuable insights into the factors that influence homeownership, as well as potential strategies for increasing access to safe, stable, and affordable housing for all.

References

James, G., Witten, D., Hastie, T., & Tibshirani, R. (n.d.).An Introduction to Statistical Learning. Retrieved April 26, 2023, from https://www.statlearning.com/

R Notebook

Libraries Required for the Analysis

```
library(tidyverse)
```

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

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.2.1     v dplyr   1.1.2
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

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

```
library(e1071)
library(dplyr)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(ggplot2)
library(tictoc)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

Uploading the dataset

```
load("/Users/kat/Documents/SeattleU/Spring 23/Written Homework-2/Housing.Rdata")
```

Data Cleaning:

```
data = data %>% filter(PERNUM == 1)

#The Response Variable
data $OWNERSHP <- as.factor(data $OWNERSHP)

#The Predictor Variable
data $MARST <- as.factor(data $MARST)
data $NCOUPLES <- as.factor(data $NCOUPLES)
data $NFAMS <- as.factor(data $NFAMS)

#Subsetting the variables
Housing_subset <- data %>% droplevels()%>% select('OWNERSHP','DENSITY','COSTFUEL','COSTWATR','MARST','H

# Data Pre-processing

Housing_subset$COSTFUEL <- replace(Housing_subset$COSTFUEL, Housing_subset$COSTFUEL %in% c(9993,9994,99
Housing_subset$COSTWATR <- replace(Housing_subset$COSTWATR, Housing_subset$COSTWATR %in% c(9993,9995,99

# Filter Income greater than 0

Housing_subset <- Housing_subset%>% filter(Housing_subset$HHINCOME > 0)

# Fitering the people with marital status status as single

Housing_lin_s  <- Housing_subset %>% select('OWNERSHP','DENSITY','COSTFUEL','COSTWATR','MARST','HHINCOM
```

Model 1 : SVM with Linear Kernel

```
set.seed(5)
train <- sample(1:nrow(Housing_lin_s),nrow(Housing_lin_s)*0.7)
train.house <- Housing_lin_s[train,]
test.house <- Housing_lin_s[-train,]

# Linear Kernel
set.seed(1)
tune.out <- tune(svm, OWNERSHP ~., data = train.house, kernel = "linear",
    ranges = list(cost = c(0.0001,0.001,0.01, 0.1, 1, 5, 10,20,30,40,50,100)))

summary(tune.out$best.model)
```

```
##
## Call:
## best.tune(method = svm, train.x = OWNERSHP ~ ., data = train.house,
##     ranges = list(cost = c(1e-04, 0.001, 0.01, 0.1, 1, 5, 10, 20,
```

```
##         30, 40, 50, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  50
##
## Number of Support Vectors:  2112
##
##  ( 1058 1054 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

```
x <- tune.out$best.model

w <- t(x$coefs) %*% x$SV
w
```

```
##        DENSITY    COSTFUEL   COSTWATR MARST1 MARST2 MARST3 MARST4 MARST5
## [1,] 0.2351062 -0.04487531 -0.3473065      0      0      0      0      0
##             MARST6   HHINCOME      ROOMS NCOUPLES1  NCOUPLES2 NCOUPLES3
## [1,] -2.121858e-12 -0.4518164 -0.8887249 0.2888172 -0.5609543         0
##         NFAMS2   NFAMS3   NFAMS4   NFAMS5   NFAMS6   NFAMS7 NFAMS8 NFAMS10
## [1,] 0.4500848 1.310066 2.858907 2.913517 2.007158 4.285946      0       0
##      NFAMS13        AGE   BUILTYR2
## [1,]       0 -0.5008149 0.04152502
```

```
ypred <- predict(tune.out$best.model, test.house)
print(paste("The test error rate of model is ",mean(ypred != test.house$OWNERSHP)*100,"%"))
```

```
## [1] "The test error rate of model is  22.7033352837917 %"
```

```
print(paste("Accuracy of the model for the test dataset: ",mean(ypred == test.house$OWNERSHP)*100,"%"))
```

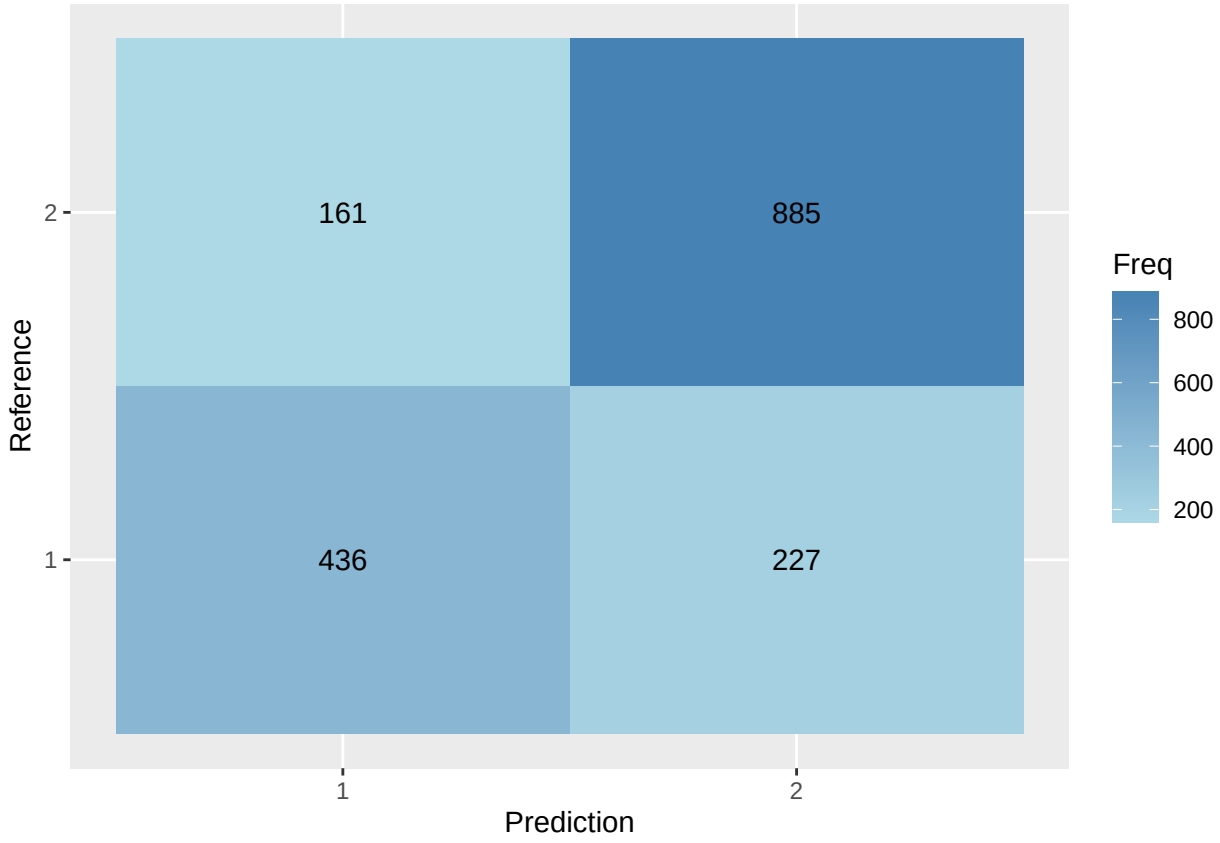
```
## [1] "Accuracy of the model for the test dataset:  77.2966647162083 %"
```

Confusion matrix for the Linear Kernel

```
# Create a sample actual and predicted vector
actual <- test.house$OWNERSHP
predicted <-ypred

# Create a confusion matrix
confusion_matrix <- confusionMatrix(predicted, actual)
```

```
# Plot the confusion matrix
ggplot(data = as.data.frame(confusion_matrix$table),
       aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq)) +
  scale_fill_gradient(low = "lightblue", high = "steelblue")
```



Computational Time

```
tic()
svmfit_l <- svm(OWNERSHP ~ . , data = train.house, kernel = "linear", cost = 50, scale = FALSE)
toc()
```

```
## 44.36 sec elapsed
```

```
w <- t(svmfit_l$coefs) %*% svmfit_l$SV
w
```

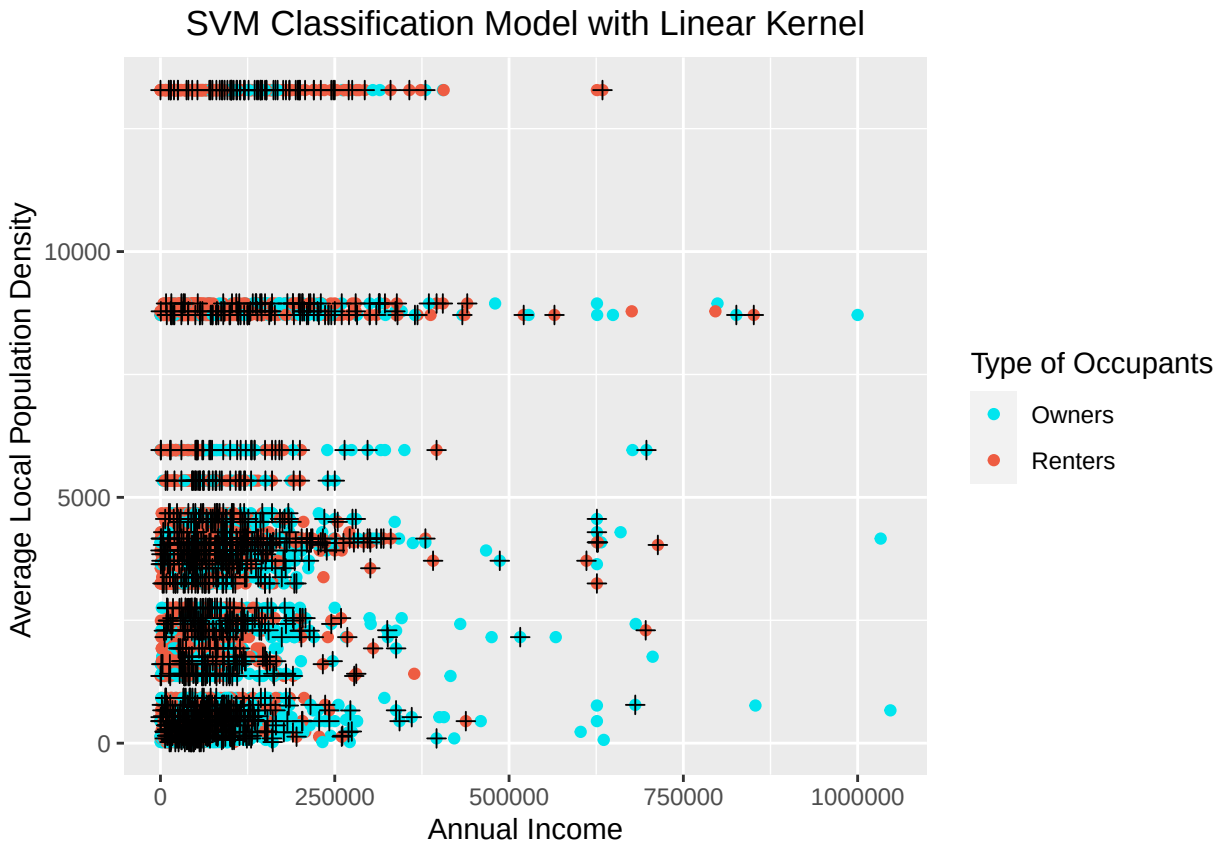
```
##       DENSITY  COSTFUEL  COSTWATR MARST1 MARST2 MARST3 MARST4 MARST5
## [1,] 42.29051 -47.03679 -196.1285      0      0      0      0      0
##            MARST6 HHINCOME  ROOMS NCOUPLES1 NCOUPLES2 NCOUPLES3  NFAMS2
## [1,] 2.273737e-12 6.903841 -36674  1381.551       -50         0 1386.33
##        NFAMS3 NFAMS4 NFAMS5 NFAMS6 NFAMS7 NFAMS8 NFAMS10 NFAMS13       AGE
```

```
## [1,] 896.9457    550      0      0      0      0       0       0 -10701.45
##      BUILTYR2
## [1,] 7592.687
```

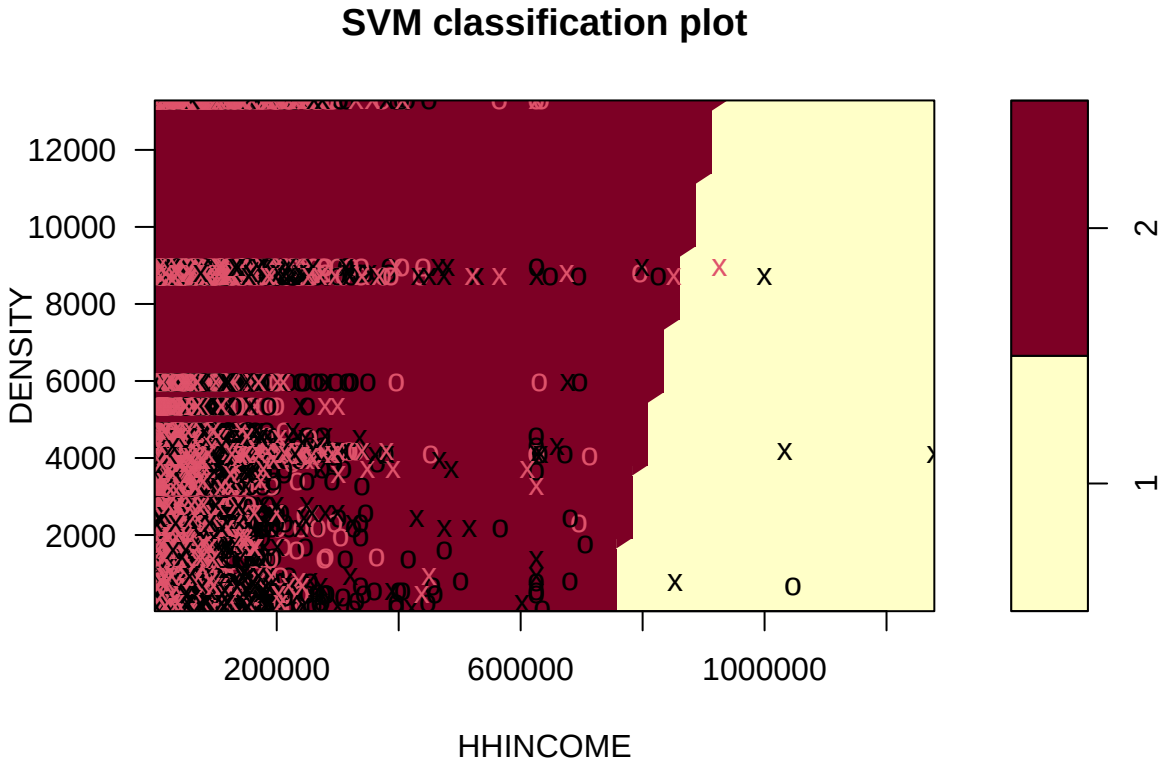
```
# pick the x and y coefficients
bx = w[10] # bill depth
by = w[1] # body mass
b0 = svmfit_l$rho # equivalent to beta_0 in e1071 implementation

# plot
base = ggplot(data=train.house, mapping= aes(x = HHINCOME, y = DENSITY))
base + geom_point(aes(color=OWNERSHP)) +
  geom_point(data=data.frame(svmfit_l$SV), mapping = aes(x = HHINCOME, y = DENSITY), shape=3, size=2)+la
```

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




```
plot(tune.out$best.model,Housing_lin_s,DENSITY ~ HHINCOME )
```



Radial Kernel

```
set.seed(1)
tune.radial <- tune(svm, OWNERSHP ~ ., data = train.house, kernel = "radial",
    ranges = list(cost = c( 1, 10, 100),
      gamma = c(0.5, 1, 2, 3,4)))

summary(tune.radial$best.model)
```

```
##
## Call:
## best.tune(method = svm, train.x = OWNERSHP ~ ., data = train.house,
##     ranges = list(cost = c(1, 10, 100), gamma = c(0.5, 1, 2, 3, 4)),
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  2313
##
##  ( 1150 1163 )
##
##
## Number of Classes:  2
```

```
##
## Levels:
##  1 2
```

```
tune.radial$best.model$gamma
```

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

```
tune.radial$performances$gamma
```

```
##  [1] 0.5 0.5 0.5 1.0 1.0 1.0 2.0 2.0 2.0 3.0 3.0 3.0 4.0 4.0 4.0
```

Confusion matrix for the Radial Kernel

```
ypred <- predict(tune.radial$best.model, test.house)
table(predict = ypred, truth = test.house$OWNERSHP)
```

```
##        truth
## predict   1   2
##       1 463 183
##       2 200 863
```

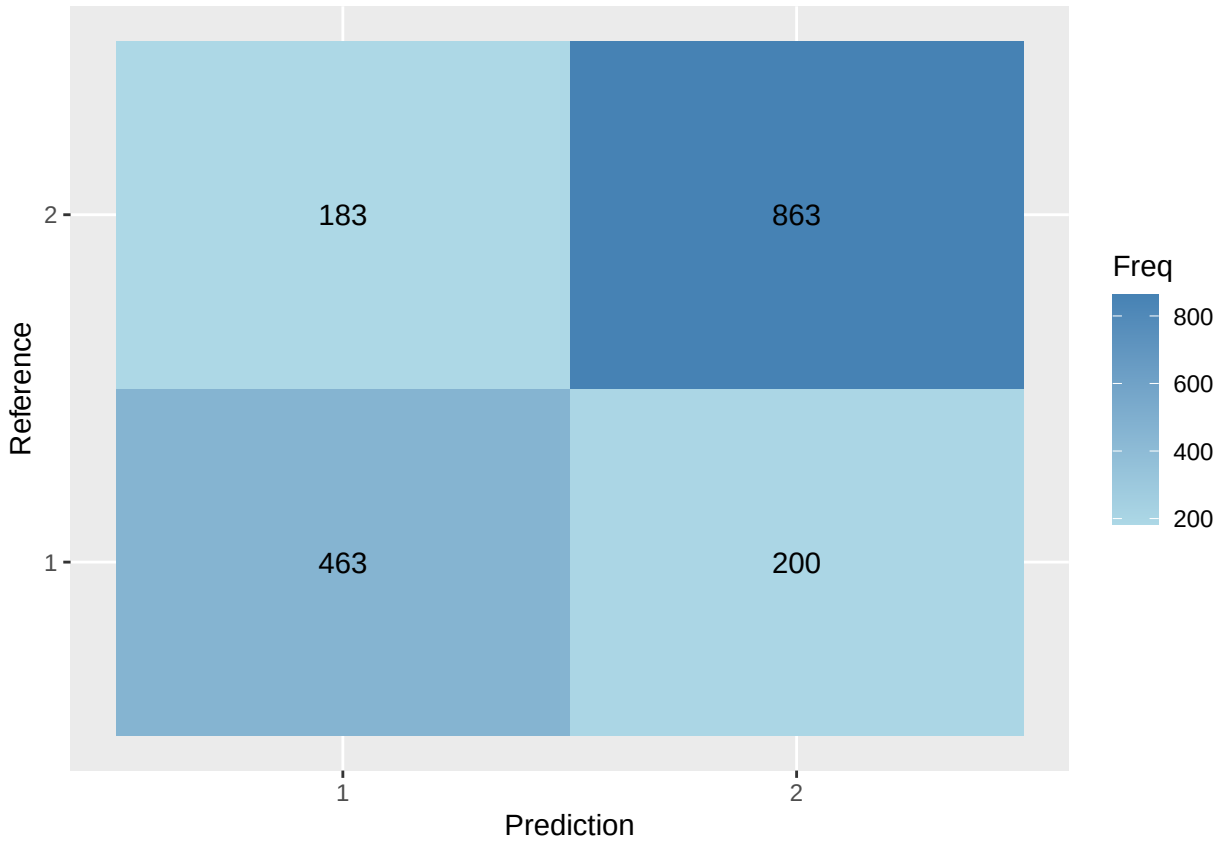
```
print(paste("The training error rate of model is ",mean(ypred != test.house$OWNERSHP)*100,"%"))
```

```
## [1] "The training error rate of model is  22.4107665301346 %"
```

```
# Create a sample actual and predicted vector
actual <- test.house$OWNERSHP
predicted <-ypred

# Create a confusion matrix
confusion_matrix <- confusionMatrix(predicted, actual)

# Plot the confusion matrix
ggplot(data = as.data.frame(confusion_matrix$table),
       aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq)) +
  scale_fill_gradient(low = "lightblue", high = "steelblue")
```




```
tic()
svm_radial <- svm(OWNERSHP ~ . , data = train.house, kernel = "radial", cost = 1,gamma=0.5, scale = FALS
toc()
```

```
## 1.169 sec elapsed
```

```
w1 <- t(svm_radial$coefs) %*% svm_radial$SV
w1
```

```
##      DENSITY COSTFUEL  COSTWATR MARST1 MARST2 MARST3 MARST4 MARST5
## [1,] 1866491 -59043.3 -643577.3      0      0      0      0      0
##             MARST6  HHINCOME     ROOMS NCOUPLES1 NCOUPLES2 NCOUPLES3   NFAMS2
## [1,] -5.002221e-12 -53421020 -3396.958  10.30098 -3.630859          0 31.79565
##        NFAMS3  NFAMS4   NFAMS5    NFAMS6    NFAMS7 NFAMS8 NFAMS10 NFAMS13
## [1,] 18.12053 14.8504 3.791048 -0.631804 0.6841062      0       0      -1
##            AGE BUILTYR2
## [1,] -17584.96 1789.069
```

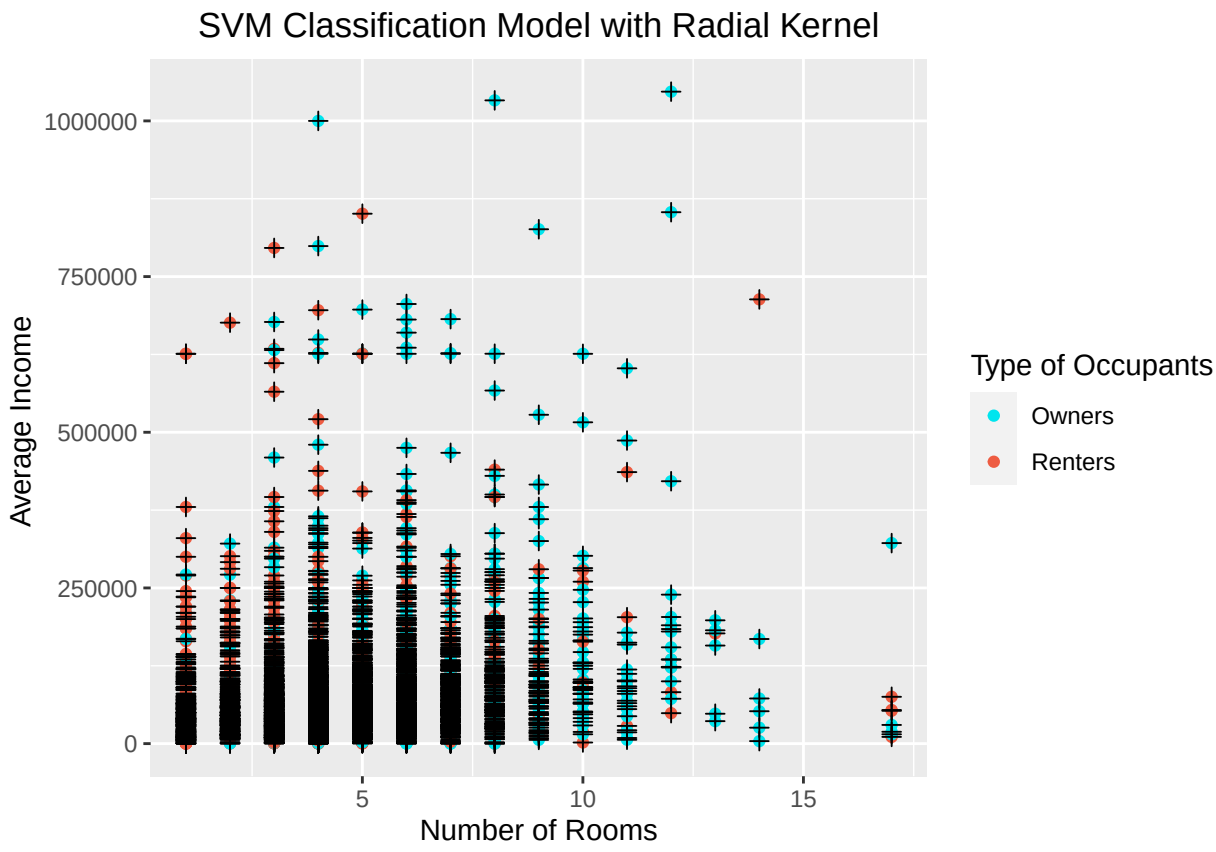
```
# pick the x and y coefficients
bx = w[11] # Rooms
by = w[10] # Income
b0 = svm_radial$rho # equivalent to beta_0 in e1071 implementation

# plot
base = ggplot(data=train.house, mapping= aes(x = ROOMS, y = HHINCOME))
```

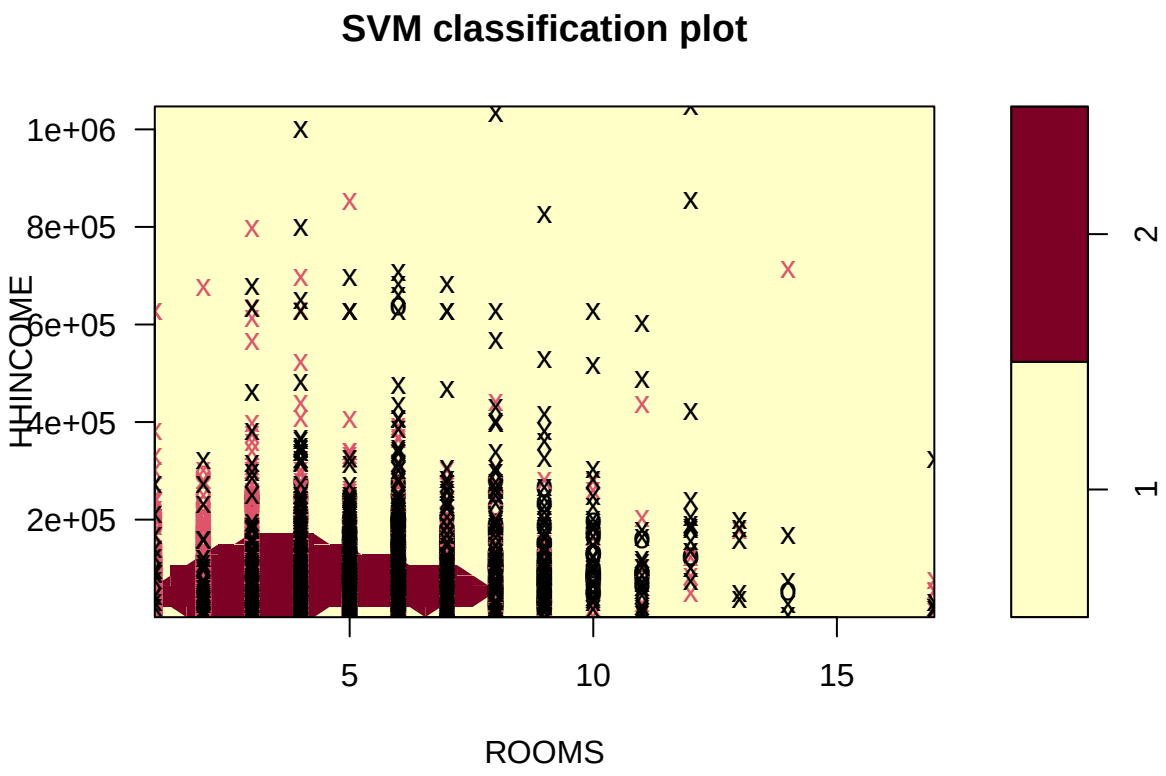
```
base + geom_point(aes(color=OWNERSHP)) +
  geom_point(data=data.frame(svm_radial$SV), mapping = aes(x = ROOMS, y = HHINCOME), shape=3, size=2)+la
```

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

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




```
plot(tune.radial$best.model,train.house,HHINCOME ~ ROOMS )
```



Polynomial Kernel

```
tune.poly <- tune(svm, OWNERSHP ~ ., data = train.house,
    kernel = "polynomial",
    ranges = list(
      degree = c(2,3,4,5,6,7),
      coef0 = c(0,1),
      cost =c(0.01,0.05,0.1,0.5,1,5,10)
    )
  )
tune.poly$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = OWNERSHP ~ ., data = train.house,
##     ranges = list(degree = c(2, 3, 4, 5, 6, 7), coef0 = c(0, 1),
##         cost = c(0.01, 0.05, 0.1, 0.5, 1, 5, 10)), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  5
##      degree:  3
##      coef.0:  1
##
## Number of Support Vectors:  1983
```

```
ypred <- predict(tune.poly$best.model, test.house)
table(predict = ypred, truth = test.house$OWNERSHP)
```

```
##        truth
## predict   1   2
##       1 447 147
##       2 216 899
```

```
print(paste("The test error rate of Polynomial model is ",mean(ypred != test.house$OWNERSHP)*100,"%"))
```

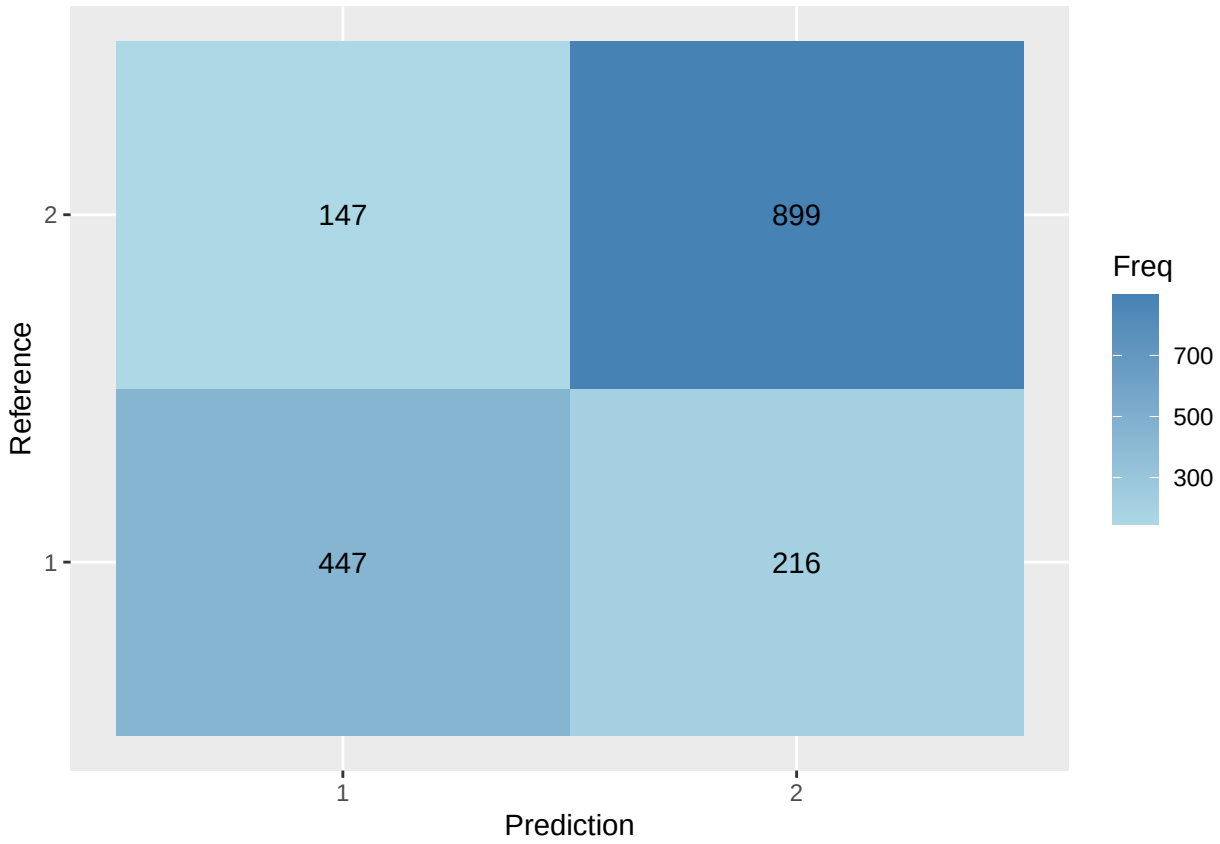
```
## [1] "The test error rate of Polynomial model is  21.2404915155061 %"
```

Confusion matrix for Polunomial Kernel

```
# Create a sample actual and predicted vector
actual <- test.house$OWNERSHP
predicted <-ypred

# Create a confusion matrix
confusion_matrix <- confusionMatrix(predicted, actual)

# Plot the confusion matrix
ggplot(data = as.data.frame(confusion_matrix$table),
       aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq)) +
  scale_fill_gradient(low = "lightblue", high = "steelblue")
```



Computational Time

```
tic()
svm_poly <- svm(OWNERSHP ~ . , data = train.house, kernel = "polynomial", cost = 10,degree=3,coef0= 1,s
toc()
```

```
## 45.56 sec elapsed
```

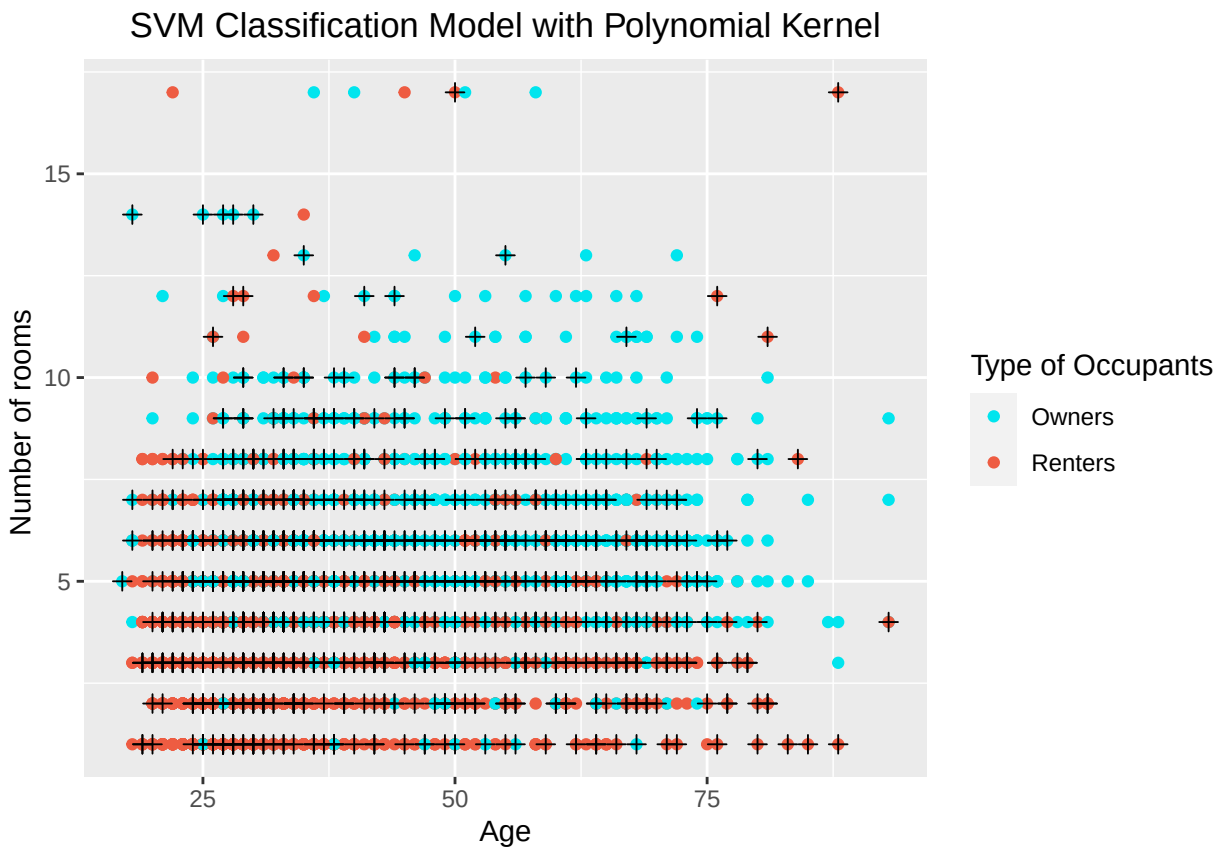
```
w <- t(svm_poly$coefs) %*% svm_poly$SV
w
```

```
##       DENSITY  COSTFUEL COSTWATR MARST1 MARST2 MARST3 MARST4 MARST5
## [1,] -2118115 -149044.1 -37768.9      0      0      0      0      0
##            MARST6  HHINCOME     ROOMS NCOUPLES1  NCOUPLES2 NCOUPLES3   NFAMS2
## [1,] 1.804779e-12 -69244583 -8506.185  16.11879 -0.1386146          0 84.52831
##        NFAMS3 NFAMS4 NFAMS5    NFAMS6 NFAMS7 NFAMS8 NFAMS10 NFAMS13      AGE
## [1,] 47.48072     40     30 -9.912348      0      0       0       0 17530.63
##      BUILTYR2
## [1,] 7219.991
```

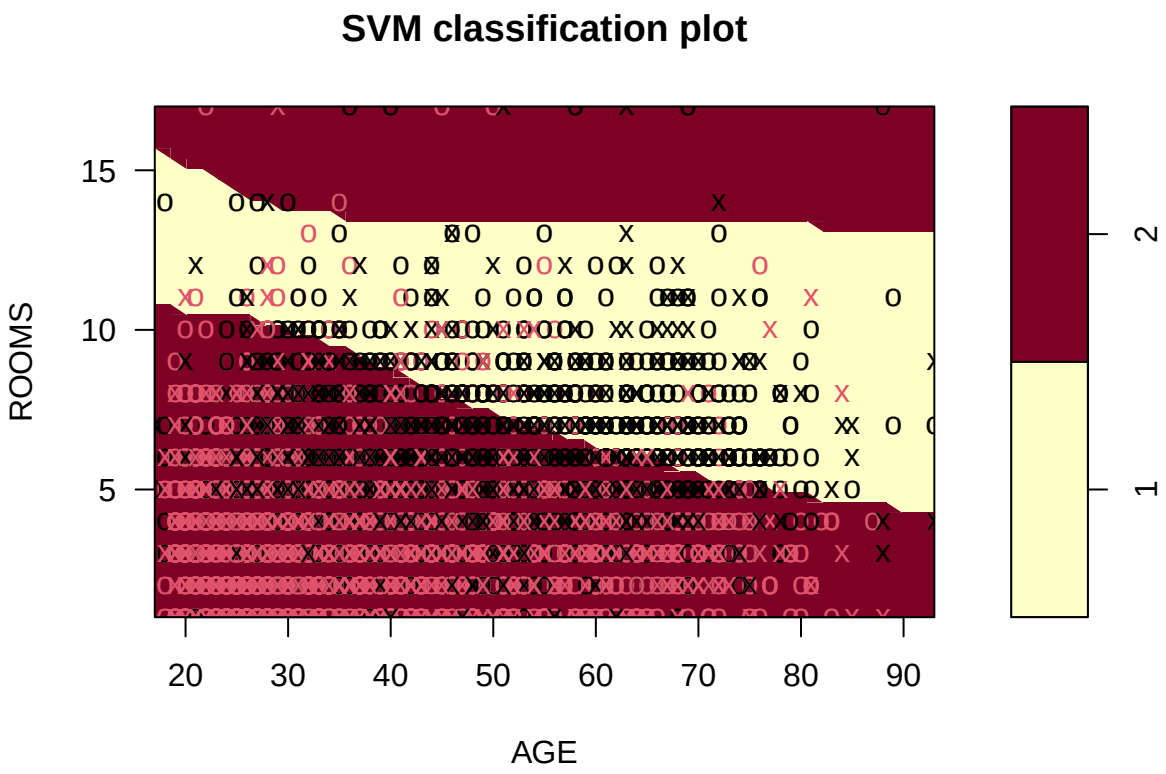
```
# pick the x and y coefficients
bx = w[24] # Rooms
by = w[11] # Income
b0 = svm_poly$rho # equivalent to beta_0 in e1071 implementation
```

```
# plot
base = ggplot(data=train.house, mapping= aes(x = AGE, y = ROOMS))
base + geom_point(aes(color=OWNERSHP)) +
  geom_point(data=data.frame(svm_poly$SV), mapping = aes(x = AGE, y = ROOMS), shape=3, size=2)+labs(tit
```

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




```
plot(tune.poly$best.model,Housing_lin_s,ROOMS ~ AGE )
```


SVM classification plot
ROOMS
15
10
5
20
30
40
50
60
70
80
90
AGE
2
1