How Many People Make Greater Than $50K A Year?

John Peterson

April 9, 2017

## 1. Introduction

The US Census conducted by the US Census Bureau is the leading source for data about the people and economy of the United States. The census collects a wealth of information about the United States. This data can be used to make predictions about income, demographics, education and employment. The goal of this project is to use the census data to develop a prediction model to predict whether a person makes over 50K per year. A machine learning approach will be used to make the prediction. The machine learning technique to be used will be SVM (Support Vector Machine). SVM is particularly good for classification and regression analysis. The goal is to use SVM to develop a prediction model that shows the number of people who made over 50K per year. The model created can be used on larger data sets and different census years data.

## 2. Data Set and Wrangling

The data set used for this project is the Census Income Data Set from the Machine Learning Repository, University of California Irvine: <http://archive.ics.uci.edu/ml/datasets/Census+Income>. Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0)).

One initial observation of the data set was there were no column names. The column names were added when loading the data set into R. The column names are:

## [1] "age" "workclass" "fnlwgt"   
## [4] "education" "education.num" "marital.status"   
## [7] "occupation" "relationship" "race"   
## [10] "sex" "capital.gain" "capital.loss"   
## [13] "hours.per.week" "native.country" "greater.less.than"

The second observation of the data set showed the majority of the data set are categorical values. The focus of the project is on the greater.less.than variable which is a factor but for future analysis it makes sense to make the rest of the data easier to use. The variables workclass, education, marital.status, occupation, relationship, race, sex, native.country do not have number values and will be dummy coded. To perform the dummy coding the Psych package was used (the dummy coding code can be found in the appendix).

The third observation was noticed after creating the initial train and test data sets. The data set is very large it contains 48842 observations. Performing SVM on this large of a data set on a PC was taking hours upon hours to run and that's if it completed. To make it easier on the PC a sample of the original data set was taken; 2000 observations were sampled for this project.

The fourth observation was observed after running SVM on the train data and predicting on the train model. The error table was showing unbalanced and all the data was for <50K. Reviewing the data frame almost all of the prediction column data was <50K. Since a sample was being pulled the data had to be balanced. To balance the data the SMOTE algorithm was utilized by installing the DMwR package. A description of the SMOTE function can be found at <http://amunategui.github.io/smote/>. Using SMOTE to over sample the greater.less.than variable that was unbalanced the data was able to be balanced. This increased our number of observations to 3662 which is acceptable for analysis on the PC.

##   
## <=50K >50K   
## 1842 1820

## 3. Analysis

The complete R code can be found in the appendix of this report and on GitHub([Capstone R Code](https://github.com/jpeterson28/Foundations_of_Data_Science/tree/master/Capstone)). To perform SVM the e1071 package was used.

### 3.1 Subset Data into Train and Test

The balanced data was split into train and test data sets. The train data (trainData) set comprised of 70% of the data and the test data set (testData) contains the remaining 30%.

### 3.2 SVM trainModel Creation

Using the train data set xTrain and yTrain were created. yTrain is the variable we are interested in predicting greater.less.than and the xTrain is the rest of the data minus greater.less.than. Using yTrain and xTrain with SVM we get the trainModel.

**summary**(trainModel)

##   
## Call:  
## svm.default(x = xTrain, y = yTrain, scale = FALSE, type = "C-classification")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 0.009259259   
##   
## Number of Support Vectors: 2439  
##   
## ( 1177 1262 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## <=50K >50K

### 3.3 Prediction

The next step is to run the prediction using the trainModel on the xTest data. The results can be displayed in a table to see how well the model predicts this is also referred to as a confusion matrix.

**table**(pred,yTest)

## yTest  
## pred <=50K >50K  
## <=50K 304 2  
## >50K 239 554

A total of 1099 predictions were made. The model predicted 556 people made over 50k and 543 people made less than 50K. In reality 793 people made over 50K. The accuracy of the model is 78% or we have an error of 22%. The equation for Accuracy is

### 3.4 Tuning the Model

Our accuracy on our initial model was 78%. The model can be improved by adjusting the Cost and Gamma in the SVM. In summary of the initial model it showed a Cost = 1 and Gamma = 0.009259259. By adjusting these values the model can look to be improved. To tune the model SVM will be run with a Cost = 2^seq(-3,5) and Gamma = 2^seq(-3,5). As in the initial model we run tune SVM with the train data.

**summary**(svm\_tune)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 1 0.125  
##   
## - best performance: 0.2309931   
##

The summary shows the best Cost = 1 and Gamma = 0.125.

### 3.5 Model update after tuning

Plugging in the best Cost = 1 and best Gamma = 0.125 into SVM a new model is created.

### 3.6 Prediction Updates

New predictions are calculated for the train and test data.

### 3.7 Validation of tuned model

Using the tuned model on the test data will show the accuracy of the new model.

**table**(testPred, yTest)

## yTest  
## testPred <=50K >50K  
## <=50K 300 1  
## >50K 243 555

## 4. Results and Further Analysis

A total of 1099 predictions were made. The model predicted 556 people made over 50K and 543 people made less than 50K. In reality 798 people made over 50K per year. This resulted in an Accuracy of 78%.

Further analysis can be performed on this data set by: \* Trying different machine learning techniques to try and produce a more accurate result \* Perform analysis using the other variables and how they factor into who makes >$50K a year \* Adjust the Cost and Gamma to run more iterations during tuning

## 5. Citation

Lichman, M. (2013). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science.

## 6. Appendix R-Code

*# Install Packages and Load Libraries*  
**install.packages**('psych')  
**library**(psych)  
**install.packages**('e1071')  
**library**(e1071)  
  
*# Load data set*  
rawData <- **read.table**("/Users/johnpeterson/Desktop/Foundations of Data Science/Capstone/Raw Data/adult.data", sep = ",", col.names = **c**("age", "workclass", "fnlwgt", "education", "education.num", "marital.status", "occupation", "relationship", "race", "sex", "capital.gain", "capital.loss", "hours.per.week", "native.country", "greater.less.than"))  
rawData <- rawData[**sample**(**nrow**(rawData), 2000),]  
  
*# Data Review*  
**dim**(rawData)  
**head**(rawData)  
  
*# Dummy Coding Categorical Values*  
categoricalIndex <- **c**("workclass", "education", "marital.status", "occupation", "relationship", "race", "sex", "native.country")  
  
for(i in 1:**length**(categoricalIndex)){  
 rawData <- **cbind**(rawData,**dummy.code**(rawData[,categoricalIndex[i]]))  
}  
  
*# Dropping categorical index columns*  
rawData <- rawData[, !(**colnames**(rawData) %in% **c**(categoricalIndex))]  
  
*#Smote for dealing with unbalanced data, install DMwR package*  
**install.packages**('DMwR')  
**library**(DMwR)  
*# Call table to see difference between factors (<=50K and >50K)*  
**table**(rawData$greater.less.than)  
rawData <- **SMOTE**(greater.less.than ~ ., rawData, perc.over =300, perc.under = 135) *#Update perc.over and perc.under based on sampling of rawData*  
  
*# Subset Data into Train and Test*  
idxTrain <- **sample**(**seq**(1,**nrow**(rawData)), 0.70\***nrow**(rawData), replace = FALSE)  
idxTest <- **setdiff**(**seq**(1,**nrow**(rawData)), idxTrain)  
trainData <- rawData[idxTrain,]  
testData <- rawData[idxTest,]  
  
*# SVM trainModel Creation*  
xTrain <- **subset**(trainData, select=-greater.less.than)  
yTrain <- **subset**(trainData, select=greater.less.than)  
yTrain <- **as.factor**(**as.character**(yTrain[,1]))  
  
trainModel <- **svm**(xTrain,yTrain, type = "C-classification", scale = FALSE)  
**summary**(trainModel)  
  
xTest <- **subset**(testData, select=-greater.less.than)  
yTest <- **subset**(testData, select=greater.less.than)  
yTest <- **as.factor**(**as.character**(yTest[,1]))  
  
*# trainModel Prediction*  
pred <- **predict**(trainModel, xTest)  
**system.time**(pred <- **predict**(trainModel, xTest))  
   
**table**(pred,yTest)  
  
*# Tune Model*  
svm\_tune <- **tune**(svm, train.x = xTrain, train.y = yTrain,  
 kernel="radial", type = "C-classification", ranges=**list**(cost=2^**seq**(-3,5), gamma=2^**seq**(-3,5)), scale = FALSE)  
  
**print**(svm\_tune)  
  
*# Update the model after tuning*  
trainModel\_after\_tune <- **svm**(xTrain,yTrain, type = "C-classification", kernel = "radial", cost=1, gamma=0.125, scale = FALSE)  
**summary**(trainModel\_after\_tune)  
  
*# Update predictions*  
testPred <- **predict**(trainModel\_after\_tune, xTest)  
trainPred <- **predict**(trainModel\_after\_tune, xTrain)  
  
*# Validation Tables*  
**table**(trainPred, yTrain)  
  
**table**(testPred, yTest)