# ‘Show Me The Money’
# Predicting Income
# via Support Vector Machine

by Mark Healy

# Table of Contents

## Abstract

Binary classification it is a highly useful predictive modelling scenario applicable in diverse disciplines, examples of which include: (i) **Financial:** Credit Risk to (ii) **Marketing:** Customer Relationship Management (CRM) and (iii) **Medical:** Patient Outcome. The aim of this work, was to better understand the theory behind support vector machines (SVM's) and to construct model which will aid classification with a high level of accuracy. The accuracy of the model was then assessed using some of the more popular evaluation methods e.g. confusion matrix, k-folds cross validation and receiver-operator characteristic (ROC) curves. The chosen model was evaluated using both Train and Test data. The latter being an important aspect of machine learning – as it tested on 'unseen' data, thereby providing an indication of how well it would be as a prediction tool. resulting in a Training Error Rate of 15.02%, a (10-Fold) Cross-Validation Error Rate of 16.5% and a Test Error Rate of 16.68%, indicating a model fit for purpose.

**Keywords:** Support vector machine (SVM), radial kernel, k-folds cross validation, train, test, confusion matrix, receiver-operator characteristic (ROC) curve.

## Introduction

Many questions posed are binary i.e. answerable by either a Yes or No, e.g. *"Will this customer default on their loan"*, *"Will this customer buy a certain product"*, *"Will this patient die"*. To answer these questions, binary classification methods are used. More traditional techniques to studying these problems would be logistic regression and classification and regression trees (CART) analysis.

SVM's are a more modern technique, stemming from machine learning. It is a supervised learning model which 'learns' on the train data. Studies conducted have found SVM's to offer better models than logistic regression, as they are less sensitive to outliers[1], able to deal with non-linearity[1] (through the use of kernel functions) and high correlation patterns within the data are less problematic for SVM's than logistic regression models[2].

Using data extracted from the US Census, the aim of this Capstone Project is to build a classification model to answer the following research question: *"Based on a respondent's demographics, do they earn (i) less than or equal to $50k per annum, or (ii) in excess of $50k per annum".*

## Methods

The technique used in this project is a (supervised learning) classification algorithm: Support Vector Machine (SVM). This technique is a discriminative classifier defined by a separating hyperplane. It learns from a portion of the data (training data), and the algorithm outputs an optimal hyperplane which categorises new examples (test data). The particular model in this work (the non-linear case), to segregate individuals into an income level of either ≤ $50k or > $50k, requires an additional aspect – a kernel function.

## Non-Linear Case

In the case of non-linearly separable cases; a kernel function is implemented. Fig 1.1 below illustrates a simple example of a linearly inseparable case. A Kernel function can be applied to this, to non-linearly map the data to a higher dimensional space, where it is then separable, as shown in Fig 1.2 below.

Fig 1.1: Linearly Inseparable

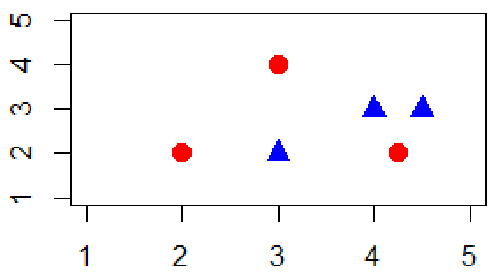
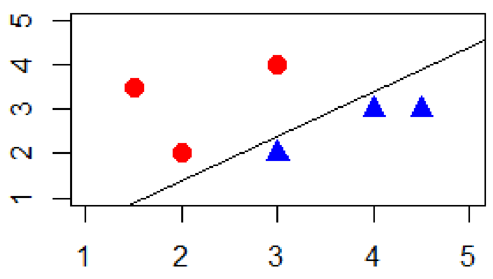


Fig 1.2: Linearly Separable



# Data Overview

## Data Source:

The data is extracted from a census (an official survey of the entire population of a country). This particular extract comes from the 1994 United States Census & is provided courtesy of the UCI repository[3]: https://archive.ics.uci.edu/ml/datasets/Census+Income.

## Data Structure:

The raw dataset contains the information of 32,561 individuals and contains a total of 15 variables. Table 1.1 below provides an overview of the variable types.

| Variable | Variable Name | Type | Levels |
|---|---|---|---|
| 1 | Age | Integer | - |
| 2 | WorkType | Factor | 9 |
| 3 | FnlWgt | Integer | - |
| 4 | Education | Factor | 16 |
| 5 | Education Numerical | Integer | - |
| 6 | Marital Status | Factor | 7 |
| 7 | Occupation | Factor | 15 |
| 8 | Relationship | Factor | 6 |
| 9 | Race | Factor | 5 |
| 10 | Sex | Factor | 2 |
| 11 | Capital Gains Recorded | Integer | - |
| 12 | Capital Losses Recorded | Integer | - |
| 13 | Hours Worked Per Week | Integer | - |
| 14 | Native Country | Factor | 42 |
| 15 | Salary | Factor | 2 |

**Table 1.1:** UCI Census Income Data Structure

## Limitations of the Data

The source of the data does not specify which U.S State it represents. This coupled with the fact that variables such as cost of living is not included, it makes it difficult ascertain what cohort might have a high disposable income. Should one wish to use this data for commercial reasons e.g. to predict mortgage default rates, or those with high disposable incomes, this data lacks the depth in detail to do so.

## Cleaning/Wrangling Required:

### Data Cleansing

*(i)* ***'FnlWgt'*** variable was discarded. This metric was the weighting that a particular respondent was believed to represent. Deemed irrelevant in this study. *[Function used: Null]*

*(ii)* ***'Education'*** (Ordinal) and ***'Education Numerical'*** (Int) both convey the same information. The integer form of this metric was retained *[Function used: Null]*

*(iii)* ***'CapGain'*** (Ordinal), records Capital Gains made. 92% had a value of zero for this *[Function used: Null]*

*(iv)* ***'CapLoss*** (Ordinal), records Capital Losses made. 95% had a value of zero for this *[Function used: Null]*

*(v)* ***'MaritalStatus'*** (Nominal) was recoded into fewer categories, simplifying this factor to three levels: 'Married', 'Not Married' and 'Widowed'.

*(vi)* ***'WorkType'*** *(Nominal)* was recoded. However, it was felt that not too much collapsing of categories could be done here without making assumptions on the terminology. Hence the nine levels were only reduced to seven in this instance. *[Function used: gsub]*

*(vii)* ***'Occupation'*** *(Nominal)* was reduced from a factor with 15 levels to 9 levels. *[Function used: gsub]*

*(viii)* ***'NativeCountry'*** *was a factor with 42 levels. Two options were considered when recoding this: (i) Geographical Region and (ii) World Bank Economic Indicator*[4]*. The latter approach was decided upon, for two reasons: (i) Being an economic metric, it might have more of an interpretable influence in the classification goal of this project and (ii) the values were decided by the World Bank and their research. This resulted in four levels:* (Low-Income, Lower-Middle Income, Upper-Middle Income and High Income).

## Limitations of the Data

Figures 2.1 – 2.3 below provide an overview of the continuous variables in the dataset, with Age being positively skewed (median = 37, SIQR = 10), Education appears to be bi-modal (most likely due to peaks at high-school and college education), while Hours worked appears slightly negatively skewed albeit the large peak at 40 (representing regular-hour full-time work) overshadows this feature of the data (median = 40, SIQR = 2.5).

Fig 2.1: Age

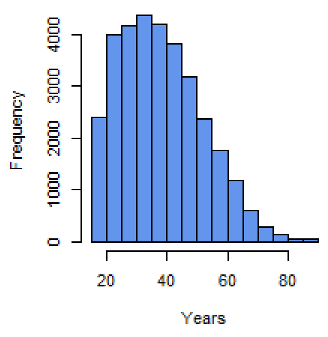


Fig 2.2: Years of Education

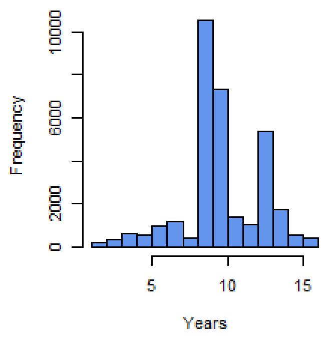


Fig 2.3: Hours worked per Week

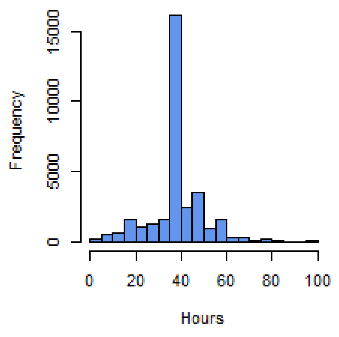


Fig 3.1 – 3.3 below, show the new levels in our recoded categorical variables, simplifying the complexity of the original data.

Fig 3.1: Marital Status

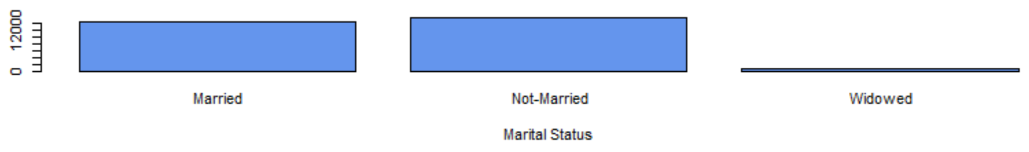


Fig 3.2: Occupation

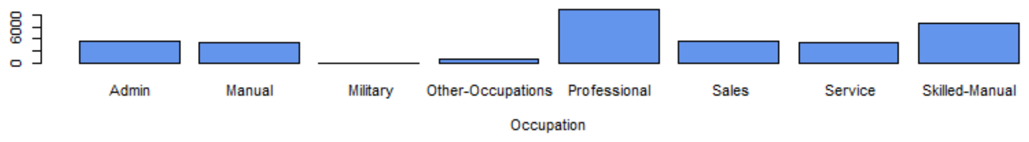
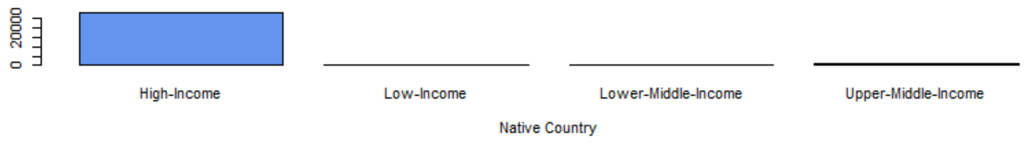


Fig 3.3: Native Country



### Missing Values

A small number of missing values were spotted in the dataset. Once all '?' responses are discarded, the data reduces from 32,561 to 30,091.

## Data Partitioning

The universal approach in machine learning is to split the data into two groupings; Training and Test data. As the aim is to build a predictive model, and not generalise on the given data at hand, the use of a Test dataset allows for a more honest assessment of the predictive ability of the model.

In essence, the data is randomly split into two groupings: Training (70%) and Test data (30%). The model is built using the Training data i.e. it 'learns' from this data and it validated using the Test data. Once the optimal model is decided upon, the accuracy of said model is assessed using 'new' or 'unseen' data i.e. the Test data.

# Analysis

## Base Model

A base non-linear SVM model was fitted to the data, with a starting point of (1,1) for the Cost and Gamma parameters, an interpretation of which, are as follows:

- **Cost (C):** Penalises the misclassification cost (a small C equates to a lower miscslassification cost).
- **Gamma (γ):** This relates to the kernel applied (a small γ results in low bias and high variance).

With a base-model built, the next stage is optimizing the parameters.

## SVM Tuning

A medley of SVM models will be assessed with the best combination for both C and γ determined for the final model. This will be done using the `tune.svm()` function in R. The error rates of each different permutation are provided, from which the user selects the model parameters which result in the lowest error rate. Using these values, the optimal model is coded and the evaluation of the chosen model commences.

Two tuning stages were taken, as the initial process chose the minimum combination of parameters (C = 1, γ = 0.5) to be the optimal combination over the chosen grid. Adjusting the tune command to iterate through a lower range, the optimal combination was determined to be C = 1.1 and γ = 0.3. The chosen SVM model was then ran with these parameters in place.

## Model Evaluation

### K-Folds Cross Validation

K-Folds Cross Validation is a more realistic evaluation technique as opposed to say performing residual analysis – as it gives an indication of how well the model would do when asked to make new predictions.

The training data is divided into k folds. One of these folds is used as the test data, with the other k-1 folds forming the train dataset. This is done k times, which allows each data point to be in the test data once and the training data k-1 times. The biggest advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once. The cross-validation error rate for the chosen model was found to be 16.5%. An outline of this iterative process is shown in Table 1.3 below:

| Iteration | ← Training Data: 10 Folds (K = 10) → | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| K = 1 | Test | Train | Train | Train | Train | Train | Train | Train | Train | Train |
| K = 2 | Train | Test | Train | Train | Train | Train | Train | Train | Train | Train |
| K = 3 | Train | Train | Test | Train | Train | Train | Train | Train | Train | Train |
| K = 4 | Train | Train | Train | Test | Train | Train | Train | Train | Train | Train |
| K = 5 | Train | Train | Train | Train | Test | Train | Train | Train | Train | Train |
| K = 6 | Train | Train | Train | Train | Train | Test | Train | Train | Train | Train |
| K = 7 | Train | Train | Train | Train | Train | Train | Test | Train | Train | Train |
| K = 8 | Train | Train | Train | Train | Train | Train | Train | Test | Train | Train |
| K = 9 | Train | Train | Train | Train | Train | Train | Train | Train | Test | Train |
| K = 10 | Train | Train | Train | Train | Train | Train | Train | Train | Train | Test |

**Table 2.1:** K-Folds Cross Validation Overview (k=10)

## Confusion Matrix

A confusion matrix is a table which allows the reader to interpret the performance of a classification model. For any binary classification model, there are four ways a model can classify an individual as seen in table 2.1 below i.e. True Positives, True Negatives, False Positives and False Negatives. The first two are the correctly classified groupings, with the other two being incorrectly classified.

| | | **Actual** | |
|---|---|---|---|
| | | **Yes** | **No** |
| **Predicted** | **Yes** | True Positive (TP) | False Positive (FP) |
| | **No** | False Negative (FN) | True Negative (TN) |

**Table 2.2:** Confusion Matrix Layout

From this matrix, numerous performance metrics can be derived, as follows:

- **Overall Accuracy**: How often the classification model is correct = $\frac{TP+TN}{N}$
- **Sensitivity**: % of True Positives correctly classified = $\frac{TP}{TP+FN}$
- **Specificity:** % of True Negatives correctly classified = $\frac{TN}{TN+FP}$

- **Overall Error:** How often is the model wrong = $\frac{FP+FN}{N}$
- **False Negative Error Rate:** Predicted as 'no, but are actually 'yes (Type II Error) = $\frac{FN}{TP+FN}$
- **False Positive Error Rate:** Predicted as 'yes', but are actually 'no' (Type I Error) = $\frac{FP}{TN+FP}$

In the next two sections, Tables 2.1.1 and 2.1.2 will contain the confusion matrices for both the Train and Test data respectively, along with some of the key metrics derived from the matrices.

## Training Data

| | | Actual | |
|---|---|---|---|
| | | <=50k | >50k |
| Predicted | <=50k | 14,672 | 2,026 |
| | >50k | 1,138 | 3228 |

**Table 2.2.1:** Training Data Confusion Matrix

Training Accuracy = 84.98%

Training Error = 15.02%

Sensitivity = 92.80%

Specificity = 61.44%

## Test Data

| | | Actual | |
|---|---|---|---|
| | | <=50k | >50k |
| Predicted | <=50k | 6,205 | 924 |
| | >50k | 582 | 1316 |

**Table 2.2.2:** Test Data Confusion Matrix

Test Accuracy = 83.32%

Test Error = 16.68%

Sensitivity = 91.42%

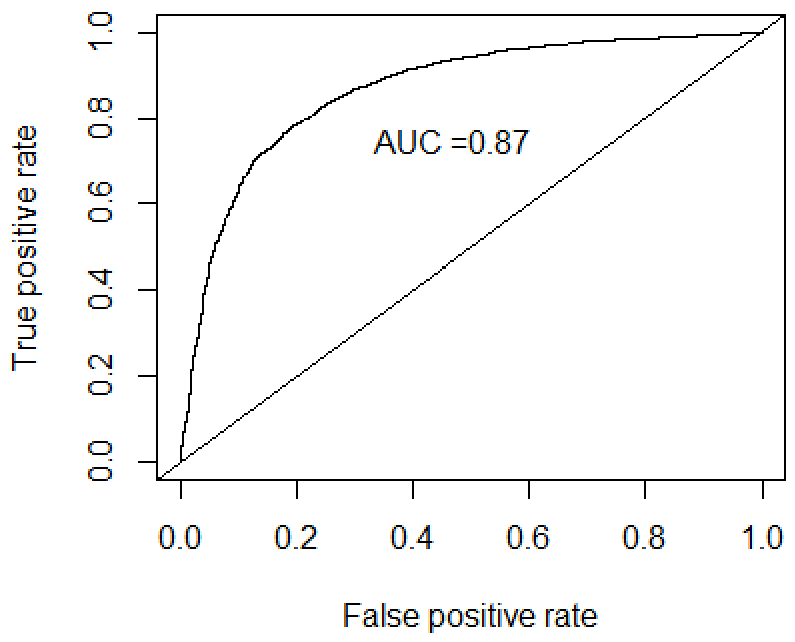
Specificity = 58.75%

## Receiver-Operator Characteristic (ROC) Curve and Area under the Curve (AUC)

A ROC curve is a visual tool used to assess the ability of a classification model. It is obtained by plotting the False Positive Rate (1 – Specificity) along the X-axis against the True Positive Rate (Sensitivity). The resulting graph shows the trade-off between the sensitivity and specificity of the model. In addition to the visual aid, there is an accompanying metric called Area under the Curve; with an AUC = 1 representing the perfect model (visually this curve would hug the left hand border and upper margin).

Fig 4.1 below displays the ROC curve for this model, along with its corresponding AUC value (0.87) indicating good discriminatory ability. In contrast to this, the value of a useless model is 0.5 – and is displayed in Fig 4.1 as the diagonal line. This value of 0.5 can be thought of as that particular model having the equivalent classification ability of flipping a coin.

Fig 4.1: ROC Curve for SVM model



## Conclusions and Future Work

Support Vector Machines have been shown to be yet another effective technique to the classification task, an approach to be added to a data scientist's toolkit, along with more commonly applied techniques such as logistic regression and classification and regression tree (CART) analysis.

The SVM model proved to be a good classifier, as evidenced by the results from the analysis in this project. With many variables being recoded into fewer variables, there appears to be minimal (if any information) lost.

One strength of the SVM approach, is that it allows even a novice user to attain optimal model parameters through the use of the svm.tune() command. This function mitigates the chance of human error in deciding on suitable parameters.

Future work will include producing both a logistic regression model and a CART analysis, and formally compare the deterministic ability of these three approaches.

# Appendix: R-Code

```
##########################################
##########   Set up the Working Directory
##########################################
setwd("C:/Users/Mark's/Desktop/Data Science/CapstoneProject")
getwd()

###############################
##########   Read in the Data
###############################
CensusData = read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data",
                 header=FALSE, sep=",",
                 col.names=c("Age","WorkType","FnlWgt","Education","EducationNum","MaritalStatus",
                       "Occupation", "Relationship", "Race", "Sex","CapGain","CapLoss","HoursPerWeek",
                       "NativeCountry","Salary"))

## Check Dimension of Dataset
dim(CensusData)

## Laod dplyr and create local dataframe
library(dplyr)
CensusDF<-tbl_df(CensusData)

###############################
##########   Data Cleaning
###############################

## Remove Eduacation from Dataset - EducationNum provides same info in integer form
CensusDF$Education<-NULL
## Remove FnlWgt from Dataset - Weighting for how many people this observation represents
CensusDF$FnlWgt<-NULL

## Remove CapGain from Dataset
length(CensusDF$CapGain)
CensusDF$CapGain<-NULL

## Remove CapLoss from Dataset
table(CensusDF$CapLoss)
CensusDF$CapLoss<-NULL

## Check Dimension of Dataset (4 variables less than previously)
dim(CensusDF)
names(CensusDF)
```

```
###################################
### Plot the  data

par(mfrow=c(1,3))
hist(CensusDF$Age, col="cornflowerblue",main="Fig 2.1: Age", xlab="Years")
hist(CensusDF$EducationNum, col="cornflowerblue",main="Fig 2.2: Years of Education", xlab="Years")
hist(CensusDF$HoursPerWeek, col="cornflowerblue",main="Fig 2.3: Hours worked per Week",
xlab="Hours")

median_age<-median(CensusDF$Age) ; siqr_age<-IQR(CensusDF$Age)/2
median_age; siqr_age

median_age<-median(CensusDF$Age) ; siqr_age<-IQR(CensusDF$Age)/2
median_age; siqr_age

median_HoursPerWeek<-median(CensusDF$HoursPerWeek) ; siqr_HoursPerWeek<-
IQR(CensusDF$HoursPerWeek)/2
median_HoursPerWeek; siqr_HoursPerWeek



## Tabulate Categorical Variables to investgate possible groupings
table(CensusDF$WorkType)
table(CensusDF$MaritalStatus)
table(CensusDF$Occupation)
table(CensusDF$Relationship)
table(CensusDF$Race)
table(CensusDF$Sex)
table(CensusDF$NativeCountry)
table(CensusDF$Salary)

##################################################
###### Recode WorkType into fewer categories
##################################################

CensusDF$WorkType = as.character(CensusDF$WorkType)

CensusDF$WorkType = gsub("Private","Private",CensusDF$WorkType)
CensusDF$WorkType = gsub("Self-emp-inc","Self-Employed",CensusDF$WorkType)
CensusDF$WorkType = gsub("Self-emp-not-inc","Self-Employed",CensusDF$WorkType)
CensusDF$WorkType = gsub("Without-pay","Not-Working",CensusDF$WorkType)
CensusDF$WorkType = gsub("Never-worked","Not-Working",CensusDF$WorkType)

CensusDF$WorkType = factor(CensusDF$WorkType)

table(CensusDF$WorkType)
str(CensusDF$WorkType)
```

```
##################################################
###### Recode MaritalStatus into fewer categories
##################################################

CensusDF$MaritalStatus = as.character(CensusDF$MaritalStatus)

CensusDF$MaritalStatus = gsub("Never-married","Not-Married",CensusDF$MaritalStatus)
CensusDF$MaritalStatus = gsub("Married-AF-spouse","Married",CensusDF$MaritalStatus)
CensusDF$MaritalStatus = gsub("Married-civ-spouse","Married",CensusDF$MaritalStatus)
CensusDF$MaritalStatus = gsub("Married-spouse-absent","Not-Married",CensusDF$MaritalStatus)
CensusDF$MaritalStatus = gsub("Separated","Not-Married",CensusDF$MaritalStatus)
CensusDF$MaritalStatus = gsub("Divorced","Not-Married",CensusDF$MaritalStatus)
CensusDF$MaritalStatus = gsub("Widowed","Widowed",CensusDF$MaritalStatus)

CensusDF$MaritalStatus = factor(CensusDF$MaritalStatus)

table(CensusDF$MaritalStatus)
str(CensusDF$MaritalStatus)



##################################################
###### Recode Occupation into fewer categories
##################################################

CensusDF$Occupation = as.character(CensusDF$Occupation)
CensusDF$Occupation <- gsub("?",NA,CensusDF$Occupation, fixed = TRUE)

CensusDF$Occupation = gsub("Adm-clerical","Admin",CensusDF$Occupation)
CensusDF$Occupation = gsub("Armed-Forces","Military",CensusDF$Occupation)
CensusDF$Occupation = gsub("Craft-repair","Skilled-Manual",CensusDF$Occupation)
CensusDF$Occupation = gsub("Exec-managerial","Professional",CensusDF$Occupation)
CensusDF$Occupation = gsub("Farming-fishing","Skilled-Manual",CensusDF$Occupation)
CensusDF$Occupation = gsub("Handlers-cleaners","Manual",CensusDF$Occupation)
CensusDF$Occupation = gsub("Machine-op-inspct","Manual",CensusDF$Occupation)
CensusDF$Occupation = gsub("Other-service","Service",CensusDF$Occupation)
CensusDF$Occupation = gsub("Priv-house-serv","Service",CensusDF$Occupation)
CensusDF$Occupation = gsub("Prof-specialty","Professional",CensusDF$Occupation)
CensusDF$Occupation = gsub("Protective-serv","Other-Occupations",CensusDF$Occupation)
CensusDF$Occupation = gsub("Sales","Sales",CensusDF$Occupation)
CensusDF$Occupation = gsub("Tech-support","Professional",CensusDF$Occupation)
CensusDF$Occupation = gsub("Transport-moving","Skilled-Manual",CensusDF$Occupation)

table(CensusDF$Occupation)
nlevels(CensusDF$Occupation)
str(CensusDF$Occupation)

CensusDF$Occupation = factor(CensusDF$Occupation)
```

```
#################################################
###### Recode NativeCountry into fewer categories
#################################################

CensusDF$NativeCountry = as.character(CensusDF$NativeCountry)
##
http://econ.worldbank.org/WBSITE/EXTERNAL/DATASTATISTICS/0,,contentMDK:20421402~menuPK:64
133156~pagePK:64133150~piPK:64133175~theSitePK:239419,00.html#Low_income

CensusDF$NativeCountry <- gsub("?",NA,CensusDF$NativeCountry, fixed = TRUE)
CensusDF$NativeCountry[CensusDF$NativeCountry==" Cambodia"] = "Low-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Canada"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" China"] = "Lower-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Columbia"] = "Upper-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Cuba"] = "Upper-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Dominican-Republic"] = "Upper-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Ecuador"] = "Lower-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" El-Salvador"] = "Lower-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" England"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" France"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Germany"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Greece"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Guatemala"] = "Lower-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Haiti"] = "Low-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Holand-Netherlands"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Honduras"] = "Lower-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Hong"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Hungary"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" India"] = "Lower-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Iran"] = "Lower-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Ireland"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Italy"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Jamaica"] = "Upper-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Japan"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Laos"] = "Low-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Mexico"] = "Upper-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Nicaragua"] = "Lower-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Outlying-US(Guam-USVI-etc)"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Peru"] = "Upper-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Philippines"] = "Lower-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Poland"] = "Upper-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Portugal"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Puerto-Rico"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Scotland"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" South"] = " ?"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Taiwan"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Thailand"] = "Lower-Middle-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Trinadad&Tobago"] = "High-Income"
```

```
CensusDF$NativeCountry[CensusDF$NativeCountry==" United-States"] = "High-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Vietnam"] = "Low-Income"
CensusDF$NativeCountry[CensusDF$NativeCountry==" Yugoslavia"] = "Upper-Middle-Income"



CensusDF$NativeCountry = factor(CensusDF$NativeCountry)
str(CensusDF$NativeCountry)
table(CensusDF$NativeCountry)


#########################################################################
#########################################################################
## Recoded categorical data plotted

CensusCleaned<-na.omit(CensusDF)
dim(CensusCleaned)

par(mfrow=c(3,1))
counts_MaritalStatus <- table(CensusCleaned$MaritalStatus)
barplot(counts_MaritalStatus, main="Fig 3.1: Marital Status",
        xlab="Marital Status", col="cornflowerblue")

counts_Occupation <- table(CensusCleaned$Occupation)
barplot(counts_Occupation, main="Fig 3.2: Occupation",
        xlab="Occupation", col="cornflowerblue")

counts_NativeCountry <- table(CensusCleaned$NativeCountry)
barplot(counts_NativeCountry, main="Fig 3.3: Native Country",
        xlab="Native Country", col="cornflowerblue")



################################################################################
#### Split into Test and Train Data (30% and 70% respectively)
################################################################################

## Set seed to ensure reproducible results
set.seed(888)
## Sample Indexes
indexes = sample(1:nrow(CensusCleaned), size=0.3*nrow(CensusCleaned))
```

```
# Split the data into Test and Train
Test = CensusCleaned[indexes,]
dim(Test)  # 9027 11
Train = CensusCleaned[-indexes,]
dim(Train) # 21064 11

TestX<-Test %>% select(-Salary)
TestY<-Test %>% select(Salary)

TrainX<-Train %>% select(-Salary)
TrainY<-Train %>% select(Salary)


## Export the Datasets
write.csv(TestX, "TestX.csv")
write.csv(TestY, "TestY.csv")
write.csv(TrainX, "TrainX.csv")
write.csv(TrainY, "TrainY.csv")


library(e1071)
## Build a base SVM on TrainX and TrainY, Predict on TestX
#svmfit=svm(Train$Salary~., data=Train, kernel="radial",  gamma=1, cost=1)
#TestPred<-predict(svmfit, Test[,-11])
#length(TestPred) # 9027

## Build an SVM on TrainX and TrainY, Predict on TrainX
#svmfit=svm(Train$Salary~., data=Train, kernel="radial",  gamma=1, cost=1)
TrainPred<-predict(svmfit, Train[,-11])
length(TrainPred) # 21,064

## Calculate Train Error

## Contingency Tables
#tablePred<-table(pred=TrainPred, true=unlist(TrainY))
#table(pred=TestPred, true=unlist(TestY))

## Confusion Matrices for Train and Test
install.packages("caret")
#caret::confusionMatrix(TrainPred,unlist(TrainY))
#caret::confusionMatrix(TestPred,unlist(TestY))
```

```
########################################################################
#### Use tune.svm to pick optimal values for Cost and Gamma parameters
########################################################################
str(Train)


#obj <- tune.svm(Salary~., data = Train, gamma =seq(0.5, 2, by = 0.5), cost = seq(1,8, by = 2))
#obj
#summary(obj)
#class(obj)

### the smallest combo was best in the above tuning
#### - best performance: 0.164546

###- Detailed performance results:
###       gamma cost     error  dispersion
###   1    0.5    1     0.1645460 0.007409911
### so will edit Cost and Gamma to search a lower bound

#############################################################################
#############################################################################
### Second Tune

### - sampling method: 10-fold cross validation

### - best parameters:
###  gamma cost
###  0.3  1.1

obj <- tune.svm(Salary~., data = Train, gamma =seq(0.1, 0.5, by = 0.1), cost = seq(0.1,1.1, by = .2))
obj
summary(obj)
class(obj)

### Run singular model with optimal parameter values inputted

svm_model_after_tune <- svm(Salary~., data = Train, gamma =0.3, cost = 1.1, probability=TRUE)
summary(svm_model_after_tune)

trainpred <- predict(svm_model_after_tune,TrainX)
#table(trainpred,Train$Salary)

testpred <- predict(svm_model_after_tune,TestX)
#table(testpred,Test$Salary)
```

```
########################################
#Tabulate Test and Train Errors

caret::confusionMatrix(trainpred,unlist(TrainY))
caret::confusionMatrix(testpred,unlist(TestY))

########################################

#Using Test Data here

###########################################
TestSalary<-Test$Salary
TestX ### Variable Salary discarded
table(Test$Salary)

# obsolete i think
##svm.model <- svm(TestSalary~., data = TestX, gamma =0.3, cost = 1.1,probability=TRUE)

### replace below (svm_model_after_tune) with svm.model from above

# obsolete i think
#svm.pred<-predict(svm.model, TestX, decision.values = TRUE,
 #              probability = TRUE) ###### re-run from here

svm.pred<-predict(svm_model_after_tune, TestX, decision.values = TRUE,
              probability = TRUE) ###### re-run from here

library(ROCR)
svm.roc <- prediction(attributes(svm.pred)$decision.values, TestSalary, label.ordering=c(" >50K", "
<=50K"))
svm.auc <- performance(svm.roc, 'tpr', 'fpr')
par(mfrow=c(1,1))
plot(svm.auc, main="Fig 4.1: ROC Curve for SVM model")
text(0.47, 0.75, "AUC =0.87", cex=1)
abline(a=0, b= 1)

summary(svm.auc)

performance(svm.roc, "auc")
```

# References

1. Pochet, N. L. M. M., and J. A. K. Suykens. "Support vector machines versus logistic regression: improving prospective performance in clinical decision-making." *Ultrasound in Obstetrics & Gynecology* 27.6 (2006): 607-608.
2. Salazar, Diego Alejandro, Jorge Iván Vélez, and Juan Carlos Salazar. "Comparison between SVM and logistic regression: Which one is better to discriminate?" *Revista Colombiana de Estadística* 35.2 (2012): 223-237.
3. Excerpt of U.S. Census Data. Retrieved from https://archive.ics.uci.edu/ml/datasets/Census+Income.
4. http://econ.worldbank.org/WBSITE/EXTERNAL/DATASTATISTICS/0,,contentMDK:20421402~menuPK:64133156~pagePK:64133150~piPK:64133175~theSitePK:239419,00.html#Low_income