Classification with SVM

Zachary Canoot [*] Gray Simpson [†]

23 October, 2022

SVM Classification

Support Vector Machines can divide data into classes by a hyperplane in multidimensional space. This line separates classes by finding minimum distance of margins between support vectors. Once we calculate support vectors for our model (given an input of slack in the margins optimized with validation data), we can then classify the data in relation to the margins on the hyperplane.

We are going to apply this classification model to data we have used in the past, census data from 1994, and hope to improve previous results at predicting income class.

Exploring Our Data

As before, the data is stored as two files, with rows just delimited by commas, so we read them in to one whole data frame, and label the headers manual using our source as a reference. It's worth noting that this data was extracted with the intention of creating a classification model, so the two files are meant to be training and test data, but we are going to re-distribute the data sets to train and test later.

Factoring and splitting our data, we can explore the data with a bit more ease. We are going to sample down the data size to 10,000 for shorter compilation times as well.

```
income_train <- read.table("adult.data", sep=",", header=FALSE)
income_test <- read.table("adult.test", sep=",", header=FALSE)
income <- rbind(income_test, income_train)
colnames(income) <- c("Age", "WorkClass", "Weight", "Education", "YearsEdu", "Marital-Status", "Job", "I
# Note here that while sapply returns a vector, lapply returns a list
income[, sapply(income, is.character)] <- lapply(income[, sapply(income, is.character)], as.factor)
levels(income$IncomeClass) <- c("<=50k", "<=50k", ">50k", ">50k")
# Then remove the attribute weight using it's index
set.seed(8)
income <- income[sample(1:nrow(income),10000,replace=FALSE),]
spec <- c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(income), nrow(income)*cumsum(c(0,spec)), labels=names(spec)))
train <- income[i=="train",]
test <- income[i=="test",]
vald <- income[i=="validate",]
# Cleaning up earlier data
rm("income", "income_test", "income_train")
```

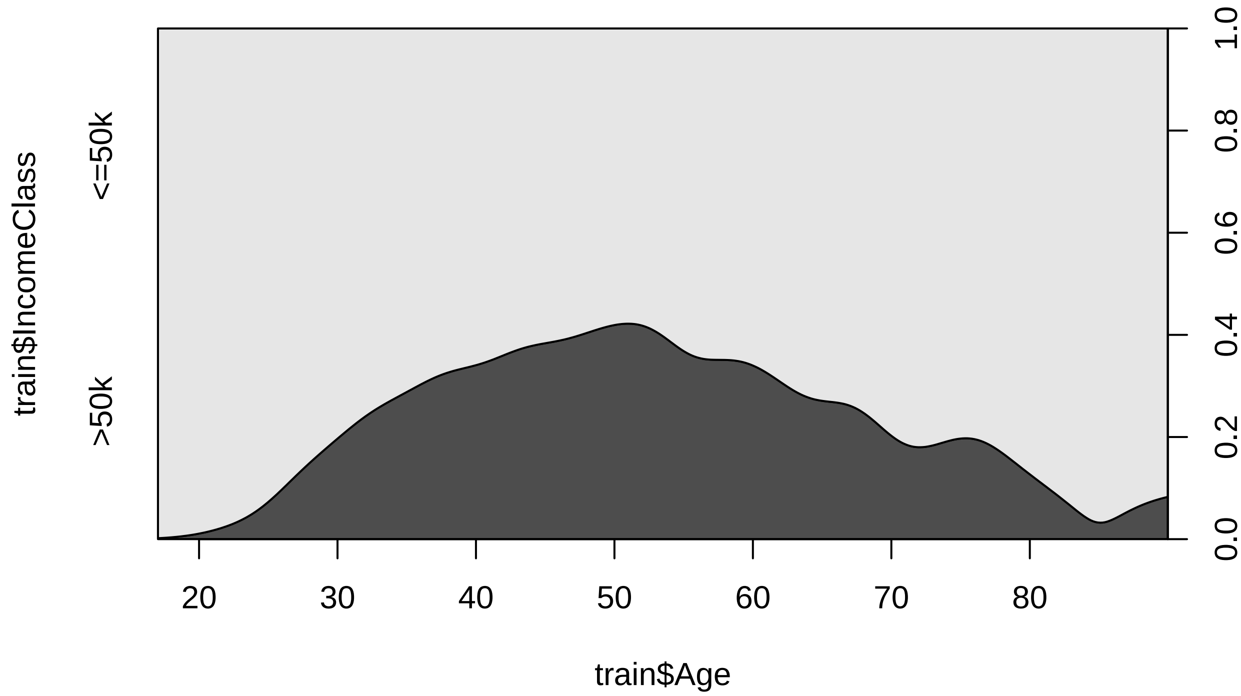
[*] Zaiquiri's Portfolio
[†] Gray's Porfolio

```r
summary(train)
```

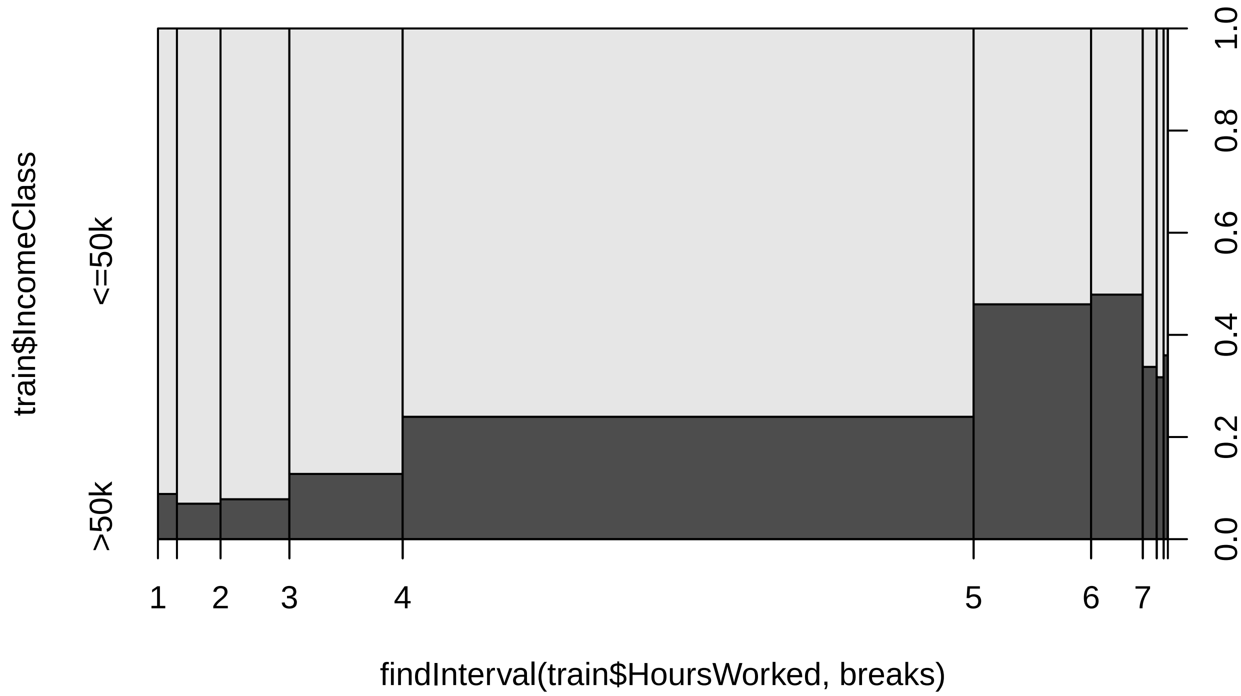
```
##       Age                     WorkClass        Weight
##  Min.   :17.00    Private         :4228   Min.   :  13769
##  1st Qu.:28.00    Self-emp-not-inc: 479   1st Qu.: 119885
##  Median :37.00    Local-gov       : 385   Median : 179607
##  Mean   :38.49    ?               : 325   Mean   : 190855
##  3rd Qu.:48.00    State-gov       : 233   3rd Qu.: 238203
##  Max.   :90.00    Self-emp-inc    : 192   Max.   :1366120
##                  (Other)          : 158
##          Education       YearsEdu                  Marital-Status
##   HS-grad     :1972   Min.   : 1.00    Divorced             : 762
##   Some-college:1300   1st Qu.: 9.00    Married-AF-spouse    :   4
##   Bachelors   :1004   Median :10.00    Married-civ-spouse   :2798
##   Masters     : 305   Mean   :10.03    Married-spouse-absent:  62
##   Assoc-voc   : 251   3rd Qu.:12.00    Never-married        :1990
##   11th        : 248   Max.   :16.00    Separated            : 206
##  (Other)      : 920                    Widowed              : 178
##                Job                Relationship                 Race
##   Exec-managerial: 784    Husband       :2443    Amer-Indian-Eskimo:  55
##   Craft-repair   : 751    Not-in-family :1502    Asian-Pac-Islander: 193
##   Prof-specialty : 739    Other-relative: 179    Black             : 535
##   Sales          : 687    Own-child     : 953    Other             :  45
##   Adm-clerical   : 662    Unmarried     : 615    White             :5172
##   Other-service  : 601    Wife          : 308
##  (Other)         :1776
##       Sex        CapitalGain     CapitalLoss      HoursWorked
##   Female:2004   Min.   :    0   Min.   :   0.0   Min.   : 1.00
##   Male  :3996   1st Qu.:    0   1st Qu.:   0.0   1st Qu.:40.00
##                 Median :    0   Median :   0.0   Median :40.00
##                 Mean   : 1061   Mean   :  78.1   Mean   :40.15
##                 3rd Qu.:    0   3rd Qu.:   0.0   3rd Qu.:45.00
##                 Max.   :99999   Max.   :3004.0   Max.   :99.00
##
##         NativeCountry  IncomeClass
##   United-States:5407   <=50k:4523
##   Mexico       : 114   >50k :1477
##   ?            :  96
##   Philippines  :  39
##   Canada       :  31
##   Puerto-Rico  :  26
##  (Other)       : 287
```

While the data is complex, we can see in the summary that there are of course averages we can determine the average person who recorded census data. He is a man with some high school experience about to enter his 40's, married, and born and raised in the USA. There is some skew in the data, but in the interest of time we'll not dig into stratifying the data right now.

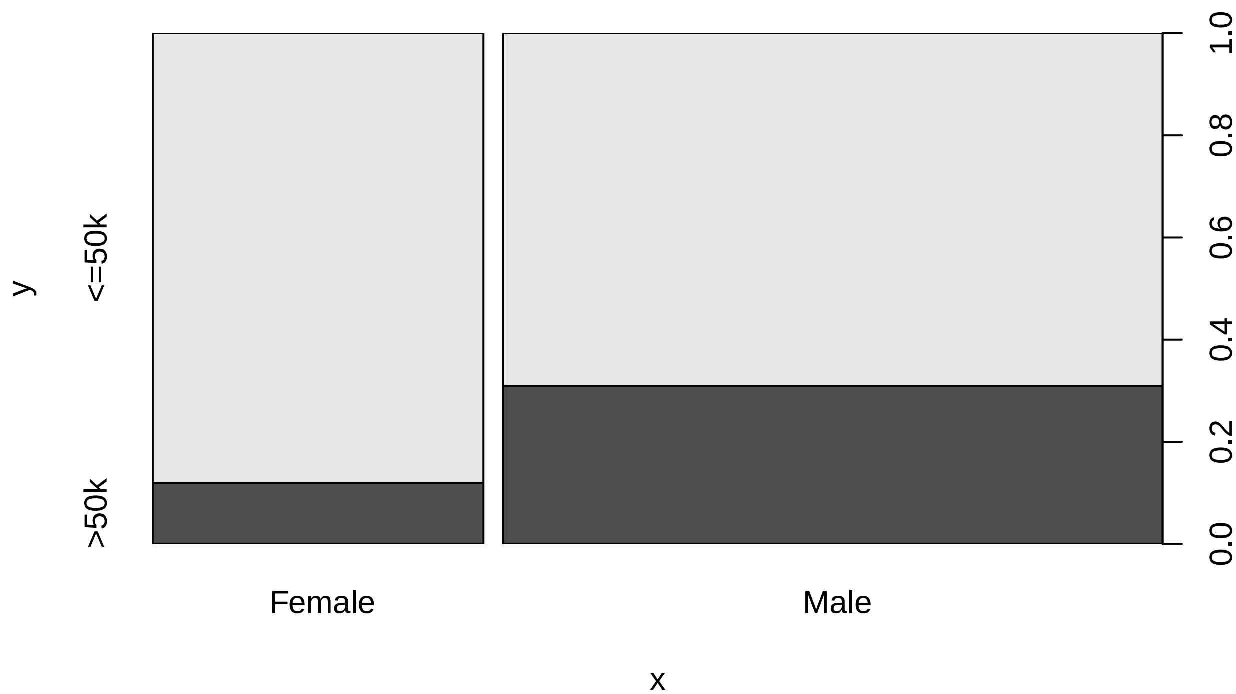
```r
cdplot(train$Age, train$IncomeClass)
```




```r
breaks <- (0:10)*10
plot(train$IncomeClass ~ findInterval(train$HoursWorked, breaks))
```




```
plot(train$Sex, train$IncomeClass)
```



Just as a reminder as well, while ever predictor helps improve the model, some relationships are more clear/obvious:

- Men make more then women!
- The longer you work, the more money you make
- People make the most of their money in their 40's and 50's (if they are making money)

Truly because the data has so many factors, exploring the data doesn't help too well getting the whole picture that our eventual model will produce. At least in our opinion.

Baseline Naive Bayes

We are going to compare our results to Naive Bayes this time for analysis, as we are most interested in the comparison to the performance to the radial kernel (for their ability to handle overlapping data).

```
library(e1071)
nb1 <- naiveBayes(train$IncomeClass~., data=train)
pred1 <- predict(nb1, newdata=test, type="class")
cm <- caret::confusionMatrix(as.factor(pred1), test$IncomeClass)
cm
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction <=50k >50k
##      <=50k  1404  254
##      >50k    113  229
##
##                Accuracy : 0.8165
##                  95% CI : (0.7988, 0.8332)
##     No Information Rate : 0.7585
##     P-Value [Acc > NIR] : 2.539e-10
##
##                   Kappa : 0.4438
##
##  Mcnemar's Test P-Value : 2.713e-13
##
##             Sensitivity : 0.9255
##             Specificity : 0.4741
##          Pos Pred Value : 0.8468
##          Neg Pred Value : 0.6696
##              Prevalence : 0.7585
##          Detection Rate : 0.7020
##    Detection Prevalence : 0.8290
##       Balanced Accuracy : 0.6998
##
##        'Positive' Class : <=50k
##
```

We are trying to beat a baseline accuracy of ~81 percent, and considering the skew in our data, a kappa of ~.44. Reducing our data set to reduce compilation times for SVM did lower our original accuracy from a previous notebook (82 percent), but it will hopefully have returns in our final predictions.

Performing SVM Classification

Linear Kernel

```r
svmlin <- svm(IncomeClass~., data=train, kernel="linear", cost=10, scale=TRUE)
summary(svmlin)
```

```
##
## Call:
## svm(formula = IncomeClass ~ ., data = train, kernel = "linear", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  2009
##
##  ( 1012 997 )
```

```
##
##
## Number of Classes:  2
##
## Levels:
##  <=50k >50k
```

```
pred1 <- predict(svmlin, newdata=test)
cmlin <- caret::confusionMatrix(as.factor(pred1), test$IncomeClass)
cmlin
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50k >50k
##      <=50k  1411  202
##      >50k    106  281
##
##                Accuracy : 0.846
##                  95% CI : (0.8294, 0.8616)
##     No Information Rate : 0.7585
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5491
##
##  Mcnemar's Test P-Value : 6.193e-08
##
##             Sensitivity : 0.9301
##             Specificity : 0.5818
##          Pos Pred Value : 0.8748
##          Neg Pred Value : 0.7261
##              Prevalence : 0.7585
##          Detection Rate : 0.7055
##    Detection Prevalence : 0.8065
##       Balanced Accuracy : 0.7560
##
##        'Positive' Class : <=50k
##
```

We see an increase in accuracy, but lets tune anyway.

```
tune_svmlin <- tune(svm, IncomeClass~., data = vald, kernel="linear", ranges=list(cost=c(0.001, 0.01, 0
tune_svmlin$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = IncomeClass ~ ., data = vald, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
```

```
##        cost:  0.1
##
## Number of Support Vectors:  732
```

It estimates quite a low cost function, which bodes well for an increase in our accuracy

```r
svmlin <- svm(IncomeClass~., data=train, kernel="linear", cost=.1, scale=TRUE)
pred2 <- predict(svmlin, newdata=test)
tuned_cmlin <- caret::confusionMatrix(as.factor(pred2), test$IncomeClass)
tuned_cmlin
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50k >50k
##      <=50k  1412  208
##      >50k    105  275
##
##                Accuracy : 0.8435
##                  95% CI : (0.8268, 0.8592)
##     No Information Rate : 0.7585
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5393
##
##  Mcnemar's Test P-Value : 8.147e-09
##
##             Sensitivity : 0.9308
##             Specificity : 0.5694
##          Pos Pred Value : 0.8716
##          Neg Pred Value : 0.7237
##              Prevalence : 0.7585
##          Detection Rate : 0.7060
##    Detection Prevalence : 0.8100
##       Balanced Accuracy : 0.7501
##
##        'Positive' Class : <=50k
##
```

There was an increase with a bit of tuning. Still hopeful for better results.

Polynomial Kernel

```r
svmpoly <- svm(IncomeClass~., data=train, kernel="polynomial", cost=.1, scale=TRUE)
summary(svmpoly)
```

```
##
## Call:
## svm(formula = IncomeClass ~ ., data = train, kernel = "polynomial",
##     cost = 0.1, scale = TRUE)
##
```

```
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.1
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  2965
##
##  ( 1511 1454 )
##
##
## Number of Classes:  2
##
## Levels:
##  <=50k >50k
```

```r
pred3 <- predict(svmpoly, newdata=test)
cmpoly <- caret::confusionMatrix(as.factor(pred3), test$IncomeClass)
cmpoly
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50k >50k
##      <=50k  1517  474
##      >50k      0    9
##
##                Accuracy : 0.763
##                  95% CI : (0.7437, 0.7815)
##     No Information Rate : 0.7585
##     P-Value [Acc > NIR] : 0.3298
##
##                   Kappa : 0.028
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.00000
##             Specificity : 0.01863
##          Pos Pred Value : 0.76193
##          Neg Pred Value : 1.00000
##              Prevalence : 0.75850
##          Detection Rate : 0.75850
##    Detection Prevalence : 0.99550
##       Balanced Accuracy : 0.50932
##
##        'Positive' Class : <=50k
##
```

Well… we didn't expect a radically low kappa but that was because the default degree value is quite extreme, lets tune

```r
tune_svmpoly <- tune(svm, IncomeClass~., data = vald, kernel="polynomial", ranges=list(cost=c(0.001, 0.0
tune_svmpoly$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = IncomeClass ~ ., data = vald, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100), degree = c(1, 2, 3)), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  1
##      coef.0:  0
##
## Number of Support Vectors:  733
```

It found a linear result, but it did raise the cost from our previous linear test which is quite interesting

```r
svmpoly <- svm(IncomeClass~., data=train, kernel="polynomial", cost=10, degree=1, scale=TRUE)
pred4 <- predict(svmpoly, newdata=test)
tuned_cmpoly <- caret::confusionMatrix(as.factor(pred4), test$IncomeClass)
tuned_cmpoly
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50k >50k
##      <=50k  1413  209
##      >50k    104  274
##
##                Accuracy : 0.8435
##                  95% CI : (0.8268, 0.8592)
##     No Information Rate : 0.7585
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5386
##
##  Mcnemar's Test P-Value : 4.142e-09
##
##             Sensitivity : 0.9314
##             Specificity : 0.5673
##          Pos Pred Value : 0.8711
##          Neg Pred Value : 0.7249
##              Prevalence : 0.7585
##          Detection Rate : 0.7065
##    Detection Prevalence : 0.8110
##       Balanced Accuracy : 0.7494
##
##        'Positive' Class : <=50k
##
```

This is a solid result, with really a statistically insignificant result compared to our other models. Lets test a Radial Kernel!

Radial Kernel

```
svmrad <- svm(IncomeClass~., data=train, kernel="radial", cost=.1, gamma=1, scale=TRUE)
summary(svmrad)
```

```
##
## Call:
## svm(formula = IncomeClass ~ ., data = train, kernel = "radial", cost = 0.1,
##     gamma = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.1
##
## Number of Support Vectors:  5360
##
##  ( 3883 1477 )
##
##
## Number of Classes:  2
##
## Levels:
##  <=50k >50k
```

```
pred5 <- predict(svmrad, newdata=test)
cmrad <- caret::confusionMatrix(as.factor(pred5), test$IncomeClass)
cmrad
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50k >50k
##      <=50k  1514  471
##      >50k      3   12
##
##                Accuracy : 0.763
##                  95% CI : (0.7437, 0.7815)
##     No Information Rate : 0.7585
##     P-Value [Acc > NIR] : 0.3298
##
##                   Kappa : 0.0341
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99802
##             Specificity : 0.02484
```

```
##          Pos Pred Value : 0.76272
##          Neg Pred Value : 0.80000
##              Prevalence : 0.75850
##          Detection Rate : 0.75700
##    Detection Prevalence : 0.99250
##       Balanced Accuracy : 0.51143
##
##        'Positive' Class : <=50k
##
```

Well... we didn't expect a radically low kappa but that was because the default degree value is quite extreme, lets tune

```
tune_svmrad <- tune(svm, IncomeClass~., data = vald, kernel="radial", ranges=list(cost=c(0.001, 0.01, 0
tune_svmrad$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = IncomeClass ~ ., data = vald, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1), gamma = c(0.1, 0.5, 1)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  926
```

Inputting the tuned parameters into the radial model one final time:

```
svmrad <- svm(IncomeClass~., data=train, kernel="radial", cost=1, gamma=.1, scale=TRUE)
pred6 <- predict(svmrad, newdata=test)
tuned_cmrad <- caret::confusionMatrix(as.factor(pred6), test$IncomeClass)
tuned_cmrad
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <=50k >50k
##      <=50k  1429  209
##      >50k     88  274
##
##                Accuracy : 0.8515
##                  95% CI : (0.8352, 0.8668)
##     No Information Rate : 0.7585
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5568
##
##  Mcnemar's Test P-Value : 3.329e-12
##
```

```
##             Sensitivity : 0.9420
##             Specificity : 0.5673
##          Pos Pred Value : 0.8724
##          Neg Pred Value : 0.7569
##              Prevalence : 0.7585
##          Detection Rate : 0.7145
##    Detection Prevalence : 0.8190
##       Balanced Accuracy : 0.7546
##
##        'Positive' Class : <=50k
##
```

That is only marginally better then our Naive Bayes base line result

Analysis

Briefly describing the kernels:

- The linear kernel is simple, it fits a hyperplane to the data
- The polynomial kernel transforms the data in such a way to mimic adding more features to the data set, really just by mapping the input data to a polynomial of a higher degree. By mapping values in a higher degree space, say, to the second degree, what really is a circular data set classification can now have a straight line drawn through it.
- The radial kernel compares the distance between every 2 values in the input data, and scales the data by the value of it's distance. This mimics nearest neighbor, where the model predicts every value with increasing weight supplied to its neighbors. The kernel can then map the input to a higher (infinite) dimensional space where it is easiest to fit a hyperplane that best maximizes the margins of the model… it's not exactly easy to wrap a brain around

For all three of these kernels we got increasingly better results, slowly growing more accurate then our last attempts to fit the data to a model. This could be a result of really the complexity of the data, and how hard it is to truly predict something like someone's income bracket based on a snapshot of their socioeconomic status. While it is almost always the case that SVM is better than Naive Bayes, perhaps we were hitting the upper bound of what we could predict, meaning only a 3% increase in accuracy

By any case, 3% more accuracy could be a meaningful increase based on what the model is used for. We really should be aiming for 99% accuracy though. This would perhaps require trimming the data, or running some ensemble methods!