Support Vector Machines(SVMs) Tutorial

Sonali Narang

11/12/2019

Support Vector Machines(SVMs)

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. Given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples.

The Breast Cancer Dataset

699 Observations, 11 variables Predictor Variable: Class–benign or malignant

```
data(BreastCancer)

#bc = BreastCancer %>%
#  mutate_if(is.character, as.numeric)
#bc[is.na(bc)] = 0

BreastCancer_num = transform(BreastCancer, Id = as.numeric(Id),
                         Cl.thickness = as.numeric(Cl.thickness),
                         Cell.size = as.numeric(Cell.size),
                         Cell.shape = as.numeric(Cell.shape),
                         Marg.adhesion = as.numeric(Marg.adhesion),
                         Epith.c.size = as.numeric(Epith.c.size),
                         Bare.nuclei = as.numeric(Bare.nuclei),
                         Bl.cromatin = as.numeric(Bl.cromatin),
                         Normal.nucleoli = as.numeric(Normal.nucleoli),
                         Mitoses = as.numeric(Mitoses))

BreastCancer_num[is.na(BreastCancer_num)] = 0

train_size = floor(0.75 * nrow(BreastCancer_num))
train_pos <- sample(seq_len(nrow(BreastCancer_num)), size = train_size)

train_classification <- BreastCancer_num[train_pos, ]
test_classification <- BreastCancer_num[-train_pos, ]
```

##SVM

```
set.seed(1112)
control = trainControl(method = "repeatedcv", repeats = 5, classProbs = T, savePredictions = T)

svm = train(Class ~ Id + Cl.thickness + Cell.size + Cell.shape + Marg.adhesion + Epith.c.size + Bare.nu

svm
```

```
## Support Vector Machines with Linear Kernel
##
## 524 samples
##  10 predictor
##   2 classes: 'benign', 'malignant'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 473, 471, 471, 472, 472, 471, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9640831  0.9178939
##
## Tuning parameter 'C' was held constant at a value of 1
```

##Receiver operating characteristic(ROC) curve

```
roc(predictor = svm$pred$malignant, response = svm$pred$obs)$auc
```

```
## Setting levels: control = benign, case = malignant
```

```
## Setting direction: controls < cases
```

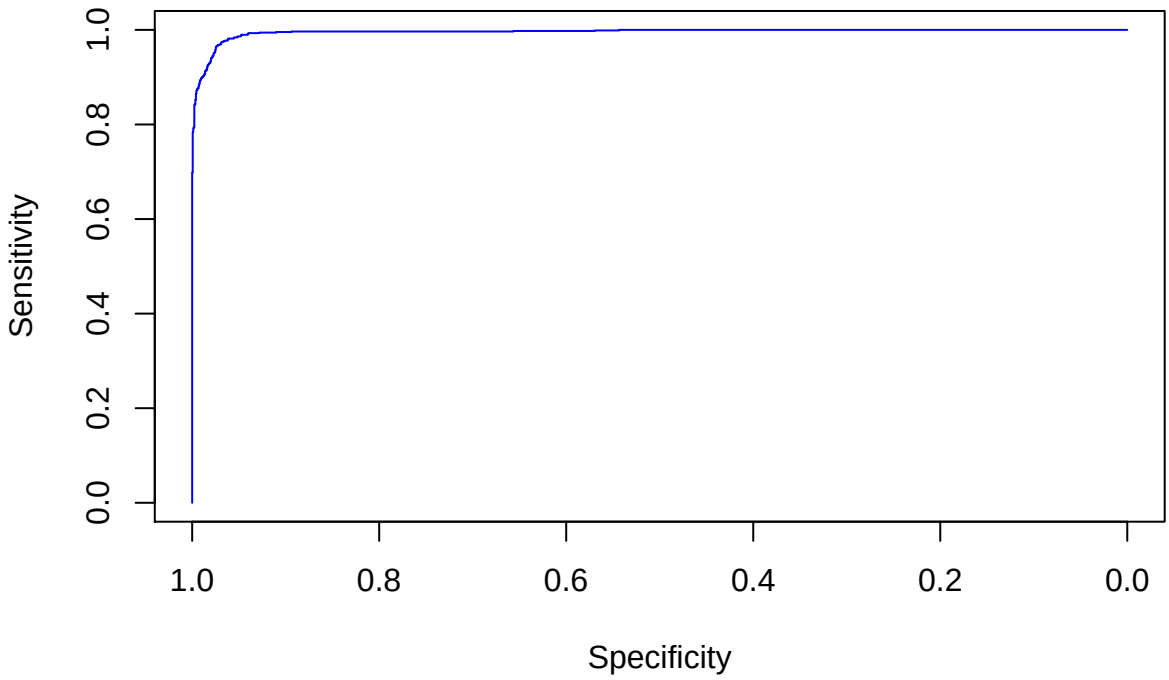
```
## Area under the curve: 0.9954
```

```
plot(x = roc(predictor = svm$pred$malignant, response = svm$pred$obs)$specificities, y = roc(predictor =
```

```
## Setting levels: control = benign, case = malignant
## Setting direction: controls < cases
```

```
## Setting levels: control = benign, case = malignant
```

```
## Setting direction: controls < cases
```



##

Test Set

```
svm_test = predict(svm, newdata = test_classification)
confusionMatrix(svm_test, reference = test_classification$Class)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
```

```
##   benign       103         2
##   malignant      6        64
##
##                Accuracy : 0.9543
##                  95% CI : (0.9119, 0.9801)
##     No Information Rate : 0.6229
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9038
##
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.9450
##             Specificity : 0.9697
##          Pos Pred Value : 0.9810
##          Neg Pred Value : 0.9143
##              Prevalence : 0.6229
##          Detection Rate : 0.5886
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.9573
##
##        'Positive' Class : benign
##
```

SVM with a radial kernel

```
set.seed(1112)
control = trainControl(method = "repeatedcv", repeats = 5, classProbs = T, savePredictions = T)

svm = train(Class ~ Id + Cl.thickness + Cell.size + Cell.shape + Marg.adhesion + Epith.c.size + Bare.nu

svm
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 524 samples
##  10 predictor
##   2 classes: 'benign', 'malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 473, 471, 471, 472, 472, 471, ...
## Resampling results across tuning parameters:
##
##   C       Accuracy   Kappa
##     0.25  0.9496558  0.8900864
##     0.50  0.9504105  0.8914669
##     1.00  0.9542793  0.8994983
##     2.00  0.9511800  0.8920917
##     4.00  0.9511731  0.8920753
##     8.00  0.9496561  0.8887717
##    16.00  0.9511800  0.8921669
##    32.00  0.9500337  0.8893805
##    64.00  0.9511948  0.8920506
```

```
##   128.00  0.9511728  0.8920443
##
## Tuning parameter 'sigma' was held constant at a value of 0.5467371
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.5467371 and C = 1.
```

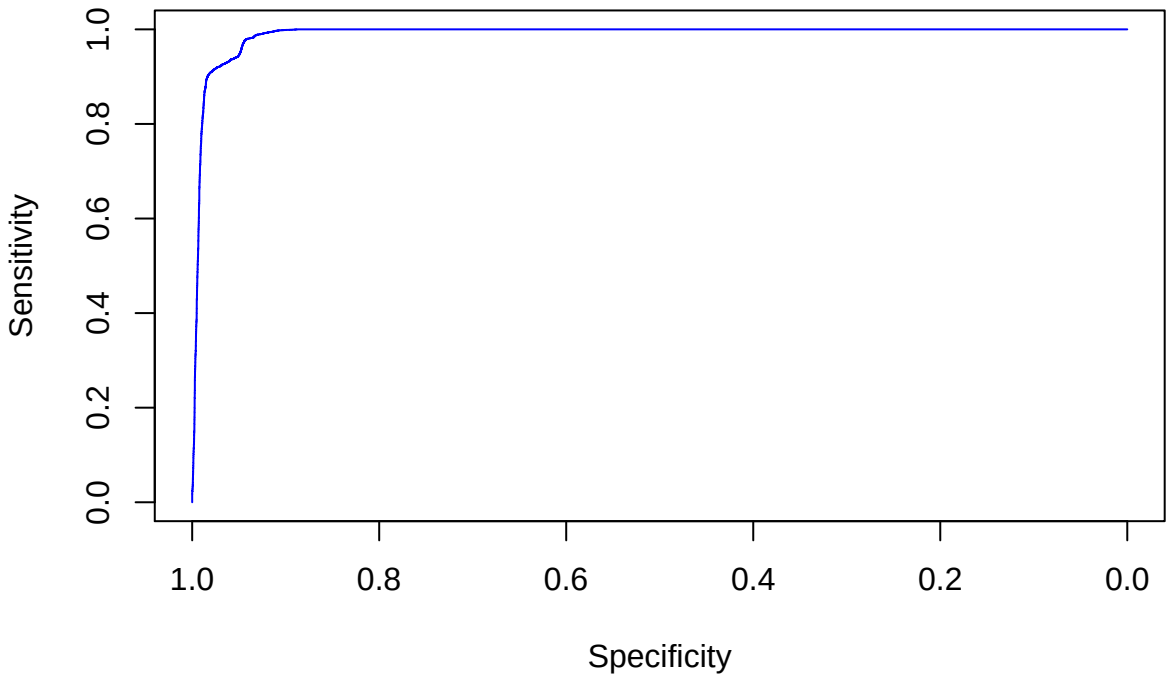
##Receiver operating characteristic(ROC) curve

```
roc(predictor = svm$pred$malignant, response = svm$pred$obs)$auc
```

```
## Setting levels: control = benign, case = malignant
## Setting direction: controls < cases
## Area under the curve: 0.99
```

```
plot(x = roc(predictor = svm$pred$malignant, response = svm$pred$obs)$specificities, y = roc(predictor =
```

```
## Setting levels: control = benign, case = malignant
## Setting direction: controls < cases
## Setting levels: control = benign, case = malignant
## Setting direction: controls < cases
```



Test Set

```
svm_test = predict(svm, newdata = test_classification)
confusionMatrix(svm_test, reference = test_classification$Class)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       103         1
##   malignant      6        65
```

```
##
##                Accuracy : 0.96
##                  95% CI : (0.9193, 0.9838)
##     No Information Rate : 0.6229
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9161
##
##  Mcnemar's Test P-Value : 0.1306
##
##             Sensitivity : 0.9450
##             Specificity : 0.9848
##          Pos Pred Value : 0.9904
##          Neg Pred Value : 0.9155
##              Prevalence : 0.6229
##          Detection Rate : 0.5886
##    Detection Prevalence : 0.5943
##       Balanced Accuracy : 0.9649
##
##        'Positive' Class : benign
##
```

##Homework

1. Choose an appropriate machine learning dataset and use SVM with two different kernels. Campare the results.

The Pima Indians Diabetes dataset is a decent dataset on which to try SVM methods. There are a total of 8 predictor variables for the binary outcome of diabetic status being positive or negative.

There are a total of 768 observations.

```
##load in the data
data("PimaIndiansDiabetes")

##clean the data to make values numeric and remove any possible missing
pima = PimaIndiansDiabetes %>%
  mutate_if(is.character, as.numeric)
pima[is.na(pima)] = 0

##set train and test sets
train_size = floor(0.75 * nrow(pima))
train_pos <- sample(seq_len(nrow(pima)), size = train_size)

train_classification <- pima[train_pos, ]
test_classification <- pima[-train_pos, ]
```

Once the data is prepped, the SVM with a linear kernal can be built and visualized.

```
##set seed to make reproduceable
set.seed(1112)
##define control
control = trainControl(method = "repeatedcv", repeats = 5, classProbs = T, savePredictions = T)

##create SVM Linear model
svm = train(diabetes ~ .,  data = train_classification, method = "svmLinear", tuneLength = 10, trControl
```

```
##view SVM model
svm
```

```
## Support Vector Machines with Linear Kernel
##
## 576 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 519, 519, 519, 519, 517, 518, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.789732  0.4968437
##
## Tuning parameter 'C' was held constant at a value of 1
```

Area under the ROC shows decent model performance of 0.8388.

```
##calculate AUC ROC
roc(predictor = svm$pred$pos, response = svm$pred$obs)$auc
```

```
## Setting levels: control = neg, case = pos
```

```
## Setting direction: controls < cases
```

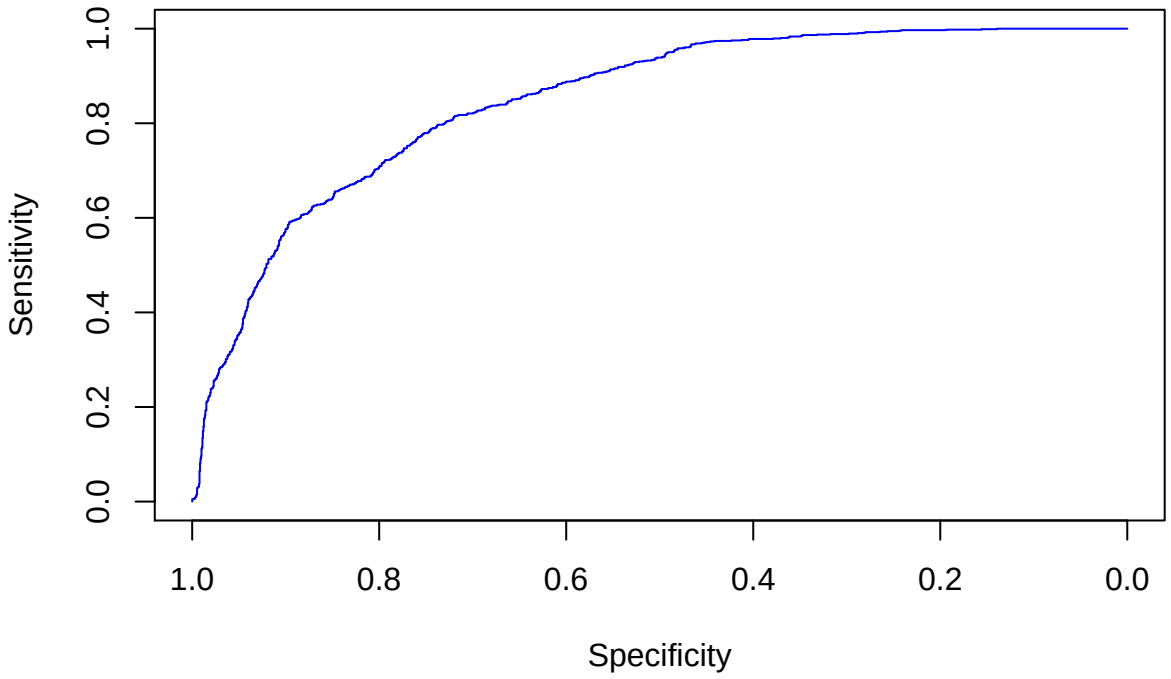
```
## Area under the curve: 0.8477
```

```
##visualize ROC plot
plot(x = roc(predictor = svm$pred$pos, response = svm$pred$obs)$specificities, y = roc(predictor = svm$p
```

```
## Setting levels: control = neg, case = pos
## Setting direction: controls < cases
```

```
## Setting levels: control = neg, case = pos
```

```
## Setting direction: controls < cases
```



To get a more detailed view of model performance, a confusion matrix can be built to see comparisons between predictive classes and any possible bias.

```
##calculate confusion matrix to visualize classificaiton performance
svm_test = predict(svm, newdata = test_classification)
confusionMatrix(svm_test, reference = test_classification$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 109  40
##        pos   8  35
##
##                Accuracy : 0.75
##                  95% CI : (0.6826, 0.8096)
##     No Information Rate : 0.6094
##     P-Value [Acc > NIR] : 2.776e-05
##
##                   Kappa : 0.4313
##
##  Mcnemar's Test P-Value : 7.660e-06
##
##             Sensitivity : 0.9316
##             Specificity : 0.4667
##          Pos Pred Value : 0.7315
##          Neg Pred Value : 0.8140
##              Prevalence : 0.6094
##          Detection Rate : 0.5677
##    Detection Prevalence : 0.7760
##       Balanced Accuracy : 0.6991
##
##        'Positive' Class : neg
```

##

To compare, the same steps as above will be run, but with a radial kernel for the SVM.

```
##set seed to make reproduceable
set.seed(1112)
##define control
control = trainControl(method = "repeatedcv", repeats = 5, classProbs = T, savePredictions = T)

##create SVM Radial model
svm = train(diabetes ~ .,  data = train_classification, method = "svmRadial", tuneLength = 10, trControl

##view SVM model
svm
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 576 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 519, 519, 519, 519, 517, 518, ...
## Resampling results across tuning parameters:
##
##   C       Accuracy   Kappa
##     0.25  0.7762339  0.4761118
##     0.50  0.7747761  0.4617988
##     1.00  0.7712915  0.4502763
##     2.00  0.7573224  0.4113422
##     4.00  0.7565975  0.4084282
##     8.00  0.7468989  0.3820017
##    16.00  0.7402808  0.3596802
##    32.00  0.7279011  0.3271562
##    64.00  0.7268241  0.3210938
##   128.00  0.7202413  0.2985867
##
## Tuning parameter 'sigma' was held constant at a value of 0.1318509
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1318509 and C = 0.25.
```

```
##calculate AUC ROC
roc(predictor = svm$pred$pos, response = svm$pred$obs)$auc
```

```
## Setting levels: control = neg, case = pos
```

```
## Setting direction: controls < cases
```

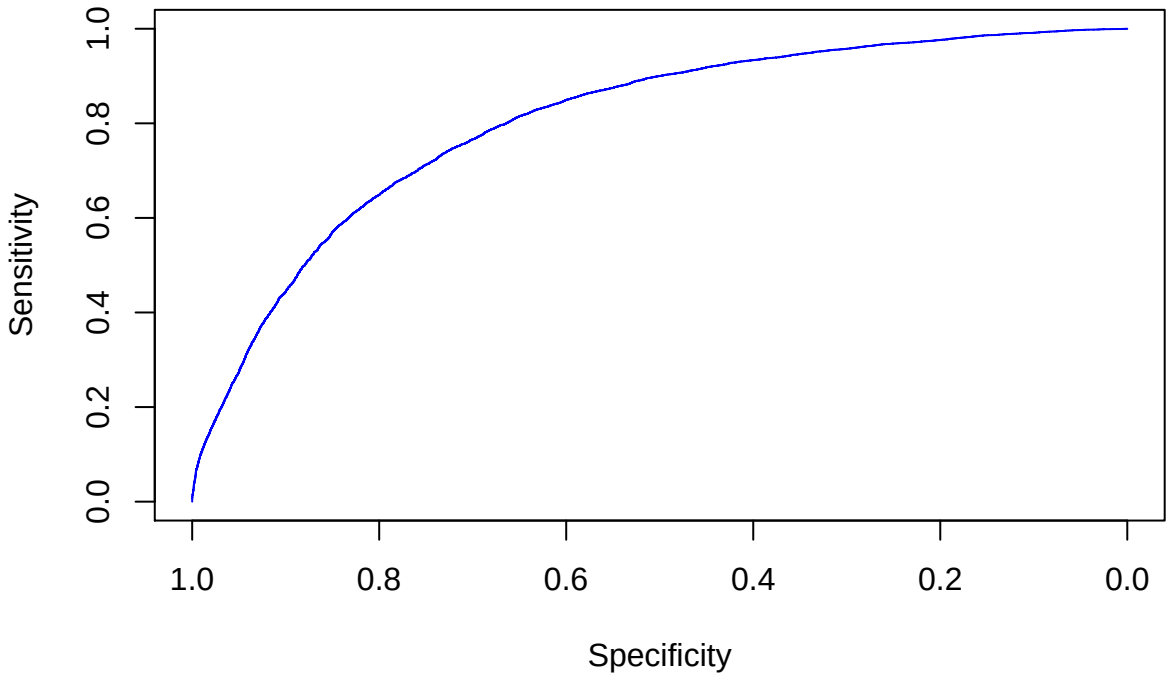
```
## Area under the curve: 0.8041
```

```
##visualize ROC plot
plot(x = roc(predictor = svm$pred$pos, response = svm$pred$obs)$specificities, y = roc(predictor = svm$p
```

```
## Setting levels: control = neg, case = pos
## Setting direction: controls < cases
```

```
## Setting levels: control = neg, case = pos
```

```
## Setting direction: controls < cases
```




```
##calculate confusion matrix to visualize classificaiton performance
svm_test = predict(svm, newdata = test_classification)
confusionMatrix(svm_test, reference = test_classification$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 104  37
##        pos  13  38
##
##                Accuracy : 0.7396
##                  95% CI : (0.6715, 0.8001)
##     No Information Rate : 0.6094
##     P-Value [Acc > NIR] : 0.0001014
##
##                   Kappa : 0.4197
##
##  Mcnemar's Test P-Value : 0.0011432
##
##             Sensitivity : 0.8889
##             Specificity : 0.5067
##          Pos Pred Value : 0.7376
##          Neg Pred Value : 0.7451
##              Prevalence : 0.6094
##          Detection Rate : 0.5417
##    Detection Prevalence : 0.7344
##       Balanced Accuracy : 0.6978
##
##        'Positive' Class : neg
##
```

From both the area under the ROC and the confusion matrix, you can see that a linear kernel has better performance than a radial one. This indicates that the Pima Indians Diabetes dataset is likely split along a linear space, and does not require the radial kernel method, which could overfit the training set if used in a linear problem. It can be hard to determine what the spatial arangement is in datasets such as these with multiple predictors, but if there is doubt, construction of both models can help clarify and the simplest method is always going to be preferred.

2. Attempt using SVM after using a previously covered feature selection method. Do the results improve? Explain.

Since this dataset performs better with a linear kernel, I chose to employ another linear-based feature selection model first, to see if the model performance could be improved. Below, Least Absolute Shrinkage and Selection Operator (LASSO) regression is performed on the same dataset, resulting in the elimination of 2 predictor variables: triceps measurement and insulin level.

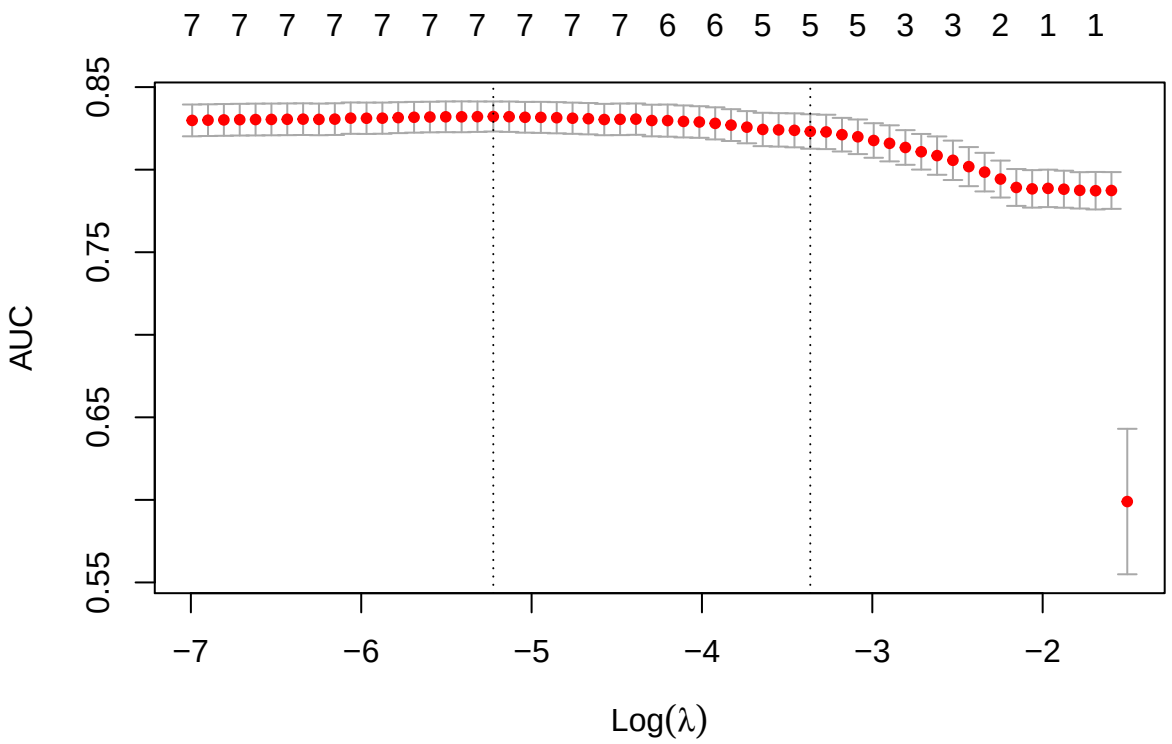
```
set.seed(24)

#convert data
x = x <- as.matrix(pima[,1:8])
y = as.double(as.matrix(ifelse(pima[,9]=='neg', 0, 1)))

#fit Lasso model
cv.lasso <- cv.glmnet(x, y, family='binomial', alpha=1, parallel=TRUE, standardize=TRUE, type.measure='a
```

```
## Warning: executing %dopar% sequentially: no parallel backend registered
```

```
plot(cv.lasso)
```




```
cat('Min Lambda: ', cv.lasso$lambda.min, '\n 1Sd Lambda: ', cv.lasso$lambda.1se)
```

```
## Min Lambda:  0.005382165
##  1Sd Lambda:  0.03459695
```

```
df_coef <- round(as.matrix(coef(cv.lasso, s=cv.lasso$lambda.min)), 2)

# See all contributing variables
df_coef[df_coef[, 1] != 0, ]
```

```
## (Intercept)    pregnant     glucose    pressure        mass    pedigree
##       -7.90        0.11        0.03       -0.01        0.08        0.80
##         age
##        0.01
```

With this reduction in features, a new SVM model can be built with only the remaining 6 predictors, and performance can be visualized as before.

```
##set seed to make reproduceable
set.seed(1112)
##define control
control = trainControl(method = "repeatedcv", repeats = 5, classProbs = T, savePredictions = T)

##create SVM Linear model
svm = train(diabetes ~ pregnant + glucose + pressure + mass + pedigree + age,  data = train_classificati

##view SVM model
svm
```

```
## Support Vector Machines with Linear Kernel
##
## 576 samples
##   6 predictor
##   2 classes: 'neg', 'pos'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 519, 519, 519, 519, 517, 518, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.789399  0.4952965
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
##calculate AUC ROC
roc(predictor = svm$pred$pos, response = svm$pred$obs)$auc
```

```
## Setting levels: control = neg, case = pos
```

```
## Setting direction: controls < cases
```

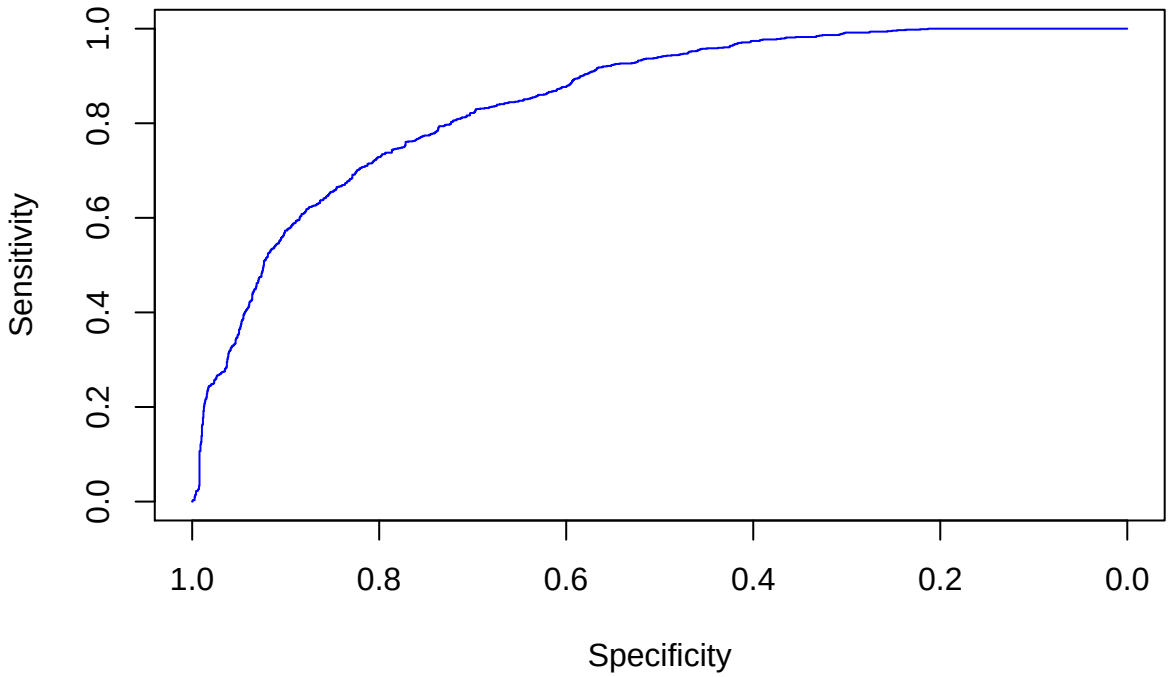
```
## Area under the curve: 0.8484
```

```
##visualize ROC plot
plot(x = roc(predictor = svm$pred$pos, response = svm$pred$obs)$specificities, y = roc(predictor = svm$p
```

```
## Setting levels: control = neg, case = pos
## Setting direction: controls < cases
```

```
## Setting levels: control = neg, case = pos
```

```
## Setting direction: controls < cases
```




```
##calculate confusion matrix to visualize classificaiton performance
svm_test = predict(svm, newdata = test_classification)
confusionMatrix(svm_test, reference = test_classification$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 105  36
##        pos  12  39
##
##                Accuracy : 0.75
##                  95% CI : (0.6826, 0.8096)
##     No Information Rate : 0.6094
##     P-Value [Acc > NIR] : 2.776e-05
##
##                   Kappa : 0.4429
##
##  Mcnemar's Test P-Value : 0.0009009
##
##             Sensitivity : 0.8974
##             Specificity : 0.5200
##          Pos Pred Value : 0.7447
##          Neg Pred Value : 0.7647
##              Prevalence : 0.6094
##          Detection Rate : 0.5469
##    Detection Prevalence : 0.7344
##       Balanced Accuracy : 0.7087
##
##        'Positive' Class : neg
##
```

The resulting model shows a very tiny increase in performance. In fact, after LASSO regression, the model

predicts ONE true negative case that it before incorrectly classified. While this objective boost in performance may signal that optimisation can improve SVM model performance, the computational toll should be factored in based on the gains expected from the dataset. SVM can be hindered by unnecessary features, but with only 8 predictors, this model is unlikely to suffer too much from that fact. I would imagine that feature selection prior to SVM would be more impactful on a dataset with higher dimensionality, or one with a significant number of features with very low predictive weight.