Project 1: Bioinformatics of Gene Expression

Merrimack College DSE6630: Healthcare & Life Sciences Analytics

Adeline Casali, Samantha Harper, Scott Eugley, Tiffany Michel Acosta

2024-05-16

# Questions

##### **Question 1** (SH): Look up these rules and try to explain what they are doing for yourself.

A classification tree predicts classes for observations by selecting the most common class for that node of the tree. The Gini index is a measure of the variance across the classes within a single region of the tree and is therefore a measure of ‘node purity’. The smaller the Gini index, the more of that node corresponds to a singular class. (James et al., 2021) When using Gini Impurity as a splitting rule, the algorithm will choose a split that has the largest decrease in Gini Impurity (since our ideal impurity is 0) (Lin, 2023, Ch 11.2).

The extratrees splitting rule is based on the Extra-Tree Algorithm by Geurts et al. (2005). The algorithm splits the ensemble of trees completely at random and uses the whole training sample to grow each tree. The goal is to reduce variation by using randomization when selecting cut-points and to reduce bias by using the whole training sample (Geurts et al, 2005).

##### **Question 2** (SH): Take a moment to go look at the function and the documentation I have written. Also try playing with the parameters a bit - how many more or fewer genes do you get when you lower or raise the filterCutoff, respectively?

splits <- doSplits(vst = vsData, algorithm = "rf", splitRatio = 0.8, filterCutoff = 5)

## [1] "After filtering, the number of genes remaining in the dataset are: 7676"

splits <- doSplits(vst = vsData, algorithm = "rf", splitRatio = 0.8, filterCutoff = 3)

## [1] "After filtering, the number of genes remaining in the dataset are: 12927"

When the filterCuttoff is reduced to 3 from 5 the number of genes remaining in the dataset increases to 12,927 from 7,676.

splits <- doSplits(vst = vsData, algorithm = "rf", splitRatio = 0.8, filterCutoff = 7)

## [1] "After filtering, the number of genes remaining in the dataset are: 4476"

When the filterCutoff is increased to 7, the number of genes remaining in the dataset decreases to 4476.

##### **Question 3** (SH): Do some research on **data leakage** and try to explain what that means in this context. Why do we care about data leakage?

Data leakage happens when a model is trained on data that will not be available ‘at the moment of prediction’. One type of data leakage is target leakage; any features informed by a label cannot be used for predicting that label. For example, the treatment drugs taken by a patient cannot be used for diagnosing a patient, since they are prescribed after diagnosis has taken place. Another type of data leakage occurs when any type of data processing takes place before splitting the data into training and validation sets. Train-test data leakage causes the results obtained from the validation set to become invalid, since they were part of informing the model originally (Cook).

In this context, doing feature selection on the whole data set, instead of just the training set, will falsely increase the accuracy of the validation set. Training and test error are often very different and can help us identify when the model has been overfit. If the validation or test set is subject to data leakage, it will be much harder to identify overfitting and ultimately produce a model that hasn’t been tested and is less accurate.

##### **Question 4** (SH): What combination of hyperparameters gives the highest accuracy in the **training model**? (Use plot above to answer.)

The accuracy rises very quickly and then slowly decreases and the number of randomly chosen predictors increases. This increase is steeper for the gini splitting rule than it is for the extratrees splitting rule; the accuracy for gini rises to over 71% while the accuracy for extratrees only rises to about 67.5%. The peak for both is fairly low, at closer to 100 randomly selected predictors. The best model had a gini splitrule, mtry = 123, and a minimum node size of 1.

##### **Question 5** (SH): Can you see an impact of the pre-processing (to remove lowly expressed genes)?

The accuracy of the second model (after filtering for low expression) is much more accurate (from 43% to 50% accuracy).

##### **Question 6** (SE): Can you see an impact of the 10-fold cross-validation?

By comparing the two confusion matrices provided in table 2, it does not appear as though the 10-fold CV had any significant impact on the RF model. This is most likely due to RF’s internal mechanisms providing robust performance without the need for additional CV.

##### **Question 7** (SE): How many different combinations are we going to search with this grid?

mtry has 5 possible values, splitrule has 2 possible values, and min.node.size has 4 possible values. 5 \* 2 \* 4 = 40 different combinations.

##### **Question 8** (SE): What combination of hyperparameters gives the highest accuracy in the **training model**? (Use plot above to answer.)

The highest accuracy is achieved with the following hyperparameter values: Number Randomly Selected Predictors (mtry) = ~2000 splitrule = “extratrees” min.node.size = 5

##### **Question 9** (SE): How well did hyperparameter tuning with the grid search perform?

Hyperparameter tuning with grid search actually performed worse than k-fold CV and OOB RF. Accuracy decreased from 0.571 to 0.500 and the kappa coef. dropped to 0! This potentially points to grid search hyperparameter tuning leading to overfitting.

##### **Question 10** (SE): Use the modelLookup function to help you figure out what hyperparameters are available for the rf vs. ranger methods for performing random forest algorithm with caret. What are the key differences you observce in the hypers you can tune?

rf\_info <- modelLookup("rf")  
ranger\_info <- modelLookup("ranger")  
  
# Hyperparameters for rf  
print(rf\_info)

## model parameter label forReg forClass probModel  
## 1 rf mtry #Randomly Selected Predictors TRUE TRUE TRUE

# Hyperparameters for ranger  
print(ranger\_info)

## model parameter label forReg forClass probModel  
## 1 ranger mtry #Randomly Selected Predictors TRUE TRUE TRUE  
## 2 ranger splitrule Splitting Rule TRUE TRUE TRUE  
## 3 ranger min.node.size Minimal Node Size TRUE TRUE TRUE

Ranger provides additional options to tune (splitting rule and minimal node size), this offers more flexibility in hyperparameter tuning. rf only offers mtry, which ranger also offers.

##### **Question 11** (TMA): What combination of hyperparameters gives the highest accuracy in the **training model**? (Use plot above to answer.) How does it compare to previous results?

gama = 0.0001302253 and cost = 1 in the training model the yield accuracy is 64.29%. This accuracy match with the cross validation svm model but it is lower than the initial SVM.The initial SVM performed better without the hyperparameter. Accuracy (71.43%) and Kappa (0.4286) this provided a more stable model by avoiding overfitting.

##### **Question 12** (TMA): How well did hyperparameter tuning with the grid search perform?

The tuned Hyperparameter did not improve while the default did.

##### **Question 13** (TMA): Notice that the y-axis is now **log-Loss** and not **accuracy**. You may need to do some digging, but can you figure out why? Sources can be helpful if you’re not sure you’re on the right track!

The switch from accuracy to log-loss on the y-axis means the plot now looks at a different aspect of the SVM model’s performance. Log-loss measures how well the model predicts probabilities compared to the actual outcomes. It’s important because SVMs don’t directly give probabilities but instead provide decision values. The plot likely shows the ROC curve, which is about how well the model balances true positives and false positives. This change tells us the focus is now on the model’s ability to estimate probabilities accurately, not just its overall correctness.(Palnt.J.C, 1999)

##### **Question 14** (TMA): The ROC and precision-recall curves are suggesting very high accuracy. Woohoo!!… Right?! What is the confusion matrix telling us? How do you know?

It is telling us that high P- value is close to 1 and there were no diferences between performances.

##### **Question 15** (TMA): Should we have any confidence in these results? Why or why not?

The dataset looks large enough and balanced, so the results should be reliable.

# Background and Question (SH)

As described by Verble et al. in 2024, cold temperatures can have a range of impacts on insects and their gene expressions. This study used standard statistical methods to analyze the large amounts of data resulting from the genetic material collected from the cold-exposed bees. Our aim is to conduct a benchmarking study, i.e. we wish to use the same data and apply new methods of analysis in order to learn more about the gene expression of cold-exposed bumble bees. This study will compare two machine learning models, Support Vector Machines and Random Forests. Our aim is to identify significant genes by identifying which genes have had large changes in expression in the cold-exposed samples compared to the control groups. Once these genes are identified we can compare the results with the original Verble study to compare the advantages of each model and learn more about the genetic impact of cold on bumble bees.

# Data Source (SH)

The data used was collected by Verble et al. in their 2024 study “A rapid return to normal: temporal gene expression patterns following cold exposure in the bumble bee Bombus impatients.” The data contain variables for the different colonies of bees, the control or chill exposure that the individual received, and the duration of the treatment. The count data recorded the ‘counts per million’ for each individual gene found in the samples.

The counts and other features were used to create a DESeq design object which stores the data. We then used a variance-stabilizing transformation to normalize the data. This allows us to compare samples accurately. This VST data is fed into our machine learning models.

Verble, K. M., Keaveny, E. C., Rahman, S. R., Jenny, M. J., Dillon, M. E., & Lozier, J. D. (2024). A rapid return to normal: temporal gene expression patterns following cold exposure in the bumble bee Bombus impatiens. The Journal of experimental biology, 227(9), jeb247040. <https://doi.org/10.1242/jeb.247040>

# Analysis (AC)

In this section, we delve into the methods employed and results derived from the analysis of the dataset using Support Vector Machine (SVM) models. The analysis includes the performance evaluation of SVM models with various configurations, a comparison of their accuracies, and insights into the effectiveness of different model configurations.

#### Data Preprocessing

Before training the models, it was essential to preprocess the dataset to ensure its quality and readiness for analysis. This involved loading the VST (variance stabilizing transformation) data for further manipulation. The data was then split into a test and training dataset using a custom function (see Appendix 2) with 80% of the dataset in the training set, and the remaining 20% in the test set.

#### Model Training

The first step in the model training process was fitting an SVM model with a linear kernel using a basic, out-of-the-box approach. This initial model yielded an accuracy of 78.57%, providing a solid baseline for comparison. Subsequently, SVM models with various kernels (linear, sigmoid, radial, and polynomial) were trained using k-fold cross-validation (k = 10). Among these, the radial kernel emerged as the top performer, achieving an accuracy of 64.29%.

#### Model Tuning

To optimize the SVM model further, hyperparameter tuning was conducted separately for gamma and cost parameters using a grid search. This process involved systematically exploring different combinations of hyperparameter values to identify the optimal configuration. This step was broken into two sub-steps to reduce the computational power required to create the model. These optimal values of gamma and cost were then implemented into an SVM model trained with a radial kernel and using k-fold validation (k = 10), for a resulting accuracy of 64.28%. This fine-tuning step aimed to enhance the model’s performance and generalization capabilities, ensuring it could effectively capture the underlying patterns in the data while avoiding overfitting.

#### Model Evaluation

Once the models were trained and tuned, confusion matrices were generated for each model to assess their predictive accuracy. Additionally, an ROC (receiver operating characteristic) curve was created, and the AUC (area under the curve) metric was calculated to be 0.5714 for the tuned SVM model, providing insights into its discriminative power (Figure 1). Finally, a comparative table was created to highlight differences in accuracies, kappa values, and statistical significance among the various models (Figure 2).

A graph with a line

Description automatically generated

Figure 1: ROC Curve for the SVM model created with cross-validation (k = 10) and a radial kernel.

A table with numbers and text

Description automatically generated

Figure 2: Table comparison of accuracies for each model (out-of-box, cross-validation, and grid search tuned).

# Interpretations and Conclusions (SE)

In this analysis three SVM models were evaluated based on their ability to classify treatment outcomes:

* OOB Model with linear kernel
* K-fold (k=10) CV Model using a radial kernel
* Tuned Model using Grid Search to Optimize Hyperparameters

An attempt at a fourth model with nested CV was performed, but not completed in this analysis. The performance of each model was studied using confusion matrices, accuracy scores, and other related statistics, such as sensitivity, specificity, ROC curve, and AUC.

Interestingly, the OOB SVM model performed the best with the highest accuracy of 0.7857. This model also showed good sensitivity and specificity with values 0.7143 and 0.8571, respectively. The k-fold CV model had lower accuracy at 0.6429 and lower sensitivity, which dropped to 0.4286. Specificity remained high at 0.8571, indicating that the model struggled in identifying true positives, but did well at identifying true negatives. Despite performing grid search to optimize hyperparameters, the tuned model did not show improvement over the k-fold CV model. Accuracy remained the same at 0.6429 and so did sensitivity and specificity. Also, both the k-fold CV model and tuned model produced the same ROC curve. The ROC curves did not “hug” the upper-left area of the graph and did not have a large area under the curve. AUC for the tuned model was 0.5714, indicating fairly poor discriminative ability. While AUC was not calculated for the k-fold CV model, given the ROC curves, it is likely the same as the tuned model.

The reason the OOB model may have performed best in this analysis could be purely due to its simplicity. The high accuracy score and Kappa suggests that it did generalize well. The kernel for the OOB was also set to linear and linear models can be less prone to overfitting, especially if the underlying relationships in the data are approximately linear. A non-linear kernel, such as radial, could potentially lead to overfitting. Another potential issue could be with the size of the dataset. In more sparse datasets, simpler models often outperform more complex models. This is because more complex models need more data to capture underlying relationships more effectively. Since RNA-seq data is prone to having “big p little n” issues, it is likely that the more complex models overfit the data, as there were not enough observations for the complex models to capture relationships.

# Recommendations and Next Steps (SE)

While the OOB SVM model performed best in this analysis, it is recommended that some further steps be taken to yield a more complete analysis. An attempt at Nested CV was made in this analysis, however, it was not completed due to encountering various errors. Further troubleshooting of this model is recommended. Additionally, it would be helpful to determine a method capable of extracting a list of gene importances from the finalized SVM model. The SVM approach should also be compared with ensemble methods, such as random forest, which will help refine the overall approach in determining significant cold-expressed genes. Finally, to complete the benchmarking study, these results should be compared with the results from the original study, Verble et al. “A rapid return to normal: temporal gene expression patterns following cold exposure in the bumble bee Bombus impatients”.

# Appendix 1: References

James, G. et al. (2021) An Introduction to Statistical Learning with Applications in R. (Second Edition). Springer. Lin, H. & Li, M. (2023) Practitioners Guide to Data Science. <https://scientistcafe.com/ids/#preface> Geurts, P. et al. (2005) Extremely randomized trees. Mach Learn (63: 3- 42). Springer. Cook, A., n.d. Data Leakage. <https://www.kaggle.com/code/alexisbcook/data-leakage>

# Appendix 2: Code

## [1] "After filtering, the number of genes remaining in the dataset are: 7676"

# Code Author: AC  
# Creating a basic (OOB) svm model  
svmOOB <- svm(Treatment ~ .,   
 data = train,  
 kernel = "linear",  
 na.action = na.omit  
)  
paste0("The total number of support vectors was: ", svmOOB$tot.nSV)

## [1] "The total number of support vectors was: 56"

pred.test.svm <- predict(svmOOB, test, type = "response")  
confMat\_OOB <- caret::confusionMatrix(pred.test.svm, test$Treatment)  
confMat\_OOB

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Control Cold  
## Control 5 1  
## Cold 2 6  
##   
## Accuracy : 0.7857   
## 95% CI : (0.492, 0.9534)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.02869   
##   
## Kappa : 0.5714   
##   
## Mcnemar's Test P-Value : 1.00000   
##   
## Sensitivity : 0.7143   
## Specificity : 0.8571   
## Pos Pred Value : 0.8333   
## Neg Pred Value : 0.7500   
## Prevalence : 0.5000   
## Detection Rate : 0.3571   
## Detection Prevalence : 0.4286   
## Balanced Accuracy : 0.7857   
##   
## 'Positive' Class : Control   
##

# Code Author: AC  
# Creating svm models with CV k = 10 and grid search for kernel: linear, sigmoid, radial, and polynomial  
# Create the grid  
searchGrid <- expand.grid(  
 .kernel = c("linear", "sigmoid", "radial", "polynomial")  
)  
# Create the model (k = 5 for computational purposes)  
svm\_tune <- tune(svm,   
 Treatment ~ .,   
 data = train,   
 ranges = searchGrid,   
 tunecontrol = tune.control(  
 sampling = "cross",   
 cross = 5  
 )  
 )  
svm\_CV <- svm\_tune$best.model  
key <- svm\_CV$kernel  
kernel\_key <- list(  
 "0" = "Linear kernel",  
 "1" = "Polynomial kernel",  
 "2" = "Radial kernel",  
 "3" = "Sigmoid kernel"  
)  
kernel\_description <- kernel\_key[[as.character(key)]]  
print(kernel\_description)

## [1] "Radial kernel"

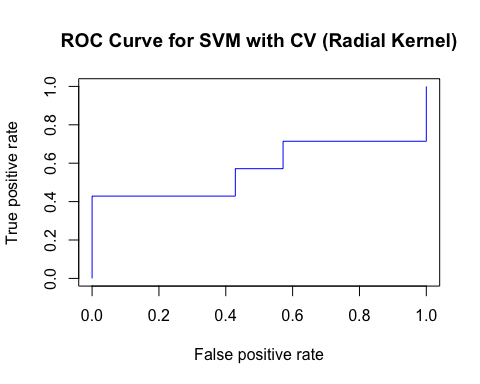
# Run chosen kernel with k = 10  
kFoldCtrl <- trainControl(method = "cv", # for k-fold CV  
 number = 10) # k  
svmCV <- svm(Treatment ~ .,  
 data = train,  
 kernel = "radial",   
 na.action = na.omit,   
 trControl = kFoldCtrl)  
paste0("The total number of support vectors was: ", svmCV$tot.nSV)

## [1] "The total number of support vectors was: 60"

pred.test.svm\_CV <- predict(svmCV, test, type = "response")  
confMat\_CV <- caret::confusionMatrix(pred.test.svm\_CV, test$Treatment)  
confMat\_CV

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Control Cold  
## Control 3 1  
## Cold 4 6  
##   
## Accuracy : 0.6429   
## 95% CI : (0.3514, 0.8724)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.2120   
##   
## Kappa : 0.2857   
##   
## Mcnemar's Test P-Value : 0.3711   
##   
## Sensitivity : 0.4286   
## Specificity : 0.8571   
## Pos Pred Value : 0.7500   
## Neg Pred Value : 0.6000   
## Prevalence : 0.5000   
## Detection Rate : 0.2143   
## Detection Prevalence : 0.2857   
## Balanced Accuracy : 0.6429   
##   
## 'Positive' Class : Control   
##

# Visualize performance of chosen model (Radial)  
decision\_values <- attributes(predict(svmCV, test, decision.values = TRUE))$decision.values  
pred <- prediction(decision\_values, test$Treatment)  
roc\_curve <- performance(pred, "tpr", "fpr")  
plot(roc\_curve, main = "ROC Curve for SVM with CV (Radial Kernel)", col = "blue")



# Code Author: AC  
# Creating a tuned SVM model (two separate tunes due to computational power restrictions)  
# Define the grid for gamma  
searchGrid\_gamma <- expand.grid(  
 .gamma = c(0.01, 0.1, 1),   
 .kernel = c("radial")  
)  
  
# Tune the model for gamma  
svm\_tune\_gamma <- tune(svm,   
 Treatment ~ .,   
 data = train,   
 prediction = TRUE,   
 probability = TRUE,   
 ranges = searchGrid\_gamma,   
 tunecontrol = tune.control(  
 sampling = "fix",   
 fix = 1  
 )  
 )  
  
# Extract the best gamma  
best\_gamma <- svm\_tune\_gamma$best.model$gamma  
cat("Best Gamma:", best\_gamma, "\n")

## Best Gamma: 0.0001302253

# Define the grid for cost  
searchGrid\_cost <- expand.grid(  
 .C = c(0.1, 1, 10),  
 .kernel = c("radial")  
)  
  
# Tune the model for cost  
svm\_tune\_cost <- tune(svm,   
 Treatment ~ .,   
 data = train,   
 prediction = TRUE,   
 probability = TRUE,   
 ranges = searchGrid\_cost,   
 tunecontrol = tune.control(  
 sampling = "fix",   
 fix = 1  
 )  
 )  
  
# Extract the best cost  
best\_cost <- svm\_tune\_cost$best.model$cost  
cat("Best Cost:", best\_cost, "\n")

## Best Cost: 1

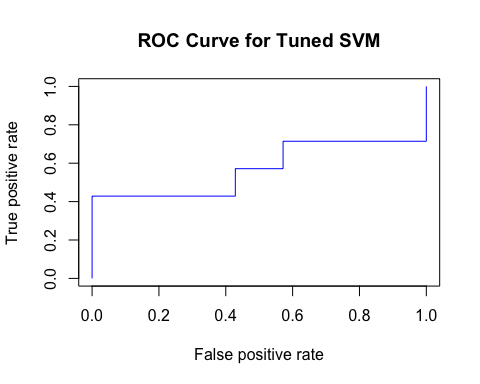
# Create the final model  
kFoldCtrl <- trainControl(method = "cv", # for k-fold CV  
 number = 10) # k  
svm\_tuned <- svm(Treatment ~ .,  
 data = train,  
 kernel = "radial",   
 C = best\_cost,   
 gamma = best\_gamma,   
 probability = TRUE,   
 trControl = kFoldCtrl)  
  
# Print the total number of support vectors  
paste0("The total number of support vectors was: ", svm\_tuned$tot.nSV)

## [1] "The total number of support vectors was: 60"

# Predict on the test set  
pred.test\_svm\_tuned <- predict(svm\_tuned, test, type = "response")  
  
# Create the confusion matrix  
confMat\_tuned <- caret::confusionMatrix(pred.test\_svm\_tuned, test$Treatment)  
confMat\_tuned

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Control Cold  
## Control 3 1  
## Cold 4 6  
##   
## Accuracy : 0.6429   
## 95% CI : (0.3514, 0.8724)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.2120   
##   
## Kappa : 0.2857   
##   
## Mcnemar's Test P-Value : 0.3711   
##   
## Sensitivity : 0.4286   
## Specificity : 0.8571   
## Pos Pred Value : 0.7500   
## Neg Pred Value : 0.6000   
## Prevalence : 0.5000   
## Detection Rate : 0.2143   
## Detection Prevalence : 0.2857   
## Balanced Accuracy : 0.6429   
##   
## 'Positive' Class : Control   
##

# Code Author: AC  
# Visualize performance  
decision\_values <- attributes(predict(svm\_tuned, test, decision.values = TRUE))$decision.values  
pred <- prediction(decision\_values, test$Treatment)  
roc\_curve <- performance(pred, "tpr", "fpr")  
plot(roc\_curve, main = "ROC Curve for Tuned SVM", col = "blue")



# Calculating AUC  
pred\_prob <- predict(svm\_tuned, test, probability = TRUE)  
pred\_prob\_pos <- attr(pred\_prob, "probabilities")[, "Control"]  
auc <- roc(test$Treatment, pred\_prob\_pos)$auc

## Setting levels: control = Control, case = Cold

## Setting direction: controls > cases

print(auc)

## Area under the curve: 0.5714

# Code Author: AC  
# Comparing the models  
compareConfusion(confusionList = list(confMat\_OOB, confMat\_CV, confMat\_tuned)) %>%  
 knitr::kable(format = "pandoc", caption = "Table 1. Comparing Accuracy - Support Vector Machine Models") %>%  
 kableExtra::kable\_styling(bootstrap\_options = "striped", full\_width = FALSE)

## Warning in kableExtra::kable\_styling(., bootstrap\_options = "striped",  
## full\_width = FALSE): Please specify format in kable. kableExtra can customize  
## either HTML or LaTeX outputs. See https://haozhu233.github.io/kableExtra/ for  
## details.

Table 1. Comparing Accuracy - Support Vector Machine Models

| Metric | Confusion Matrix 1 | Confusion Matrix 2 | Confusion Matrix 3 |
| --- | --- | --- | --- |
| Accuracy | 0.7857143 | 0.6428571 | 0.6428571 |
| Kappa | 0.5714286 | 0.2857143 | 0.2857143 |
| AccuracyLower | 0.4920243 | 0.3513801 | 0.3513801 |
| AccuracyUpper | 0.9534207 | 0.8724016 | 0.8724016 |
| AccuracyNull | 0.5000000 | 0.5000000 | 0.5000000 |
| AccuracyPValue | 0.0286865 | 0.2119751 | 0.2119751 |
| McnemarPValue | 1.0000000 | 0.3710934 | 0.3710934 |

# Nested CV Model  
# Code Author: SE/SH  
  
# Set up grid  
searchGrid <- expand.grid(  
 .sigma = c(0.01, 0.1, 1),  
 .C = c(0.1, 1, 10)  
)  
selected\_columns <- train[, !grepl("^LOC", names(train))]  
print(selected\_columns)  
train$Time <- as.integer(sub("m$", "", train$Time))  
train[, -which(names(train) == "Treatment")] <- sapply(train[, -which(names(train) == "Treatment")], as.integer)  
train$Treatment <- as.factor(train$Treatment)  
  
# Set up training control with probability = TRUE  
train\_control <- trainControl(method = "cv", number = 10, savePredictions = "final", classProbs = TRUE)  
  
# Train the model using nested cross-validation  
ncv <- nestcv.train(y = train$Treatment, x = train,  
 method = 'svmRadial',  
 tuneGrid = searchGrid,  
 trControl = train\_control,  
 probability = TRUE)  
pred.test\_svm\_nested <- predict(svm\_tuned, test, type = "response")  
  
# Create the confusion matrix  
confMat\_nested <- caret::confusionMatrix(pred.test\_svm\_nested, test$Treatment)  
print(confMat\_nested)

# Visualize  
# Code Author: SE  
ggplot(ncv$outer\_result[[1]]$fit) +  
 scale\_x\_log10() +  
 ggtitle("Results of Nested CV with hyperparameter tuning") +  
 theme\_bw()  
  
# Plot ROC and Precision-Recall curves  
op <- par(mfrow = c(1, 2))  
  
# Outer CV ROC  
plot(ncv$roc,   
 main = "Outer-folds ROC",   
 col = 'blue')  
legend("bottomright",   
 legend = paste0("AUC = ", signif(pROC::auc(ncv$roc), 3)),   
 bty = 'n')  
  
# Inner CV ROC  
inroc <- innercv\_roc(ncv)  
plot(inroc,   
 main = "Inner-folds ROC",   
 col = 'red')  
legend("bottomright",   
 legend = paste0("AUC = ", signif(pROC::auc(inroc), 3)),   
 bty = 'n')

# Code Author: SE  
# Comparing the models  
compareConfusion(confusionList = list(confMat\_OOB, confMat\_CV, confMat\_tuned, conMat\_nested)) %>%  
 knitr::kable(format = "pandoc", caption = "Table 1. Comparing Accuracy - Support Vector Machine Models") %>%  
 kableExtra::kable\_styling(bootstrap\_options = "striped", full\_width = FALSE)