Development of an Optimized Statistical Learning Model for Breast Cancer Classification Using the Wisconsin Diagnostic Dataset

Zach Ginder

Overview

Cancer is a devastating disease effecting millions of people a year in the United States and abroad. Studies are constantly being performed to determine better diagnostic guidelines and tools to increase the ability for early detection, something that is known to lower the risk of mortality. One area of research that has seen great progress in the past few decades is detection and treatment of breast cancer. Even though breast cancer detection has significantly progressed, there is always need for improvement in accurate detection of benign (non-cancerous) versus malignant (cancerous) tumors to ensure accurate medical treatment plans.

Utilizing data from the University of California, Irvine, Machine Learning Repository, this project seeks to develop an optimized statistical learning model for classifying cells from breast masses as benign (non-cancerous) or malignant (cancerous) tumors. The data utilized is diagnostic breast cancer data originating from patients in Wisconsin. The features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. Nuclei are analyzed per patient and ten features are analyzed for each with the mean (labeled as 1), standard error (labeled as 2), and worst/largest (mean of the three largest values labeled as 3) considered for each of the ten features. This results in a total of thirty features per patient. Along with the thirty cell features the data includes the diagnosis class of the breast mass as malignant or benign, and the patient ID.

Data

```
# Read in the CSV data
celldata <-
  read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsi
```

Data Cleaning

```
#Data Cleaning

#Change the column/feature names to match that of the UCI database
colnames(celldata) <- c("ID","Diagnosis",
                        "Radius1","Texture1","Perimeter1","Area1",
                        "Smoothness1","Compactness1","Concavity1","Concave_Points1",
                        "Symmetry1","Fractal_Dimension1",
                        "Radius2","Texture2","Perimeter2","Area2",
                        "Smoothness2","Compactness2","Concavity2","Concave_Points2",
                        "Symmetry2","Fractal_Dimension2",
                        "Radius3","Texture3","Perimeter3","Area3",
                        "Smoothness3","Compactness3","Concavity3","Concave_Points3",
                        "Symmetry3","Fractal_Dimension3")

#Determine if any missing values are in the dataset
sum_of_nas <- sum(is.na(celldata))

#Check the class of each variable
class <- sapply(celldata,class)

#Change Diagnosis to a factor
celldata$Diagnosis <- as.factor(celldata$Diagnosis)

#Remove ID Column as it isn't necessary as a predictor
celldata <- celldata[,-1]
```

Splitting the Data into a Train and Test Set

```
#Set a seed for reproducible results
set.seed(8675309)

#Sample from the row indices to include in the test set
n <- nrow(celldata)
index <- sample(x = 1:n,
                size= round(0.7*n),
                replace=FALSE)

#Test and training sets
train <- celldata[index, ]
test <- celldata[-index, ]
```

Training Models

K-Nearest Neighbors Model

The K-Nearest Neighbors (KNN) model is a type of nonparametric model. The tuning parameter for the model is k, the number of neighbors used when predicting the response. This type of model is not used for inference but is for prediction. In terms of variable selection, the KNN model does not perform variable selection. Since the KNN model depends on distances to neighbors, it is important to standardize the predictors.

```
#Set a seed for reproducible results
set.seed(8675309)

#Standardize the predictors
train_standard <- as.data.frame(scale(train[,sapply(train,is.numeric)]))
test_standard <- as.data.frame(scale(test[,sapply(test,is.numeric)]))

#Add back in the Diagnosis Columns
train_standard$Diagnosis <-train$Diagnosis
test_standard$Diagnosis <-test$Diagnosis
```

```
#Fit the KNN Model on the Training Set
#K values for tuning
kgrid <- expand.grid(k=seq(1,51,by=2))

#5-fold CV, repeated, tuning
tr <- trainControl(method="repeatedcv",
                   number=5,
                   repeats=50)
#Train the classifier
fit_knn <- train(Diagnosis ~.,
             data=train_standard,
             method="knn",
             tuneGrid=kgrid,
             trControl=tr)
fit_knn$bestTune$k
```

```
[1] 3
```

Using accuracy as our metric, the optimal tuning parameter k is 3.

```
#Refit the model with the optimal K=3.
tuned_knn <- train(Diagnosis~.,
                   data=train_standard,
                   method="knn",
                   tuneGrid=expand.grid(k=3),
                   trControl=trainControl(method="none"))
```

Single Tree Models

The single tree model is a type of nonparametric model. The tuning parameter for the model is the complexity parameter denoted cp. This type of model is not used for inference but is for prediction. In terms of variable selection, the use of the tuning parameter will cause variable selection. For tree based models, it is not necessary to standardize the predictors.

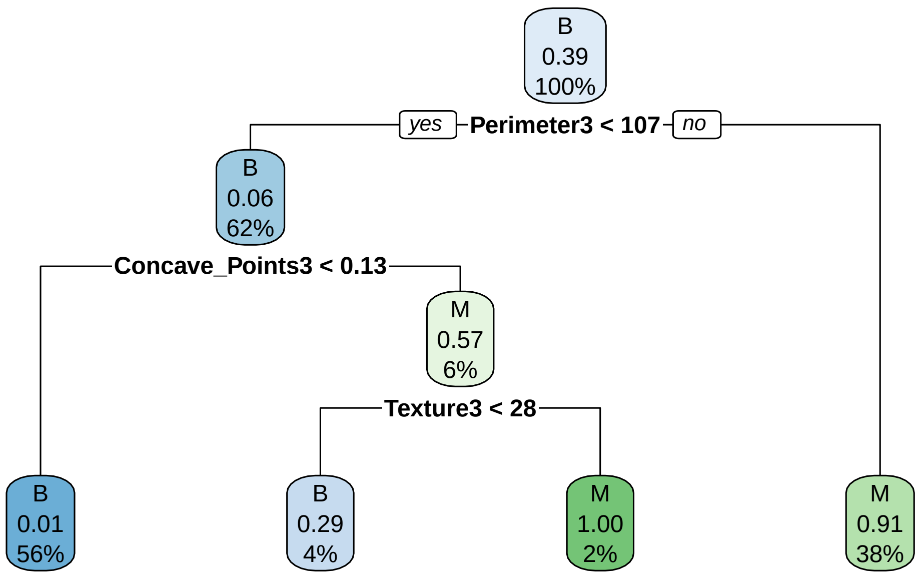
```
#Set a seed for reproducible results
set.seed(8675309)

#Find optimal tuning parameter
cancer_rpart <- rpart(Diagnosis ~.,
                      data=train,
                      method="class",
                      parms = list(split="information"),
                      control=rpart.control(xval=10,
                                            minbucket=2,
                                            cp=0))
cancer_rpart$cptable
```

```
          CP nsplit  rel error    xerror       xstd
1 0.81168831      0 1.00000000 1.0000000 0.06309472
2 0.02922078      1 0.18831169 0.2597403 0.03895016
3 0.01948052      3 0.12987013 0.2272727 0.03668809
4 0.00000000      8 0.02597403 0.2337662 0.03715723
```

```r
cp<- cancer_rpart$cptable
```

Using the CV error rate as the metric, the optimal cp tuning parameter is the cp value of 0.01948052

```r
#Refit the model with the optimal cp=0.01948052
tuned_tree <- prune(cancer_rpart, cp=cp[3,1])
rpart.plot(tuned_tree)
```



Ensemble Tree Models (Random Forest)

The random forest model is a type of nonparametric model. The tuning parameter for the model is m, the number of predictors to consider at each split. This type of model is not used for inference but is for prediction. In terms of variable selection, random forest models conceptually perform variable selection. For tree based models, it is not necessary to standardize the predictors.

```r
#Set a seed for reproducible results
set.seed(8675309)

#Find optimal tuning parameter for the random forest model
```

```r
cancer_rforest <- train(Diagnosis~.,
                        method="rf",
                        data=train,
                        tuneGrid=data.frame(mtry=1:(ncol(train)-1)),
                        trControl=trainControl(method="oob",number=3000))
cancer_rforest$results
```

```
    Accuracy     Kappa mtry
1  0.9547739 0.9037511    1
2  0.9547739 0.9042144    2
3  0.9497487 0.8935715    3
4  0.9547739 0.9042144    4
5  0.9547739 0.9042144    5
6  0.9597990 0.9148572    6
7  0.9522613 0.8990144    7
8  0.9497487 0.8935715    8
9  0.9547739 0.9039833    9
10 0.9572864 0.9096445   10
11 0.9522613 0.8990144   11
12 0.9497487 0.8935715   12
13 0.9547739 0.9044443   13
14 0.9623116 0.9202746   14
15 0.9572864 0.9096445   15
16 0.9547739 0.9044443   16
17 0.9547739 0.9039833   17
18 0.9497487 0.8938270   18
19 0.9497487 0.8938270   19
20 0.9497487 0.8935715   20
21 0.9522613 0.8990144   21
22 0.9522613 0.8992566   22
23 0.9522613 0.8990144   23
24 0.9447236 0.8832097   24
25 0.9522613 0.8992566   25
26 0.9497487 0.8938270   26
27 0.9522613 0.8992566   27
28 0.9497487 0.8935715   28
29 0.9522613 0.8992566   29
30 0.9547739 0.9042144   30
```

```r
cancer_rforest$bestTune
```

```
   mtry
```

```
14    14
```

Using kappa, the adjusted accuracy, as the metric, the optimal mtry tuning parameter is 14.

```r
#Refit the model with the optimal mtry=14
tuned_rforest <- train(Diagnosis~.,
                       method="rf",
                       data=train,
                       tuneGrid=data.frame(mtry=14),
                       trControl=trainControl(method="none"))
varImp(tuned_rforest)
```

```
rf variable importance

  only 20 most important variables shown (out of 30)

                 Overall
Perimeter3      100.0000
Concave_Points3  68.3836
Concave_Points1  64.4544
Area3            56.1940
Radius3          45.5349
Concavity1        9.5247
Concavity3        6.2645
Area2             5.8108
Texture3          5.4735
Texture1          4.8980
Symmetry3         4.2366
Perimeter1        4.2073
Smoothness3       4.0484
Compactness3      3.8773
Area1             2.8336
Radius2           1.8064
Perimeter2        1.3417
Radius1           0.8318
Concavity2        0.8151
Compactness1      0.6674
```

Regularized Logistic Regression Model Using Lasso

The regularized logistic regression model using lasso is a type of parametric model. The tuning parameter for the model is λ , the penalty parameter. This type of model is used for inference.

In terms of variable selection, the use of lasso in this model will perform variable selection by effectively shrinking coefficients of less important variables to zero. By using lasso, we do need to standardize the predictors, however, the use of glmnet() will standardize the predictors automatically.

```r
#Set a seed for reproducible results
set.seed(8675309)

#CV to choose lambda
logit_cv <- cv.glmnet(x=as.matrix(train |> dplyr::select(Perimeter3, Concave_Points3,
                                                          Concave_Points1, Area3, Radius3,
                                                          Concavity1, Concavity3,
                                                          Texture3, Perimeter1)),
                      y=train$Diagnosis,
                      family=binomial(),
                      alpha=1)
lambda <- logit_cv$lambda.1se
```

The optimal tuning parameter lambda that is the largest withing 1 standard error the the minimum lambda that minimizes the cross-validation error, is $\lambda = 0.0149111608370971$. The predictors considered for the logistic regression model were those found from the tuned random forest model that had an overall importance greater than 5%. This includes: Perimeter3, Concave_Points3, Concave_Points1, Area3, Radius3, Concavity1, Concavity3, Texture3, and Perimeter1.

```r
#Set a seed for reproducible results
set.seed(8675309)

#Refit the model with the optimal lambda
tuned_logit <- cv.glmnet(x=as.matrix(train |> dplyr::select(Perimeter3, Concave_Points3,
                                                          Concave_Points1, Area3, Radius3,
                                                          Concavity1, Concavity3,
                                                          Texture3, Perimeter1)),
                      y=train$Diagnosis,
                      family=binomial(),
                      alpha=1,
                      lamda=logit_cv$lambda.1se)
#Estimated Coefficients
coef(tuned_logit)
```

```
10 x 1 sparse Matrix of class "dgCMatrix"
                         s1
```

```
(Intercept)      -14.3098086
Perimeter3         .
Concave_Points3   18.7797158
Concave_Points1   23.7120278
Area3              .
Radius3            0.4626055
Concavity1         .
Concavity3         0.2122881
Texture3           0.1093809
Perimeter1         .
```

Generalized Additive Model Using Natural Cubic Splines

The generalized additive model is a type of nonparametric model. The tuning parameter for the model is the degrees of freedom. This type of model is not used for inference but is for prediction. In terms of variable selection, this model does not perform variable selection. For generalized additive models, it is not necessary to standardize the predictors.

```r
#Set a seed for reproducible results
set.seed(8675309)

#Test different gams by changing the degrees of freedom uniformly
glm_3 <- glm(Diagnosis ~ ns(Perimeter3, df=3)+ns(Concave_Points3,df=3)
             +ns(Concave_Points1,df=3)+ns(Area3,df=3)+ns(Radius3,df=3)
             +ns(Concavity1,df=3)+ns(Concavity3,df=3)+ns(Texture3,df=3)
             +ns(Perimeter1,df=3),
             data=train,
             family=binomial())
glm_4 <- glm(Diagnosis ~ ns(Perimeter3, df=4)+ns(Concave_Points3,df=4)
             +ns(Concave_Points1,df=4)+ns(Area3,df=4)+ns(Radius3,df=4)
             +ns(Concavity1,df=4)+ns(Concavity3,df=4)+ns(Texture3,df=4)
             +ns(Perimeter1,df=4),
             data=train,
             family=binomial())
glm_5 <- glm(Diagnosis ~ ns(Perimeter3, df=5)+ns(Concave_Points3,df=5)
             +ns(Concave_Points1,df=5)+ns(Area3,df=5)+ns(Radius3,df=5)
             +ns(Concavity1,df=5)+ns(Concavity3,df=5)+ns(Texture3,df=5)
             +ns(Perimeter1,df=5),
             data=train,
             family=binomial())

#Compare the AIC of the three models
AIC(glm_3,glm_4,glm_5)
```

```
      df      AIC
glm_3 28 560.6111
glm_4 37  74.0000
glm_5 46  92.0000
```

Using AIC to compare the performance of the three models, model 4 with degrees of freedom equal to 4 has the lowest AIC. Thus the optimal degrees of freedom will be equal to 4. The predictors considered for the logistic regression model were those found from the tuned random forest model that had an overall importance greater than 5%. This includes: Perimeter3, Concave_Points3, Concave_Points1, Area3, Radius3, Concavity1, Concavity3, Texture3, and Perimeter1.

```
#The optimal model has degrees of freedom of 4
tuned_gam <- glm(Diagnosis ~ ns(Perimeter3, df=4)+ns(Concave_Points3,df=4)
             +ns(Concave_Points1,df=4)+ns(Area3,df=4)+ns(Radius3,df=4)
             +ns(Concavity1,df=4)+ns(Concavity3,df=4)+ns(Texture3,df=4)
             +ns(Perimeter1,df=4),
             data=train,
             family=binomial())
```

Support Vector Machine Model Using LASSO Penalty

The support vector machine model is a type of nonparametric model as it uses both parametric and nonparametric methods. The tuning parameter for the model depends on the model, but in the case of a LASSO penalty we consider the tuning parameter of lambda λ. This type of model is mainly used for prediction. In terms of variable selection, this model does perform variable selection when using the penalized method of LASSO. For support vector machines, it is not required to standardize the predictors though we commonly will standardize them. In this case we will standardize the predictors.

```
#Set a seed for reproducible results
set.seed(8675309)

#Turn training Diagnosis into Binary
train_standard$Diagnosis_Binary <- as.numeric(ifelse(train_standard$Diagnosis == "M", 1,0))

#CV to choose lambda
spr.cv <- cv.sparseSVM(X=as.matrix(train_standard[,-c(31,32)]),
                       y=as.matrix(train_standard[,32]),alpha=1)
```

```
#Minimum Lambda
min_lambda <-spr.cv$lambda.min
min_lambda
```

```
[1] 0.05886309
```

```
as.matrix(coef(spr.cv, lambda=spr.cv$lambda.min))
```

```
                           [,1]
(Intercept)          0.97377049
Radius1             -0.29484716
Texture1            -0.09004880
Perimeter1          -0.30565999
Area1               -0.36592934
Smoothness1         -0.07253032
Compactness1        -0.23813156
Concavity1          -0.27718222
Concave_Points1     -0.34306618
Symmetry1           -0.07712168
Fractal_Dimension1   0.00000000
Radius2             -0.36202022
Texture2             0.00000000
Perimeter2          -0.39902926
Area2               -0.57005185
Smoothness2          0.00000000
Compactness2        -0.07191397
Concavity2          -0.03607388
Concave_Points2     -0.10041265
Symmetry2            0.00000000
Fractal_Dimension2 -0.00296905
Radius3             -0.34008470
Texture3            -0.10176041
Perimeter3          -0.34458732
Area3               -0.41644271
Smoothness3         -0.10508970
Compactness3        -0.25863951
Concavity3          -0.21121293
Concave_Points3     -0.27386062
Symmetry3           -0.17007879
Fractal_Dimension3 -0.10837483
```

The optimal tuning parameter lambda is $\lambda = 0.0588630937602297$. The predictors chosen from the use of the LASSO penalty are shown above in the table where the coefficients are different than zero.

```
#Optimal model has an optimal lambda defined from before
tuned_svm <- cv.sparseSVM(X=as.matrix(train_standard[,-c(31,32)]),
                       y=as.matrix(train_standard[,32]),lambda=spr.cv$lambda.min,
                       alpha=1)
```

Model Testing

```
#KNN Prediction
knn_pred <- predict(tuned_knn, newdata=test_standard)
#KNN Accuracy
knn_accuracy <- mean(knn_pred == test_standard$Diagnosis)

#Single Tree Prediction
tree_pred <- predict(tuned_tree,  type="class", newdata = test)
#Single Tree Accuracy
tree_accuracy <- mean(tree_pred == test$Diagnosis)

#Random Forest Prediction
rforest_pred <- predict(tuned_rforest, newdata=test)
rforest_accuracy <- mean(rforest_pred == test$Diagnosis)

#Logistic Regression Prediction
logit_pred <- predict(tuned_logit,
                      newx=as.matrix(test |> dplyr::select(Perimeter3, Concave_Points3,
                                                           Concave_Points1, Area3, Radius3,
                                                           Concavity1, Concavity3,
                                                           Texture3, Perimeter1)),
                      s=logit_cv$lambda.1se,
                      type="response")
#Convert Logistic Regression Predictions to Class
logit_pred_class <- ifelse(logit_pred>0.5, "M", "B")
logit_accuracy <- mean(logit_pred_class == test$Diagnosis)

#Generalized Additive Model Prediction
gam_pred <- predict(tuned_gam, newdata = test, type="response")
#Convert Generalized Additive Model Predictions to Class
gam_pred_class <- ifelse(gam_pred>0.5, "M", "B")
```

```r
gam_accuracy <- mean(gam_pred_class == test$Diagnosis)

#Support Vector Machine Prediction
svm_pred<- predict(spr.cv, X = as.matrix(test_standard[,-31]), type="class",
                   lambda=spr.cv$lambda.min)
svm_pred_class <- ifelse(svm_pred==1, "M", "B")
svm_accuracy <- mean(svm_pred_class == test_standard$Diagnosis)

#Print table of all of the accuracies
accuracies <- data.frame(
  Model=c("K-Nearest Neighbors","Single Tree", "Ensemble Tree (Random Forest)",
          "Regularized Logistic Regression with LASSO",
          "Generalized Additive Model with Natural Cubic Splines",
          "Support Vector Machine with LASSO"),
  Accuracy=c(knn_accuracy,tree_accuracy,rforest_accuracy,logit_accuracy,
             gam_accuracy,svm_accuracy)
)

print(accuracies)
```

```
                                                  Model  Accuracy
1                                   K-Nearest Neighbors 0.9649123
2                                           Single Tree 0.9122807
3                         Ensemble Tree (Random Forest) 0.9473684
4            Regularized Logistic Regression with LASSO 0.9707602
5 Generalized Additive Model with Natural Cubic Splines 0.9590643
6                     Support Vector Machine with LASSO 0.9473684
```

Best Model

Based on accuracy, the best model to predict breast cancer using the data from this data set is the Regularized Logistic Regression with LASSO. The accuracy for this optimal model is 0.9707602 which means that 97.07602% of the time the model made a correct prediction on the test data set. This model is also useful as it is highly interpretable.

```r
#Fit the best model to the entire data set
#Set a seed for reproducible results
set.seed(8675309)

#Refit the model with the optimal lambda
```

```r
final_logit <- cv.glmnet(x=as.matrix(celldata |> dplyr::select(Perimeter3, Concave_Points3,
                                                         Concave_Points1, Area3, Radius3,
                                                         Concavity1, Concavity3,
                                                         Texture3, Perimeter1)),
                      y=celldata$Diagnosis,
                      family=binomial(),
                      alpha=1,
                      lamda=logit_cv$lambda.1se)
#Estimated Coefficients
coef(final_logit)
```

```
10 x 1 sparse Matrix of class "dgCMatrix"
                         s1
(Intercept)     -19.2280165
Perimeter3        .
Concave_Points3  20.3213701
Concave_Points1  26.3840129
Area3             .
Radius3           0.6218394
Concavity1        .
Concavity3        0.8293441
Texture3          0.1731363
Perimeter1        .
```

Log Odds of Tumor Malignancy (Y=1) fitted to the entire dataset:

$$log(\frac{P(Y=1)}{P(Y=0)}) = -19.2280165 + 20.3213701(ConcavePoints3) + 26.3840129(ConcavePoints1)$$

$$+0.6218394(Radius3) + 0.8293441(Concavity3) + 0.1731363(Texture3)$$

It can be noticed that a one unit increase in both feature Concave Points 3 and feature Concave Points 1 significantly increase the log odds. This makes sense as malignant tumors are associated with having more rigid perimeters.