Project 2

Asif Hasan - 301376671

2023-11-26

Section 1: ROC Curves and Binary Classification

1.1 Importance of Thresholds in Binary Classification

In binary classification, the default threshold is typically set at 0.5. According to this threshold, any observation with a predicted probability greater than or equal to 0.5 is classified as the positive class, while those below 0.5 are classified as the negative class. However, in domain-specific situations, it may be more desirable to adjust this threshold. For instance, if the goal is to avoid incorrectly classifying negative cases as positive, the threshold can be raised. Conversely, if the aim is to prevent the misclassification of positive cases as negative, the threshold can be lowered.

1.2 Impact of Threshold Choice on Error Rate

The threshold selection is pivotal in determining the error rates of False Positive (Type I), where actual negative cases are classified as positive, and False Negative (Type II), where actual positive cases are classified as negative.

When the threshold is raised from its default value of 0.5, there is a tendency to reduce Type I errors, as the likelihood of classifying negative cases as positive decreases. However, this adjustment may lead to an increase in Type II errors, as there is a higher chance of missing positive cases.

Conversely, lowering the threshold from the default value of 0.5 may decrease Type II errors, as positive cases are more likely to be correctly identified. Nevertheless, this adjustment could result in an increase in Type I errors, as there is a greater likelihood of classifying negative cases as positive.

1.3 True Positive Rate and False Positive Rate

The True Positive Rate (TPR), also referred to as sensitivity, represents the proportion of actual positives correctly classified by the model. It can be calculated using the formula:

$$TPR = \frac{TruePositive}{TruePositive + FalseNegative}$$

On the other hand, the False Positive Rate (FPR), also known as one minus specificity, is the proportion of actual negatives incorrectly classified as positives, given a specific threshold. The calculation is expressed as:

$$FPR = \frac{FalsePositive}{FalsePositive + TrueNegative}$$

1.4 Understanding ROC Curves

The ROC curve serves as a visual representation of a classifier's performance across a range of threshold values from 0.0 to 1.0. It illustrates the relationship between True Positive Rate (TPR), displayed on the y-axis, and False Positive Rate (FPR), presented on the x-axis. The curve showcases how these rates evolve as the threshold value changes. An ideal ROC curve hugs the top-left corner, reflecting an optimal classifier with high TPR and low FPR across various thresholds.

1.5 Area Under the ROC Curve (AUC)

The AUC, or Area Under the ROC Curve, serves as a comprehensive metric summarizing the overall performance of a binary classifier across all possible threshold values. It is particularly useful for comparing different classifiers, as it considers their performance over the entire range of thresholds. A larger AUC corresponds to a better classifier. The AUC value ranges between 0.0 and 1.0, where 1.0 indicates the best possible performance, 0.5 signifies a classifier that performs no better than random chance, and a value below 0.5 indicates a classifier that performs worse than random chance.

1.6 Comparison of Two Methods

For the binary classification task of identifying breast cancer, we employed two methods: LASSO-λ_{min} version of Logistic Regression and Default Random Forest. The dataset comprises nine integer features, each ranging from a minimum value of 1 to a maximum value of 10; hence, feature scaling was not applied.

Using the test set, the AUC value for LASSO Logistic Regression was found to be 0.9924, with an optimal threshold of 0.1154. Similarly, for Default Random Forest, the AUC value was 0.9915, with an optimal threshold of 0.2. It is noteworthy that low threshold values also work well for this classification problem, as we want to avoid classifying cancer cases as non-cancer cases.

While the AUC values of both methods are very similar, indicating comparable performance, a notable distinction arises in the optimal threshold values between the two methods. The misclassification rate for both methods on the test set is 0.0244, aligning with the AUC value obtained from the ROC curve.

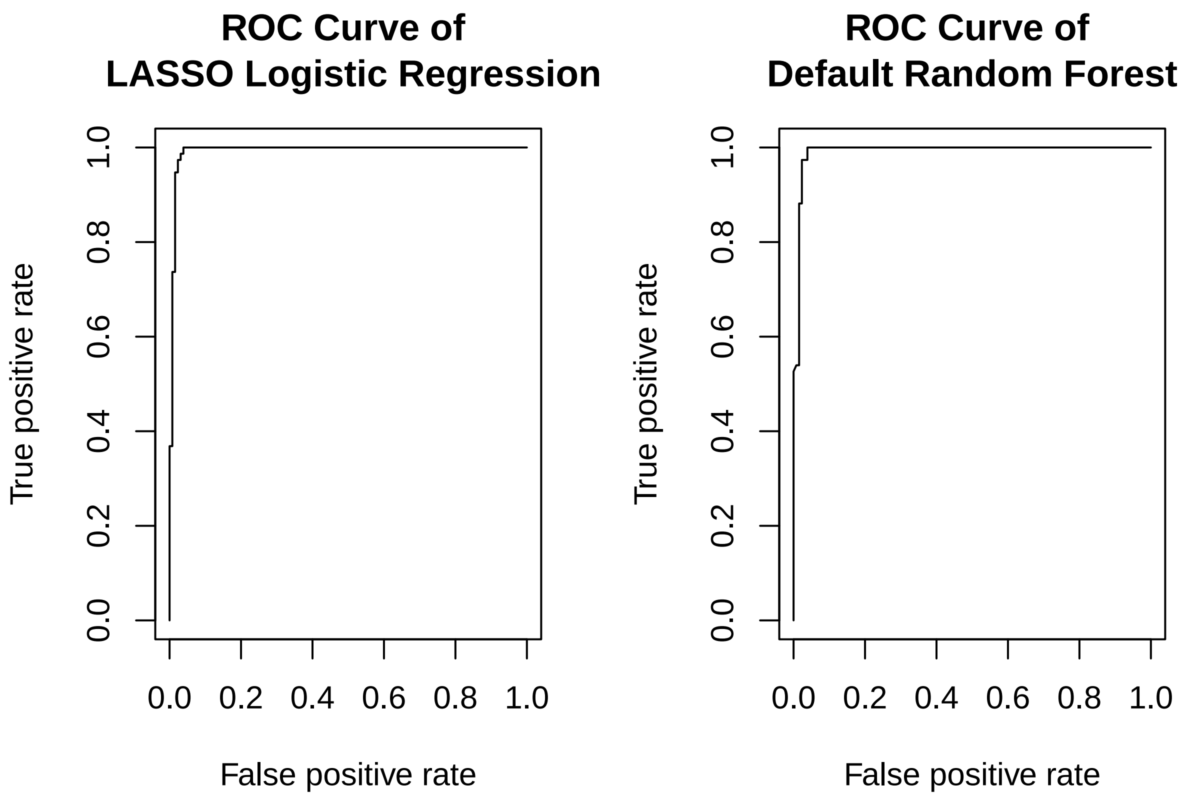
The disparity in optimal threshold values, coupled with a consistent test error, highlights the reliability of the ROC curve as a tool for identifying optimal thresholds in classification tasks and achieving the best test error.

1.7 Code Illustration with Breast Cancer Wisconsin (Original) Data

```r
# data from: https://datahub.io/machine-learning/breast-w#data
# read data and clean the data
cancer <- read.csv("breast-w.csv", header=TRUE)
cancer2 <- na.omit(cancer)
cancer2$Class <- as.factor(cancer2$Class)
# set seed
set.seed(4099183)
# generate sample using sampling fraction of 0.7:0.3
reorder <- sample.int(nrow(cancer2))
set <- ifelse(test = (reorder < 0.70 * nrow(cancer2)), yes = 1, no=2)
# fit lasso version of logistic regression
model.logit.cv <- cv.glmnet(
  x = as.matrix(cancer2[set==1, 1:9]),
  y = cancer2[set==1, 10], family = "binomial")
```

```r
# predict probabilities using lamda-min of logit on the test set
logit.pred.test <- predict(model.logit.cv, type = "response",
                           s = model.logit.cv$lambda.min,
                           newx = as.matrix(cancer2[set==2, 1:9]))
# fit random forest using default parameters
model.rf <- randomForest(data=cancer2[set==1, ], Class ~ .)
# predict probabilities of rf on the test set
rf.pred.test <- predict(model.rf, newdata=cancer2[set==2, 1:9], type="vote")
```

```r
# plot roc curve of both models
par(mfrow = c(1, 2))
plot(performance(prediction(logit.pred.test, cancer2[set==2, 10]),"tpr","fpr"),
     main="ROC Curve of \n LASSO Logistic Regression")
plot(performance(prediction(rf.pred.test[, 2], cancer2[set==2, 10]),"tpr","fpr"),
     main="ROC Curve of \nDefault Random Forest")
```



```r
# compute auc of logit
auc.logit <- auc(roc(cancer2[set==2, 10], logit.pred.test[, 1]))
print(paste("The AUC of LASSO Version of Logistic Regression:", round(auc.logit, 4)))
```

```
## [1] "The AUC of LASSO Version of Logistic Regression: 0.9924"
```

```r
# get the best threshold for logit
threshold.logit <- coords(roc(cancer2[set==2, 10], logit.pred.test[, 1]), "best", ret = "threshold")
print(paste("The optimal threshold for LASSO Logistic Regression:", round(threshold.logit, 4)))
```

```
## [1] "The optimal threshold for LASSO Logistic Regression: 0.1154"
```

```
# get predicted class for logit
logit.pred.class <- ifelse(logit.pred.test[, 1] > threshold.logit[, 1], "malignant", "benign")
# compute misclassification rate of logit
misclass.test.logit <- mean(ifelse(logit.pred.class == cancer2[set==2, 10], yes=0, no=1))
print(paste("The misclassification rate of LASSO Logistic Regression:", round(misclass.test.logit, 4)))
```

```
## [1] "The misclassification rate of LASSO Logistic Regression: 0.0244"
```

```
# compute auc of rf
auc.rf <- auc(roc(cancer2[set==2, 10], rf.pred.test[, 2]))
print(paste("The AUC of Default Random Forest:", round(auc.rf, 4)))
```

```
## [1] "The AUC of Default Random Forest: 0.9915"
```

```
# get the best threshold for rf
threshold.rf <- coords(roc(cancer2[set==2, 10], rf.pred.test[, 2]), "best", ret = "threshold")
print(paste("The optimal threshold for Default Random Forest:", round(threshold.rf, 4)))
```

```
## [1] "The optimal threshold for Default Random Forest: 0.2"
```

```
# get predicted class for rf
rf.pred.class <- ifelse(rf.pred.test[, 2] > threshold.rf[, 1], "malignant", "benign")
# compute misclassification rate of rf
misclass.test.rf <- mean(ifelse(rf.pred.class == cancer2[set==2, 10], yes=0, no=1))
print(paste("The misclassification rate of Default Random Forest:",
            round(misclass.test.rf, 4)))
```

```
## [1] "The misclassification rate of Default Random Forest: 0.0244"
```

Section 2: Support Vector Machines (SVM) for Binary Classification

2.1 Overview of SVM Classifier

Support Vector Machine (SVM) classifier aims to find the hyperplane that best separates different classes in the feature space. The feature space can be enlarged using kernel functions that enable SVM to implicitly map the input features into a higher-dimensional space, making it possible to find a hyperplane that separates classes even when they are not linearly separable in the original feature space. The optimization objective is to maximize the margin between classes, which is the smallest perpendicular distance from the observations from either class to the hyperplane. SVM seeks to find the hyperplane that not only separates the classes but also maximizes this margin.

2.2 Difference from Other Classifiers

The support vector machine (SVM) classifier's decision rule is based only on a potentially small subset of the training observations, also known as support vectors (observations that lie directly on the margin, or on the wrong side of the margin for their class). This property of SVM that only the support vectors have a direct impact the decision boundary is distinct from some of the other classification methods that we have seen so far in class.

2.3 Advantages and Disadvantages

Advantages of SVM:

- Robust to outliers
- Effective in higher-dimensional feature space
- Kernel functions allow accommodation of both linear and versatile non-linear decision boundaries between classes

Disadvantages of SVM:

- Difficult to understand and interpret the fitted model
- Choice of kernel and other tuning parameters may impact predictive performance
- Computationally intensive, especially with large datasets; even though the use of kernel functions decreases the overall computational load

2.4 Implementation in R and Test Error

In R, the ‘e1071’ library provides the ‘svm’ function for implementing a Support Vector Machine (SVM) classifier. Initial tuning parameters include the cost parameter (C), which determines the width of the margins and controls the bias-variance trade-off, and the kernel function (K). Depending on the kernel function, there are additional tuning parameters. For instance, the polynomial kernel requires tuning the degree (d), representing the degree of the polynomial to be used. In contrast, the radial kernel requires tuning gamma (γ), which controls the shape of the radial kernel function.

Given the two-step tuning parameters, we conducted two iterations of 5-fold cross-validation. In the first iterations, we explored C values form 1 to 3 and three functions of K: “linear”, “polynomial”, and “radial.” We found that $C = 1$ and $K = polynomial$ provides the lowest misclassification error. In the final iteration, we utilized the tuned values of parameters C and K and explored d values from 1 to 10, and found that $d = 3$ provides the lowest misclassification error among other values (code available in *rmd* file).

After tuning the parameters, the optimal values are $C = 1$, $K = polynomial$, and $d = 3$, yielding the best misclassification error for this method. The code below utilizes the same training and test sets from Section 1, and employs the ROC curve metric to determine the optimal threshold for classification. On the test set, the SVM classifier achieves a misclassification rate of 0.0146 using a threshold of 0.1619. This represents the best test error among the three methods used for the classification problem. Furthermore, the SVM method achieves the highest AUC value of 0.995 on the test set, consistent with the observed test error. Thus, the SVM demonstrates superior performance in making predictions for this classification problem.

```r
# fit svm
model.svm <- svm(x = as.matrix(cancer2[set==1, 1:9]), y = cancer2[set==1, 10],
                 kernel = "polynomial", cost = 1, degree = 3, probability=TRUE)
# get predict probabilities of svm on the test set
svm.pred.test <- predict(model.svm, probability = TRUE,
                         newdata = as.matrix(cancer2[set==2, 1:9]))
svm.pred.test <- attributes(svm.pred.test)$probabilities
```

```r
# compute auc of svm
auc.svm <- auc(roc(cancer2[set==2, 10], svm.pred.test[, 2]))
print(paste("The AUC of SVM:", round(auc.svm, 4)))
```

```
## [1] "The AUC of SVM: 0.995"
```

```r
# get the best threshold for svm
threshold.svm <- coords(roc(cancer2[set==2, 10], svm.pred.test[, 2]), "best", ret = "threshold")
print(paste("The optimal threshold for SVM:", round(threshold.svm, 4)))
```

```
## [1] "The optimal threshold for SVM: 0.1619"
```

```r
# get predicted class for svm
svm.pred.class <- ifelse(svm.pred.test[, 2] > threshold.svm[, 1], "malignant", "benign")
# compute misclassification rate of svm
misclass.test.svm <- mean(ifelse(svm.pred.class == cancer2[set==2, 10], yes=0, no=1))
print(paste("The misclassification rate of optimal SVM:", round(misclass.test.svm, 4)))
```

```
## [1] "The misclassification rate of optimal SVM: 0.0146"
```

References

Meyer, D. (2023, February 1). svm: Support Vector Machines. In RDocumentation. Retrieved from https://www.rdocumentation.org/packages/e1071/versions/1.7-13/topics/svm

Rickert, J. (2019, March 1). Some R Packages for ROC Curves. In R Views. Retrieved from https://rviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/

Robin, X. (2023, November 1). pROC. In RDocumentation. Retrieved from https://www.rdocumentation.org/packages/pRO

Wikipedia contributors. (2023, November 4). Sensitivity and specificity. In Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Sensitivity_and_specificitybe