Analysis of Cervical Cancer Risk Factors using SVM

Mariana Canelas Pais

2024-04-02

Introduction

Cervical cancer is one of the leading causes of cancer-related deaths among women worldwide. The early identification of associated risk factors can significantly contribute to the prevention and effective treatment of this disease. This report focuses on analyzing the dataset concerning cervical cancer risk factors made available by the UCI repository, collected at 'Hospital Universitario de Caracas' in Caracas, Venezuela. The dataset comprises demographic information, habits, and historic medical records of 858 patients. Some patients chose not to answer certain questions due to privacy concerns, leading to missing values in the dataset.

Dataset Characteristics

The dataset is multivariate, covering the health and medicine domain, specifically aimed at classification tasks. It includes both integer and real feature types across various variables.

Variables Description

The dataset contains the following variables, among others, providing a comprehensive overview of each patient's demographic background, habits, and medical history:

- **Age** (int)
- **Number of sexual partners** (int)
- **First sexual intercourse** (int)
- **Number of pregnancies** (int)
- **Smokes** (bool)
- **Smokes (years)** (bool)
- **Smokes (packs/year)** (bool)
- **Hormonal Contraceptives** (bool)
- **Hormonal Contraceptives (years)** (int) contraceptives.
- **IUD** (bool)
- **IUD (years)** (int)
- **STDs** (bool)
- **STDs (number)** (int)
- **STDs: condylomatosis** (bool)
- **STDs: cervical condylomatosis** (bool)
- **STDs: vaginal condylomatosis** (bool)
- **STDs: vulvo-perineal condylomatosis** (bool)
- **STDs: syphilis** (bool)
- **STDs: pelvic inflammatory disease** (bool)
- **STDs: genital herpes** (bool)
- **STDs: molluscum contagiosum** (bool)
- **STDs: AIDS** (bool)
- **STDs: HIV** (bool)

- **STDs: Hepatitis B** (bool)
- **STDs: HPV** (bool)
- **STDs: Number of diagnosis** (int)
- **STDs: Time since first diagnosis** (int)
- **STDs: Time since last diagnosis** (int)
- **Dx:Cancer** (bool) - target variable
- **Dx:CIN** (bool) - target variable
- **Dx:HPV** (bool) - target variable
- **Dx** (bool) - target variable
- **Hinselmann** (bool) - target variable
- **Schiller** (bool) - target variable
- **Cytology** (bool) - target variable
- **Biopsy** (bool) - target variable

Methods

The analysis conducted in this report aims to utilize Support Vector Machines (SVM) to classify instances into categories indicating the presence or absence of cervical cancer based on the dataset's variables.

Imports and Data Preparation

The following R packages were integral to our analysis: - **e1071**: This package is crucial for implementing Support Vector Machines (SVM) in R. It provides functionalities for both linear and non-linear SVM models, including those with radial basis function (RBF) kernels, enabling complex classification tasks. - **caret**: The 'Classification And REgression Training' package offers a comprehensive framework for training and evaluating machine learning models in R. It simplifies model tuning, cross-validation, and performance assessment, supporting a wide array of models including SVM. caret's extensive set of tools for preprocessing data, such as scaling and balancing, are essential for preparing our dataset for analysis. - **pROC**: Specializing in Receiver Operating Characteristic (ROC) analysis, the pROC package is used to evaluate the performance of our SVM models. It provides advanced capabilities for computing and comparing ROC curves, an important aspect of understanding model efficacy in binary classification problems like ours. - **ROSE**: Dealing with imbalanced datasets is a common challenge in classification tasks. The 'Random OverSampling Examples' (ROSE) package is designed to address this issue by generating synthetic samples. In our analysis, ROSE is employed to balance the classes in our dataset, ensuring that our models learn to identify both majority and minority classes effectively.

```
library("e1071")
library("caret")
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library("pROC")
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library("ROSE")
```

```
## Loaded ROSE 0.0-4
```

Reading the Dataset

The dataset is loaded from a CSV file into an R dataframe for manipulation and analysis.

```
fp <- "../data/cervical.csv"
ds <- read.csv(fp)
```

Data Cleaning

To ensure the quality and usability of the data, we perform several cleaning steps. This includes replacing "?" with NA for missing values, converting certain columns to factors to reflect their categorical nature, and transforming other columns to numeric types to enable quantitative analysis.

Handling Missing Values and Data Types

First, we replace any "?" values with NA across specified columns, acknowledging the presence of missing data in both categorical and numeric fields.

```
# List of columns to convert to factors
col_to_factor <- c(
  "Smokes",
  "Hormonal.Contraceptives",
  "IUD",
  "STDs",
  "STDs.condylomatosis",
  "STDs.cervical.condylomatosis",
  "STDs.vaginal.condylomatosis",
  "STDs.vulvo.perineal.condylomatosis",
  "STDs.syphilis",
  "STDs.pelvic.inflammatory.disease",
  "STDs.genital.herpes",
  "STDs.molluscum.contagiosum",
  "STDs.AIDS",
  "STDs.HIV",
  "STDs.Hepatitis.B",
  "STDs.HPV",
  "Dx.Cancer",
  "Dx.CIN",
  "Dx.HPV",
  "Dx",
  "Hinselmann",
  "Schiller",
  "Citology",
  "Biopsy"
)

# List of columns to convert to numeric
col_to_numeric <- c(
  "Number.of.sexual.partners",
  "First.sexual.intercourse",
  "Num.of.pregnancies",
  "Smokes..years.",
  "Smokes..packs.year.",
  "Hormonal.Contraceptives..years.",
  "STDs..Number.of.diagnosis",
  "STDs..Time.since.first.diagnosis",
```

```
  "STDs..Time.since.last.diagnosis",
  "IUD..years.",
  "STDs..number."
)

# Replace "?" with NA and convert data types
ds[c(col_to_factor, col_to_numeric)] <- lapply(ds[c(col_to_factor, col_to_numeric)], function(x) {
  x[x == "?"] <- NA
  return(x)
})
```

Converting Data Types

After handling missing values, we convert the specified columns to their appropriate data types: factors for categorical variables and numeric for continuous variables.

```
# Converting columns to factors
ds[col_to_factor] <- lapply(ds[col_to_factor], factor)

# Converting columns to numeric
ds[col_to_numeric] <- lapply(ds[col_to_numeric], as.numeric)
```

Adjusting Output Variable Category names

```
ds$Dx.Cancer <- factor(ds$Dx.Cancer,
                       levels = levels(ds$Dx.Cancer),
                       labels = make.names(levels(ds$Dx.Cancer), unique = TRUE))
```

Renaming Columns

To improve readability and simplify future references to the dataset columns, we rename them using a more consistent naming convention.

```
# Renaming the columns for better readability
names(ds) <- c("Age", "NumSexualPartners", "FirstSexualIntercoarse", "NumPregnancies", "Smokes", "Smoke
```

Variable Selection

To ensure the integrity of our analysis and prevent data leakage, we will remove variables that could introduce bias because they represent outcomes of diagnostics or are directly related to the target variable 'Dx.Cancer'.

```
# Excluding target variables from the dataset
ds <- ds[ , !(names(ds) %in% c("DxCIN", "DxHPV", "Dx", "Hinselmann", "Schiller", "Citology", "Biopsy"))]
colnames(ds)[colnames(ds) == "DxCancer"] <- "Dx"
```

In refining our dataset for the current exercise, we also undertook further variable selection.

```
# Final selection of variables for the current analysis
ds <- ds[ , (names(ds) %in% c("Dx", "Age", "NumSexualPartners", "FirstSexualIntercoarse", "NumPregnanci
```

Results

In this analysis, we aimed to balance our dataset to address the significant class imbalance present and subsequently trained Support Vector Machine (SVM) models using different kernels, namely linear and radial kernels. Here we present the process and findings of our modeling efforts.

Splitting Data into Training and Testing Sets

```
set.seed(7)
training_indices <- createDataPartition(ds$Dx, p = 0.8, list = FALSE)
training_data <- ds[training_indices, ]
testing_data <- ds[-training_indices, ]
```

Remove rows with any NA values and constant columns from the training data

```
# Remove NAs
training_data_cleaned <- na.omit(training_data)

# Remove constant variables from the dataset
constant_vars <- sapply(training_data_cleaned, function(x) length(unique(x))) == 1
training_data_cleaned <- training_data_cleaned[, !constant_vars]
```

Using ROSE to balance the classes in the cleaned training data

```
set.seed(123)
training_data_cleaned <- ovun.sample(Dx ~ ., data = training_data_cleaned, method = "both", p = 0.5, se

# Check the new distribution of the target variable after balancing
table(training_data_cleaned$Dx)
```

```
##
##  X0  X1
## 817 783
```

SVM Model Construction

```
# Setup for cross-validation
control <- trainControl(method="repeatedcv",
                        number=10,
                        summaryFunction=twoClassSummary,
                        classProbs=TRUE,
                        savePredictions=TRUE,
                        repeats=3)

# Train SVM with Linear Kernel on balanced data
set.seed(7)
svm_linear <- train(Dx ~ ., data = training_data_cleaned,
                    method="svmLinear",
                    metric="ROC",
                    trControl=control,
                    tuneLength=10)

# Train SVM with Radial Kernel on balanced data
set.seed(7)
svm_radial <- train(Dx ~ ., data = training_data_cleaned,
                    method="svmRadial",
                    metric="ROC",
                    trControl=control,
                    tuneLength=10)
```

Compare Model Performance

```
# Summarize and compare model performance
fit_models <- list(Linear = svm_linear, Radial = svm_radial)
results <- resamples(fit_models)
print(summary(results))
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: Linear, Radial
## Number of resamples: 30
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.     Max. NA's
## Linear 0.6824504 0.7373749 0.7581073 0.7554819 0.7722405 0.804878    0
## Radial 0.9762272 0.9887311 1.0000000 0.9947210 1.0000000 1.000000    0
##
## Sens
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Linear 0.6463415 0.7195122 0.7439024 0.7454632 0.7774390 0.8271605    0
## Radial 0.9506173 0.9756098 0.9878049 0.9836595 0.9969512 1.0000000    0
##
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu.     Max. NA's
## Linear 0.5443038 0.6038218 0.6666667 0.6684193 0.7079276 0.835443    0
## Radial 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.000000    0
```

```
# ROC plots
roc_data_linear <- roc(response=svm_linear$pred$obs, predictor=svm_linear$pred$X1)
```

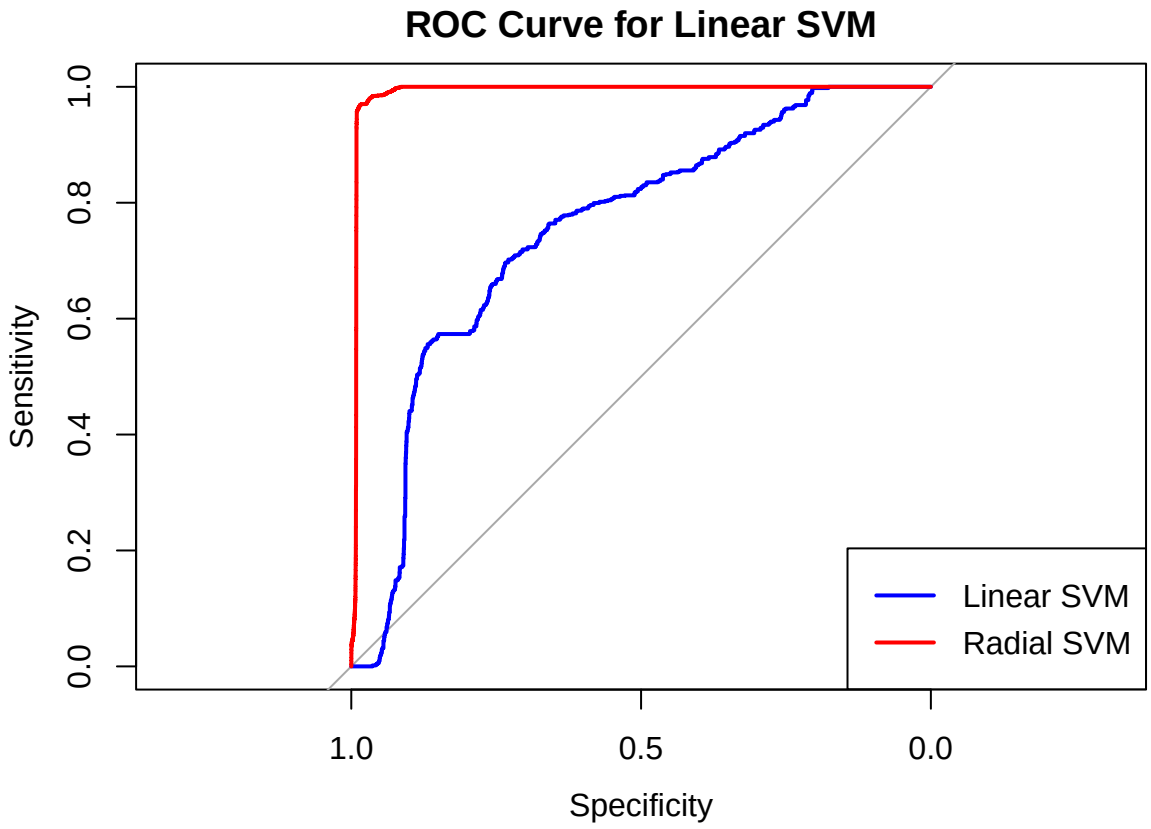
```
## Setting levels: control = X0, case = X1
```

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

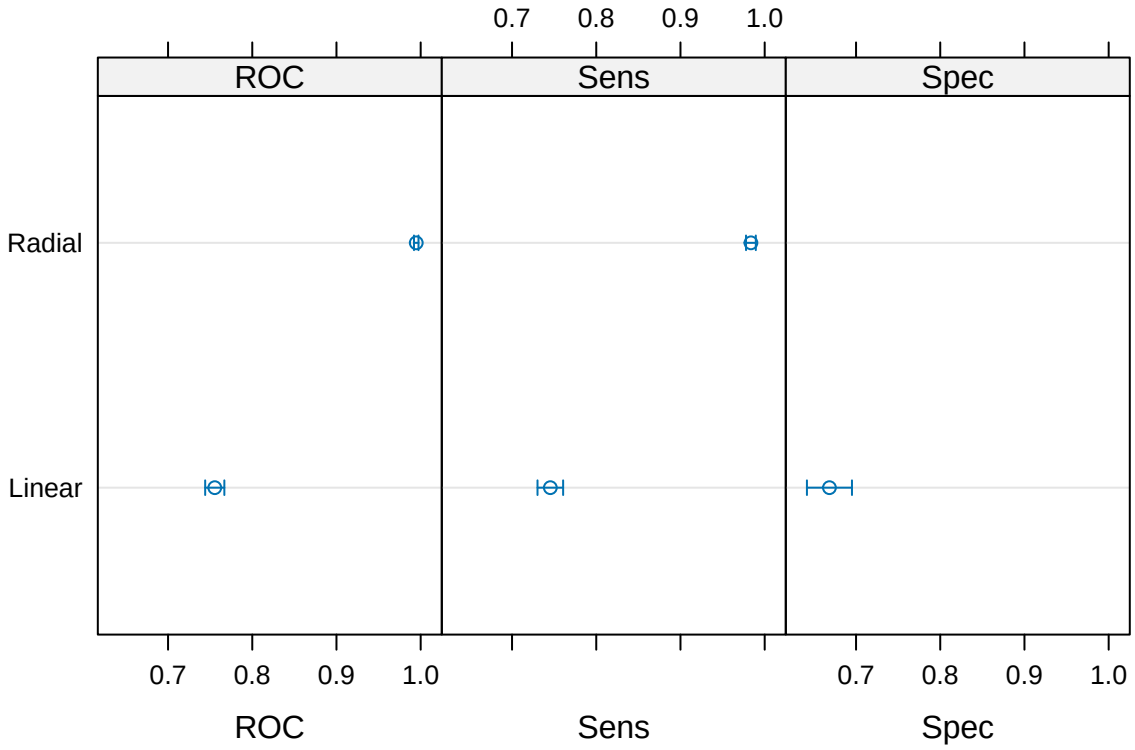
```
roc_data_radial <- roc(response=svm_radial$pred$obs, predictor=svm_radial$pred$X1)
```

```
## Setting levels: control = X0, case = X1
## Setting direction: controls < cases
```

```
plot(roc_data_linear, main="ROC Curve for Linear SVM", col="blue")
plot(roc_data_radial, add=TRUE, col="red")
legend("bottomright", legend=c("Linear SVM", "Radial SVM"), col=c("blue", "red"), lwd=2)
```




```
dotplot(results)
```



Confidence Level: 0.95

```r
# Inspect Models
print(svm_linear)
```

```
## Support Vector Machines with Linear Kernel
##
## 1600 samples
##    8 predictor
##    2 classes: 'X0', 'X1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1440, 1440, 1440, 1440, 1440, 1440, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7554819  0.7454632  0.6684193
##
## Tuning parameter 'C' was held constant at a value of 1
```

```r
getModelInfo("svmLinear")$svmLinear$parameters
```

```
##   parameter   class label
## 1         C numeric  Cost
```

```r
print(svm_radial)
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1600 samples
##    8 predictor
##    2 classes: 'X0', 'X1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1440, 1440, 1440, 1440, 1440, 1440, ...
## Resampling results across tuning parameters:
##
##   C       ROC        Sens       Spec
##     0.25  0.9729845  0.9396919  0.9012550
##     0.50  0.9820534  0.9425073  0.9285081
##     1.00  0.9883222  0.9600120  0.9447041
##     2.00  0.9883865  0.9649202  1.0000000
##     4.00  0.9906746  0.9791830  1.0000000
##     8.00  0.9918693  0.9800060  1.0000000
##    16.00  0.9910846  0.9800060  1.0000000
##    32.00  0.9946751  0.9799859  1.0000000
##    64.00  0.9947210  0.9836595  1.0000000
##   128.00  0.9947047  0.9898123  1.0000000
##
## Tuning parameter 'sigma' was held constant at a value of 0.1653719
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1653719 and C = 64.
```

```r
getModelInfo("svmRadial")$svmRadial$parameters
```

```
##   parameter   class label
```

```
## 1     sigma numeric Sigma
## 2         C numeric  Cost
```

Tuning the radial SVM model

```
myGrid <- expand.grid(C = c(1, 2, 4, 8), sigma = 0.09382649)

set.seed(7)
fit_svm_radial_tune <- train(Dx ~ ., data = training_data_cleaned,
                             method = "svmRadial",
                             metric = "ROC",
                             trControl = control,
                             tuneGrid = myGrid)

# Summarize accuracy of models including the tuned radial model
fit_models <- list(Linear = svm_linear, Radial = svm_radial, TunedRadial = fit_svm_radial_tune)
results <- resamples(fit_models)
print(summary(results))
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: Linear, Radial, TunedRadial
## Number of resamples: 30
##
## ROC
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Linear      0.6824504 0.7373749 0.7581073 0.7554819 0.7722405 0.8048780    0
## Radial      0.9762272 0.9887311 1.0000000 0.9947210 1.0000000 1.0000000    0
## TunedRadial 0.9585873 0.9777709 0.9875704 0.9866993 0.9974203 0.9992183    0
##
## Sens
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Linear      0.6463415 0.7195122 0.7439024 0.7454632 0.7774390 0.8271605    0
## Radial      0.9506173 0.9756098 0.9878049 0.9836595 0.9969512 1.0000000    0
## TunedRadial 0.8765432 0.9419226 0.9634146 0.9563234 0.9756098 1.0000000    0
##
## Spec
##                  Min.   1st Qu.    Median      Mean   3rd Qu.     Max. NA's
## Linear      0.5443038 0.6038218 0.6666667 0.6684193 0.7079276 0.835443    0
## Radial      1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.000000    0
## TunedRadial 0.8607595 0.9233204 0.9487179 0.9527372 1.0000000 1.000000    0
```

```
# ROC plots
roc_data_linear <- roc(response=svm_linear$pred$obs, predictor=svm_linear$pred$X1)
```

```
## Setting levels: control = X0, case = X1
```

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

```
roc_data_radial <- roc(response=svm_radial$pred$obs, predictor=svm_radial$pred$X1)
```

```
## Setting levels: control = X0, case = X1
## Setting direction: controls < cases
```

```
roc_data_radial_tuned <- roc(response=fit_svm_radial_tune$pred$obs, predictor=fit_svm_radial_tune$pred$X
```

```
## Setting levels: control = X0, case = X1
## Setting direction: controls < cases
```

```
plot(roc_data_linear, main="ROC Curve for Linear SVM", col="blue")
plot(roc_data_radial, add=TRUE, col="red")
plot(roc_data_radial_tuned, add=TRUE, col="green")
legend("bottomright", legend=c("Linear SVM", "Radial SVM", "Radial Tuned SVM"), col=c("blue", "red", "g
```