Project 2 Report

Author: Vibhuti Gandhi
Date: 2023-11-29
I read and acknowledge the SFU student academic integrity given here -
https://www.sfu.ca/policies/gazette/student/s10-01.html
For further detail regarding the code chunks and plots contained in this report, refer to the 452_Project2.Rmd

Problem & Data Description

The Pima Indians Diabetes database serves as the basis of the report. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The datasets consist of 8 medical explanatory variables and one response variable, which diagnostically tests whether or not a patient has diabetes. Explanatory variables include - Number of times pregnant, Plasma glucose concentration a 2 hours in an oral glucose tolerance test, Diastolic blood pressure (mm Hg), Triceps skin fold thickness (mm), 2-Hour serum insulin (mu U/ml), Body mass index (weight in kg/(height in m)^2), Diabetes pedigree function, and Age (years). Response variable is Diabetes Class, where class value 1 is interpreted as "tested positive for diabetes" and class value 0 is interpreted as "tested negative for diabetes". The objective is to utilize this dataset to build binary classification models and explain ROC (Receiver Operating Characteristic) curves and introduce SVM (Support Vector Machine) model for the task of binary classification.

Section 1. ROC Curves

1.1 Model Training

To understand ROC curves and its various concepts, 3 different classification models have been selected. These include - Logistic Regression, K-Nearest Neighbours (KNN), and Linear Discriminant Analysis (LDA).

```
log.fit = glm(diabetes~., train_df , family="binomial")
summary(log.fit)
```

Note that datasets were appropriately scaled to mean 0 and sd 1 before implementing KNN and LDA.

```
pred.knn <- knn(X.train, X.test, Y.train, k = 1)
```

```
fit.lda <- lda(X.train, Y.train)
pred.lda <- predict(fit.lda, X.test)$class
```

1.2 Confusion Matrix

The confusion matrix is a frequently used table in binary classification problems. It is used to evaluate the performance of a classification model. The columns in the confusion matrix represent the observed values and the rows represent the predicted values generated by the model. Comparing the observed values to the predicted values is useful in showing how many correct and incorrect predictions were made by the model.

The confusion matrix is composed of 4 main elements -

1. True Positive (TP): Instances that are actually positive and are correctly predicted as positive
2. False Positive (FP): Instances that are actually negative but are incorrectly predicted as positive
3. True Negative (TN): Instances that are actually negative and are correctly predicted as negative

4. False Negative (FN): Instances that are actually positive but are incorrectly predicted as negative

The confusion matrix for our 3 different classification models is as follows -

```
confusion.matrix.log <- table(pred.log, test_df[,9], dnn=c('Predicted', 'Observed'))
print(confusion.matrix.log)
```

```
##          Observed
## Predicted neg pos
##       neg 100  28
##       pos  16  27
```

```
confusion.matrix.knn <- table(pred.knn, Y.test, dnn=c('Predicted', 'Observed'))
print(confusion.matrix.knn)
```

```
##          Observed
## Predicted neg pos
##       neg  85  28
##       pos  20  37
```

```
confusion.matrix.lda <- table(pred.lda, Y.test, dnn=c('Predicted', 'Observed'))
print(confusion.matrix.lda)
```

```
##          Observed
## Predicted neg pos
##       neg 104  27
##       pos   7  32
```

1.3 Error Metrics

Once the confusion matrix and its 4 components have been plotted, they can be used to calculate a variety of error metrics. These error metrics are as follows-

1. Model accuracy: (TP + TN) / (TP + FP + FN + TN)
2. Precision (Positive Predictive Value): TP / (TP + FP)
3. Recall (Sensitivity or True Positive Rate): TP / (TP + FN)
4. Specificity (True Negative Rate): TN / (TN + FP)
5. F1 Score: 2 * (Precision * Recall) / (Precision + Recall)

Along with aforementioned metrics, 1 more error metrics are used to evaluate models. This is the misclassification rate, calculated using the formula: (FP + FN) / (TP + TN + FP + FN).

Logistic Regression	KNN	LDA
Misclassification Rate: 0.2573099	Misclassification Rate: 0.2823529	Misclassification Rate: 0.2
Model Accuracy: 0.7426901	Model Accuracy: 0.7176471	Model Accuracy: 0.8
Precision: 0.4909091	Precision: 0.5692308	Precision: 0.5423729
Recall: 0.627907	Recall: 0.6491228	Recall: 0.8205128
Specificity: 0.78125	Specificity: 0.7522124	Specificity: 0.7938931
F1 Score: 0.5510204	F1 Score: 0.6065574	F1 Score: 0.6530612

(Code found in Rmd file)

1.4 Classification Threshold

Classification threshold is a key parameter in Binary Classification which affects how the 2 classes are separated. When a binary classification model is trained on a dataset it outputs a number between 0 (negative) and 1 (positive) that signifies the closeness of that prediction to its respective class. The threshold indicates the boundary of separation between the two classes. By default, the probability threshold for most classification models is 0.5. This means, any prediction equal to or over 0.5 is regarded as 1 (or positive) and any prediction lower than 0.5 is regarded as 0 (or negative).

Classification threshold, however, is not a fixed value and it can be changed depending on the problem. Figuring out the optimum classification threshold is important to reducing errors and achieving a balance between True Positives and False Positives. This can be achieved by plotting ROC curves.

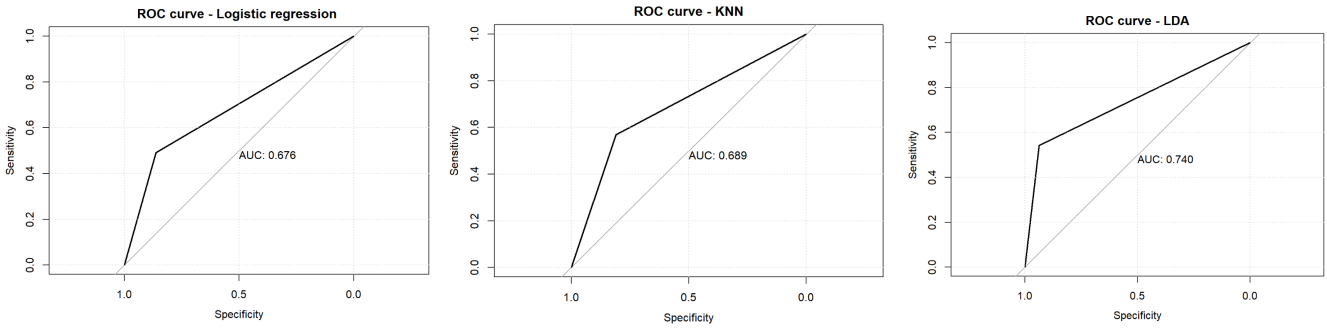
1.5 ROC curves

ROC curve stands for Receiver Operator Characteristic and it is a statistical plot that is used to evaluate the performance of binary classification models. This curve is generated by changing the classification threshold of the model and observing the trade-off between True positive rate against the False Positive rate. The area under the ROC curve is given by AUC (Area Under Curve) which gives a value representing the overall performance of the model across various. The higher the AUC-ROC value, the better the model as it will have better discriminatory power. AUC values can exist between 0 and 1. AUC of 0.5 indicates completely random performance and is given by a diagonal line. AUC < 0.5 indicates that the model's performance is worse than random chance showing that the model is making incorrect predictions. AUC > 0.5 indicates that the model's performance is better than random and it is able to make correct predictions. An AUC = 1.0 represents that a model is able to make 100% correct predictions and has perfectly understood the data. However, this is usually an indication of overfitting.

ROC can be plotted for all 3 models using the code -

```
test_roc <- roc(test_df$diabetes ~ pred.svm,
                plot = TRUE,
                print.auc = TRUE,
                main ="ROC curve - SVM")
```

ROC-AUC for all 3 models -



These ROC-AUC in combination with the error metrics calculated above show that the best performance is the LDA, followed by KNN and then Logistic Regression

Section 2. Support Vector Machines

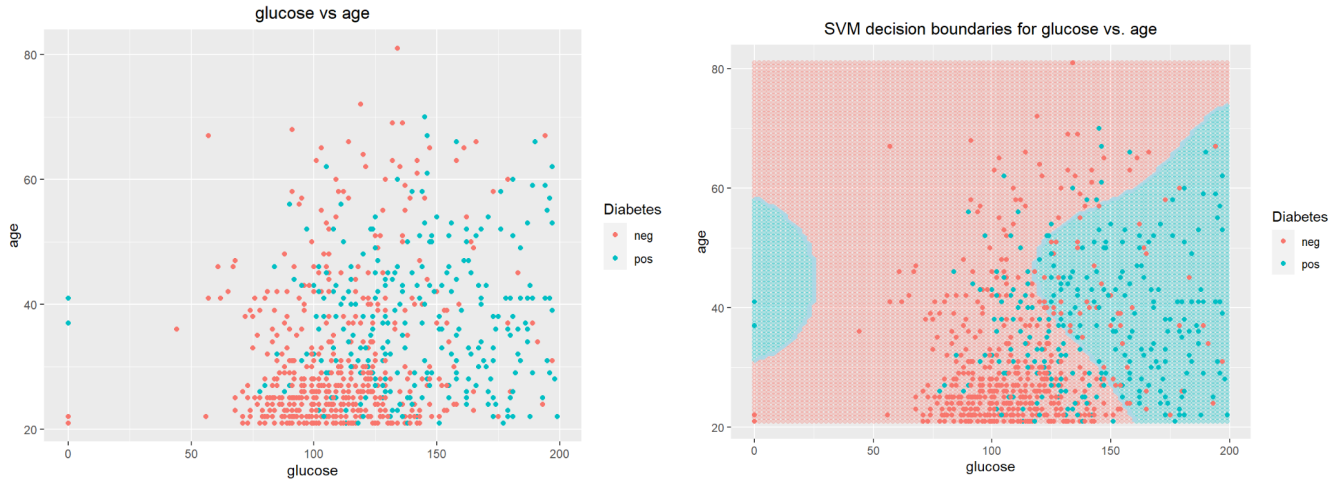
2.1 Introduction

Support Vector Machine (SVM(is a machine learning algorithm that can be utilized for classification and regression problems. SVMs are different from other types of classification algorithms like Logistic Regression and LASSO regression as they create non-linear boundaries and can work with complex class structures. SVMs are highly robust models that are sensitive to outliers and their performance can significantly increase by tuning their parameters. This report will be looking to demonstrate the application of SVM on a binary classification problem.

2.2 Support Vectors

When classification data is input into SVMs, the model likes to draw a boundary between the classes that best separates them. This line is called hyperplane and it is the decision boundary that separates the data points of one class from another. SVMs tries to find the hyperplane that will maximize the distance between itself and the nearest data points of each class (the margin). Support vectors are the data points that lie closest to the hyperplane. These are crucial in determining the position and orientation of the hyperplane. SVM focuses on these support vectors for defining the decision boundary.

The following graph shows the decision boundaries created by an SVM model on the glucose and age variables-



(Code found in Rmd file)

2.2 Kernel Trick

The kernel trick is a technique used by SVMs to handle non-linear decision boundaries. Kernel trick transforms the explanatory features into a higher-dimensional space. There are 3 distinct kernel functions - linear, polynomial, and radial basis function (RBF) kernels. Different kernel types are used to model diverse relationships between classes. The linear kernel is used when the data is linearly separable and the decision boundary is a straight line. Polynomial kernel is used to capture non-linear relationships in the data. The RBF kernel can model complex, non-linear decision boundaries.

2.3 Gaussian Parameter and Cost function

In our problem, we have implemented a radial (RBF) kernel. The formula for this is given by -

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

The parameter σ in this equation is called the Gaussian parameter and it controls the width of the Gaussian function used in the kernel formula. A small σ results in a narrow Gaussian function and more complex models. This leads to a more complex decision boundary and the SVM starts capturing outliers and noise. A larger σ produces a broader Gaussian function, smoothing the decision boundary. This leads to a more generalized model that is less sensitive to local variations in the data. The optimal σ is selected by keeping the bias-variance tradeoff in mind.

Cost function (C) refers to the penalty assigned to misclassified instances by the model. The formula for this is given by -

$$(\theta) = C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1 - y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_i^2$$

This function is meant to minimize θ and penalize the SVM for making mistakes. As the SVM finds a decision boundary that separates the data into different classes while maximizing the margin and minimizing classification errors, the cost function plays a crucial role in balancing the trade-off between achieving a wider margin and minimizing misclassifications.

In order to select the optimum values we tune are SVM model by supplying it a list of gaussian parameters and cost function-

```
tune.svm <- tune(svm, diabetes~., data = train_df,
                 kernel = "radial",
                 ranges = list(cost = c(0.1,1,10,100,1000),
                               gamma = c(0.5,1,2,3,4,5)))
```

```
svm.fit <- tune.svm$best.model
svm.fit
```

```
##
## Call:
## best.tune(METHOD = svm, train.x = diabetes ~ ., data = train_df,
##     ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5,
##         1, 2, 3, 4, 5)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  431
```

This best fit model subsequently allows us to calculate the error metrics and the generate the ROC-AUC-

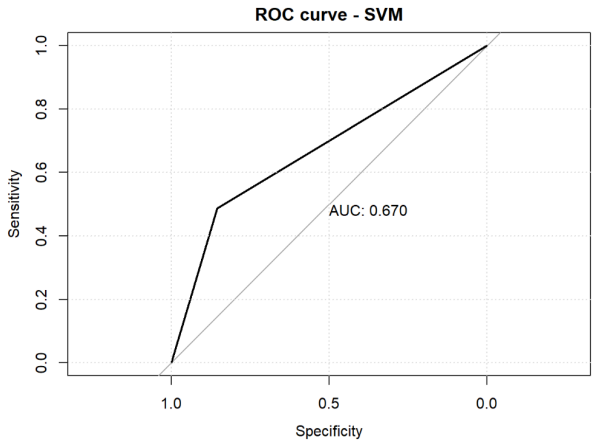
SVM
Misclassification Rate: 0.3058824
Model Accuracy: 0.6941176

Precision: 0.4864865	
Recall:	0.72
Specificity: 0.6833333	
F1 Score: 0.5806452	

(Code found in Rmd file)

ROC-AUC of SVM best fit is as follows -

```
test_roc <- roc(test_df$diabetes ~ pred.svm,
                plot = TRUE,
                print.auc = TRUE,
                main ="ROC curve - SVM")
```



2.4 Advantages and Disadvantages of SVMs

Like every algorithm, SVMs have their unique set of advantages and disadvantages. SVMs perform well in high-dimensional feature spaces and complex class structures. These models have different kernel functions (e.g., linear, polynomial, Gaussian) that allow the model to handle non-linear decision boundaries and capture complex relationships in the data. SVMs are less prone to overfitting due to the existence of cost function and thus are suitable for tasks with a large number of features. The disadvantages of SVMs are that these models are computationally expensive and memory intensive to run. SVMs are also affected greatly by imbalance datasets. Overall SVMs is a versatile tool that can aid a programmer in finding meaningful insights into their binary classification problem.