|  |
| --- |
| Saint peters University |
| Summary Report – Breast Cancer Diagnosis Classification |
| DS:630- HMWK 3 |
|  |
| **PranayKatta** |
| **10/17/2017** |

|  |
| --- |
|  |

Summary

This project aims at predicting whether a cancer is benign or malignant based on 30 Variables. We compare and contrast the outcome of diagnosis from four different algorithms. Four different algorithms are as follows:

* GLM – Generalized Linear Model
* LDA – Linear Discriminant Analysis
* SVM – Support Vector Machine
* KNN – K Nearest Neighbors

Each of the ten variables are divided into three different measurements namely – mean, standard error and worst values making them 30 variables.

Ten variables are as follows:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness (perimeter^2 / area - 1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension ("coastline approximation" - 1)

With variables ID Number and Diagnosis outcome total variables are aggregated to 32 and have 569 observations. Observations are divided into 2 datasets 75% of observations for training i.e. 426 and 25% for testing i.e. 143. Each algorithm behaves differently in different setting and SVM comes out to be having best accuracy at 99.3007.

Cleaning the data

Before dividing the data into training and test datasets we remove ID\_number since this doesn’t contribute to our diagnosis outcome. Removing NA values using na.omit() if there are any. Data can be divided into training and test by randomly and the way it is given (first 75% for training and the remaining for testing). In the first part, we divide the data the way it is given. In the second part, we analyze how different algorithms behave when data is divided randomly.

Modeling the data – I

GLM – Logistic Regression

We use the following code to model logistic regression.

logmodel <- glm(Diagnosis~.,family =binomial(),data=data\_cancer\_train) test\_glm <- predict.glm(logmodel,data\_cancer\_test,type='response') test\_prediction\_glm <- ifelse(test\_glm > 0.5, "M","B" ) confusion\_matrix\_glm <-table(test\_prediction\_glm,data\_cancer\_test$Diagnosis,dnn=c('Predicted','Actual')) accuracy\_glm <- ((confusion\_matrix\_glm[1,1] + confusion\_matrix\_glm[2,2])/sum(confusion\_matrix\_glm))\*100

|  |  |  |
| --- | --- | --- |
|  | Actual | |
| Predicted | B | M |
| B | 98 | 3 |
| M | 10 | 32 |

Confusion Matrix :

With 3 false negatives and 10 false positives, its accuracy comes out to be 90.90%. When analyzing the significant values of glm all variables seem to have similar significance.

LDA – Linear Discriminant Analysis

We use the following code to model lda using package “MASS”.

ldamodel <- lda(Diagnosis~.,data=data\_cancer\_train)

pred\_lda <- predict(ldamodel,data\_cancer\_test)

confusion\_matrix\_lda <- table(pred\_lda$class,data\_cancer\_test$Diagnosis,dnn=c('Predicted','Actual'))

accuracy\_lda <- ((confusion\_matrix\_lda[1,1] + confusion\_matrix\_lda[2,2])/sum(confusion\_matrix\_lda))\*100

|  |  |  |
| --- | --- | --- |
|  | Actual | |
| Predicted | B | M |
| B | 107 | 3 |
| M | 1 | 32 |

Confusion Matrix:

With 3 false negatives and 1 false positives, its accuracy comes out to be 97.20%

SVM – Support Vector Machine

We use the following code to model svm using package“e1071”.

model\_svm <- svm(Diagnosis ~ . , data= data\_cancer\_train)

test\_svm <- predict(model\_svm,data\_cancer\_test)

confusion\_matrix\_svm <- table(test\_svm,data\_cancer\_test$Diagnosis,dnn = c('predicted','actual'))

accuracy\_svm <- ((confusion\_matrix\_svm[1,1] + confusion\_matrix\_svm[2,2])/sum(confusion\_matrix\_svm))\*100

|  |  |  |
| --- | --- | --- |
|  | Actual | |
| Predicted | B | M |
| B | 105 | 0 |
| M | 3 | 35 |

Confusion Matrix:

With 0 false negatives and 3 false positives, its accuracy comes out to be 97.90%

KNN – K Nearest Neighbors

We use the following code to model knn using package“class”.

Changing M to 1 and B to 0

data\_cancer\_copy$Diagnosis <- ifelse(data\_cancer\_copy$Diagnosis == 'M', 1 , 0)

data\_cancer\_train\_knn <- data\_cancer\_copy[1:floor(nrow(data\_cancer\_copy) \* 0.75),]

data\_cancer\_test\_knn <- data\_cancer\_copy[ceiling(nrow(data\_cancer\_copy) \* 0.75):nrow(data\_cancer\_copy),]

prev\_accuracy\_knn <- 0

k\_value <- 1

Looping through the model for selecting the most accurate model with changing K.

repeat{

model\_knn <- knn(data\_cancer\_train\_knn, data\_cancer\_test\_knn, data\_cancer\_train\_knn$Diagnosis, k = k\_value)

summary(model\_knn)

confusion\_matrix\_knn <- table(model\_knn,data\_cancer\_test$Diagnosis,dnn = c('prediceted','actual'))

confusion\_matrix\_knn

accuracy\_knn <- ((confusion\_matrix\_knn[1,1] + confusion\_matrix\_knn[2,2])/sum(confusion\_matrix\_knn)) \* 100

if(prev\_accuracy\_knn >= accuracy\_knn){

accuracy\_knn <- prev\_accuracy\_knn

k\_value <- k\_value - 2

break

}

prev\_accuracy\_knn <- NULL

prev\_accuracy\_knn <- accuracy\_knn

k\_value <- k\_value + 2

}

|  |  |  |
| --- | --- | --- |
|  | Actual | |
| Predicted | B | M |
| B | 100 | 2 |
| M | 8 | 33 |

Confusion Matrix:

With 2 false negatives and 8 false positives its accuracy comes out to be 93.70% at K = 5.

Modeling the data – II

Accuracy of a model depends on the data that is being used to train the model. In the previous process of modeling we did not randomize and divide the data into training and testing. The outcome of the process has SVM the accuracy at 97.90%. Let’s analyze what happens when we randomize the data or order the data with respect to a particular variable.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Accuracy when data is selected randomly** | | | | | | |
| **GLM** | **LDA** | **SVM** | **KNN** | **K Value** | **Seed** | **Avg** |
| 97.9021 | 97.9021 | 97.9021 | 93.00699 | 1 | 11 | 96.67832 |
| 93.7063 | 95.1049 | 97.9021 | 93.00699 | 1 | 12 | 94.93007 |
| 96.5035 | 95.8042 | 97.2028 | 93.70629 | 5 | 13 | 95.8042 |
| 94.4056 | 95.1049 | 98.6014 | 93.70629 | 5 | 14 | 95.45455 |
| 95.1049 | 94.40559 | 99.3007 | 92.30769 | 3 | 15 | 95.27972 |
| 95.5245 | 95.6643 | 98.1818 | 93.1469 | **Average** | | 95.62937 |

Table i

Above table describes how different algorithms behave when we randomize the data. On an average SVM seems to behave really well at 98.18% than all other algorithms. And at Seed 11 all algorithms have the highest accuracy 96.67%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Accuracy** | | | | | |
| **Order By Ascending** | **GLM** | **LDA** | **SVM** | **KNN** | **K Value** | **Avg** |
| **Default** | 90.90909 | 97.2028 | 97.9021 | 93.70629 | 5 | **94.93007** |
| **radius\_mean** | 98.6014 | 96.5035 | 77.62238 | 97.2028 | 1 | **92.48252** |
| **texture\_mean** | 88.11189 | 93.00699 | 95.8042 | 86.01399 | 1 | **90.73427** |
| **perimeter\_mean** | 98.6014 | 99.3007 | 68.53147 | 96.5035 | 3 | **90.73427** |
| **area\_mean** | 98.6014 | 97.9021 | 60.13986 | 97.2028 | 1 | **88.46154** |
| **smoothness\_area** | 90.90909 | 96.5035 | 95.8042 | 92.30769 | 3 | **93.88112** |
| **compactness\_mean** | 91.60839 | 95.1049 | 81.11888 | 88.11189 | 3 | **88.98602** |
| **concavity\_mean** | 92.30769 | 98.6014 | 69.23077 | 87.41259 | 1 | **86.88811** |
| **concave\_points\_mean** | 97.2028 | 98.6014 | 71.32867 | 90.20979 | 3 | **89.33567** |
| **symmetry\_mean** | 92.30769 | 96.5035 | 95.8042 | 90.90909 | 5 | **93.88112** |
| **fractal\_dimension\_mean** | 92.30769 | 92.30769 | 90.90909 | 88.81119 | 3 | **91.08392** |
| **Average** | **93.76986** | **96.50349** | **82.19962** | **91.671965** |  | **91.0362** |

Table ii

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Accuracy** | | | | | |
| **Order By Ascending** | **GLM** | **LDA** | **SVM** | **KNN** | **K Value** | **Avg** |
| **radius\_SE** | 91.60839 | 98.6014 | 54.54545 | 95.8042 | 3 | **85.13986** |
| **texture\_SE** | 93.00699 | 95.1049 | 95.1049 | 93.00699 | 1 | **94.05595** |
| **area\_SE** | 95.8042 | 98.6014 | 46.85315 | 97.2028 | 3 | **84.61539** |
| **perimeter\_SE** | 96.5035 | 97.2028 | 57.34266 | 93.70629 | 3 | **86.18881** |
| **smoothness\_SE** | 96.5035 | 97.2028 | 57.34266 | 93.70629 | 3 | **86.18881** |
| **compactness\_SE** | 89.51049 | 92.30769 | 84.61538 | 86.01399 | 3 | **88.11189** |
| **concavity\_SE** | 93.00699 | 93.00699 | 88.11189 | 86.01399 | 1 | **90.03497** |
| **concave\_points\_SE** | 92.30769 | 91.60839 | 94.40559 | 88.11189 | 1 | **91.60839** |
| **symmetry\_SE** | 91.60839 | 90.90909 | 90.90909 | 93.00699 | 5 | **91.60839** |
| **fractal\_dimension\_SE** | 93.00699 | 94.40559 | 80.41958 | 89.51049 | 5 | **89.33566** |
| **Average** | **93.286713** | **94.895105** | **74.965035** | **91.608392** |  | **88.68881** |

Table iiI

Surprisingly from table i and ii, it can be seen that on an average SVM has lowest accuracy when ordered with respect to different variables. And LDA has the highest accuracy at **99.3** when ordered with respect to perimeter\_mean. So it can be inferred that there is some kind of dependency. Apart from SVM, all other algorithms are really good at finding the patterns when ordered with a specific variable.

Best performance of each algorithm is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy** | **Ordered By** | **Accuracy when randomly ordered** | **Seed** |
| GLM | 98.6 | radius \_mean | 97.9 | 11 |
| LDA | 99.3 | perimeter\_mean | 97.9 | 11 |
| SVM | 99.3 | seed 15 | 99.3 | 15 |
| KNN | 97.2 | area\_SE | 93.7 | 13 |

Conclusion

SVM has the highest accuracy at 99.3 when randomly sampled. All other algorithms have a high level of accuracy ranging from 93 to 99. Even though, all columns were showing the same level of significance it could be because of the fact that we were using many variables and overfitting the data. We can improve the models by implementing dimensionality reduction techniques which will reduce the variables from 30 to a significant number. Increasing the number of observations and testing models would help us to learn how well algorithms are behaving. Last but not the least it is important to convey implications of the false positives and false negatives to a patient when employing a specific model to predict his/her diagnosis outcome.