STUDENT NAME: SRINIVAS VEGISETTI
STUDENT ID: 21080840
My GitHub link: Click Here…

Predictive Analysis for Prime Indians Diabetes

Data used for Predictive Analysis:

PimaIndiansDiabetes dataset, a built-in dataset in R's mlbench package, is the dataset I utilized.

Question 1: Description of Prima Indian Dataset

All of the adult female Pima Indian individuals in this dataset are the subject of this analysis, which compares several approaches to using the data's information to determine whether or not a person has diabetes based on certain diagnostic measurements included in the dataset.

The Pima Indian Diabetes dataset consists of 768 instances out of which 268 tested positive. A description of the attributes and their values is given in below.

ATTRIBUTES	DESCRIPTION	VALUES
Preg	Pregnancies count	From 0 - 17
Gluc	Glucose levels	From 0 - 199
Pres	Blood Pressure levels	From 0 - 122
Tric	Triceps skin thickness	From 0 - 99
Insu	Insulin levels in the body	From 0 - 846
Mass	Body Mass measures	From 0 - 67
Pedi	Pedigree function	From 0 – 2.45
Age	Individuals Age	From 21 - 81
class	Tested Positive / Negative	0 - Neg, 1 - Pos

- ***Response Variable**:* Here our Response/Dependent/Outcome variable is the “Diabetes” Variable (Diabetes Positive/Negative) as whose variation can be accounted for by other variables in our dataset because the outcome of our experiment in which the explanatory variable is altered is the response variable.
- ***Explanatory Variables**:* Here our Explanatory/Independent/Predictor variables would be Pregnant, Glucose, Blood Pressure, Triceps Skin Thickness, Insulin, Body Mass, Pedigree, and Age. These are the factors/causes that help to predict the other factor/cause (response variable).

The following solutions are Predictive analytics to find patterns in this data to identify risks and opportunities for Diabetes.

Question 2: Preparing the Dataset for Analysis

- ***Response Variable to Binary:*** Converting Response variable values to Binary values is the best way for a logistic regression model based on proportional data of a certain result, the response variable (which needs to be binary with a simple yes or no) needs to be a binary yes or no. A binary-response variable can also be good for many regression models and algorithms (Ex: Naive Bayes, Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Decision Tree, Bagging Decision Tree, Boosted Decision Tree, Random Forest, Voting Classification, Neural Networks) in which the dependent variable takes only the values zero and one.
- ***Standard Score Transform:*** The standardization procedure known as Z- Score transformation enables the comparison of results from various distributions. Z-Score transformations combine distinct distributions into a standardized distribution using the distribution's mean and standard deviation, enabling the comparison of metrics with different properties. To find how many standard deviations are above or below the mean of the distribution. As my data follow a z-distribution, I have used this Z-Score/ Standard Score Transform to put data from different sources onto the same scale.

  Ex: To plot change over time in Age and Blood pressure or Glucose and Diabetes on the same graph we can transform the two factors, raw measurements into Z scores and plot them on the same scale.

Q2. Output:

```
Console   Terminal   Background Jobs
R 4.2.2 · ~/
> # To Normalise all the explanatory variables using the standard score transformation
>
> # Z -Score Transform for Pregnant variable
> z_preg <- (pid_data$pregnant - mean(pid_data$pregnant))/sd(pid_data$pregnant)
> round(mean(z_preg), 5)
[1] 0
>
> sd(z_preg) # To find Standard Deviation
[1] 1
>
> head(z_preg)
[1]  0.6395305 -0.8443348  1.2330766 -0.8443348 -1.1411079  0.3427574
>
>
> # Z -Score Transform for Glucose variable
> z_gluc <- (pid_data$glucose - mean(pid_data$glucose))/sd(pid_data$glucose)
> round(mean(z_preg), 5)
[1] 0
>
> sd(z_gluc) # To find Standard Deviation
[1] 1
>
>
> # Z -Score Transform for Pressure variable
> z_pres <- (pid_data$pressure - mean(pid_data$pressure))/sd(pid_data$pressure)
> round(mean(z_preg), 5)
[1] 0
>
> sd(z_pres) # To find Standard Deviation
[1] 1
>
>
> # Z -Score Transform for Triceps variable
> z_tric <- (pid_data$triceps - mean(pid_data$triceps))/sd(pid_data$triceps)
> round(mean(z_preg), 5)
[1] 0
>
> sd(z_tric) # To find Standard Deviation
[1] 1
```

```
Console   Terminal ×   Background Jobs ×
R 4.2.2 · ~/
> # To split I used sample.split() and subset() function to do so.
>
> # Syntax: sample.split(X = , SplitRatio = )
> # Where: X = target variable
> # SplitRatio = no of train observation divided by the total number of test observation.
> # for eg. SplitRatio for 70%:30% (Train:Test) is 0.7.Here The observations are chosen randomly.
>
> pid_split = sample.split(pid_data$pregnant, SplitRatio = 0.7)
>
> #subsetting into Train data
> pid_train = pid_data[pid_split,]
>
> #subsetting into Test data
> pid_test = pid_data[!pid_split,]
>
> # Now, checking the dimensions of the train and test data created
> # to check whether the splitting is done correct.
>
> dim(pid_train)
[1] 538   9
> dim(pid_test)
[1] 230   9
>
>
```

Question 3: Feature Selection Tests (Fisher and Wilcoxon Scores)

Undergone Feature selection process which identifies and selects a subset of explanatory/input variables that are most informative/relevant to the response/target variable. This process can achieve a Decrease in Data over-fitting, Improves Accuracy, and Reduces Training Time.

In our data, based on the train and test subset data, I have applied Fisher and Wilcoxon's scores feature selection tests which gave me the result of glucose, mass, and age explanatory variables which are more informative for predicting diabetes and removed non-informative explanatory variables pregnant, pressure, triceps, insulin, pedigree.

Hence, based on the above scores I have reduced the train and test subsets to reduced train and reduced test subsets with information that only contains the most effective features responsible for diabetes, leaving the redundant variables. This helps to our regression model to be more accurate.

Q3. Output:

```
Console   Terminal ×   Background Jobs ×
R 4.2.2 · ~/
+   return(list(score=J))
+ }
>
> # Calling the function Fisher Score
> fis_calc(pid_data[ , c(1, 2, 3, 4, 5, 6, 7, 8)], pid_data$diabetes, 3)
$score
   glucose       mass        age 
0.27750834 0.09345735 0.06008179 


>
>
> # Wilcoxon Scores Calculation_______________
>
> wils_calc = function(X, Y, k) # X - matrix with predictors, Y - binary outcome, 
+ {
+   J<- rep(NA, ncol(X))
+   names(J)<- colnames(X)
+   for (i in 1:ncol(X))
+   {
+     X_rank<- apply(data.matrix(X[,i]), 2, function(c) rank(c))
+     X1_rank<- X_rank[which(Y==0)]
+     X2_rank<- X_rank[which(Y==1)]
+     mu1<- mean(X1_rank); mu2<- mean(X2_rank); mu<- mean(X_rank)
+     n1<- length(X1_rank); n2<- length(X2_rank); N<- length(X_rank)
+     num<- (n1*(mu1-mu)^2+ n2*(mu2-mu)^2)
+     denom<- 0
+     for (j in 1:n1)
+       denom<- denom+(X1_rank[j]-mu)^2
+     for (j in 1:n2)
+       denom<- denom+(X2_rank[j]-mu)^2
+     J[i]<- (N-1)*num/denom
+   }
+   J<- sort(J, decreasing=TRUE)[1:k]
+   return(list(score=J))
+ }
>
> wils_calc(pid_data[ , c(1, 2, 3, 4, 5, 6, 7, 8)], pid_data$diabetes, 3)
$score
  glucose      mass       age 
173.62049  73.56931  73.25301 
```

Question 4: Logistic Regression:

Performed Logistic Regression Model on my Reduced Training Dataset and observed the following results:

The more significant the p-value, the more compelling the case is to reject the null hypothesis.

A p-value of 0.05 or less (usually 0.05) indicates statistical significance; a p-value of 0.05 or more (> 0.05) does not and suggests substantial evidence supporting the null hypothesis.

Result: Here, as we can see from my data the p-value is < 0.05 for mass and age variables, neither mass nor age is insignificant in my logistic regression model. Thus, I rejected the Null Hypothesis and we can consider that our model is the best fit to predict diabetes in our data using those 2 explanatory variables (mass, and age).

```
Console   Terminal ×   Background Jobs ×
R 4.2.2 · ~/
>
>
> # Logistic regression with all the predictors included
> mylogit1 <- glm(diabetes~ glucose+mass+age, data = pid_red_train, family = "binomial")
> summary(mylogit1)

Call:
glm(formula = diabetes ~ glucose + mass + age, family = "binomial",
    data = pid_red_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2732  -0.6790  -0.4151   0.7047   2.6417

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -8.433345   0.920923  -9.157  < 2e-16 ***
glucose      0.034233   0.004944   6.923 4.41e-12 ***
mass         0.064714   0.018591   3.481  0.00050 ***
age          0.034608   0.011169   3.098  0.00195 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 473.06  on 382  degrees of freedom
Residual deviance: 350.09  on 379  degrees of freedom
AIC: 358.09

Number of Fisher Scoring iterations: 5
```

Question 5: Random Forest:

Used a random forest model because it allowed me to build the model using a variety of various features. I have values for the dataset's feature variables, which will increase the classifier's likelihood of correctly predicting outcomes rather than just offering an estimate.

Following are the results observed from the training and reduced training subsets.

Random Forest Training Model1:

Accuracy = 73.4 %, Sensitivity = 79 %, Specificity = 68 %

Random Forest Training Model2:

Accuracy = 73.4 %, Sensitivity = 77 %, Specificity = 66 %

We can examine the observed proportions of events in model 1 are similar to the predicted probabilities of occurrence in model 2 of the data set while conducting a logistic regression model. Here, as both the models have similar accuracy, sensitivity, and specificity we can conclude that our model is the best fit.

```
        OOB estimate of  error rate: 26.58%
Confusion matrix:
    0   1 class.error
0 280  63   0.1836735
1  80 115   0.4102564
>
> importance(rfm_train)
                0          1 MeanDecreaseAccuracy MeanDecreaseGini
pregnant 3.0946363  0.1520541            2.6312785         21.06837
glucose  9.7192984 11.5063287           13.5350657         70.31870
pressure 0.7578142 -1.8210259           -0.2003921         19.88125
triceps  0.8592201 -0.6360831            0.4786462         15.45912
insulin  2.7768488  3.2532654            5.3828336         17.06419
mass     4.2727797  5.3758199            5.8374036         38.43303
pedigree 2.1333632  0.8345061            2.1712930         33.31962
age      5.1572274  3.2743087            6.4439396         34.15089
>
> # Evaluating Model Accuracy & Building Confusion Matrix
> cnf_matrix <- table(predict(rfm_train), pid_train$diabetes) # conditional RF
> cnf_matrix

      0   1
  0 280  80
  1  63 115
>
> # Calculating Accuracy
>
> sum(diag(cnf_matrix))/sum(cnf_matrix)*100
[1] 73.42007
>
> # Calculating Sensitivity and Specifity
> # Sensitivty = TP/TP+FN and specifity = TN/TN+FN
>
> sensitivity1 = 286/(286+73)
> specificity1 = 122/(122+57)
>
> sensitivity1
[1] 0.7966574
> specificity1
[1] 0.6815642
```

```
Call:
 randomForest(formula = factor(diabetes) ~ ., data = pid_red_train,      importance = TRUE, ntree =
 50, mtry = 3, replace = TRUE)
               Type of random forest: classification
                     Number of trees: 50
No. of variables tried at each split: 3

        OOB estimate of  error rate: 26.11%
Confusion matrix:
    0  1 class.error
0 205 39   0.1598361
1  61 78   0.4388489
>
> importance(rfm_red_train)
               0        1 MeanDecreaseAccuracy MeanDecreaseGini
glucose 9.948536 9.848545            14.118551         84.94069
mass    3.153410 3.860617             5.646686         49.63921
age     4.843410 1.759788             4.649333         39.96861
>
> # Evaluating Model Accuracy & Building Confusion Matrix
> cnf_matrix2 <- table(predict(rfm_red_train), pid_red_train$diabetes) # conditional RF
> cnf_matrix2

      0   1
  0 205  61
  1  39  78
>
> # Calculating Accuracy
> sum(diag(cnf_matrix2))/sum(cnf_matrix2)*100
[1] 73.89034
>
> # Calculating Sensitivity and Specifity
> # Sensitivty = TP/TP+FN and specifity = TN/TN+FN
>
> sensitivity2 = 203/(203+59)
> specificity2 = 80/(80+41)
>
> sensitivity2
[1] 0.7748092
> specificity2
[1] 0.661157
>
```

Question 6: ROC Curves:

AUC stands for the level or measurement of separability, and ROC is a probability curve that lies between 0 and 1. Here, our data, reveals how well the model can differ across classes. Our model is more accurate at classifying 0 classes as 0, and classifying 1 class as 1, the higher the AUC. The AUC and probability graphs below display how well a classification model performs at various threshold levels.