Diabetes Prediction in PIMA INDIANS

Unnaty Gandhi

1. INTRODUCTION

Diabetes Mellitus is a rising concern for many countries today because of its crippling effects on human life. Onset of diabetes is on the rise in many nations. It leads to blindness, kidney failure, hampers nervous system, digestion, and causes skin issues. If not treated, it can cause hitches. It has become a universal problem as many parts of the world don't have optimal access to health services and the right medicines. There are roughly 300 million people in the world who are affected by diabetes. By leveraging the data, it is possible to bridge the gap between human knowledge and humongous datasets. There is too much information in the universe which stems from raw data that exists in mind boggling amounts. If this information isn't soldered, connected and put to effective use, the purpose of technology is defeated. This means data must be converted to actionable data which can optimize the capabilities of industry. This can be done by using machine learning and data mining. The intention is to predict diabetes in PIMA INDIANS in this project. The source of the dataset is Pima Indians Diabetes Database of National Institute of Diabetes and Digestive and Kidney Diseases which was donated by Vincent Sigillito. This database has a collection of medical diagnostic reports. There are 768 observations from a population living near Phoenix, Arizona, USA. PIMA INDIANS are Native Americans who lived along the Gila and Salt rivers in Arizona. This area was dominated by prehistoric Hohokam culture. Pima people address themselves as RIVER PEOPLE and are considered to be descendants of the Hohokam. According to various studies, it is noted that the PIMA INDIANS have highest prevalence of diabetes as compared to any population of the world. Occurrence of type 2 diabetes is quite high among PIMA INDIANS. Thus, they are the most studied group to understand the causes and consequences of diabetes.

Studies show that type 2 diabetes and obesity are associated with genes and environment factors. There is compelling evidence that lifestyle changes coupled with Westernization have brought the global epidemic of type 2 diabetes.

In the dataset, there are 738 observations in the dataset and 9 variables out of which 8 are independent variables (predictors) and one is the dependent variable (outcome). The outcome variable is binary in nature. i.e. 0s and 1s (for ex. if a dependent variable predicts Diabetes then it takes the value 0 if a patient does not have diabetes, 1 otherwise). All patients are females at least 21 years old of Pima Indian heritage. For the purposes of this dataset, diabetes was diagnosed according to World Health Organization Criteria. The criteria stated that a patient is diagnosed with diabetes if the 2 hour postload glucose was at least 200 mg/dl at any survey exam.

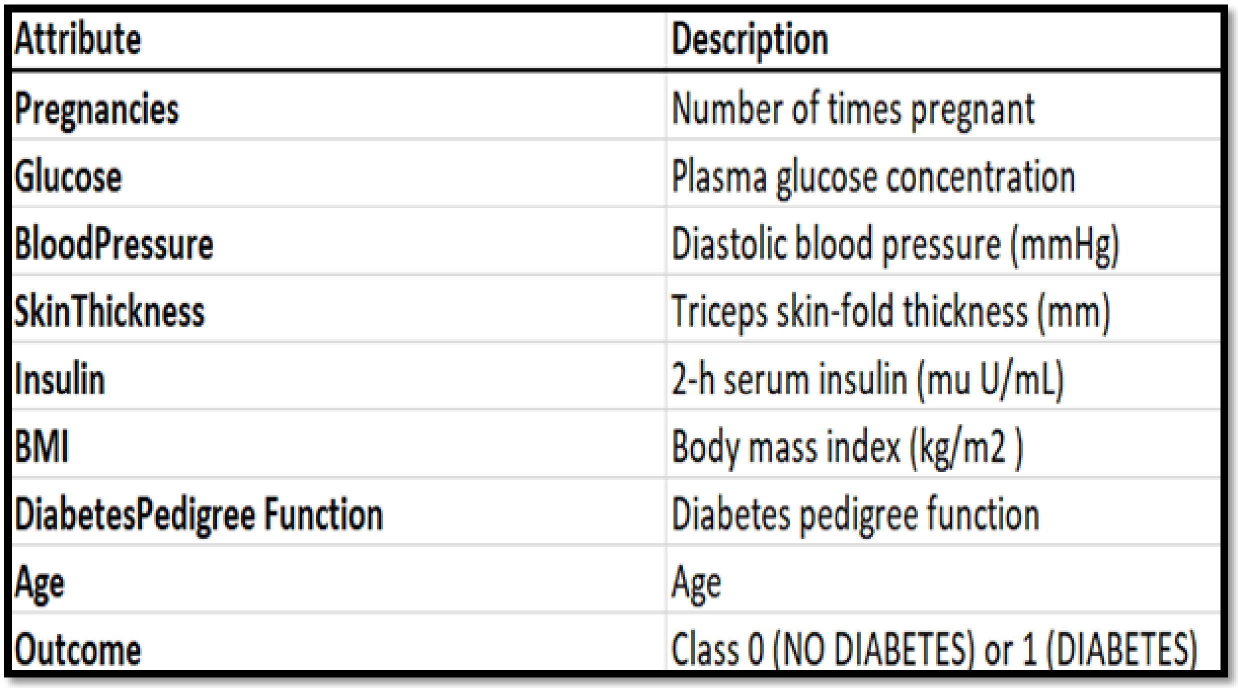
Research Question: To diagnostically predict whether or not a patient has diabetes.

WHO raised a concern that by 2030, diabetes will be 7th leading cause of death in the world. It is a universal problem which significantly impacts society and economy. Given the medical data of people, it is possible to make prediction on how likely a person is to suffer the onset of diabetes. It is medically accepted that each patient has different case and different therapies work on them. Thus, appropriate action and medication can be given without any time loss.

By analyzing the data and experimenting with algorithms, the onset of diabetes in Pima Indians can be predicted.

2. Methods

The PIMA Indians Database contains information on whether or not a patient is suffering from diabetes. All patients in the data base are females and are at least 21 years old of PIMA Indian Heritage. The data set has been taken from Kaggle, and was initially contributed by UCI Machine Learning. It consists of 768 observations and 9 variables, that are briefly explained in the table below:



Attribute	Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration
BloodPressure	Diastolic blood pressure (mmHg)
SkinThickness	Triceps skin-fold thickness (mm)
Insulin	2-h serum insulin (mu U/mL)
BMI	Body mass index (kg/m2)
DiabetesPedigree Function	Diabetes pedigree function
Age	Age
Outcome	Class 0 (NO DIABETES) or 1 (DIABETES)

Besides the Diabetes Pedigree Function, the above-mentioned health indicators are self-explanatory. A Diabetes Pedigree Function is a continuous variable that quantifies the hereditary risk of a patient suffering from diabetes. The hereditary risk factor increases if a patient's relative has a history of carrying the disease. In our case study, our dependent variable is 'Outcome', which is a categorical variable, consisting of binary values i.e. 0 and 1, where it takes the value 0 when a given patient does not have diabetes, and it takes 1, otherwise. Therefore, our objective aims at predicting if a given patient has diabetes or not, through the given data set. To predict our 'Outcome' based on the values of the predictor/independent variables, we have used the following three approaches:

1. **Logistic Regression Analysis**
   A logistic Regression technique models the probability of an event occurring depending on the values of the predictor variables which can be categorical/numerical. It classifies observations by estimating the probability that an observation is in a particular category such as having the disease (1) or not (0). Below is the estimated Logistic Regression Equation:

$$\hat{p} = \frac{\exp(b_0 + b_1X_1 + b_2X_2 + \ldots + b_pX_p)}{1 + \exp(b_0 + b_1X_1 + b_2X_2 + \ldots + b_pX_p)},$$

Where, p-hat gives us the estimated probability of an event occurring, such as the probability that a given patient has diabetes.
X1, X2,….Xp are predictor/independent variables such as Pregnancies, BMI, Insulin, Glucose, etc.
b0, b1,….bp represent coefficient estimates of the respective predictor variables.
The dependent variable in logistic regression follows the Bernoulli distribution having an unknown probability, p. Bernoulli distribution is a special case of Binomial distribution where there are two outcomes, success and failure. The probability for success is 1, whereas it is 0 for failure. Similarly, in our data set, as we have a dependent variable with two binary values, 0 and 1, a Logistic Regression model is a good fit to solve our research question i.e. predicting weather a given patient has diabetes or not. The coefficient estimates in a logistic regression model are computed using the Maximum Likelihood Estimation (MLE) method.
It is important to note that a binary variable does not follow a normal distribution and the associated probabilities are often not linear (like the dependent variable is linear in case of a linear regression model). In logistic regression analysis, probabilities are calculated using the odds ratio. The odds ratio for a variable represents how the odds change with a 1-unit increase in that variable, holding other factors constant.
In logistic regression, the coefficient estimates are computed in terms of natural log odds which is equivalent to a linear function of the independent variables. The antilog of the logit function allows us to find the estimated regression equation.

We have built a logistic regression model using glm() function in R, to predict our outcome variable. The model is first trained on the training data and later on validated on test data. We have used a confusion matrix to interpret our prediction accuracy on test data. To increase the efficiency of the model, we have used the 'OptimalCutOff()' function to increase the prediction accuracy of our model. We have, then, reviewed the performance of the logistic regression model on test data using performance metrics such as sensitivity, specificity and AUROC curve.

2. Decision Tree Analysis

A Decision Tree approach segments or stratifies the predictor space into a number of simple regions using which we can make predictions for the observations in our data set. The mean value of training observations in each region is computed, based on which splitting is performed. The rules of splitting, determined by the mean of each region, can be summarized in an output that takes the form of a tree. Hence, these methods are typically referred to as Decision Tree methods that can be used in regression as well as classification problems. Tree-based methods can be applied with ease as they require minimum assumptions to be fulfilled and are simple to implement. As our research question poses a classification problem, we have tried to predict our outcome using a decision tree model.
We will use the C50 package in R, to form a decision tree. The output of this algorithm is the cross-table which will provide us with the value of sensitivity, specificity and the overall accuracy of the model. The accuracy, prioritizing the sensitivity, would provide us with the percentage of patients that the model predicted correctly for the patients having diabetes, which is basically our objective.

The algorithm uses “c5.0” function on the training dataset and the response variable explicitly mentioned apart from the rest of the variables. The ‘outcome’ variable was in the numeric form, thus, we converted it into a factor variable using as.factor() function since the algorithm only accepts factor variables.
Next, we have used the predict() function using the model on the test dataset to check on how our model performs. The next was to use the find the accuracy with the values of the true positive and true negative. The result of the crosstable() function provides us with the knowledge of how our model performed when introduced to a completely new dataset.
Finally, we have performed the boosting technique in order to increase the accuracy of the model by tuning it.
Note - The boosting technique will not necessarily improve the model in every case. The outputs are compared, and a decision is taken if the boosting was worth or not.

3. KNN Classification

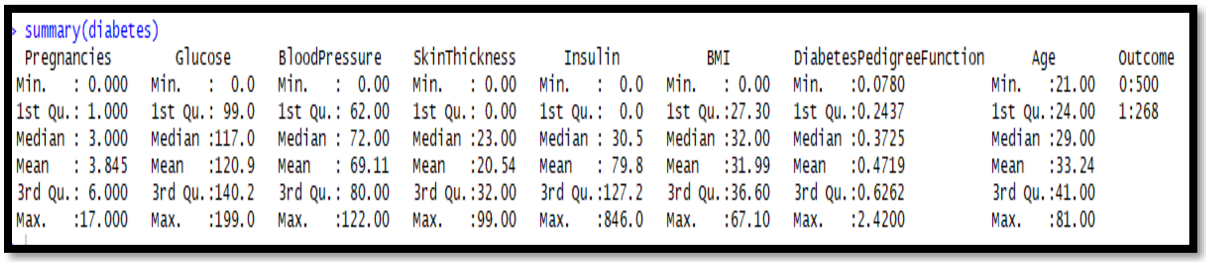
K-nearest-neighbor is a data classification algorithm that attempts to determine what group a data point is in, by looking at the data points around it. To label a new point, it looks at the labeled points closest to that new point (those are its nearest neighbors), and then all those neighbors vote, so the maximum votes for whichever neighbor is, becomes the new data point. K is the number of neighbors it checks). If the value for K = 1, then the object is assigned to the class of its nearest neighbor. K-nearest-neighbor classification is commonly used for its easy interpretation and low calculation time.
In R, the syntax and method of a KNN classification are different from other methods such as decision tree and logistic regression. In other models, we first predict our model using the training data and then using the predict() function and testing data, we predict our model to ensure there is no overfitting or underfitting whereas, in the case of KNN classification, we just use a simple KNN() function which takes both the training and testing data as inputs, along with the qualitative field.

After building the predictive models, we attempt to increase the efficiency of each model using different boosting techniques. We have used certain performance metrics such as Sensitivity, Specificity and AUROC curve, to comment on the accuracy of each model, before and after improving the efficiency, and, therefore, select the best classification method. Our model selection is based on the metric that gives us the highest True Positive Rate (TPR) i.e. Sensitivity, such that patients suffering from the disease, get immediate treatment.

3. Results

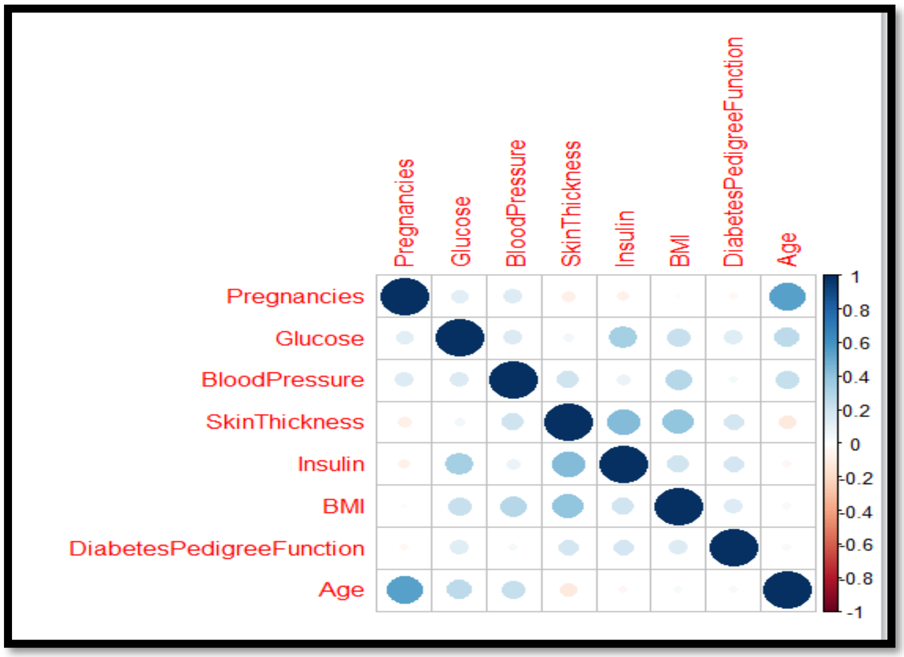
3.1 Exploratory Data Analysis (EDA)

Below is the summary statistic of each variable in our dataset.

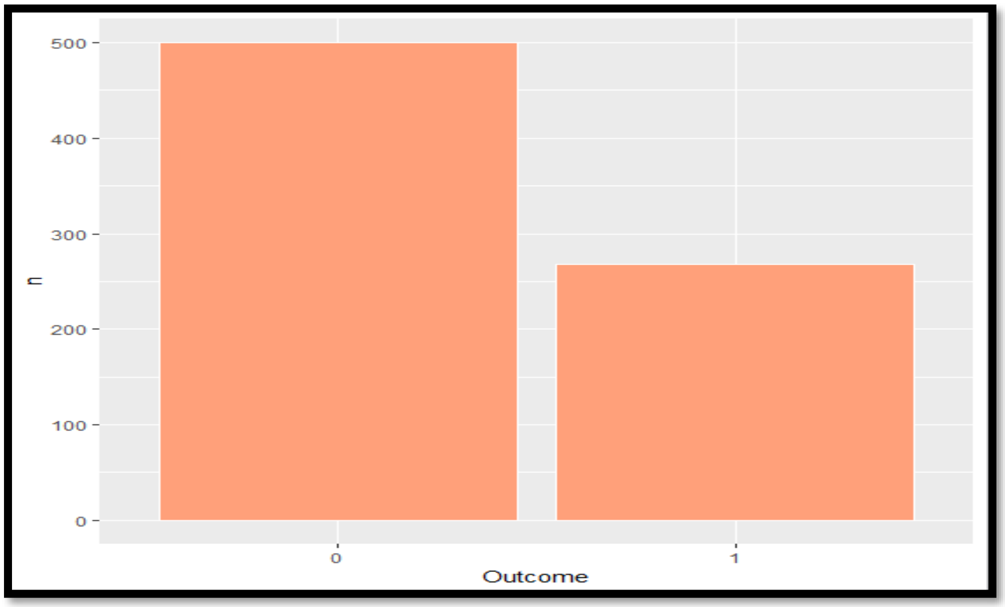


```
> summary(diabetes)
  Pregnancies       Glucose      BloodPressure    SkinThickness     Insulin          BMI        DiabetesPedigreeFunction      Age       Outcome
Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.0   Min.   : 0.00   Min.   :0.0780          Min.   :21.00   0:500
1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00   1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437          1st Qu.:24.00   1:268
Median : 3.000   Median :117.0   Median : 72.00   Median :23.00   Median : 30.5   Median :32.00   Median :0.3725          Median :29.00
Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54   Mean   : 79.8   Mean   :31.99   Mean   :0.4719          Mean   :33.24
3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00   3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262          3rd Qu.:41.00
Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.0   Max.   :67.10   Max.   :2.4200          Max.   :81.00
```

From the above output, we can infer that the minimum number of pregnancies is 1 whereas maximum number of pregnancies are 17. Our outcome variable is binary in nature, having 500 observations of 0's(non-diabetic) and 268 observations of 1's(diabetic). Similarly, we can interpret all the other variables. Next, we have created a correlation plot showing the correlation between our variables using the corrplot() function in R. Below is the screenshot for the same:



From the above output we can interpret that pregnancies and age are strongly correlated with each other. Also, Insulin and BMI are strongly correlated with Skin thickness. Next, we have plotted a bar plot of our outcome variable to understand the distribution.

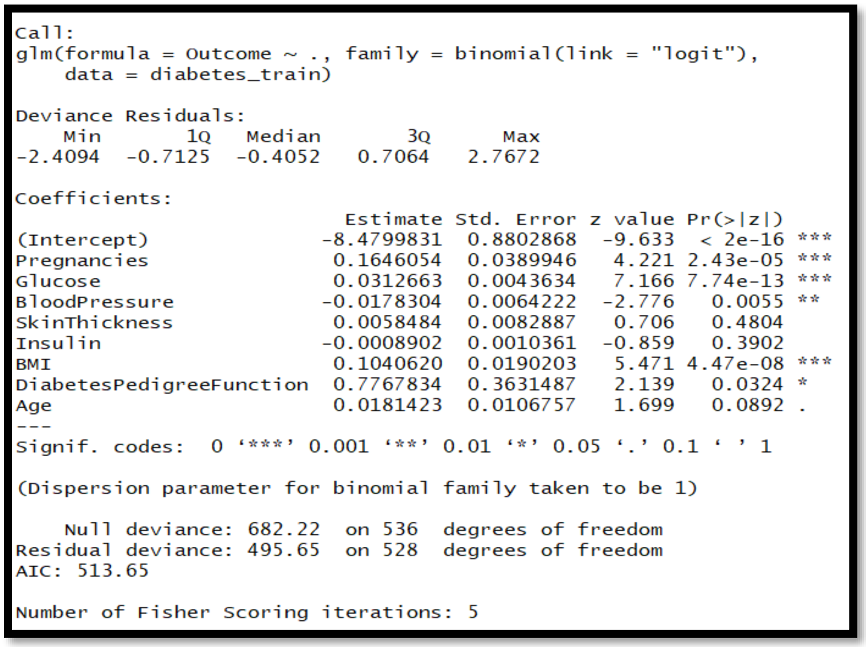


From the above output, we can infer that patients that do not have diabetes are almost double as compared to the patients that have diabetes.

3.2 Model Results

1. Logistic Regression

Using glm() function in R, we first trained a logistic regression model on our training data and then performed validation on our test data using predict() function. It predicts the outcome variable based on the values of the predictor variables. Below is the summary of the output that shows the coefficient estimates, intercept, p-values, and standard errors.

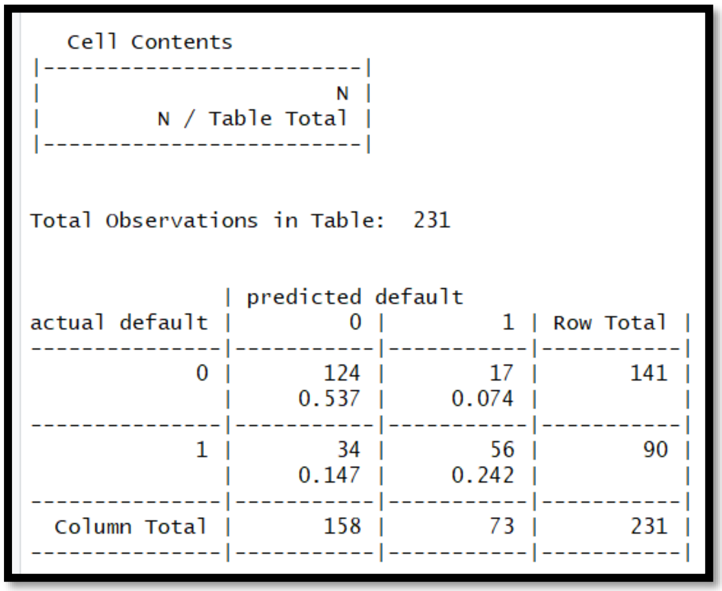


```
Call:
glm(formula = Outcome ~ ., family = binomial(link = "logit"), 
    data = diabetes_train)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.4094  -0.7125  -0.4052   0.7064   2.7672  

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)    
(Intercept)              -8.4799831  0.8802868  -9.633  < 2e-16 ***
Pregnancies               0.1646054  0.0389946   4.221 2.43e-05 ***
Glucose                   0.0312663  0.0043634   7.166 7.74e-13 ***
BloodPressure            -0.0178304  0.0064222  -2.776   0.0055 ** 
SkinThickness             0.0058484  0.0082887   0.706   0.4804    
Insulin                  -0.0008902  0.0010361  -0.859   0.3902    
BMI                       0.1040620  0.0190203   5.471 4.47e-08 ***
DiabetesPedigreeFunction  0.7767834  0.3631487   2.139   0.0324 *  
Age                       0.0181423  0.0106757   1.699   0.0892 .  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 682.22  on 536  degrees of freedom
Residual deviance: 495.65  on 528  degrees of freedom
AIC: 513.65

Number of Fisher Scoring iterations: 5
```

The above output can be interpreted in the following way:

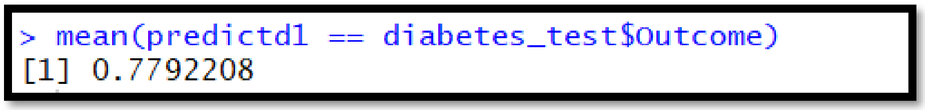
1. The estimated logistic regression line has an intercept of -8.48 approx. which is significant at 0.05 level as its corresponding p-value is extremely low and lesser than 0.05.
2. The coefficient estimate of the variable 'Pregnancies' is 0.165 approx. which implies that an increase in the number of pregnancies by 1, increases the log odds of having Diabetes by 0.165 approx., which is significant at 0.05 level (as p-value < 0.05), holding all other factors constant.
3. Similarly, other predictor variables that are positively related with the outcome variable are Glucose, Skin Thickness, BMI and Diabetes Pedigree Function. Hence, their coefficient estimate can be interpreted in terms of log odds as explained above in part (2).
4. Insulin level has a negative relation with the outcome variable i.e. with every unit increase in insulin level, the log odds of having diabetes decreases by 0.00089 approx., holding all other variables constant, significant at 0.05 level. Low insulin levels, in the real world, are attributed to the cause of Type 1 diabetes (Healthline editorial 2017). Blood Pressure is another factor that has a negative relation with the outcome variable.
5. In the above output summary, No. of Pregnancies, Glucose, Blood Pressure, BMI and Diabetes pedigree function are the variables that have a significant effect on determining if a given patient has diabetes or not (as they are significant at 0.05 level).

We have assigned probabilities of 0 and 1, where our function takes the value 0 if probability < 0.5 and 1 otherwise. Based on the outcome prediction on our test data, we can evaluate the efficiency of our logistic regression model with help of the following confusion matrix generated with the CrossTable() function:



```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  231


               | predicted default
actual default |         0 |         1 | Row Total |
---------------|-----------|-----------|-----------|
             0 |       124 |        17 |       141 |
               |     0.537 |     0.074 |           |
---------------|-----------|-----------|-----------|
             1 |        34 |        56 |        90 |
               |     0.147 |     0.242 |           |
---------------|-----------|-----------|-----------|
  Column Total |       158 |        73 |       231 |
---------------|-----------|-----------|-----------|
```

According to the confusion matrix above, 17 negative observations were incorrectly predicted as positive, where 0 = positive and 1 = negative. On the other hand, 34 positive values were incorrectly predicted as negative. Therefore, the model predicts 124 negative, and 56 positive values correctly.

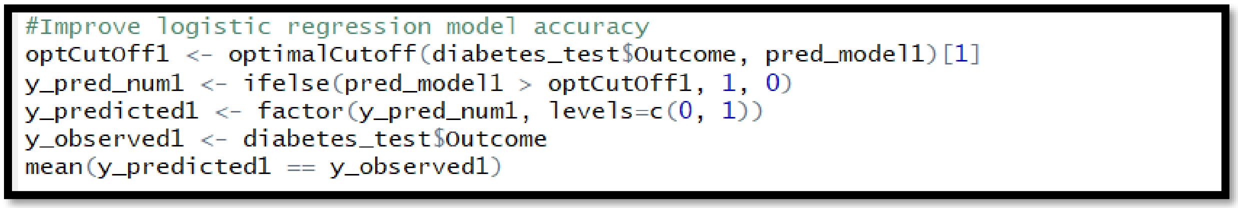


```
> mean(predictd1 == diabetes_test$Outcome)
[1] 0.7792208
```

The above prediction result shows that the model has an accuracy of 77.92% approx., which is fairly good. However, we need to consider the accuracy rate of negatives and positives, individually.

Improving Accuracy

We can improve the accuracy of our logistic regression model by using the OptimalCutOff() function using the Information Value package in R. It selects optimal cut off points for probabilities such that binary classification metrics can be maximized and minimized i.e. classification errors are minimized.



```
#Improve logistic regression model accuracy
optCutOff1 <- optimalCutoff(diabetes_test$Outcome, pred_model1)[1]
y_pred_num1 <- ifelse(pred_model1 > optCutOff1, 1, 0)
y_predicted1 <- factor(y_pred_num1, levels=c(0, 1))
y_observed1 <- diabetes_test$Outcome
mean(y_predicted1 == y_observed1)
```

Based on the optimized model algorithm above, we get the same model accuracy of approx. 78% of prediction on our test/validation data. Therefore, optimization has no effect on our logistic regression model.

Performance metrics such as Sensitivity and Specificity can be utilized to review the accuracy of the negatives and positives in our model.

Sensitivity (also called the True Positive Rate) is a measure of the proportion of correctly identified patients who have the disease, while Specificity refers to the proportion of correctly identified patients who do not have the disease. These are calculated using the formula:

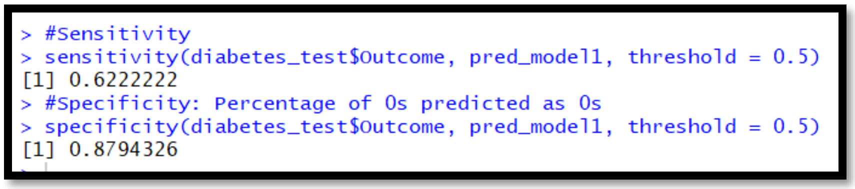
$$\text{Sensitivity} = \frac{\text{TP}}{(\text{TP} + \text{FN})},$$

$$\text{Specificity} = \frac{\text{TN}}{(\text{TN} + \text{FP})},$$

,where TP = True Positive

FN = False Negative

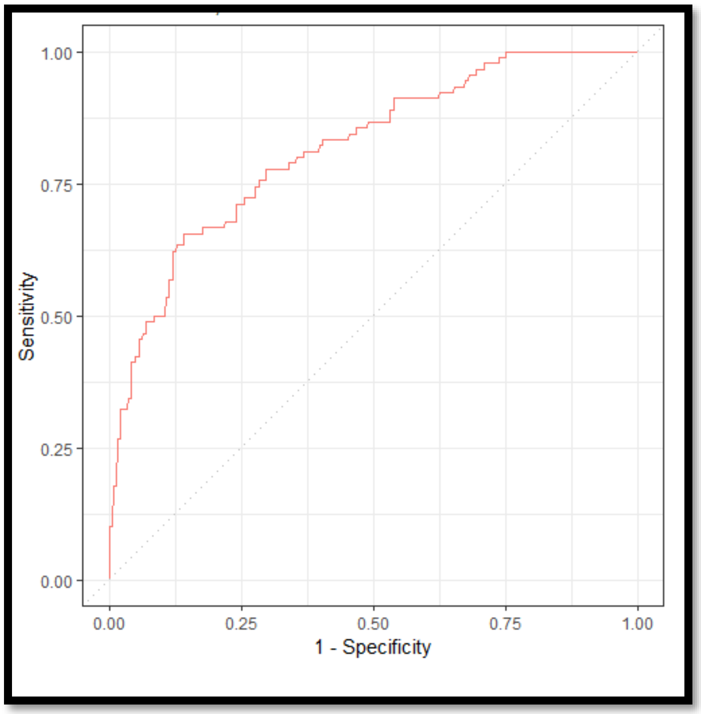
TN = True Negative



```
> #Sensitivity
> sensitivity(diabetes_test$Outcome, pred_model1, threshold = 0.5)
[1] 0.6222222
> #Specificity: Percentage of 0s predicted as 0s
> specificity(diabetes_test$Outcome, pred_model1, threshold = 0.5)
[1] 0.8794326
```

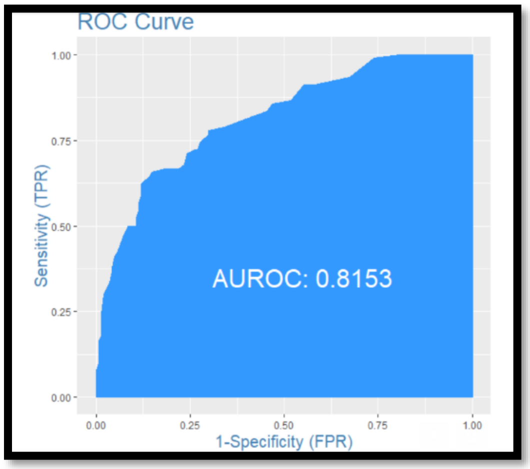
Our prediction model has a sensitivity of 62.22%, while the specificity is 88% approx.. While our specificity is high, the True Positive Rate or sensitivity is quite low. A low TPR may be a serious concern in medical practice as it is crucial to correctly identify those patients who are actually suffering from the disease. We can later use other models to predict our outcome variable and then review if the prediction accuracy has increased or not.

We can plot an ROC curve to show the relationship between Sensitivity (True Positive Rate) and the False Positive Rate (1-Specificity). A Receiver Operating Characteristic (ROC) curve describes a trade-off between the True Positive Rate (Sensitivity) and False Positive Rate (1-Specificity) which corresponds to a decision threshold. The 45 degree line lying along the diagonal shows that TPR = FPR, and, hence, there is no prediction accuracy. Therefore, an ROC curve closer to the upper left corner represents good model accuracy, while the one that is closer to the 45 degree line reflects poor accuracy.



The ROC curve for our logistic regression model prediction is away from the 45 degree line and somewhat close to the left hand border. Therefore, the curve reflects a fairly good prediction accuracy on our test data.

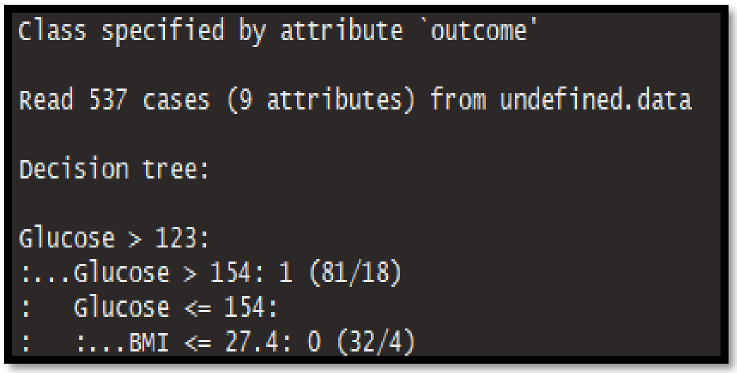
We can also perform AUROC analysis or the analysis of the Area Under the Curve that represents the degree of separability. In other words, it tells us that how well can we classify our patients correctly based on the decision threshold. As already stated above, a curve with no better accuracy will have an AUC of 0.5, where FPR = TPR, whereas, an ideal AUC = 1.



In the above plot, AUC = 0.8153 which indicates that our model does fairly well at classifying the patients, correctly, based on our decision threshold.

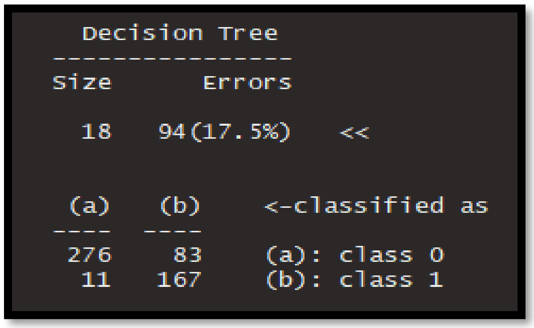
2. Decision Tree Analysis

After Training our decision tree, we get the following output:



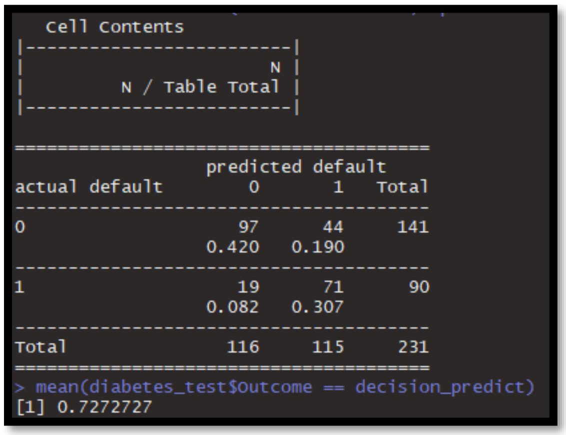
```
Class specified by attribute `outcome'

Read 537 cases (9 attributes) from undefined.data

Decision tree:

Glucose > 123:
:...Glucose > 154: 1 (81/18)
:   Glucose <= 154:
:   :...BMI <= 27.4: 0 (32/4)
```

The above output if the summary of the decision tree model. This works the same as an ‘if-else’ statement where to check if the patient has diabetes or not, the model first checks if the patient’s glucose level is greater than 123 or not. If the glucose level is greater than 123, then it checks if the glucose level is greater than 154 or not, which, here we can notice that the output is 1 for 81 cases as ‘The patient does have diabetes’ and for 18 cases ‘The patient does not have diabetes’. Considering the values 81/99, which is around 81.8% times this stands true that if the glucose level is greater than 154, then the patient will have diabetes. Similarly, the model checks for other cases such as BMI when the glucose level is less than 154.



```
  Decision Tree
----------------
Size      Errors

  18   94(17.5%)   <<


 (a)   (b)    <-classified as
----  ----
 276    83    (a): class 0
  11   167    (b): class 1
```

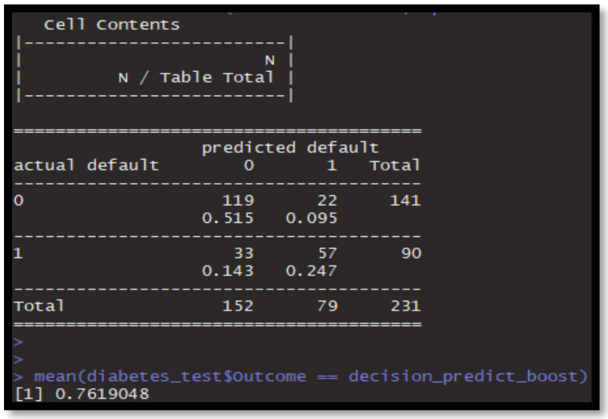
The above results show that the model was able to identify in the training dataset, that there were 94 erroneous cases, which is approximately 17.5% of the total.



```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

=======================================
                  predicted default
actual default         0       1   Total
---------------------------------------
0                     97      44     141
                   0.420   0.190
---------------------------------------
1                     19      71      90
                   0.082   0.307
---------------------------------------
Total                116     115     231
=======================================
> mean(diabetes_test$Outcome == decision_predict)
[1] 0.7272727
```

The above results are for the predictions made by the model on the test dataset. We can notice that the model was able to predict 97 correct True negative cases, which means that the patient did not have diabetes and the model identified it correctly. The specificity here is 97/141 = 0.68. The model also predicted 71 correct True positive cases, which means that the patient has diabetes and the model identified it correctly. The sensitivity here is 71/90 = 0.7888. The total accuracy of the model turned out to be 97+71/231 = 0.727 (72.73%). We will try to perform boosting and check if the accuracy of the model increases or not.

After Boosting-



```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

=======================================
                  predicted default
actual default         0       1   Total
---------------------------------------
0                    119      22     141
                   0.515   0.095
---------------------------------------
1                     33      57      90
                   0.143   0.247
---------------------------------------
Total                152      79     231
=======================================
>
>
> mean(diabetes_test$Outcome == decision_predict_boost)
[1] 0.7619048
```

The above results are for the predictions made by the model after the boosting on the test dataset. We can notice that the model was able to predict 119 correct True negative cases, which means that the patient did not have diabetes and the model identified it correctly. The specificity here is 119/141 = 0.8439. The model also predicted 57 correct True positive cases, which means that the patient has diabetes and the model identified it correctly. The sensitivity here is 57/90 = 0.6333. The total accuracy of the model turned out to be 119+57/231 = 0.76 (76.19%) which is 4% better than the accuracy of the model before the boosting. The selected number of trials that gave us the maximum accuracy was 20.

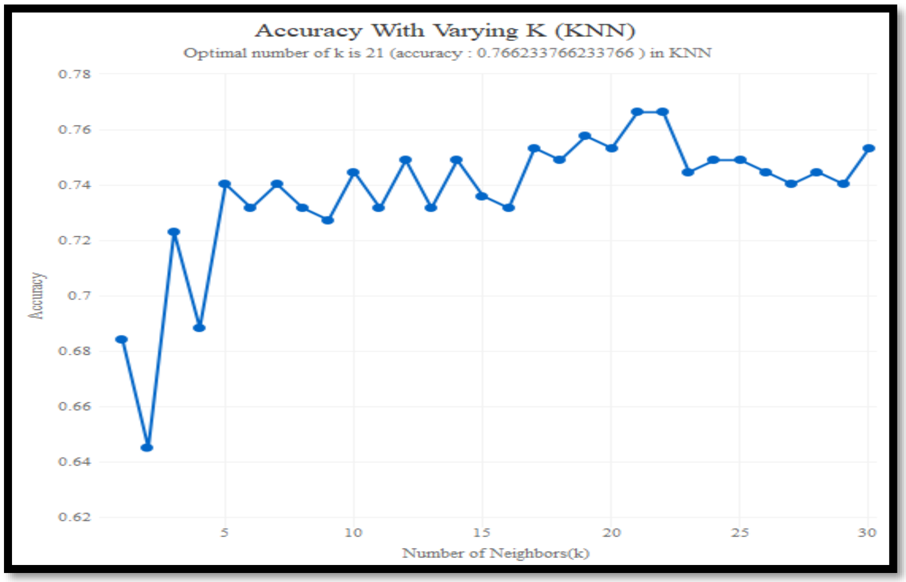
However, the increased accuracy is due to the increased specificity and not the sensitivity. In the healthcare field, sensitivity matters more than specificity. Thus, in this case, it would not be ideal to use the boosting method and stay with the actual model.

3. KNN Classification

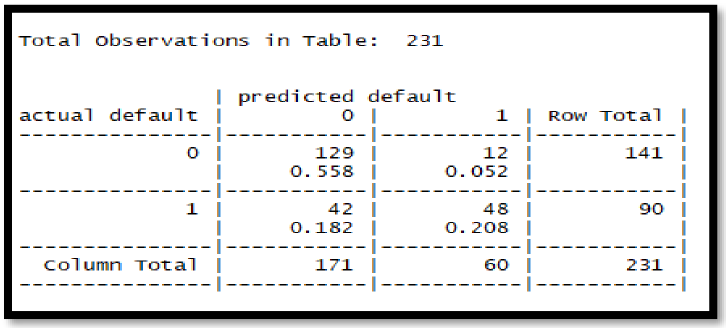
In R, the syntax and method of a KNN classification are different from other methods such as decision tree and logistic regression. In other models, we first predict our model using the training data and then using the predict() function and testing data, we predict our model to ensure there is no overfitting or underfitting whereas, in the case of KNN classification, we just use a simple KNN() function which takes both the training and testing data as inputs, along with the qualitative field.

The first parameter of the knn function includes all the training data of our predictor variables in a data frame whereas the second parameter contains testing data of our predictor variable. The third and the fourth parameters contain the training data of our outcome variable and the value for K i.e. the number of neighbors. In this project, we have applied the knn function in a for loop which iterates 30 times for the values of K(1 to 30). The purpose is to find the value of K which gives us the maximum model accuracy. After that, we calculate the accuracy, sensitivity, and the specificity of our model.

After iterating the model for the value of K =1 to K =30, we got different accuracies. Below is a plot of accuracies we got for different values of K:



From the above line chart, we can infer that for K = 21, we have the maximum model accuracy i.e. 76.23%. Now, for K = 21, the output of the confusion matrix is as follows:



```
Total Observations in Table:  231

               | predicted default
actual default |         0 |         1 | Row Total |
---------------|-----------|-----------|-----------|
             0 |       129 |        12 |       141 |
               |     0.558 |     0.052 |           |
---------------|-----------|-----------|-----------|
             1 |        42 |        48 |        90 |
               |     0.182 |     0.208 |           |
---------------|-----------|-----------|-----------|
  Column Total |       171 |        60 |       231 |
---------------|-----------|-----------|-----------|
```

From the above output of the confusion matrix, we can interpret that there are 48 true positives i.e. 48 out of 90 positives were correctly identified while 42 were incorrectly identified as negatives. Also, the model correctly identified 129 false negatives while it incorrectly identified 12 negatives as positives. Next, we can calculate the sensitivity and the specificity for our model.

Sensitivity = True Positive/ (True Positive + False Negative)

Sensitivity = 48/(42+48) = 53.33%

Specificity = True Negative/(True Negative + False Positive)

Specificity = 129/ (129 + 12) = 91.48 %

The sensitivity of the model comes out to be 53.33% which is very low in this field as it is very important to correctly predict if a patient has diabetes such that timely treatment can be provided.

4. Discussion/Conclusion

The table below summarizes our model results:

Model	Accuracy	Sensitivity	Specificity
1. Logistic Regression	77.92%	62.22%	87.94%
After Optimization	77.92%	62.22%	87.94%
2. Decision Tree	72.73%	78.88%	68.79%
After Boosting	76.19%	63.33%	84.40%
3. KNN Classification	76.23%	53.33%	91.48%
After Boosting	N/A	N/A	N/A

The Logistic Regression model gives us the best overall accuracy out of all the models i.e. 77.92%. However, it gives a True Positive Rate (TPR) or Sensitivity of 62.22% which is quite low and raises concern. Its overall accuracy, sensitivity and specificity remain the same after Optimization. On the other hand, the decision tree gives us an accuracy of 72.73% before boosting, but it gives us the highest True Positive Rate of 78% out of all the three models. Boosting technique increases its overall accuracy but lowers its sensitivity. The KNN Classification technique gives the lowest sensitivity which is a serious matter of concern. In medical practice, sensitivity is given due importance as it is crucial to correctly identify the patients who have a particular disease, to avoid delay in treatment, which otherwise can worsen their medical condition. In other words, when we are faced with a trade-off between specificity and sensitivity, it's better to have incorrectly predicted a healthy patient as having the disease rather than incorrectly identifying a sick patient as not having the disease.

Therefore, if we are concerned about the overall accuracy of a model, logistic regression is the best technique to predict if a given patient has diabetes or not. However, as there is not much difference in the overall accuracies of logistic regression and decision tree (after boosting) models, one can select the best model based on their respective True Positive Rate or Sensitivity. Hence, if we lay emphasis on sensitivity, then a Decision tree model (before boosting) works best as a classifier as it has the highest True Positive Rate of 78.88%, among all classifiers.

One of the Strengths of our data set is that there are no missing or N/A values that have significantly cut down the time taken for us to perform our analysis. The classification models such as decision trees take significantly less computation time to predict the outcome variable. In addition, the nature of the data set is such that we can apply varied number of classification techniques to select the best model for solving our research problem. Therefore, more options will help us choose the best model with the highest accuracy and sensitivity.

The discussed classification techniques help answer our research question about predicting diabetes in PIMA Indians. However, below are a few limitations associated with our models and data set:

1. Except the Decision tree model (before boosting), the sensitivity of True Positives is consistently low across all models. This can be attributed to the fact that our Outcome variable consists of the number of negatives or 0s that are almost double the number of positives or 1s in our data set (As discussed under EDA), which is also one of the major limitations of our data set. Thus, it is plausible that all our models, except decision tree model (before boosting), predict more negatives or 0s than positives or 1s.
2. The boosting technique increased the accuracy of the decision tree model, but lowered its sensitivity significantly by 15.55%. Therefore, our boosted decision tree model gives us a poor True Positive Rate.
3. The optimalcutoff() function has no effect on the efficiency of the logistic regression model, which further limits our options from which we can select the best model.

4.1 Future scope of study

We can attempt to increase the accuracy and other performance metrics of our model by using other classifiers such as Random Forests and Linear Discriminant Analysis (LDA). The model results can then be compared to the above summary table, based on which we can expand our options for model selection. Most importantly, we can use a '**Gradient Boosting Technique'** which is most commonly applied to settings where data is unbalanced. It is a procedure that produces a prediction model based on the ensemble or collection of weak learners (typically in decision trees). Additionally, we can gain better accuracy and sensitivity if we have larger data set that is fairly balanced.

Therefore, the best model out of all the classifiers is a decision tree (before boosting), based on its high sensitivity. The model can be used by Physicians to classify PIMA Indians on the basis of the outcome variable i.e. if they have the diabetes or not. However, as the sensitivity is 78.88%, there remains a scope for further improvement which implies that it is quite plausible for a physician to incorrectly identify some of the patients who have the disease as not having the disease. This can have major implications such as delay in treatment that can further worsen their medical condition and lead to significantly high treatment cost.

References:

Case Study: Predicting the Onset of Diabetes Within Five Years (part 1 of 3). (2018, June 21). Retrieved from https://machinelearningmastery.com/case-study-predicting-the-onset-of-diabetes-within-five-years-part-1-of-3/

Schulz, L. O., Bennett, P. H., Ravussin, E., Kidd, J. R., Kidd, K. K., Esparza, J., & Valencia, M. E. (2006, August 01). Effects of Traditional and Western Environments on Prevalence of Type 2 Diabetes in Pima Indians in Mexico and the U.S. Retrieved from http://care.diabetesjournals.org/content/29/8/1866

Schulz, L. O., Bennett, P. H., Ravussin, E., Kidd, J. R., Kidd, K. K., Esparza, J., & Valencia, M. E. (2006, August 01). Effects of Traditional and Western Environments on Prevalence of Type 2 Diabetes in Pima Indians in Mexico and the U.S. Retrieved from http://care.diabetesjournals.org/content/29/8/1866

Healthline. (2019). Diabetes Causes: How Do You Get Type 1 and Type 2 Diabetes. [online] Available at: https://www.healthline.com/health/diabetes-causes

How to build a logistic regression model from scratch in R. (2018, October 02). Retrieved from https://www.r-bloggers.com/how-to-build-a-logistic-regression-model-from-scratch-in-r/

Jain, R., & Jain, R. (2017, March 20). Decision Tree. It begins here. Retrieved from https://medium.com/@rishabhjain_22692/decision-trees-it-begins-here-93ff54ef134

1354257684624621. (2018, December 30). K-nearest Neighbors Algorithm with Examples in R (Simply Explained knn). Retrieved from https://towardsdatascience.com/k-nearest-neighbors-algorithm-with-examples-in-r-simply-explained-knn-1f2c88da405c

User180984, User180984user180984 477, & Jan KukackaJan Kukacka 6. (n.d.). K - NN classifier vs logistic regression for the MNIST data set. Retrieved from https://stats.stackexchange.com/questions/326012/k-nn-classifier-vs-logistic-regression-for-the-mnist-data-set/326080

Koslowsky, S., & Hanks, H. (2018, August 30). On Variable Importance in Logistic Regression - Predictive Analytics Times - machine learning & data science news. Retrieved from https://www.predictiveanalyticsworld.com/patimes/on-variable-importance-in-logistic-regression/9649/

Appendix:

```
##packages install

#install.packages("dplyr")

#install.packages("caret")

#install.packages("plotROC")

#install.packages("InformationValue")

#install.packages("gmodels")

#install.packages("pROC")

#install.packages("class")

#install.packages("highcharter")

#install.packages("precrec")

#install.packages("corrplot")

#install.packages("C50")


##importing libraries
```

```
library(dplyr)

library(caret)

library(plotROC)

library(InformationValue)

library(gmodels)

library(pROC)

library(class)

library(highcharter)

library(precrec)

library(corrplot)

library(C50)


##to maintain consistency

set.seed(5689)

##data load

diabetes <- read.csv(file = "diabetes.csv", header = TRUE)

##converting our outcome variable to factor

diabetes$Outcome <- as.factor(diabetes$Outcome)

##EDA##

##descriptive statistic

summary(diabetes)

prop.table(table(diabetes$Outcome))

##Plotting correlations between predictor variables

correlat <- cor(diabetes[, setdiff(names(diabetes), 'Outcome')])

corrplot(correlat)

##plotting outcome variable


outvar <- diabetes %>% group_by(Outcome)%>% count(Outcome)
```

```
ggplot(data= outvar, aes(x = Outcome,y = n)) +geom_bar(stat='identity',colour="white", fill =
"#FFA07A")



#split data into train/test

nrows <- NROW(diabetes)

                    # fix random value

index <- sample(1:nrows, 0.7 * nrows)   # shuffle and divide


diabetes_train <- diabetes[index,]

diabetes_test <- diabetes[-index,]


###decision trees#####




##training model of decision tree

decision_model <- C5.0(diabetes_train[-9], diabetes_train$Outcome)

summary(decision_model)

##predicting on test data

decision_predict <- predict(decision_model, diabetes_test)

##finding model accuracy

CrossTable(diabetes_test$Outcome, decision_predict,

         prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,

         dnn = c('actual default', 'predicted default'))

mean(diabetes_test$Outcome == decision_predict)

sensitivity_dt <- 71/(71+19)

specificity_dt <- 97/(97+44)

##improving model performance

grid_c50 <- expand.grid(.model = "tree",
                    .trials = c(10, 20, 30, 40),
                    .winnow = "FALSE")
ctrl <- trainControl(method = "repeatedcv",
                  number = 10, repeats = 10)
m_c50 <- train(Outcome ~ ., data = diabetes_train, method = "C5.0",
             metric = "Kappa", trControl = ctrl,
             tuneGrid = grid_c50)



##predicting on test data
decision_predict_boost <- predict(m_c50, diabetes_test)
##finding model accuracy
CrossTable(diabetes_test$Outcome, decision_predict_boost,
          prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
          dnn = c('actual default', 'predicted default'))



mean(diabetes_test$Outcome == decision_predict_boost)
sensitivity_dtb <- 57/(57+33)
specificity_dtb <- 119/(119+22)
#Standardize predictors
diabetes$Pregnancies <- scale(diabetes$Pregnancies, scale=TRUE, center = TRUE)
diabetes$Glucose <- scale(diabetes$Glucose, scale=TRUE, center = TRUE)
diabetes$BloodPressure <- scale(diabetes$BloodPressure, scale = TRUE, center = TRUE)
diabetes$SkinThickness <- scale(diabetes$SkinThickness, scale = TRUE, center = TRUE)
diabetes$Insulin <- scale(diabetes$Insulin, scale = TRUE, center = TRUE)
diabetes$BMI <- scale(diabetes$BMI, scale = TRUE, center = TRUE)
diabetes$DiabetesPedigreeFunction <- scale(diabetes$DiabetesPedigreeFunction, scale = TRUE,
center = TRUE)
```

```
diabetes$Age <- scale(diabetes$Age, scale = TRUE, center = TRUE)


#KNN CLASSIFICATION ####

acc_test <- numeric()

##for loop from k = 1 to 30 to find maximum accuracy of algorithm

for(i in 1:30){


  predict <- knn(train=diabetes_train[,-9], test=diabetes_test[,-9], cl=diabetes_train[,9], k=i,
prob=T)


  acc_test <- c(acc_test,mean(predict==diabetes_test[,9]))


}


acc <- data.frame(k= seq(1,30), cnt = acc_test)


opt_k <- subset(acc, cnt==max(cnt))[1,]

sub <- paste("Optimal number of k is", opt_k$k, "(accuracy :", opt_k$cnt,") in KNN")

##plot of accuracies using highcharter

hchart(acc, 'line', hcaes(k, cnt)) %>%


  hc_title(text = "Accuracy With Varying K (KNN)") %>%


  hc_subtitle(text = sub) %>%


  hc_add_theme(hc_theme_google()) %>%


  hc_xAxis(title = list(text = "Number of Neighbors(k)")) %>%
  hc_yAxis(title = list(text = "Accuracy"))


##creation of model using optimized value of k

pre_knn <- knn(train = diabetes_train[,-9], test = diabetes_test[,-9], cl = diabetes_train[,9],
k=opt_k$k, prob=T)

##finding model accuracy

CrossTable(diabetes_test$Outcome, pre_knn,prop.chisq = FALSE, prop.c = FALSE, prop.r =
FALSE,

                                              dnn = c('actual default', 'predicted default'))

mean(diabetes_test[,9] == pre_knn)


sensitivity_knn <- 48/(48+42)

specificity_knn <- 129/(129+12)

##logistic regression


#Train model using glm

logistic_dmodel <- glm(Outcome ~., family=binomial(link="logit"), data=diabetes_train)

summary(logistic_dmodel)


#Testing model accuracy on test data

pred_model1 <- predict(logistic_dmodel, newdata = diabetes_test, type = "response")

predictd1 <- ifelse(pred_model1 > 0.5, 1,0)


pred_table1 <- table(predicted = predictd1, actual = diabetes_test$Outcome)

pred_table1

mean(predictd1 == diabetes_test$Outcome)


#Confusion Matrix

CrossTable(diabetes_test$Outcome, predictd1,
          prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
          dnn = c('actual default', 'predicted default'))

#Sensitivity
sensitivity(diabetes_test$Outcome, pred_model1, threshold = 0.5)

#Specificity: Percentage of 0s predicted as 0s
specificity(diabetes_test$Outcome, pred_model1, threshold = 0.5)

#Improve logistic regression model accuracy
optCutOff1 <- optimalCutoff(diabetes_test$Outcome, pred_model1)[1]
y_pred_num1 <- ifelse(pred_model1 > optCutOff1, 1, 0)
y_predicted1 <- factor(y_pred_num1, levels=c(0, 1))
y_observed1 <- diabetes_test$Outcome
mean(y_predicted1 == y_observed1)



#Plot AUROC curve
plotROC(diabetes_test$Outcome, pred_model1)

precrec_obj <- evalmod(scores = pred_model1, labels = diabetes_test$Outcome)
autoplot(precrec_obj)
```

##END##