PS0002 Project

Report on the analysis of the PimaIndianDiabetes dataset

By: Sowmiya Arasi Senthilkumar, Nisharani Palani

U2040964B, U2040654B

Nanyang Technological University

1.0 Introduction

Diabetes is a prevalent disease in societies, the 'global diabetes prevalence in 2019 is estimated to be 9.3% (463 million people), rising to 10.2% (578 million) by 2030' (Saeedi, et al., 2019). We have sought to analyze diabetic data, to better understand the factors and possibility of diabetes. 'Diabetes is a chronic disease that occurs either when the pancreas does not produce enough insulin or when the body cannot effectively use the insulin it produces. Insulin is a hormone that regulates blood sugar.' (World Health Organisation, 2021). If left untreated overtime, it can lead to possibly life-threatening results such as heart failure and kidney failure (World Health Organisation, 2021).

For our report, we have utilized the programming language 'R', to analyze the data set PimaIndiansDiabetes, from the National Institute of Diabetes and Digestive and Kidney Diseases (Newman, Hettich, Blake, & Merz, 1998). Our main objective is to predict the chances of an individual being diabetic based on the dataset's variables. To supplement our objective we will be addressing the following study question: What are the relationships between the outcome of diabetes and the variables from the data set?

2.0 Data Preparation and Information

The dataset PimaIndiansDiabetes contains 8 independent numeric predictor variables and 1 categorical outcome variable, Diabetes. It has 768 observations in total. The data has been taken from individuals who are females of at least 21 years old and are of Pima Indian heritage. We retain all variables of the dataset in this stage due to their limited selection. However, some of the columns have illogical measurements, such as Glucose, Mass, Insulin, Triceps possessing values of 0, which is not realistic for a human. Hence, we will treat such occurrences of 0s as missing values and impute the mean of such columns as the new values in the subsequent analysis.

Variables	Description
Diabetes	Denotes if the person has diabetes
Pregnant	Number of times pregnant
Glucose	Plasma glucose concentration(mg/dL)
Pressure	Diastolic blood pressure (mm Hg)
Triceps	Triceps skin fold thickness (mm)
Insulin	2-Hour serum insulin (mu U/ml)
Mass	Body mass index (kg/m^2)
Pedigree	Diabetes pedigree function
Age	Age in years

Figure 1 Description of Variables

	Median	Mean	Standard Deviation	Count
Pregnant	3.00	3.84	3.37	768
Glucose	117.00	121.69	30.54	768
Pressure	72.00	72.41	12.38	768
Triceps	29.00	29.15	10.48	768
Insulin	125.00	155.55	118.78	768
Mass	32.30	32.46	6.92	768
Pedigree	0.37	0.47	0.33	768
Age	29.00	33.24	11.76	768

Figure 2 Descriptive Statistics of Predictor Variables

3.0 Preliminary Exploratory Analysis

From the correlation matrix in Figure 3, we initially concluded that out of all the variables, the best predictors are ranked in order by Glucose, Mass, and Age, which each have moderately positive correlation with diabetes. While moderately positive correlation exists between pairs of variables such as Pregnant and Age, and Triceps and Mass. The linear relationship between Triceps and Mass is shown in Figure 4. Furthermore, from Figure 5, the mean age of people having diabetes is higher than the mean age of non-diabetic people. In addition, the mean value of the other predictor variables is also higher in diabetic individuals. This can be seen from the boxplots for each predictor variable with diabetes (Figure 11 & 12 in the Appendix).

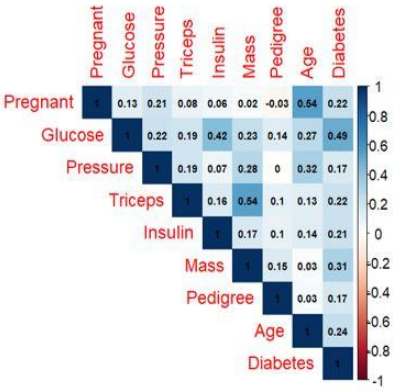

Figure 3 Correlation Matrix

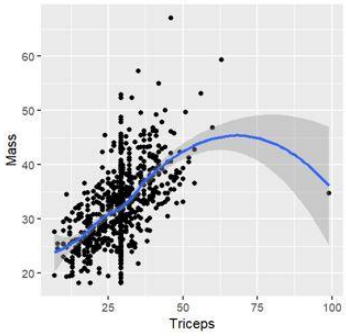

Figure 4 Scatterplot for Triceps & Mass

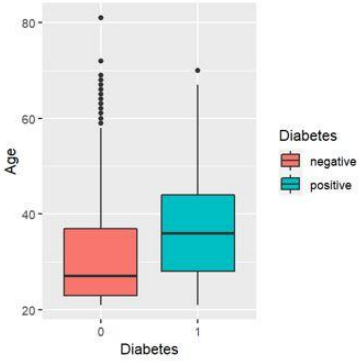

Figure 5 Boxplot of Age & Diabetes

4.0 Methodology

The dataset's response variable, Diabetes, is categorical, as such our goal of predicting chances of diabetes based on a set of predictor variables, can be labeled as a classification problem. We have utilized supervised machine learning algorithms such as K-Nearest Neighbor(kNN) and Logistic Regression. Furthermore, we have also performed Clustering as an attempt to observe how many classes the data can be organized into based on their similar traits.

4.1 Classification

For both kNN and Logistic Regression algorithms, we have set the training and test data by using the standard 80/20 splitting rule. By randomly choosing 80% of the data to be part of the training set to create a predictive model, while the rest of the data (20%) serves as the test set for model evaluation.

4.1.1 K-Nearest Neighbor(kNN)

The kNN algorithm stores the available data and classifies a new data point based on Euclidean distance measures. We have also normalized the dataset to ensure that our algorithm remains unaffected by differing variable magnitudes. Initially by running the algorithm for the predictor variables Pregnant, Age and Pedigree to predict Diabetes, we however concluded that it only gave an accuracy of 72.9% from the confusion matrix and found an optimum k-value of 42. This has a significantly lower accuracy than when running the algorithm for all the predictor variables. Figure 6 showcases a glimpse of the results of the more effective kNN algorithm involving all variables, 'Diabetes' and 'pred' compares the original outcome versus the predicted outcome, while 'prob' details the probability of the prediction. This gave an accuracy of 79.2% and an optimal k-value of 19 (Figure 7). Hence we are able to conclude that the algorithm involving the entire set of predictors gave a higher classification accuracy for Diabetes.

	Diabetes	pred	prob
3	1	1	0.7894737
5	1	1	0.5789474
8	0	1	0.5263158

Figure 6 kNN classification probability (First 3 observations of test data)

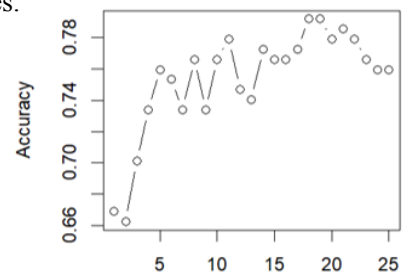

Figure 7 Optimal K

4.1.2 Logistic Regression

Logistic regression is used to model the probability of the occurrence of a certain event. It is especially useful in the case when the outcome variable is binary. Our outcome variable which is Diabetes has 2 outcomes which are either positive (diabetic) or negative (non-diabetic) which we have set to 1 and 0 respectively. From Figure 8, we can see that for only Pressure and Insulin, the odds of being diabetic decrease with every one unit change, as we can see from the estimated coefficients being negative. For every one unit change in Pedigree, the ln(odds) of being diabetic (versus non-diabetic) increases by 0.759 (from Figure 8), that is, the odds of being diabetic increase by $e^{0.759}$ = 2.14 times. Compared to the other variables, one unit change in Pedigree increases the chances of having diabetes the most. Pregnancy, Glucose, Mass and Pedigree have p-values less than 0.05 which shows that these variables have a statistically significant relationship with the outcome variable, Diabetes. The confusion matrix (Figure 9) describes the performance of the logistic regression model. By comparing the actual outcomes of the test data set alongside the predicted outcomes, the confusion matrix allows us to deduce that the accuracy of the model is 80.5%.

```
Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -8.5682242  0.8813432  -9.722  < 2e-16 ***
Pregnant     0.1122102  0.0362653   3.094  0.00197 **
Glucose      0.0353983  0.0041600   8.509  < 2e-16 ***
Pressure    -0.0055548  0.0096745  -0.574  0.56585
Triceps      0.0030979  0.0146064   0.212  0.83204
Insulin     -0.0008717  0.0012409  -0.703  0.48235
Mass         0.0785731  0.0200425   3.920 8.84e-05 ***
Pedigree     0.7589869  0.3296161   2.303  0.02130 *
Age          0.0156036  0.0105913   1.473  0.14069
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 8 Outcome of Logistic Regression

		Actual Values: Positive	Actual Values: Negative
Predicted Values	Positive	91 (true positive)	23 (false positive)
Predicted Values	Negative	7 (false negative)	33 (true negative)

Figure 9 Confusion Matrix

We can see that both kNN and Logistic regression can correctly classify around 80% of the points in the test set. From analyzing the confusion matrix of both the models, the logistic regression model gives more true positives while the kNN model gives more false negatives.

4.2 K-means Clustering

K-means clustering is an unsupervised machine learning algorithm, and requires an input for the value of k clusters in order to partition the dataset by minimizing the total within-cluster-variation. As the input k is not learnt from the data it is clear there can be varied numbers of clusters within a data set. As such, we have utilized the elbow method here to determine an optimal k of 2 for our algorithm (Figure 10). This appears to fit our dataset as they can represent the Diabetic and non-Diabetic classes. By excluding the diabetes column in the clustering and generating the confusion matrix, we determined the algorithm had an accuracy of 70.3%

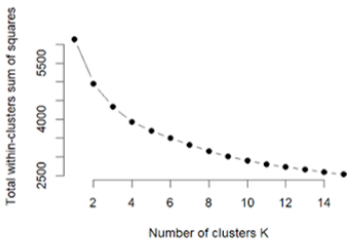

Figure 10 K-Means Clustering

5.0 Conclusion

Overall, we have concluded that among our 3 algorithms, logistic regression indeed boasts the highest accuracy percentage and singles out the most significant predictors linked to the risk of diabetes such as Pregnancy, Glucose, Mass and Pedigree. This outcome is indeed backed by studies that show factors such as glucose do have influence upon diabetes (Li, et al., 2020).

Limitations of our analysis include the imputation of missing values as average values, this can affect accuracy as such values are not representative of the actual data. In addition, the dataset that we considered only has data of females of the Pima Indian heritage, more accurate prediction can be obtained if we have data of females from other heritages. We believe that by predicting diabetes correctly, health policies and interventions can be rolled out in order to combat the prevalence of diabetes in societies.

References

Li, C., Yang, Y., Liu, X., Li, Z., Liu, H., & Tan, Q. (2020). Glucose metabolism-related gene polymorphisms as the risk predictors of type 2 diabetes. *Diabetology & Metabolic Syndrome Volume 12*, Article 97.

Newman, D., Hettich, S., Blake, C., & Merz, C. (1998). UCI Machine Learning Repository. Irvine, CA: University of California, Department of Information and Computer Science. Retrieved from MLRespository: http://www.ics.uci.edu/~mlearn/MLRepository.html

Saeedi, P., Petersohn, I., Salpea, P., Malanda, B., Karuranga, S., Unwin, N., . . . Williams, R. (2019). Global and regional diabetes prevalence estimates for 2019 and projections for 2030 and 2045: Results from the International Diabetes Federation Diabetes Atlas, 9th edition. *Diabetes Research and Clinical Practice*, Volume 157.

World Health Organisation. (2021, November 10). *WHO Health News Room*. Retrieved from World Health Organisation: https://www.who.int/news-room/fact-sheets/detail/diabetes

Appendix

Extra Graphs:

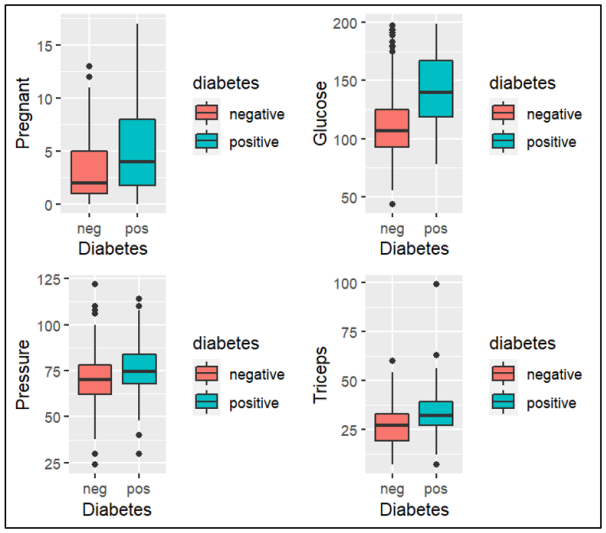

Figure 11 Boxplot of predictors against Diabetes

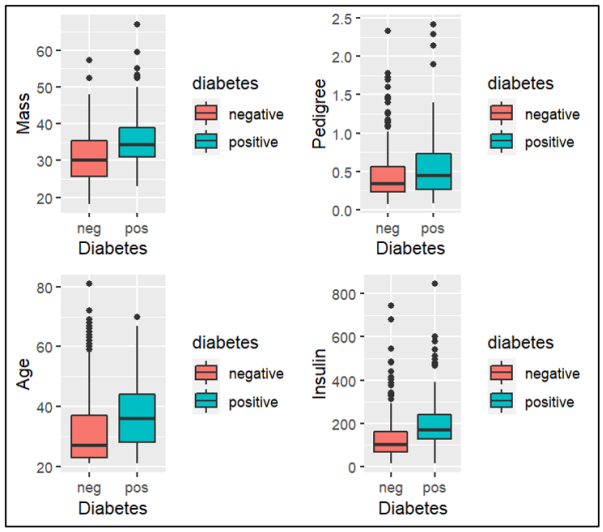

Figure 12 Boxplot of predictors against Diabetes

R code:

```
install.packages("mlbench")
install.packages("dplyr")
install.packages("ggplot2")
install.packages("corrplot")
library(mlbench)
library(dplyr)
library(ggplot2)
library(corrplot)
data("PimaIndiansDiabetes")
PimaIndianDiabetes =PimaIndiansDiabetes[, 2:9][PimaIndiansDiabetes[, 2:9] == 0] <- NA
df = data.frame(PimaIndiansDiabetes)
colnames(df)=c("Pregnant","Glucose","Pressure","Triceps","Insulin","Mass","Pedigree","Age", "Diabetes")
df
df %>% summarise(samplesize=n())
PimaIndianDiabetes= df
PID1=PimaIndianDiabetes %>% mutate(Diabetes = factor(ifelse(Diabetes == "pos",1,0)))
summary(PID1)
sum(is.na(PID1))
mean_insulin = mean(PID1$Insulin,na.rm=TRUE)
mean_glucose = mean(PID1$Glucose,na.rm=TRUE)
mean_pressure = mean(PID1$Pressure,na.rm=TRUE)
mean_triceps = mean(PID1$Triceps,na.rm=TRUE)
mean_mass = mean(PID1$Mass,na.rm=TRUE)
PID1$Insulin[is.na(PID1$Insulin)]<-mean_insulin
PID1$Glucose[is.na(PID1$Glucose)]<-mean_glucose
PID1$Pressure[is.na(PID1$Pressure)]<- mean_pressure
PID1$Triceps[is.na(PID1$Triceps)]<- mean_triceps
PID1$Mass[is.na(PID1$Mass)]<- mean_mass
```

```
str(PID1)
PIDnew = PID1
PIDnew$Diabetes = as.numeric((PIDnew$Diabetes == 1))
#Correlation matrix
corrplot(cor(PIDnew[1:9]), type="upper", method="color",addCoef.col = "black",number.cex = 0.6)
#boxplot for Age&Diabetes
df %>% ggplot(aes(x=Diabetes, y=Age, fill=Diabetes)) + geom_boxplot()+ylim(20,90) +
scale_fill_discrete(name = "diabetes", labels = c("negative", "positive"))
#scatter plot for Triceps&Mass
PID1 %>% ggplot(aes(x=Triceps, y=Mass)) + geom_point() +geom_smooth()

#initial kNN that's only on the variables Pregnant, Pedigree, Age
nor <-function(x) { (x -min(x))/(max(x)-min(x)) }
PID3 <- PID1 %>% select(Pregnant, Pedigree,Age, Diabetes)
PID3[,1:3] <- sapply(PID3[,1:3], nor)
str(PID1)
str(PID3)
#split data
set.seed(100)
training.idx <- sample(1: nrow(PID3), size=nrow(PID3)*0.8)
train.data <-PID3[training.idx, ]
test.data <- PID3[-training.idx, ]
#kNN classification
library(class)
set.seed(100)
knn1<-knn(train.data[,1:3], test.data[,1:3], cl=train.data$Diabetes, k=3)
mean(knn1 ==test.data$Diabetes)

#optimisation algorithm for finding best k value
ac<-rep(1, 50)
for(i in 1:50){
  set.seed(101)
  knn.i<-knn(train.data[,1:3], test.data[,1:3], cl=train.data$Diabetes, k=i)
  ac[i]<-mean(knn.i ==test.data$Diabetes)
  cat("k=", i, " accuracy=", ac[i], "\n")}
plot(ac, type="b", xlab="K",ylab="Accuracy")
#k=42 results in highest accuracy, knn correctly classifies 72.7% of the points in the test data set.
library(class)
set.seed(101)
knn1<-knn(train.data[,1:3], test.data[,1:3], cl=train.data$Diabetes, k=42)
mean(knn1 ==test.data$Diabetes)
table(knn1, test.data$Diabetes)
#confusion matrix gives accuracy of 72.7%

#kNN on all the variables
#Normalize numeric variables
numvar<- sapply(PID1, is.numeric)
nor <-function(x) { (x -min(x))/(max(x)-min(x)) }
PID2 <- PID1
PID2[,1:8] <- sapply(PID1[,1:8], nor)
str(PID1)
str(PID2)
#split data
```

```
set.seed(100)
training.idx <- sample(1: nrow(PID2), size=nrow(PID2)*0.8)
train.data <-PID2[training.idx, ]
test.data <- PID2[-training.idx, ]
#kNN classification
library(class)
set.seed(100)
knn1<-knn(train.data[,1:8], test.data[,1:8], cl=train.data$Diabetes, k=19, prob=TRUE)
mean(knn1 ==test.data$Diabetes)
head(data.frame(test.data, pred=knn1, prob=attr(knn1, "prob")))
mean(knn1 ==test.data$Diabetes)
table(knn1, test.data$Diabetes)
#In the confusion matrix above,
#88 and 34 are the numbers of true positive and true negative cases respectively
#10 cases are false negative while 22 cases are false positive
#Accuracy of model = (88+34)/(88+34+10+22) = 79.2% is how often the classification is correct

#optimisation algorithm for finding best k value
ac<-rep(1, 25)
for(i in 1:25){
  set.seed(100)
  knn.i<-knn(train.data[,1:8], test.data[,1:8], cl=train.data$Diabetes, k=i)
  ac[i]<-mean(knn.i ==test.data$Diabetes)
  cat("k=", i, " accuracy=", ac[i], "\n")}
plot(ac, type="b", xlab="K",ylab="Accuracy")
#k=19 results in highest accuracy, knn correctly classifies 72.7% of the points in the test data set.

#Logistic Regression
#split data
set.seed(100)
training.index <- sample(1: nrow(PID1), size=nrow(PID1)*0.8)
training.data <-PID1[training.index, ]
testing.data <- PID1[-training.index, ]
#logistic regression
mlogit <- glm(Diabetes ~., data = training.data, family = "binomial")
summary(mlogit)
Pred.p <-predict(mlogit, newdata =testing.data, type = "response")
y_pred_num <-ifelse(Pred.p > 0.5, 1, 0)
y_pred <-factor(y_pred_num, levels=c(0, 1))
#Accuracy of the classification
mean(y_pred ==testing.data$Diabetes )
#Confusion matrix with row:y_pred & column:diabetes
table(y_pred,testing.data$Diabetes)
#Accuracy of model = (91+33)/(91+33+7+23) = 80.5% is how often the classification is correct
#In the confusion matrix above,
#91 and 33 are the numbers of true positive and true negative cases respectively
#7 cases are false negative while 23 cases are false positive

#K-Means Clustering
str(PIDnew)
arr = scale(PIDnew[1:8])
k2 = kmeans(arr, centers = 2, nstart = 10)
str(k2)
```

```
k2
wcss = function(k) {kmeans(arr,k,nstart = 10)$tot.withinss}
k.values = 1:15
set.seed(100)
wcss_k = sapply(k.values,wcss)
plot(k.values, wcss_k, type="b", pch = 19, frame = FALSE, xlab="Number of clusters K",ylab="Total
within-clusters sum of squares")
table(PID1$Diabetes, k2$cluster)
```