Comparison Of Different Classification For Methods Diabetes Detection

Instructor : Dr. Anil K. Ghosh

Stat-Math Unit, Indian Statistical Institute Kolkata

Group V

Sayantan Roy (MB2128)

Arkaprava Sanki (MB2122)

Saswata Naha (MB2119)

Spandan Ghoshal (MD2124)

Contents

About the Dataset

- This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.

About the Dataset

- This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.
- The objective of the dataset is to diagnostically predict whether a patient has diabetes, based on certain diagnostic measurements included in the dataset.

About the Dataset

- This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.
- The objective of the dataset is to diagnostically predict whether a patient has diabetes, based on certain diagnostic measurements included in the dataset.
- Several constraints were placed on the selection of these instances from a larger database.

About the Dataset

- This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.
- The objective of the dataset is to diagnostically predict whether a patient has diabetes, based on certain diagnostic measurements included in the dataset.
- Several constraints were placed on the selection of these instances from a larger database.
- In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Introduction

- The dataset has the following variables :-

Introduction

- The dataset has the following variables :-
  - 1) Pregnancies 2) Glucose 3) BloodPressure

Introduction

- The dataset has the following variables :-
  - 1) Pregnancies 2) Glucose 3) BloodPressure
  - 4) SkinThickness 5) Insulin 6) BMI

Introduction

- The dataset has the following variables :-
  - 1) Pregnancies 2) Glucose 3) BloodPressure
  - 4) SkinThickness 5) Insulin 6) BMI
  - 7) DiabetesPedigreeFunction 8) Age 9) Outcome

Introduction

- The dataset has the following variables :-
  - 1) Pregnancies 2) Glucose 3) BloodPressure
  - 4) SkinThickness 5) Insulin 6) BMI
  - 7) DiabetesPedigreeFunction 8) Age 9) Outcome
- Out of these 9 variables **Outcome** is our response which denotes whether a patient has diabetes or not (1 : yes , 0 : No).

Loading Data in R

- We load the dataset in R and see a few entries through **head()** function :-

```
  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
1           6     148            72            35       0 33.6
2           1      85            66            29       0 26.6
3           8     183            64             0       0 23.3
4           1      89            66            23      94 28.1
5           0     137            40            35     168 43.1
6           5     116            74             0       0 25.6
  DiabetesPedigreeFunction Age Outcome
1                    0.627  50       1
2                    0.351  31       0
3                    0.672  32       1
4                    0.167  21       0
5                    2.288  33       1
6                    0.201  30       0
```

Data Summary

- To get idea about the values of different covariates and response, we calculate the summary :-

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

Observations

- From the summary values we can see that there are 0 entries in the variables "Glucose", "BloodPressure","SkinThickness" ,"Insulin" ,"BMI" which is not biologically possible.

Observations

- From the summary values we can see that there are 0 entries in the variables “Glucose”, “BloodPressure”,“SkinThickness” ,“Insulin” ,“BMI” which is not biologically possible.
- Clearly this indicates that NA values in those places are converted to 0.

Observations

- From the summary values we can see that there are 0 entries in the variables “Glucose”, “BloodPressure”,“SkinThickness” ,“Insulin” ,“BMI” which is not biologically possible.
- Clearly this indicates that NA values in those places are converted to 0.
- So we need to either delete those entries or do some sort of imputation method.

Observations

- From the summary values we can see that there are 0 entries in the variables "Glucose", "BloodPressure","SkinThickness" ,"Insulin" ,"BMI" which is not biologically possible.
- Clearly this indicates that NA values in those places are converted to 0.
- So we need to either delete those entries or do some sort of imputation method.
- But as we see the proportion of such missing entries for each column is :-

```
             Pregnancies                  Glucose            BloodPressure
                   0.000                    0.007                    0.046
           SkinThickness                  Insulin                      BMI
                   0.296                    0.487                    0.014
DiabetesPedigreeFunction                      Age                  Outcome
                   0.000                    0.000                    0.000
[1] 652
```

Observations

- From the summary values we can see that there are 0 entries in the variables "Glucose", "BloodPressure","SkinThickness" ,"Insulin" ,"BMI" which is not biologically possible.
- Clearly this indicates that NA values in those places are converted to 0.
- So we need to either delete those entries or do some sort of imputation method.
- But as we see the proportion of such missing entries for each column is :-

```
             Pregnancies                  Glucose            BloodPressure
                   0.000                    0.007                    0.046
           SkinThickness                  Insulin                      BMI
                   0.296                    0.487                    0.014
DiabetesPedigreeFunction                      Age                  Outcome
                   0.000                    0.000                    0.000
[1] 652
```

- Among all the predictors, SkinThickness and Insulin has the highest missing ratio so we decide to delete these two variables from the dataset and for the remaining missing observations, we use **na.omit()** function to delete the corresponding observations.

Data Summary

- After deleting the predictors and NA observations, we again calculate the summary :-

```
  Pregnancies        Glucose      BloodPressure        BMI
 Min.   : 0.000   Min.   : 44.00   Min.   : 24.0   Min.   :18.20
 1st Qu.: 1.000   1st Qu.: 99.75   1st Qu.: 64.0   1st Qu.:27.50
 Median : 3.000   Median :117.00   Median : 72.0   Median :32.40
 Mean   : 3.866   Mean   :121.88   Mean   : 72.4   Mean   :32.47
 3rd Qu.: 6.000   3rd Qu.:142.00   3rd Qu.: 80.0   3rd Qu.:36.60
 Max.   :17.000   Max.   :199.00   Max.   :122.0   Max.   :67.10
 DiabetesPedigreeFunction      Age        Outcome
 Min.   :0.0780           Min.   :21.00   0:475
 1st Qu.:0.2450           1st Qu.:24.00   1:249
 Median :0.3790           Median :29.00
 Mean   :0.4748           Mean   :33.35
 3rd Qu.:0.6275           3rd Qu.:41.00
 Max.   :2.4200           Max.   :81.00
```

Data Summary

- After deleting the predictors and NA observations, we again calculate the summary :-

```
  Pregnancies        Glucose      BloodPressure        BMI
 Min.   : 0.000   Min.   : 44.00   Min.   : 24.0   Min.   :18.20
 1st Qu.: 1.000   1st Qu.: 99.75   1st Qu.: 64.0   1st Qu.:27.50
 Median : 3.000   Median :117.00   Median : 72.0   Median :32.40
 Mean   : 3.866   Mean   :121.88   Mean   : 72.4   Mean   :32.47
 3rd Qu.: 6.000   3rd Qu.:142.00   3rd Qu.: 80.0   3rd Qu.:36.60
 Max.   :17.000   Max.   :199.00   Max.   :122.0   Max.   :67.10
 DiabetesPedigreeFunction      Age          Outcome
 Min.   :0.0780           Min.   :21.00   0:475
 1st Qu.:0.2450           1st Qu.:24.00   1:249
 Median :0.3790           Median :29.00
 Mean   :0.4748           Mean   :33.35
 3rd Qu.:0.6275           3rd Qu.:41.00
 Max.   :2.4200           Max.   :81.00
```

- Now the values seem to be ok.

Visualize the data

- We visualize the dataset to get idea about how different covariates classify the responses.

Visualize the data

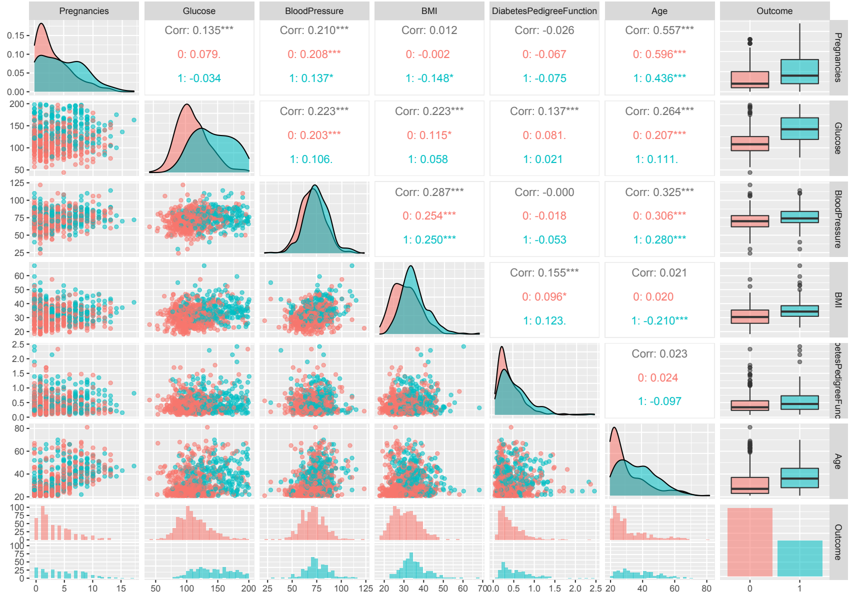
- We visualize the dataset to get idea about how different covariates classify the responses.
- We first see the parallel coordinate plot considering all the covariates which is colour coded by the Outcome variable.

Visualize the data

- We visualize the dataset to get idea about how different covariates classify the responses.
- We first see the parallel coordinate plot considering all the covariates which is colour coded by the Outcome variable.
- Next we make a pairwise scatterplot to get idea about the effects of individual predictors.

Parallel Coordinate Plot

Pairwise Plot



Conclusion

- From the two plots we can see that the two classes are not perfectly linearly seperable.

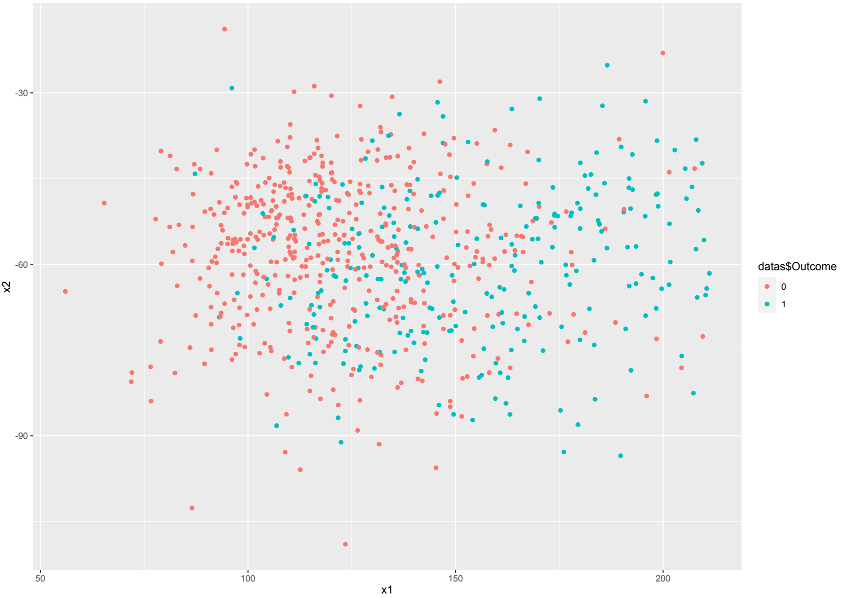
Conclusion

- From the two plots we can see that the two classes are not perfectly linearly seperable.
- But some of the covariates (Glucose,BMI etc) have good seperability which is also evident from the Parallel Coordinate Plots.

Plotting the Principal Components

- We calculate the principal components and plot upto the first 3 components which are colour coded wrt Outcome variable to see whether there is any seperability in it or not.

First Two Components



First Three Components

Splitting Data

- For fitting several classifiers we split the cleaned data set into two groups where the ratio of both the classes (0 & 1) are maintained. (The data is split in 4:1 ratio.)

Splitting Data

- For fitting several classifiers we split the cleaned data set into two groups where the ratio of both the classes (0 & 1) are maintained. (The data is split in 4:1 ratio.)
- Using this set of data we fit all the classifiers and also find optimal choice of the tuning parameters.

LDA

- We first fit the LDA classifier and obtain the following two confusion matrices for them with corresponding misclassification rates :-

```
> ### LDA ###
> ## Training error
> p = predict(model, Train.Data[,-1])$class
> (T = table(Train.Y,p))
p
Train.Y 0 1
0 338 42
1 87 112
> 1-sum(diag(T))/sum(T)
[1] 0.2227979
> # Test Error
> p = predict(model, Test.Data[,-1])$class
> (T = table(Test.Y,p))
p
Test.Y 0 1
0 82 13
1 20 30
> 1-sum(diag(T))/sum(T)
[1] 0.2275862
```

QDA

- Next for QDA we obtain the following :-

QDA

- Next for QDA we obtain the following :-
-
```
> #### QDA #####
> model.qda = qda(Y~., data = Train.Data)
> ## Training error
> p = predict(model.qda, Train.Data[,-1])$class
> (T = table(Train.Y,p))
  0 1
0 329 51
1 81 118
> 1-sum(diag(T))/sum(T)
[1] 0.2279793
> # Test Error
> p = predict(model.qda, Test.Data[,-1])$class
> (T = table(Test.Y,p))
  0 1
0 78 17
1 18 32
> 1-sum(diag(T))/sum(T)
[1] 0.2413793
```

KDA

- We try to fit a classifier based on Kernel Density estimates. First we estimate suitable choice of bandwidths for each predictor i.e. the bandwidth matrix using **Hkda()** function.

KDA

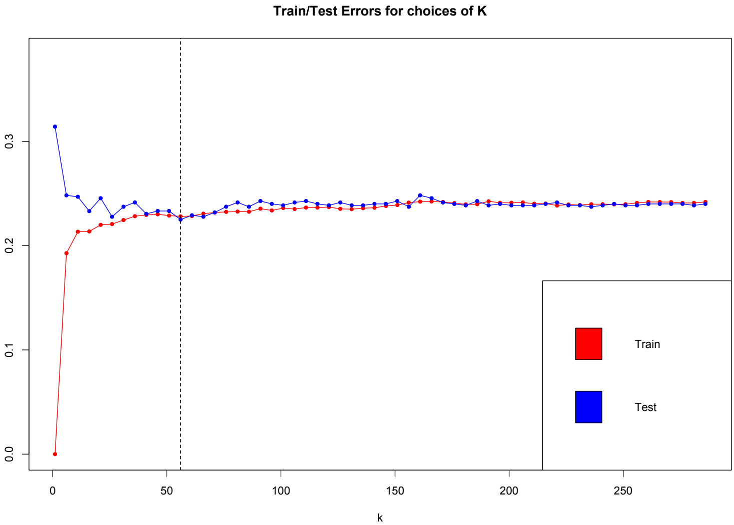
- We try to fit a classifier based on Kernel Density estimates. First we estimate suitable choice of bandwidths for each predictor i.e. the bandwidth matrix using **Hkda()** function.
- Then using this choice of bandwidth matrix we predict the classes for test observations based on kernel density estimates for each group separately. In this case we obtain the test error rates as :-

KDA

- We try to fit a classifier based on Kernel Density estimates. First we estimate suitable choice of bandwidths for each predictor i.e. the bandwidth matrix using **Hkda()** function.
- Then using this choice of bandwidth matrix we predict the classes for test observations based on kernel density estimates for each group separately. In this case we obtain the test error rates as :-
- ```
  > table(pred.train,Train.Y)
  Train.Y
  pred.train 0 1
  0 380 19
  1 0 180
  > (miss = sum(pred.train != Train.Y)/length(Train.Y))
  [1] 0.0328152
  pred.test = c()
  for(i in 1:length(test.lab))
  pred.test[i] = as.numeric(kda(test.data[i,]))
  > sum(pred.test != Test.Y)/length(Test.Y)
  [1] 0.2695937
  ```
  ```

KNN

- Next for K-NN classifier we choose the optimal value of k using cross 10 fold validation. The average misclassification rates in test data for different choices of k are given in the plot below :-



KNN

- As we can both the average misclassification rates stabilize as k increases so based on these obtained value, we got $k = 56$ as the optimal choice. For this model we obtain the following results :-

KNN

- As we can both the average misclassification rates stabilize as k increases so based on these obtained value, we got $k = 56$ as the optimal choice. For this model we obtain the following results :-
-

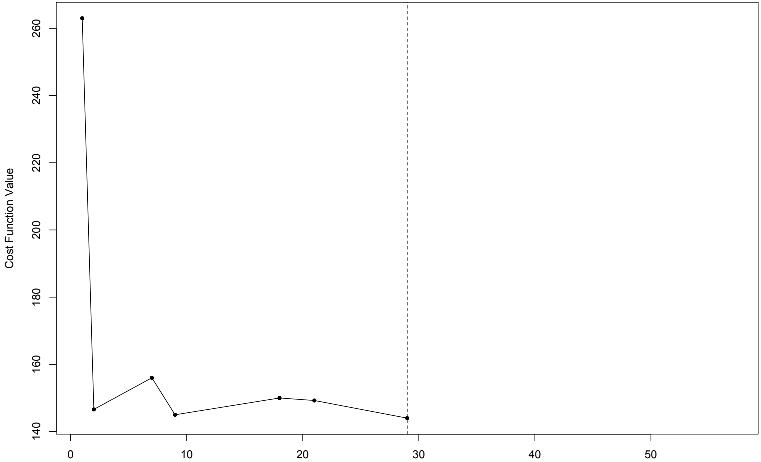
```
> ## Training error
> p = predict(model.knn.opt, Train.Data[,-1], type = "class")
> (T = table(Train.Y,p))
  0 1
0 343 37
1 92 107
> 1-sum(diag(T))/sum(T)
[1] 0.2227979
> # Test Error
> p = predict(model.knn.opt, Test.Data[,-1], type = "class")
> (T = table(Test.Y,p))
  0 1
0 77 18
1 23 27
> 1-sum(diag(T))/sum(T)
[1] 0.2827586
```

CART

- We fit a CART model for which we obtain the following values :-
-
```
> class.tree = tree(Y~.,Train.Data, split = "gini")
> ## Training Error
> tree.pred= predict(class.tree, Train.Data[,-1], type ="class")
> (T = table(tree.pred ,Train.Y))
  0 1
0 347 56
1 33 143
> 1-sum(diag(T))/sum(T)
[1] 0.1537133
> ## Test Error
> tree.pred= predict (class.tree, Test.Data[,-1],type ="class")
> (T = table(tree.pred ,Test.Y))
  0 1
0 78 22
1 17 28
> 1-sum(diag(T))/sum(T)
[1] 0.2689655
```

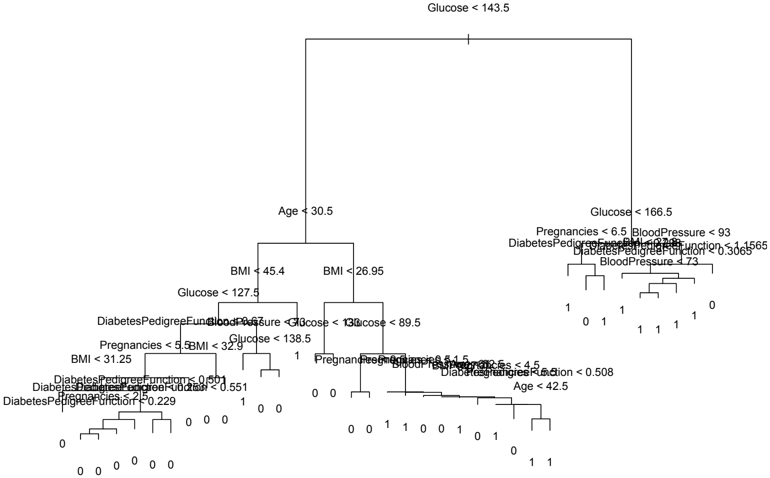
Prunning Tree

- We prune the tree to obtain optimal tree size using cost complexity criterion. We use the misclassification rate with tree size as the criterion and we obtain the following plot which gives the minimum value for a tree of size 29.



Prunning Tree

- After prunning we obtain the following tree :-



Prunning Tree

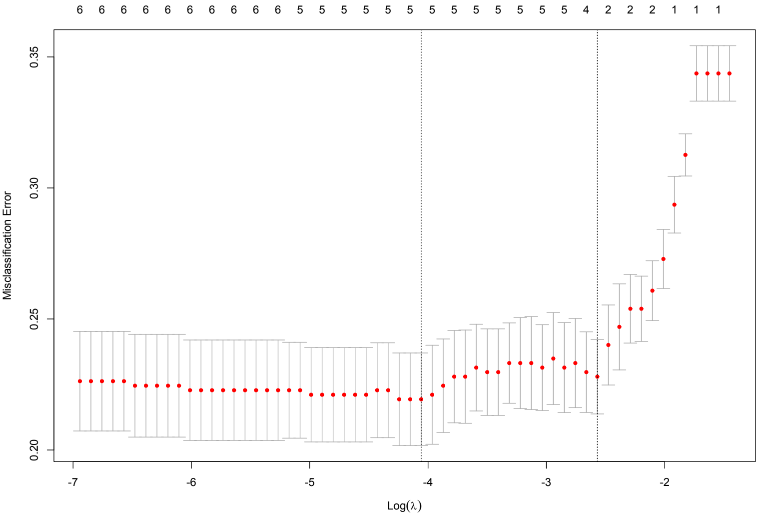
- ```
  > #prunned tree
  > pred=predict(prune.class ,newdata = Test.Data[,-1], type =
  "class")
  > (T = table(pred ,Test.Y))
  Test.Y
  pred 0 1
  0 68 19
  1 27 31
  > 1-sum(diag(T))/sum(T)
  [1] 0.3172414
  ```
- Here we note that the test error is slightly more in case of the prunned tree.
  ```

Logistic Regression

- We fit a logistic regression model and classify observations to class "1" for which estimated class probability is > 0.5. We obtain the following results :-
-
```
> ## Training Error
> (T = table(glm.pred,Train.Data$Y))
  0 1
0 333 85
1 47 114
> 1-sum(diag(T))/sum(T)
[1] 0.2279793
> ## Test Error
> glm.probs = predict(glm.fit, newdata = Test.Data ,type
="response")
> glm.pred=rep (0,length(Test.Data$Y))
> glm.pred[glm.probs>0.5]=1
> (T = table(glm.pred,Test.Data$Y))
  0 1
0 83 19
1 12 31
> 1-sum(diag(T))/sum(T)
[1] 0.2137931
```

GLM Net

- We fit a logistic model with L_1 penalty and find the best choice of tuning parameter λ using cross validation.



GLM Net

- We get $\lambda^* \approx 0.03395$ as the optimal choice and using this we make predictions and obtain the error rates as :-
-
```
> ## Training Error
> glmnet.l1.pred=predict(glmnet.l1,s=bestlam,
newx=as.matrix(Train.Data.norm[,-1]),+ type = "class")
> (T = table(glmnet.l1.pred,Train.Data.norm$Y))
> 1-sum(diag(T))/sum(T)
[1] 0.2297064
> ## Test Error
> glmnet.l1.pred=predict(glmnet.l1,s=bestlam,
newx=as.matrix(Test.Data.norm[,-1]),+ type = "class")
> (T = table(glmnet.l1.pred,Test.Data.norm$Y))
glmnet.l1.pred 0 1
0 85 23
1 10 27
> 1-sum(diag(T))/sum(T)
[1] 0.2275862
```

GLM Net

- With this choice of λ, we get the lasso estimates of the pararmeters in the logistic model as :-

```
> ind = which.min(cv.out$cvm)
> glmnet.l1$beta[,ind]
Pregnancies 1.34748053
Glucose 4.69924381
BloodPressure 0.00000000
BMI 1.63254199
DiabetesPedigreeFunction 1.19954166
Age 0.07383407
```

Random Forest

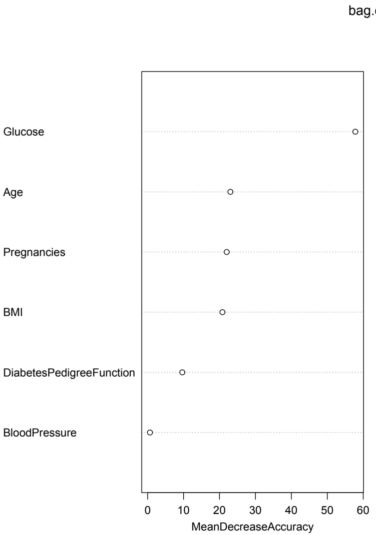
- We fit a Random Forest model considering $\sqrt{p} \approx 3$ predictors in each tree and get the following results :-
-
```
> ## Training Error
> bag.train = predict(bag.class ,newdata = Train.Data[,-1])
> (T = table(bag.train ,Train.Y))
Train.Y
bag.train 0 1
0 380 0
1 0 199
> 1-sum(diag(T))/sum(T)
[1] 0
> ## Test Error
> bag.test = predict(bag.class ,newdata = Test.Data[,-1])
> (T = table(bag.test ,Test.Y))
Test.Y
bag.test 0 1
0 79 18
1 16 32
> 1-sum(diag(T))/sum(T)
[1] 0.2344828
```

Random Forest

- Using the Random Forest model, we can also calculate the variable importance for each of the predictors and we obtain those values as :-

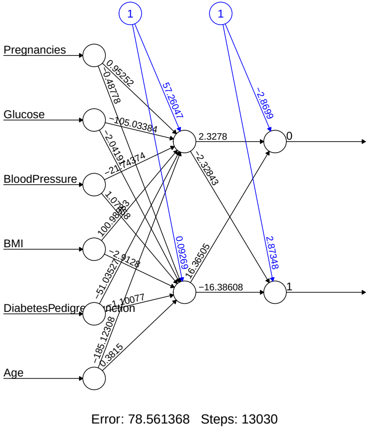
	MeanDecreaseGini
Pregnancies	20.26758313
Glucose	86.27594196
BloodPressure	24.67571276
BMI	47.53741018
Age	41.6108985

Variable Importance Plot



Neural Network

- Next we fit a neural network model with one hidden layer with 2 nodes (because it was not converging in stepmax number of iterations with higher number of nodes or layers) and obtain the fitted model as :-



Neural Network

- The error rates obtained by this model is as follows :-

Neural Network

- The error rates obtained by this model is as follows :-
-
```
## Training Error
> post = predict(nn, Train.Data.norm[,-1], type = "class")
> (T = table(Train.Y,p))
p
Train.Y 0 1
0 322 58
1 64 135
> 1-sum(diag(T))/sum(T)
[1] 0.2107081
> ## Test Error
> post = predict(nn, Test.Data.norm[,-1])
> (T = table(Test.Y,p))
p
Test.Y 0 1
0 78 17
1 23 27
> 1-sum(diag(T))/sum(T)
[1] 0.2758621
```

Support Vector Machine

- Finally we fit the SVM model initally with linear kernel.

Support Vector Machine

- Finally we fit the SVM model initally with linear kernel.
- By cross validation we obtain $C = 100$ as the optimal value for parameter C (in case of linear classifier).

Support Vector Machine

- Finally we fit the SVM model initally with linear kernel.
- By cross validation we obtain $C = 100$ as the optimal value for parameter C (in case of linear classifier).
- Finally, the error rates obtained in this case are :-

Support Vector Machine

- Finally we fit the SVM model initally with linear kernel.
- By cross validation we obtain $C = 100$ as the optimal value for parameter C (in case of linear classifier).
- Finally, the error rates obtained in this case are :-
-
```
> ## Training Error
> p = predict(svmfit, Train.Data[,-1])
> (T = table(p ,Train.Y))
Train.Y
p 0 1
0 336 86
1 44 113
> 1-sum(diag(T))/sum(T)
[1] 0.224525
```

Support Vector Machine

- ```
  > ## Test Error
  > p = predict(svmfit, Test.Data[,-1])
  > (T = table(p ,Test.Y))
  Test.Y
  p 0 1
  0 83 21
  1 12 29
  > 1-sum(diag(T))/sum(T)
  [1] 0.2275862
  ```
  ```

Support Vector Machine

- Lastly, we try fitting SVM with radial kernel. We get the error rates here as :-

Support Vector Machine

- Lastly, we try fitting SVM with radial kernel. We get the error rates here as :-
-
```
> ## Training Error
> p = predict(svmfit, Train.Data[,-1])
> (T = table(p ,Train.Y))
Train.Y
p 0 1
0 378 10
1 2 189
> 1-sum(diag(T))/sum(T)
[1] 0.02072539
> ## Test Error
> p = predict(svmfit, Test.Data[,-1])
> (T = table(p ,Test.Y))
Test.Y
p 0 1
0 69 21
1 26 29
> 1-sum(diag(T))/sum(T)
[1] 0.3241379
```

Support Vector Machine

- Since, the error rates are quite high, we prefer linear kernel here instead of radial kernel.

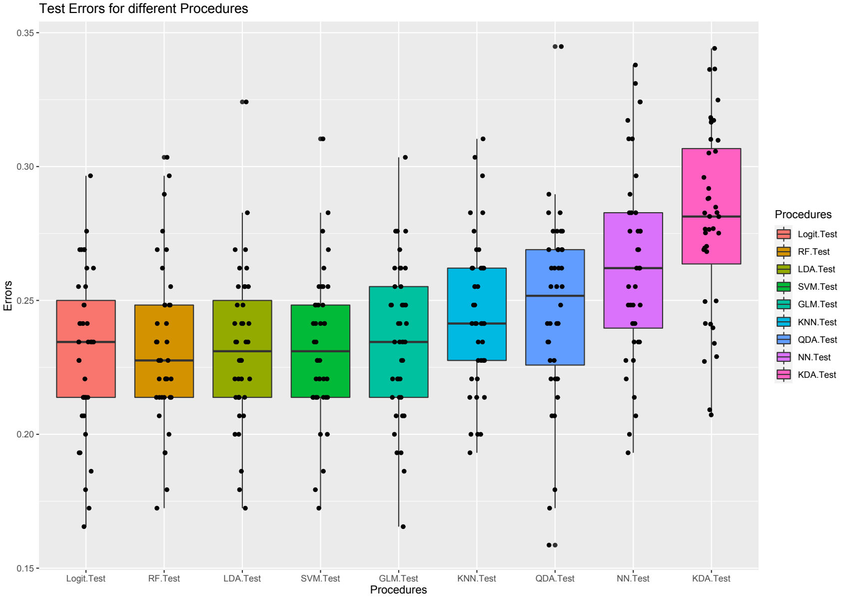
Comparison of Different Classifiers

- We draw different training and test datasets from the original dataset and fit all the classifiers (taking optimal choice of hyperparameters) to get an idea of their comparative performance.

Comparison of Different Classifiers

- We draw different training and test datasets from the original dataset and fit all the classifiers (taking optimal choice of hyperparameters) to get an idea of their comparative performance.
- We repeat this process for 40 times and plot the test errors in a boxplot as follows :-

Comparison of Different Classifiers



Comparison of Different Classifiers

- As we can see, the Random Forest works best in terms of average misclassification rate.

	train.err	test.err		train.err	test.err
LDA	0.2256	**0.2326**	SVM	0.224	**0.2328**
QDA	0.228	0.2466	KDA	0.0346	0.2801
KNN	0.2275	0.2452	Logit	0.224	**0.2303**
NN	0.2048	0.2634	GLM	0.2251	**0.235**
RF	0	**0.2324**			

Comparison of Different Classifiers

- As we can see, the Random Forest works best in terms of average misclassification rate.
- Whereas SVM, LDA, GLMNet, Logistic models also works quite good and the average misclassification rates are also very close to that of Logistic Model.

	train.err	test.err		train.err	test.err
LDA	0.2256	**0.2326**	SVM	0.224	**0.2328**
QDA	0.228	0.2466	KDA	0.0346	0.2801
KNN	0.2275	0.2452	Logit	0.224	**0.2303**
NN	0.2048	0.2634	GLM	0.2251	**0.235**
RF	0	**0.2324**			

Comparison of Different Classifiers

- As we can see, the Random Forest works best in terms of average misclassification rate.
- Whereas SVM, LDA, GLMNet, Logistic models also works quite good and the average misclassification rates are also very close to that of Logistic Model.
- Here, is the table of average training and test misclassification rates :-

	train.err	test.err		train.err	test.err
LDA	0.2256	**0.2326**	SVM	0.224	**0.2328**
QDA	0.228	0.2466	KDA	0.0346	0.2801
KNN	0.2275	0.2452	Logit	0.224	**0.2303**
NN	0.2048	0.2634	GLM	0.2251	**0.235**
RF	0	**0.2324**			

Conclusion

- From all the analysis done above we can finally conclude that most of the linear classifiers perform better than non-linear ones in terms of prediction.

Conclusion

- From all the analysis done above we can finally conclude that most of the linear classifiers perform better than non-linear ones in terms of prediction.
- Secondly, among all the predictors Glucose and BMI have shown to be having highest predicting power.

Conclusion

- From all the analysis done above we can finally conclude that most of the linear classifiers perform better than non-linear ones in terms of prediction.
- Secondly, among all the predictors Glucose and BMI have shown to be having highest predicting power.
- The above findings can be biologically explained as Glucose levels are highly correlated with Diabetes also, recent Genome-Wide Association Studies (GWAS) have shown that genes responsible for Type-2 Diabetes and BMI values are highly linked hence, that finding is also greatly supported by the above conclusion.

Books

- Hastie, T., Tibshirani, R.,, Friedman, J. (2001). The Elements of Statistical Learning. New York, NY, USA: Springer New York Inc.
- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer.

R Packages

- Venables WN, Ripley BD (2002). Modern Applied Statistics with S. Springer, New York. (Link)
- e1071 : Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. (Link)
- GGally: Extension to 'ggplot2' (Link)
- plotly: Create Interactive Web Graphics via 'plotly.js'. (Link)
- caret: Classification and Regression Training (Link)
- tree: Classification and Regression Trees (Link)
- glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models (Link)
- randomForest: Breiman and Cutler's Random Forests for Classification and Regression (Link)
- neuralnet: Training of Neural Networks (Link)
- Genz A, Bretz F, Miwa T, Mi X, Leisch F, Scheipl F, Hothorn T (2021). mvtnorm: Multivariate Normal and t Distributions. (Link)

Acknowledgement

We would like to express our **special thanks of gratitude** to our respected professor **Dr. Anil K. Ghosh** and **Mr.Bilol Banerjee**, for helping us throughout the presentation work and also for giving us this wonderful opportunity as we learned many new & interesting things during the making of this project.