MACHINE LEARNING

NAME: ADITYA RANJAN

REG. NO. 16BCE2310

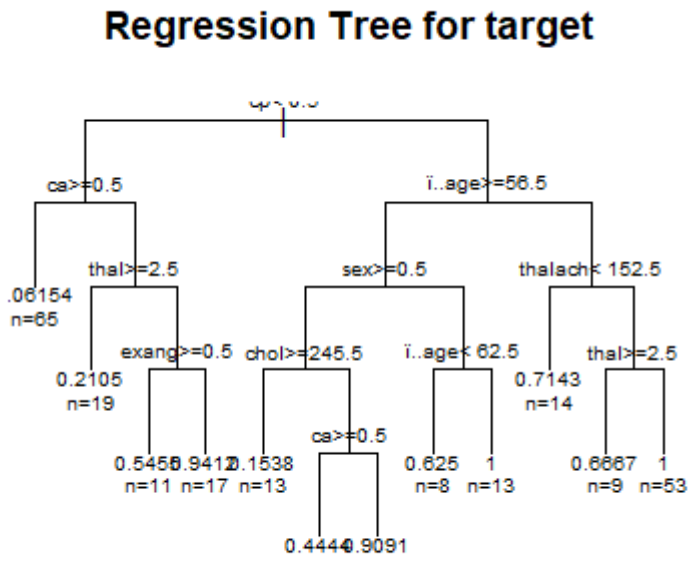
DIGITAL ASSIGNMENT 2

https://www.kaggle.com/ronitf/heart-disease-ucihttps://www.kaggle.com/ronitf/heartdisease-uci

DECISION TREE

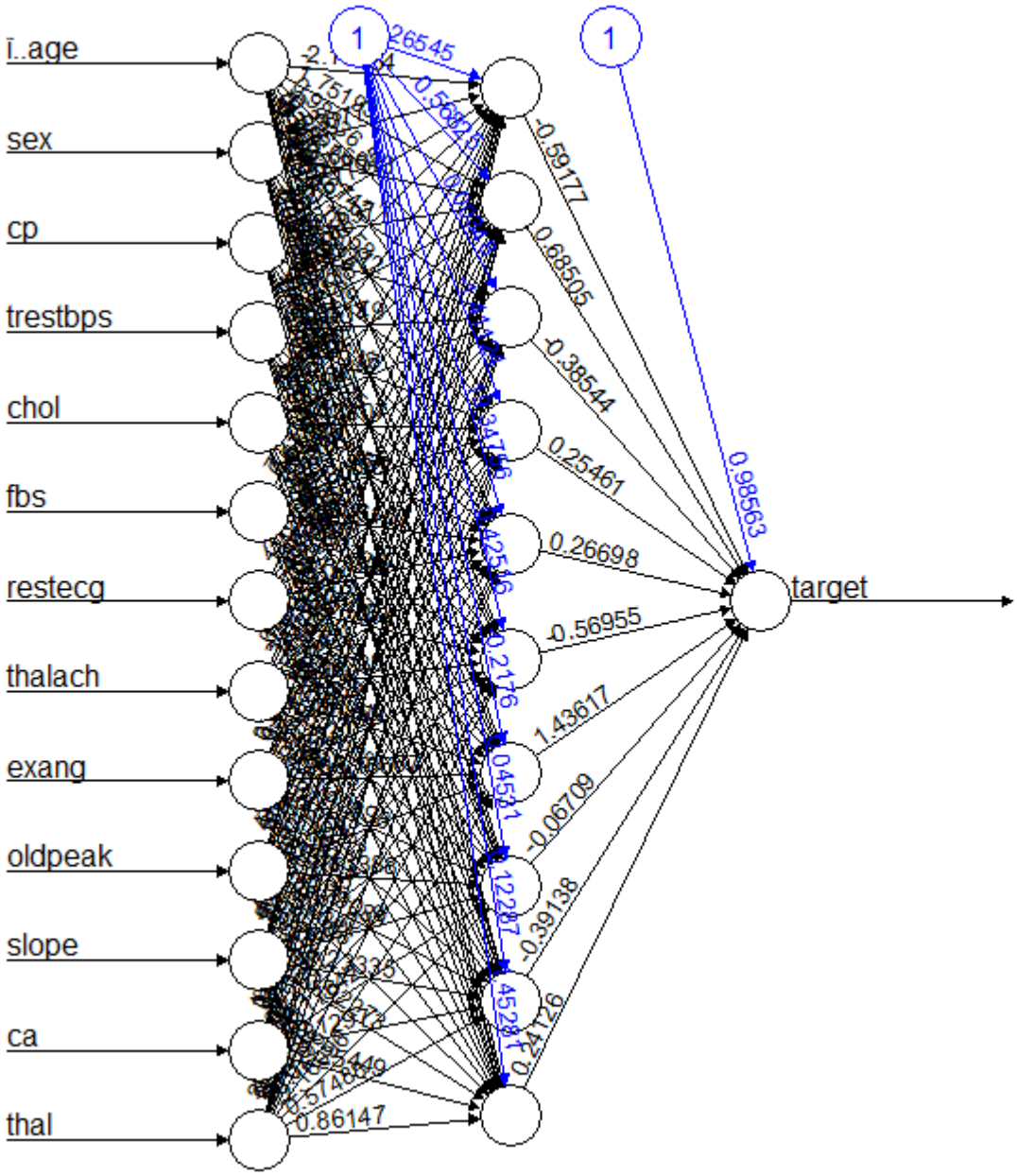
```
> dt=sample(nrow(heart), nrow(heart)*.8)
> heartTrain<- heart[dt,]
> heartTest<-heart[-dt,]
> fit <- rpart(target ~ ï..age+sex+cp+trestbps+chol+fbs+restecg+thalach+ex
ang+oldpeak+slope+ca+thal,
+              method="anova", data=heartTrain )
> plot(fit, uniform=TRUE,main="Regression Tree for target")
> text(fit, use.n=TRUE, cex = .6)
> predict(fit,heartTest)
         1          8         12         22         25
0.90909091 0.66666667 0.71428571 1.00000000 0.66666667
        29         30         31         34         38
1.00000000 0.71428571 1.00000000 0.71428571 0.66666667
        46         51         57         58         60
1.00000000 0.71428571 0.94117647 0.94117647 0.06153846
        67         70         74         81         88
0.71428571 0.94117647 0.54545455 1.00000000 0.66666667
        90         96        103        113        115
0.94117647 0.21052632 1.00000000 1.00000000 1.00000000
       118        125        127        131        134
0.66666667 1.00000000 0.94117647 1.00000000 1.00000000
       136        141        168        174        185
0.94117647 1.00000000 0.06153846 0.44444444 0.21052632
       186        191        195        196        206
0.06153846 0.21052632 0.90909091 0.21052632 0.06153846
       207        209        211        215        218
0.06153846 0.71428571 0.44444444 0.06153846 0.06153846
       221        236        238        242        246
0.06153846 0.21052632 0.06153846 0.54545455 0.21052632
       249        254        258        259        262
0.66666667 0.06153846 0.21052632 0.54545455 0.06153846
       263        264        273        278        299
0.06153846 0.06153846 0.94117647 0.15384615 0.21052632
       300
0.71428571
```

OUTPUT



NEURAL NETWORK

```
> plot(model)
> library("neuralnet")
> dt=sample(nrow(heart), nrow(heart)*.8)
> heartTrain<- heart[dt,]
> heartTest<-heart[-dt,]
> model <- neuralnet(target ~ ï..age+sex+cp+trestbps+chol+fbs+restecg+thal
ach+exang+oldpeak+slope+ca+thal,
+              data=heartTrain ,hidden=10,threshold=0.01)
```



Support Vector Machine

R code

```
> dt=sample(nrow(heart), nrow(heart)*.8)
> heartTrain<- heart[dt,]
> heartTest<-heart[-dt,]
```

```
> model <- svm(target ~ ï..age+sex+cp+trestbps+chol+fbs+restecg+thalach+ex
ang+oldpeak+slope+ca+thal,
+                     data=heartTrain )
> pred=predict(model,heartTest)
> pred
           5           13           15           19
 0.522663438  1.020409443  0.931675890  0.857341467
          22           28           33           36
 1.072740479  0.755270687  1.043667449  0.782808843
          40           50           51           52
 0.923884317  1.020954390  1.118867435  0.596243733
          58           64           69           77
 0.930471581  0.862133514  1.086894338  0.761669457
```

Logistic Regression

```
> dt=sample(nrow(heart), nrow(heart)*.8)
> heartTrain<- heart[dt,]
> heartTest<-heart[-dt,]
> model <- glm(target ~ ï..age+sex+cp+trestbps+chol+fbs+restecg+thalach+ex
ang+oldpeak+slope+ca+thal,
+                     data=heartTrain ,, family=binomial(link="logit"))
> pred=predict(model,heartTest)
> pred
          8           9          14          24          29
 1.96997783  1.95369473  0.13532507  0.06623019  2.41480209
         39          41          42          43          46
 3.84842931  1.72074448  1.38912504 -2.91576368  2.16257886
         50          54          58          59          60
 2.75828164  5.86348356  1.44670744  3.98236575  1.84632870
         61          63          64          71          72
 3.53780845  4.68161696  1.88149477  0.51427780  0.96102183
         81          90          96         109         124
 3.88070358  1.12149865 -2.89235596  3.96641538  4.92985392
        130         141         142         143         151
 0.50006978  3.87018763  0.69125720  5.66279082 -1.43116757
        154         157         162         163         182
 2.68811767  3.54884109  3.26829287  3.51082207 -2.58013883
        183         185         199         203         205
 2.62010243 -3.64763485 -5.28135539 -3.83979269 -5.63335604
        210         211         224         228         232
-0.97917684 -0.06060593 -5.94317505 -2.49650235 -4.78825722
        241         251         252         261         263
-4.07857633 -7.49515819 -4.71988345 -0.96007723 -5.65523995
        266         276         278         279         283
-1.76627383 -0.87349094  0.77309688  1.95026392  0.95380846
        284         285         290         297         299
 1.05271417 -4.02190672 -0.52242199  1.69555149 -0.02519242
        302
-3.15602743
```

```
>
>
> accuracy(heartTest$target,pred,threshold=0.5)
  threshold       AUC omission.rate sensitivity specificity
1       0.5 0.8153595     0.1470588   0.8529412   0.7777778
  prop.correct     Kappa
1    0.8196721 0.6331329
```

NAÏVE BAYES

```
dt=sample(nrow(heart), nrow(heart)*.8)

heartTrain<- heart[dt,]

heartTest<-heart[-dt,]

model <- naiveBayes(target ~
ï..age+sex+cp+trestbps+chol+fbs+restecg+thalach+exang+oldpeak+slope+
ca+thal,

             data=heartTrain )

heartTest

pred=predict(model,heartTest)

pred

accuracy(heartTest$target,pred,threshold=0.5)
```

KNN

```
> dt=sample(nrow(heart), nrow(heart)*.8)
> heartTrain<- heart[dt,]
> heartTest<-heart[-dt,]
> testlabels=heartTest$target
> trainlabels=heartTrain$target
> pred <- knn(train=heartTrain,test=heartTest,cl=trainlabels,k=10)
> pred
 [1] 0 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1
[30] 0 1 1 1 1 1 1 0 1 0 1 1 1 0 1 0 0 0 1 0 1 0 0 1 0 0 1 1 1
[59] 1 0 0
Levels: 0 1
> accuracy(heartTest$target,pred,threshold=0.5)
  threshold       AUC omission.rate sensitivity specificity
1       0.5 0.5876344     0.2580645   0.7419355   0.4333333
  prop.correct    Kappa
1    0.5901639 0.176121
```

Bagging

```
> dt=sample(nrow(heart), nrow(heart)*.8)
> heartTrain<- heart[dt,]
> heartTest<-heart[-dt,]
> model <- bagging(target ~ ï..age+sex+cp+trestbps+chol+fbs+restecg+thalac
h+exang+oldpeak+slope+ca+thal,
+                  data=heartTrain ,mfinal=5,
+                  control=rpart.control(maxdepth=5, minsplit=15))
> pred=predict(model,heartTest)
> pred
 [1] 0.96217087 0.88132961 0.88846849 0.70100231 0.66247290
0.86246171 0.43213053 0.97324763

 [9] 0.72602135 0.67504257 0.79819194 0.91766163 0.97324763
0.90433516 0.69932280 0.97136964

[17] 0.16688719 0.76030660 0.91436821 0.84672889 0.90077618
0.85465539 0.94470925 0.83760173

[25] 0.50820035 0.94251747 0.91766163 0.96880319 0.96880319
0.72415907 0.73019400 0.64760251

[33] 0.96032739 0.82861027 0.11637102 0.62121631 0.05832085
0.15589307 0.01553802 0.71694160

[41] 0.14825778 0.03191164 0.52911062 0.22358845 0.11294872
0.12573021 0.73227253 0.03760767

[49] 0.03867910 0.79694690 0.06762358 0.64472275 0.85051977
0.49347113 0.54752238 0.16825778

[57] 0.73557266 0.40756672 0.69672520 0.11324437 0.01820468

> accuracy(heartTest$target,pred,threshold=0.5)
  threshold       AUC omission.rate sensitivity specificity prop.correct
Kappa
1       0.5 0.7854031    0.05882353   0.9411765   0.6296296
0.8032787 0.588764
```

RANDOM FOREST

```
> library(randomForest)
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.
Warning message:
package ‘randomForest’ was built under R version 3.5.3
> dt=sample(nrow(heart), nrow(heart)*.8)
> heartTrain<- heart[dt,]
> heartTest<-heart[-dt,]
> model <- randomForest(target ~ ï..age+sex+cp+trestbps+chol+fbs+restecg+t
halach+exang+oldpeak+slope+ca+thal,
+                  data=heartTrain )
Warning message:
```

```
In randomForest.default(m, y, ...) :
  The response has five or fewer unique values.  Are you sure you want to
do regression?
> pred=predict(model,heartTest)
> pred
         2          3          6          7         14         15
0.61480000 0.96197500 0.61836667 0.78856667 0.59586667 0.82644167
        21         34         35         38         41         49
0.35503485 0.59496667 0.39683333 0.64496667 0.65014167 0.76744167
        64         71         73         78         82        102
0.66713333 0.45940000 0.95333333 0.83983333 0.95430000 0.19700000
       106        107        112        115        124        127
0.76020000 0.38176667 0.57766667 0.90560000 0.90547500 0.74290000
       128        129        137        138        139        140
0.91933333 0.97530000 0.72520000 0.63760000 0.26980000 0.11993333
       142        146        156        160        178        179
0.75238485 0.53746667 0.65793333 0.69616667 0.85443333 0.09753333
       182        184        185        194        197        206
0.16300000 0.30766667 0.15713333 0.00120000 0.56610000 0.26836667
       210        223        225        228        230        232
0.26223333 0.55516667 0.03280000 0.08246667 0.27046667 0.08093333
       233        248        251        258        262        264
0.02730000 0.34653333 0.03613333 0.15263333 0.51363333 0.31310000
       267        271        286        289        295        296
0.24110000 0.36571304 0.06696667 0.05496667 0.34443333 0.07096667
       298
0.09753333
> accuracy(heartTest$target,pred,threshold=0.5)
  threshold       AUC omission.rate sensitivity specificity
1       0.5 0.8229847     0.2058824   0.7941176   0.8518519
  prop.correct     Kappa
1    0.8196721 0.6386645
```

Booosting

```
> library("gbm")
Loaded gbm 2.1.5
Warning message:
package ‘gbm’ was built under R version 3.5.3
> dt=sample(nrow(heart), nrow(heart)*.8)
> heartTrain<- heart[dt,]
> heartTest<-heart[-dt,]
> model <- gbm(target ~ ï..age+sex+cp+trestbps+chol+fbs+restecg+thalach+ex
ang+oldpeak+slope+ca+thal,
+                       data=heartTrain,,distribution = "gaussian",n.trees
= 10000,
+                       shrinkage = 0.01, interaction.depth = 4 )
> model
gbm(formula = target ~ ï..age + sex + cp + trestbps + chol +
    fbs + restecg + thalach + exang + oldpeak + slope + ca +
    thal, distribution = "gaussian", data = heartTrain,
    n.trees = 10000, interaction.depth = 4, shrinkage = 0.01)
A gradient boosted model with gaussian loss function.
10000 iterations were performed.
```

```
There were 13 predictors of which 13 had non-zero influence.
> pred=predict(model,heartTest)
Error in paste("Using", n.trees, "trees...\n") :
  argument "n.trees" is missing, with no default
> pred
         2          3          6          7         14         15
0.61480000 0.96197500 0.61836667 0.78856667 0.59586667 0.82644167
        21         34         35         38         41         49
0.35503485 0.59496667 0.39683333 0.64496667 0.65014167 0.76744167
        64         71         73         78         82        102
0.66713333 0.45940000 0.95333333 0.83983333 0.95430000 0.19700000
       106        107        112        115        124        127
0.76020000 0.38176667 0.57766667 0.90560000 0.90547500 0.74290000
       128        129        137        138        139        140
0.91933333 0.97530000 0.72520000 0.63760000 0.26980000 0.11993333
       142        146        156        160        178        179
0.75238485 0.53746667 0.65793333 0.69616667 0.85443333 0.09753333
       182        184        185        194        197        206
0.16300000 0.30766667 0.15713333 0.00120000 0.56610000 0.26836667
       210        223        225        228        230        232
0.26223333 0.55516667 0.03280000 0.08246667 0.27046667 0.08093333
       233        248        251        258        262        264
0.02730000 0.34653333 0.03613333 0.15263333 0.51363333 0.31310000
       267        271        286        289        295        296
0.24110000 0.36571304 0.06696667 0.05496667 0.34443333 0.07096667
       298
0.09753333
> accuracy(heartTest$target,pred,threshold=0.5)
  threshold       AUC omission.rate sensitivity specificity
1       0.5 0.8229847     0.2058824   0.7941176   0.8518519
  prop.correct     Kappa
1    0.8196721 0.6386645
```

ADABOOST

```
> dt=sample(nrow(heart), nrow(heart)*.8)
> heartTrain<- heart[dt,]
> heartTest<-heart[-dt,]
> model <- boosting(target ~ ï..age+sex+cp+trestbps+chol+fbs+restecg+thala
ch+exang+oldpeak+slope+ca+thal,
+              data=heartTrain,boos=TRUE,
+              mfinal=3)
Error in if (nrow(object$splits) > 0) { : argument is of length zero
> model
```

PYTHON

Code::

Python-

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix,
accuracy_score,r2_score,mean_squared_error

dataset = pd.read_csv('../input/heart.csv')
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,-1].values
#Encoding Categorical Data
from sklearn.preprocessing import OneHotEncoder
#cp
oneHotEncoder = OneHotEncoder(categorical_features=[2], n_values='auto')
oneHotEncoder.fit(X)
X = oneHotEncoder.transform(X).toarray()
X = X[:, 1:]
#restecg
oneHotEncoder = OneHotEncoder(categorical_features=[8], n_values='auto')
oneHotEncoder.fit(X)
X = oneHotEncoder.transform(X).toarray()
X = X[:, 1:]
#slope
oneHotEncoder = OneHotEncoder(categorical_features=[13], n_values='auto')
oneHotEncoder.fit(X)
X = oneHotEncoder.transform(X).toarray()
X = X[:, 1:]
#ca
oneHotEncoder = OneHotEncoder(categorical_features=[15], n_values='auto')
oneHotEncoder.fit(X)
X = oneHotEncoder.transform(X).toarray()
X = X[:, 1:]
#thal
oneHotEncoder = OneHotEncoder(categorical_features=[19], n_values='auto')
oneHotEncoder.fit(X)
X = oneHotEncoder.transform(X).toarray()
X = X[:, 1:]

from sklearn.preprocessing import StandardScaler
scalerX = StandardScaler()
X = scalerX.fit_transform(X)
from sklearn.model_selection import train_test_split
XTrain, XTest, yTrain, yTest = train_test_split(X, y, test_size=0.3, random_state=0)

#Logistic Regression
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
```

```
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
r2 = r2_score(yTest,yPred)
rmse = mean_squared_error(yTest,yPred)
print("Logistic Regression :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

```
Logistic Regression :
Accuracy =  0.8461538461538461
R2 =  0.38394584139265
RMSE =  0.15384615384615385
[[34 10]
 [ 4 43]]
```

```
#K Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print("K Nearest Neighbors :")
print("Accuracy = ", accuracy)
print(cm)
```

```
K Nearest Neighbors :
Accuracy =  0.8131868131868132
[[32 12]
 [ 5 42]]
```

```
#Support Vector Machine
from sklearn.svm import SVC
classifier = SVC(kernel='linear',random_state=0)
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print("Support Vector Machine :")
print("Accuracy = ", accuracy)
print(cm)
```

```
Support Vector Machine :
Accuracy =  0.8571428571428571
[[35  9]
 [ 4 43]]
```

```
#Gaussian Naive Bayes
```

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print("Gaussian Naive Bayes :")
print("Accuracy = ", accuracy)
print(cm)
```

```
Gaussian Naive Bayes :
Accuracy =  0.8131868131868132
[[33 11]
 [ 6 41]]
```

```
#Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier as DT
classifier = DT(criterion='entropy', random_state=0)
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print("Decision Tree Classifier :")
print("Accuracy = ", accuracy)
print(cm)
```

```
Decision Tree Classifier :
Accuracy =  0.6813186813186813
[[31 13]
 [16 31]]
```

```
#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier as RF
classifier = RF(n_estimators=10, criterion='entropy', random_state=0)
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print("Random Forest Classifier :")
print("Accuracy = ", accuracy)
print(cm)
```

```
Random Forest Classifier :
Accuracy =  0.8241758241758241
[[34 10]
 [ 6 41]]
```

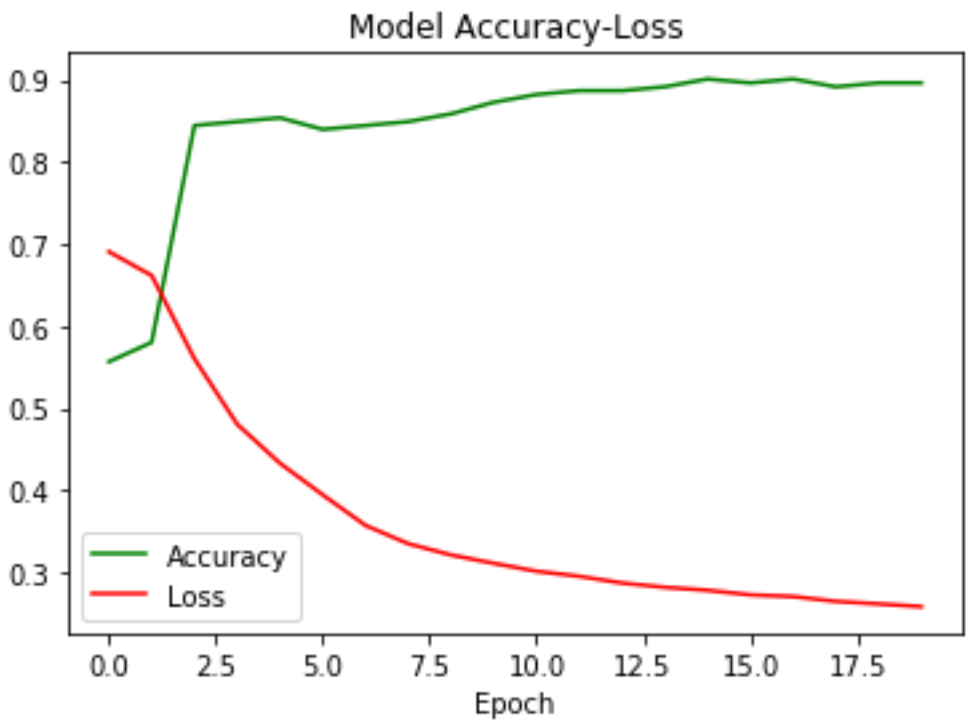
```
#Artificial Neural Network
from keras.models import Sequential
from keras.layers import Dense
```

```
#Initialising ANN
classifier = Sequential()

#Adding the first hidden layer or the input layer
classifier.add(Dense(activation='relu',
                     kernel_initializer='uniform',
                     input_dim=22,
                     units=12))
#Adding the second hidden layer
classifier.add(Dense(activation='relu',
                     kernel_initializer='uniform',
                     units=12))
#Adding the output layer
classifier.add(Dense(activation='sigmoid',
                     kernel_initializer='uniform',
                     units=1))

#Compiling the ANN
classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
print(classifier.summary())

#Fitting the ANN
history = classifier.fit(XTrain, yTrain, batch_size=5, epochs=20, verbose=1)
from matplotlib import pyplot as plt
plt.plot(history.history['acc'],'green')
plt.plot(history.history['loss'],'red')
plt.title('Model Accuracy-Loss')
plt.xlabel('Epoch')
plt.legend(['Accuracy','Loss'])
plt.show()

#Predicting the Test set Results
yPred = classifier.predict(XTest)
yPred = (yPred>0.5) #Since output is probability
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print("Artificial Neural Network Classifier :")
print("Accuracy = ", accuracy)
print(cm)
```

```
Using TensorFlow backend.

_________________________________________________________________
_____________
Layer (type)                 Output Shape
Param #
=================================================================
============
dense_1 (Dense)              (None, 12)
276
```

```
_________________________________________________________________
dense_2 (Dense)              (None, 12)
156
_________________________________________________________________
dense_3 (Dense)              (None, 1)
13
=================================================================
Total params: 445
Trainable params: 445
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/20
212/212 [==============================] - 1s
3ms/step - loss: 0.6907 - acc: 0.5566
Epoch 2/20
212/212 [==============================] - 0s
388us/step - loss: 0.6616 - acc: 0.5802
Epoch 3/20
212/212 [==============================] - 0s
356us/step - loss: 0.5603 - acc: 0.8443
Epoch 4/20
212/212 [==============================] - 0s
355us/step - loss: 0.4805 - acc: 0.8491
Epoch 5/20
212/212 [==============================] - 0s
358us/step - loss: 0.4330 - acc: 0.8538
Epoch 6/20
212/212 [==============================] - 0s
375us/step - loss: 0.3945 - acc: 0.8396
Epoch 7/20
212/212 [==============================] - 0s
351us/step - loss: 0.3572 - acc: 0.8443
Epoch 8/20
212/212 [==============================] - 0s
373us/step - loss: 0.3349 - acc: 0.8491
Epoch 9/20
```

```
212/212 [==============================] - 0s
379us/step - loss: 0.3212 - acc: 0.8585
Epoch 10/20
212/212 [==============================] - 0s
394us/step - loss: 0.3110 - acc: 0.8726
Epoch 11/20
212/212 [==============================] - 0s
390us/step - loss: 0.3014 - acc: 0.8821
Epoch 12/20
212/212 [==============================] - 0s
372us/step - loss: 0.2952 - acc: 0.8868
Epoch 13/20
212/212 [==============================] - 0s
356us/step - loss: 0.2870 - acc: 0.8868
Epoch 14/20
212/212 [==============================] - 0s
375us/step - loss: 0.2818 - acc: 0.8915
Epoch 15/20
212/212 [==============================] - 0s
372us/step - loss: 0.2782 - acc: 0.9009
Epoch 16/20
212/212 [==============================] - 0s
337us/step - loss: 0.2726 - acc: 0.8962
Epoch 17/20
212/212 [==============================] - 0s
341us/step - loss: 0.2706 - acc: 0.9009
Epoch 18/20
212/212 [==============================] - 0s
375us/step - loss: 0.2647 - acc: 0.8915
Epoch 19/20
212/212 [==============================] - 0s
363us/step - loss: 0.2616 - acc: 0.8962
Epoch 20/20
212/212 [==============================] - 0s
387us/step - loss: 0.2583 - acc: 0.8962
```




```
Artificial Neural Network Classifier :
Accuracy =  0.8571428571428571
[[35  9]
 [ 4 43]]
```

#Gradient Boosting Classifier

A Gradient Boosting classifier is used for making predictions.
This algorithm is the only one that got better results when scaling to the full dataset.
The output is a probability representing the likelihood of the
presence of heart disease.

Expected Accuray: processed.cleveland.data

The data was tested using a 10-fold cross validation technique.

The results are:

Avg Accuracy	Avg Recall	Avg Precision	Avg ROC_AUC
0.795	0.758	0.800	0.914

Expected Accuray: processed.all.data

The data was randomly shuffled and tested using a 10-fold cross validation technique.

The average results for 5 cross validations are:

Avg Accuracy	Avg Recall	Avg Precision	Avg ROC_AUC
0.813	0.854	0.817	0.900

DATASET: YOUTUBE

RCODE AND RESULTS:

PYTHON CODE AND RESULTS:

Dataset https://www.kaggle.com/mdhrumil/top-5000-youtube-channels-data-from-socialblade

Python:

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix, accuracy_score,r2_score,mean_squared_error
from sklearn.preprocessing import normalize
from sklearn.model_selection import cross_val_score
from sklearn import tree
from sklearn import linear_model
from sklearn import svm
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier

import pandas as pd
csv_file = "/content/youtube.csv"
dt = pd.read_csv(csv_file)
```

```
#REPLACING -- WITH NULL
import numpy as np
dt = dt.replace('--', np.nan, regex=True)
dt["Rank"] = dt["Rank"].replace('th', "", regex=True)
dt["Rank"] = dt["Rank"].replace('rd', "", regex=True)
dt["Rank"] = dt["Rank"].replace('nd', "", regex=True)
dt["Rank"] = dt["Rank"].replace('st', "", regex=True)
dt["Rank"] = dt["Rank"].replace(',', "", regex=True)

#REPlACING OBJECT TYPE WITH FLOAT64
dt["Subscribers"] = pd.to_numeric(dt["Subscribers"])
dt["Video Uploads"] = pd.to_numeric(dt["Video Uploads"])
dt["Video views"] = pd.to_numeric(dt["Video views"])
dt["Rank"] = pd.to_numeric(dt["Rank"])

dt["Average views"] = dt["Video views"] / dt["Video Uploads"]

#INSERTING MEDIAN VALUE TO NULL (in dt_full)
from sklearn.impute import SimpleImputer

sample_incomplete_rows = dt[dt.isnull().any(axis=1)].head()

imputer = SimpleImputer(strategy="median")

dt_num = dt.drop(["Channel name","Grade"], axis = 1)

imputer.fit(dt_num)
X = imputer.transform(dt_num)
dt_full = pd.DataFrame(X, columns=dt_num.columns)
dt_full.loc[sample_incomplete_rows.index.values]
dt_full = pd.DataFrame(X, columns=dt_num.columns)
```

```
#CATEGORICAL DATA
grade_cat = dt[["Grade"]]

from sklearn.preprocessing import OneHotEncoder
cat_encoder = OneHotEncoder()
dt_cat_1hot = cat_encoder.fit_transform(grade_cat)
dt_one_hot_array = dt_cat_1hot.toarray()

#TRAIN AND TEST DATA
dt_full_train = dt_full[:4000]
dt_full_test = dt_full[4000:]

subs_train = dt_full_train["Subscribers"].copy()

#PIPELINES
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
        ('std_scaler', StandardScaler()),
    ])

dt_num_tr = num_pipeline.fit_transform(dt_full)

from sklearn.compose import ColumnTransformer

num_attribs = list(dt_full)
cat_attribs = ["Grade"]

full_pipeline = ColumnTransformer([
        ("num", num_pipeline, num_attribs),
```

```
        ("cat", OneHotEncoder(), cat_attribs)
    ])

dt_prepared = full_pipeline.fit_transform(dt[:4000])
```

LogisticRegression:

```
lin_reg = LogisticRegression()
lin_reg.fit(dt_prepared, subs_train)

some_data = dt.iloc[:10]
some_labels = subs_train.iloc[:10]
some_data_prepared = full_pipeline.transform(some_data)

dt_predictions = lin_reg.predict(dt_prepared)
rmse = mean_squared_error(subs_train,dt_predictions)
cm = confusion_matrix(subs_train,dt_predictions)
accuracy = accuracy_score(subs_train,dt_predictions)
r2 = r2_score(subs_train,dt_predictions)

print("Logistic Regression :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

Output:

```
Logistic Regression :
Accuracy =  0.085
R2 =  -0.41283048624407837
RMSE =  23800441219721.88
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]]
```

Artificial Neural Network :

```
lin_reg = MLPClassifier()
lin_reg.set_params(hidden_layer_sizes =(100,100), max_iter = 1000,alpha = 0.01, momentum = 0.7)
lin_reg.fit(dt_prepared, subs_train)

some_data = dt.iloc[:10]
some_labels = subs_train.iloc[:10]
some_data_prepared = full_pipeline.transform(some_data)

dt_predictions = lin_reg.predict(dt_prepared)
rmse = mean_squared_error(subs_train,dt_predictions)
cm = confusion_matrix(subs_train,dt_predictions)
accuracy = accuracy_score(subs_train,dt_predictions)
r2 = r2_score(subs_train,dt_predictions)
```

```
print("Artificial Neural Network :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

OUTPUT

```
Artificial Neural Network :
Accuracy =  0.72875
R2 =  0.993882156273152
RMSE =  103060757415.6885
[[1 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]]
```

Support Vector Machine

```
rom sklearn.svm import SVC
lin_reg = SVC(kernel='linear',random_state=0)
lin_reg.fit(dt_prepared, subs_train)
```

```
some_data = dt.iloc[:10]
some_labels = subs_train.iloc[:10]
some_data_prepared = full_pipeline.transform(some_data)

dt_predictions = lin_reg.predict(dt_prepared)
rmse = mean_squared_error(subs_train,dt_predictions)
cm = confusion_matrix(subs_train,dt_predictions)
accuracy = accuracy_score(subs_train,dt_predictions)
r2 = r2_score(subs_train,dt_predictions)
print("Support Vector Machine :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
lin_reg = GaussianNB()
lin_reg.fit(dt_prepared, subs_train)

some_data = dt.iloc[:10]
some_labels = subs_train.iloc[:10]
some_data_prepared = full_pipeline.transform(some_data)

dt_predictions = lin_reg.predict(dt_prepared)
rmse = mean_squared_error(subs_train,dt_predictions)
cm = confusion_matrix(subs_train,dt_predictions)
accuracy = accuracy_score(subs_train,dt_predictions)
r2 = r2_score(subs_train,dt_predictions)
print("Gaussian Naive Bayes :")
print("Accuracy = ", accuracy)
```

```
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

OUTPUT::

```
Gaussian Naive Bayes :
Accuracy =  1.0
R2 =  1.0
RMSE =  0.0
[[1 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]
```

K nearest neighbour

```
from sklearn.neighbors import KNeighborsClassifier

lin_reg = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
lin_reg.fit(dt_prepared, subs_train)

some_data = dt.iloc[:10]
some_labels = subs_train.iloc[:10]
some_data_prepared = full_pipeline.transform(some_data)

dt_predictions = lin_reg.predict(dt_prepared)
rmse = mean_squared_error(subs_train,dt_predictions)
cm = confusion_matrix(subs_train,dt_predictions)
accuracy = accuracy_score(subs_train,dt_predictions)
```

```
r2 = r2_score(subs_train,dt_predictions)
print("K nearest neighbour :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

```
K nearest neighbour :
Accuracy =  0.1495
R2 =  0.8962040635132252
RMSE =  1748538914790.0137
[[1 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Bagging

```
lin_reg = BaggingClassifier()
lin_reg.set_params(n_estimators = 30,max_samples = 1000)
lin_reg.fit(dt_prepared, subs_train)

some_data = dt.iloc[:10]
some_labels = subs_train.iloc[:10]
some_data_prepared = full_pipeline.transform(some_data)

dt_predictions = lin_reg.predict(dt_prepared)
rmse = mean_squared_error(subs_train,dt_predictions)
cm = confusion_matrix(subs_train,dt_predictions)
```

```
accuracy = accuracy_score(subs_train,dt_predictions)
r2 = r2_score(subs_train,dt_predictions)
print("Bagging :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

```
Bagging :
Accuracy =  0.93575
R2 =  0.9894947501109713
RMSE =  176970687507.60474
[[1 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]]
```

Random Forest :

```
lin_reg = RandomForestClassifier()
lin_reg.set_params(n_estimators = 100, max_depth = 10)
lin_reg.fit(dt_prepared, subs_train)

some_data = dt.iloc[:10]
some_labels = subs_train.iloc[:10]
some_data_prepared = full_pipeline.transform(some_data)
```

```
dt_predictions = lin_reg.predict(dt_prepared)
rmse = mean_squared_error(subs_train,dt_predictions)
cm = confusion_matrix(subs_train,dt_predictions)
accuracy = accuracy_score(subs_train,dt_predictions)
r2 = r2_score(subs_train,dt_predictions)
print("Random Forest :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

```
Random Forest :
Accuracy =  0.713
R2 =  0.9199350282485733
RMSE =  1348768781875.8728
[[1 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]]
```

Gradient Boosting

```
lin_reg = GradientBoostingClassifier()
lin_reg.set_params(n_estimators = 30,learning_rate = 1)
lin_reg.fit(dt_prepared, subs_train)

some_data = dt.iloc[:10]
```

```
some_labels = subs_train.iloc[:10]
some_data_prepared = full_pipeline.transform(some_data)

dt_predictions = lin_reg.predict(dt_prepared)
rmse = mean_squared_error(subs_train,dt_predictions)
cm = confusion_matrix(subs_train,dt_predictions)
accuracy = accuracy_score(subs_train,dt_predictions)
r2 = r2_score(subs_train,dt_predictions)
print("Gradient Boosting :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)

lin_reg = AdaBoostClassifier()
lin_reg.set_params(n_estimators = 10, learning_rate = 1)
lin_reg.fit(dt_prepared, subs_train)

some_data = dt.iloc[:10]
some_labels = subs_train.iloc[:10]
some_data_prepared = full_pipeline.transform(some_data)

dt_predictions = lin_reg.predict(dt_prepared)
rmse = mean_squared_error(subs_train,dt_predictions)
cm = confusion_matrix(subs_train,dt_predictions)
accuracy = accuracy_score(subs_train,dt_predictions)
r2 = r2_score(subs_train,dt_predictions)
print("AdaBoost :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

DATASET: IRIS

RCODE AND RESULTS:

Decision Tree

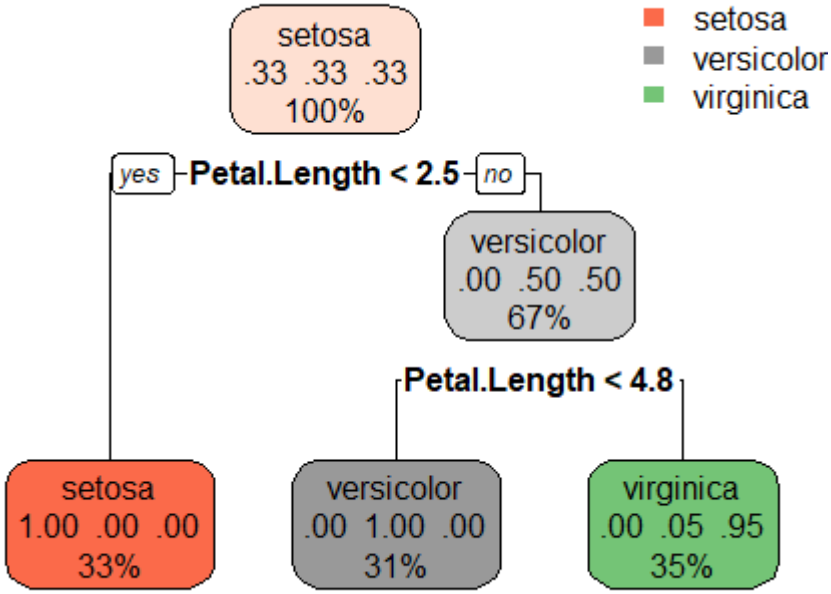
```
library("caTools")
library(rpart)
install.packages("rpart.plot")
library(rpart.plot)
data("iris")

# Split data into Train and test
set.seed(123)
split <- sample.split(iris$Species,SplitRatio = 0.7)

# split using subsetsubset(
train <- subset(iris, split==TRUE)
test <- subset(iris, split==FALSE)

#Build the model

Sptree <- rpart ( Species ~., train, method ="class")

#Plot the model
rpart.plot(Sptree)
```

#Predict using the test data set

pptree <- predict(Sptree, test, type ="class")

#Confusion matrix

table(test[,5],pptree)




```
> #Confusion matrix
> table(test[,5],pptree)
            pptree
             setosa versicolor virginica
  setosa         15          0         0
  versicolor      0         11         4
  virginica       0          1        14
```

LOGISTIC REGRESSION

library(datasets)

ir_data<- iris

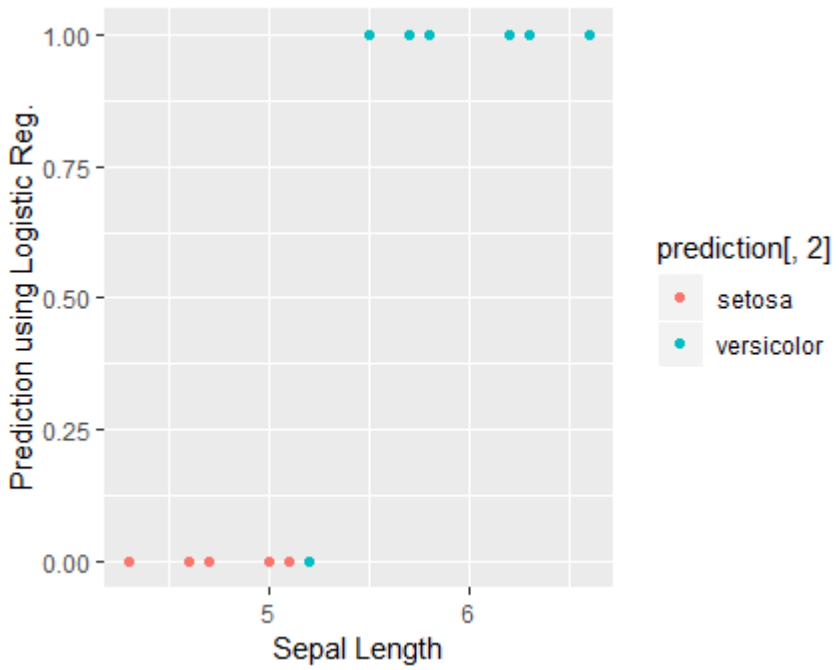
head(ir_data)

str(ir_data)

levels(ir_data$Species)

sum(is.na(ir_data))

```
ir_data<-ir_data[1:100,]

set.seed(100)

samp<-sample(1:100,80)

ir_test<-ir_data[samp,]

ir_ctrl<-ir_data[-samp,]

library(ggplot2); library(GGally)

ggpairs(ir_test)

y<-ir_test$Species; x<-ir_test$Sepal.Length

glfit<-glm(y~x, family = 'binomial')

summary(glfit)

newdata<- data.frame(x=ir_ctrl$Sepal.Length)

predicted_val<-predict(glfit, newdata, type="response")

prediction<-data.frame(ir_ctrl$Sepal.Length, ir_ctrl$Species,predicted_val)

prediction

qplot(prediction[,1], round(prediction[,3]), col=prediction[,2], xlab = 'Sepal Length', ylab = 'Prediction using Logistic Reg.')
```



SVM

```
library("e1071")

head(iris,5)
```

```
attach(iris)

x <- subset(iris, select=-Species)

y <- Species

svm_model <- svm(Species ~ ., data=iris)

summary(svm_model)

svm_model1 <- svm(x,y)

summary(svm_model1)

pred <- predict(svm_model1,x)

system.time(pred <- predict(svm_model1,x))

table(pred,y)

svm_tune <- tune(svm, train.x=x, train.y=y,

            kernel="radial", ranges=list(cost=10^(-1:2), gamma=c(.5,1,2)))


print(svm_tune)

svm_model_after_tune <- svm(Species ~ ., data=iris, kernel="radial", cost=1, gamma=0.5)

summary(svm_model_after_tune)

pred <- predict(svm_model_after_tune,x)

system.time(predict(svm_model_after_tune,x))

table(pred,y)
```

OUTPUT:

Confusion Matrix:

```
> table(pred,y)
            y
pred         setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          2        48
```

```
# load libraries

library(mlbench)

library(caret)

# load data

data("iris")

# rename dataset to keep code below generic
```

```
dataset <- iris

control <- trainControl(method="repeatedcv", number=10, repeats=3)

seed <- 7

metric <- "Accuracy"

preProcess=c("center", "scale")

# Logistic Regression

set.seed(seed)

fit.glm <- train(iris~., data=dataset, method="glm", metric=metric, trControl=control)

# SVM Radial

set.seed(seed)

fit.svmRadial <- train(iris~., data=dataset, method="svmRadial", metric=metric, preProc=c("center",
"scale"), trControl=control, fit=FALSE)

# kNN

set.seed(seed)

fit.knn <- train(iris~., data=dataset, method="knn", metric=metric, preProc=c("center", "scale"),
trControl=control)

# Naive Bayes

set.seed(seed)

fit.nb <- train(iris~., data=dataset, method="nb", metric=metric, trControl=control)


# Bagged CART

set.seed(seed)

fit.treebag <- train(iris~., data=dataset, method="treebag", metric=metric, trControl=control)

# Random Forest

set.seed(seed)

fit.rf <- train(iris~., data=dataset, method="rf", metric=metric, trControl=control)

# Stochastic Gradient Boosting (Generalized Boosted Modeling)

set.seed(seed)

fit.gbm <- train(iris~., data=dataset, method="gbm", metric=metric, trControl=control,
verbose=FALSE)

results <- resamples(list(lda=fit.lda, logistic=fit.glm, glmnet=fit.glmnet,

                          svm=fit.svmRadial, knn=fit.knn, nb=fit.nb, cart=fit.cart, c50=fit.c50,
```

```
                    bagging=fit.treebag, rf=fit.rf, gbm=fit.gbm))
# Table comparison
summary(results)
```

OUTPUT:

Models: logistic, svm, knn, nb, bagging, rf, gbm

Number of resamples: 30

Accuracy

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
logistic	0.6842	0.7639	0.7713	0.7781	0.8019	0.8701	0
svm	0.6711	0.7403	0.7582	0.7651	0.7890	0.8961	0
knn	0.6753	0.7115	0.7386	0.7465	0.7785	0.8961	0
nb	0.6316	0.7305	0.7597	0.7569	0.7869	0.8571	0
bagging	0.6883	0.7246	0.7451	0.7530	0.7792	0.8571	0
rf	0.6711	0.7273	0.7516	0.7617	0.7890	0.8571	0
gbm	0.6974	0.7273	0.7727	0.7708	0.8052	0.8831	0

PYTHON CODE AND RESULTS:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
from sklearn.model_selection import cross_val_score
data = pd.read_csv("Iris.csv")
train =data.sample(frac = 0.7)
test = data.loc[~data.index.isin(train.index)]
train
from sklearn.metrics import confusion_matrix, accuracy_score,r2_score,mean_squared_error
from sklearn import tree
```

```
from sklearn import linear_model
from sklearn import svm
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
x_train = train[train.columns[1:5]]
y_train  = train[train.columns[5]]
x_test = test[test.columns[1:5]]
y_test = test[test.columns[5]]
```

#logistic regression

```
classifier = LogisticRegression()
classifier.fit(x_train,y_train)
yPred = classifier.predict(x_test)
cm = confusion_matrix(y_test,yPred)
accuracy = accuracy_score(y_test,yPred)
r2 = r2_score(y_test,yPred)
rmse = mean_squared_error(y_test,yPred)
print("Logistic Regression :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

OUTPUT:

```
Accuracy =  0.9777777777777777
R2 =  0.9654907975460123
```

```
RMSE =  0.022222222222222223

[[15  0  0]

 [ 0 16  0]

 [ 0  1 13]]
```

#Neural Network

```
classifier = MLPClassifier()

classifier.set_params(hidden_layer_sizes =(100,100), max_iter = 1000,alpha = 0.01, momentum =
0.7)

classifier = classifier.fit(x_train,y_train)

yPred = classifier.predict(x_test)

cm = confusion_matrix(y_test,yPred)

accuracy = accuracy_score(y_test,yPred)

r2 = r2_score(y_test,yPred)

rmse = mean_squared_error(y_test,yPred)

print("Artificial Neural Network :")

print("Accuracy = ", accuracy)

print("R2 = ",r2)

print("RMSE = ",rmse)

print(cm)
```

OUTPUT:

```
Accuracy =  1.0

R2 =  1.0

RMSE =  0.0

[[15  0  0]

 [ 0 16  0]

 [ 0  0 14]]
```

#Support Vector Machine

```
from sklearn.svm import SVC

classifier = SVC(kernel='linear',random_state=0)

classifier.fit(x_train,y_train)

yPred = classifier.predict(x_test)
```

```
cm = confusion_matrix(y_test,yPred)
r2 = r2_score(y_test,yPred)
rmse = mean_squared_error(y_test,yPred)
accuracy = accuracy_score(y_test,yPred)
print("Support Vector Machine :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

OUTPUT:

```
Accuracy =  1.0
R2 =  1.0
RMSE =  0.0
[[15  0  0]
 [ 0 16  0]
 [ 0  0 14]]
```

#Gaussian Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train,y_train)
yPred = classifier.predict(x_test)
cm = confusion_matrix(y_test,yPred)
r2 = r2_score(y_test,yPred)
rmse = mean_squared_error(y_test,yPred)
accuracy = accuracy_score(y_test,yPred)
print("Gaussian Naive Bayes :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

OUTPUT:

```
Accuracy =  0.9777777777777777
R2 =  0.9654907975460123
RMSE =  0.022222222222222223
[[15  0  0]
 [ 0 15  1]
 [ 0  0 14]]
```

#K-Nearest Neighbours

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
classifier.fit(x_train,y_train)
yPred = classifier.predict(x_test)
cm = confusion_matrix(y_test,yPred)
r2 = r2_score(y_test,yPred)
rmse = mean_squared_error(y_test,yPred)
accuracy = accuracy_score(y_test,yPred)
print("K nearest neighbour :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

OUTPUT:

```
Accuracy =  0.9777777777777777
R2 =  0.9654907975460123
RMSE =  0.022222222222222223
[[15  0  0]
 [ 0 15  1]
 [ 0  0 14]]
```

#Bagging

```
classifier = BaggingClassifier()
classifier.set_params(n_estimators = 30,max_samples = 1000)
classifier = classifier.fit(x_train,y_train)
```

```
yPred = classifier.predict(x_test)
cm = confusion_matrix(y_test,yPred)
r2 = r2_score(y_test,yPred)
rmse = mean_squared_error(y_test,yPred)
accuracy = accuracy_score(y_test,yPred)
print("Bagging :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

#Random Forest

```
classifier = RandomForestClassifier()
classifier.set_params(n_estimators = 100, max_depth = 10)
classifier = classifier.fit(x_train,y_train)
yPred = classifier.predict(x_test)
cm = confusion_matrix(y_test,yPred)
r2 = r2_score(y_test,yPred)
rmse = mean_squared_error(y_test,yPred)
accuracy = accuracy_score(y_test,yPred)
print("Random Forest :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

OUTPUT:

```
Accuracy =  0.9777777777777777
R2 =  0.9654907975460123
RMSE =  0.022222222222222223
[[15  0  0]
 [ 0 15  1]
```

```
 [ 0  0 14]]
```

#Boosting

```
classifier = GradientBoostingClassifier()
classifier.set_params(n_estimators = 30,learning_rate = 1)
classifier = classifier.fit(x_train,y_train)
yPred = classifier.predict(x_test)
cm = confusion_matrix(y_test,yPred)
r2 = r2_score(y_test,yPred)
rmse = mean_squared_error(y_test,yPred)
accuracy = accuracy_score(y_test,yPred)
print("Gradient Boosting :")
print("Accuracy = ", accuracy)
print("R2 = ",r2)
print("RMSE = ",rmse)
print(cm)
```

OUTPUT:

```
Accuracy =  0.9777777777777777
R2 =  0.9654907975460123
RMSE =  0.022222222222222223
[[15  0  0]
 [ 0 15  1]
 [ 0  0 14]]
```

#AdaBoost.

```
classifier = AdaBoostClassifier()
classifier.set_params(n_estimators = 10, learning_rate = 1)
classifier = classifier.fit(x_train,y_train)
yPred = classifier.predict(x_test)
cm = confusion_matrix(y_test,yPred)
r2 = r2_score(y_test,yPred)
```

```
rmse = mean_squared_error(y_test,yPred)

accuracy = accuracy_score(y_test,yPred)

print("AdaBoost :")

print("Accuracy = ", accuracy)

print("R2 = ",r2)

print("RMSE = ",rmse)

print(cm)
```

OUTPUT:

```
Accuracy =  0.9555555555555556

R2 =  0.9309815950920245

RMSE =  0.044444444444444446

[[15  0  0]

 [ 0 15  1]

 [ 0  1 13]]
```

```
Logistic Regression :
Accuracy =  0.9777777777777777
R2 =  0.9654907975460123
RMSE =  0.022222222222222223
[[15  0  0]
 [ 0 16  0]
 [ 0  1 13]]

In [5]: classifier = MLPClassifier()
   ...: classifier.set_params(hidden_layer_sizes
=(100,100), max_iter = 1000,alpha = 0.01, momentum = 0.7)
   ...: classifier = classifier.fit(x_train,y_train)
   ...: yPred = classifier.predict(x_test)
   ...: cm = confusion_matrix(y_test,yPred)
   ...: accuracy = accuracy_score(y_test,yPred)
   ...: r2 = r2_score(y_test,yPred)
   ...: rmse = mean_squared_error(y_test,yPred)
   ...: print("Artificial Neural Network :")
   ...: print("Accuracy = ", accuracy)
   ...: print("R2 = ",r2)
   ...: print("RMSE = ",rmse)
   ...: print(cm)
Artificial Neural Network :
Accuracy =  1.0
R2 =  1.0
RMSE =  0.0
[[15  0  0]
 [ 0 16  0]
```

IPython console | History log

```
   ...: print(cm)
Support Vector Machine :
Accuracy =  1.0
R2 =  1.0
RMSE =  0.0
[[15  0  0]
 [ 0 16  0]
 [ 0  0 14]]

In [7]: from sklearn.naive_bayes import GaussianNB
   ...: classifier = GaussianNB()
   ...: classifier.fit(x_train,y_train)
   ...: yPred = classifier.predict(x_test)
   ...: cm = confusion_matrix(y_test,yPred)
   ...: r2 = r2_score(y_test,yPred)
   ...: rmse = mean_squared_error(y_test,yPred)
   ...: accuracy = accuracy_score(y_test,yPred)
   ...: print("Gaussian Naive Bayes :")
   ...: print("Accuracy = ", accuracy)
   ...: print("R2 = ",r2)
   ...: print("RMSE = ",rmse)
   ...: print(cm)
Gaussian Naive Bayes :
Accuracy =  0.9777777777777777
R2 =  0.9654907975460123
RMSE =  0.022222222222222223
[[15  0  0]
 [ 0 15  1]
```

DATASET: CAR EVALUATION

RCODE AND RESULTS:

#load the data

Code:

car_eval = read.csv("F:/ML DA/carsevaluation.csv")
colnames(car_eval)<-c("buying","maint","doors","persons","lug_boot","safety","class")
head(car_eval)

Output:

```
> #load the data
> car_eval = read.csv("F:/ML DA/carsevaluation.csv")
> colnames(car_eval)<-c("buying","maint","doors","persons","lug_boot","saf
ety","class")
> head(car_eval)
  buying maint doors persons lug_boot safety class
1  vhigh vhigh     2       2    small    low unacc
2  vhigh vhigh     2       2    small    med unacc
3  vhigh vhigh     2       2    small   high unacc
4  vhigh vhigh     2       2      med    low unacc
5  vhigh vhigh     2       2      med    med unacc
6  vhigh vhigh     2       2      med   high unacc
```

#Exploratory Data Analysis

Code:

summary(car_eval)
str(car_eval)

Output:

```
> #Exploratory Data Analysis
> summary(car_eval)
   buying      maint       doors     persons     lug_boot    safety      c
lass
 high :432   high :432   2    :432   2   :576   big  :576   high:576   acc
: 384
 low  :432   low  :432   3    :432   4   :576   med  :576   low :576   goo
d :  69
 med  :432   med  :432   4    :432   more:576   small:576   med :576   una
cc:1210
 vhigh:432   vhigh:432   5more:432                                      vgo
od:  65
> str(car_eval)
'data.frame':	1728 obs. of  7 variables:
 $ buying  : Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4
...
 $ maint   : Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4
...
 $ doors   : Factor w/ 4 levels "2","3","4","5more": 1 1 1 1 1 1 1 1 1 1 .
..
 $ persons : Factor w/ 3 levels "2","4","more": 1 1 1 1 1 1 1 1 1 2 ...
 $ lug_boot: Factor w/ 3 levels "big","med","small": 3 3 3 2 2 2 1 1 1 3 .
..
 $ safety  : Factor w/ 3 levels "high","low","med": 2 3 1 2 3 1 2 3 1 2 ..
.
 $ class   : Factor w/ 4 levels "acc","good","unacc",..: 3 3 3 3 3 3 3 3 3
3 ...
```

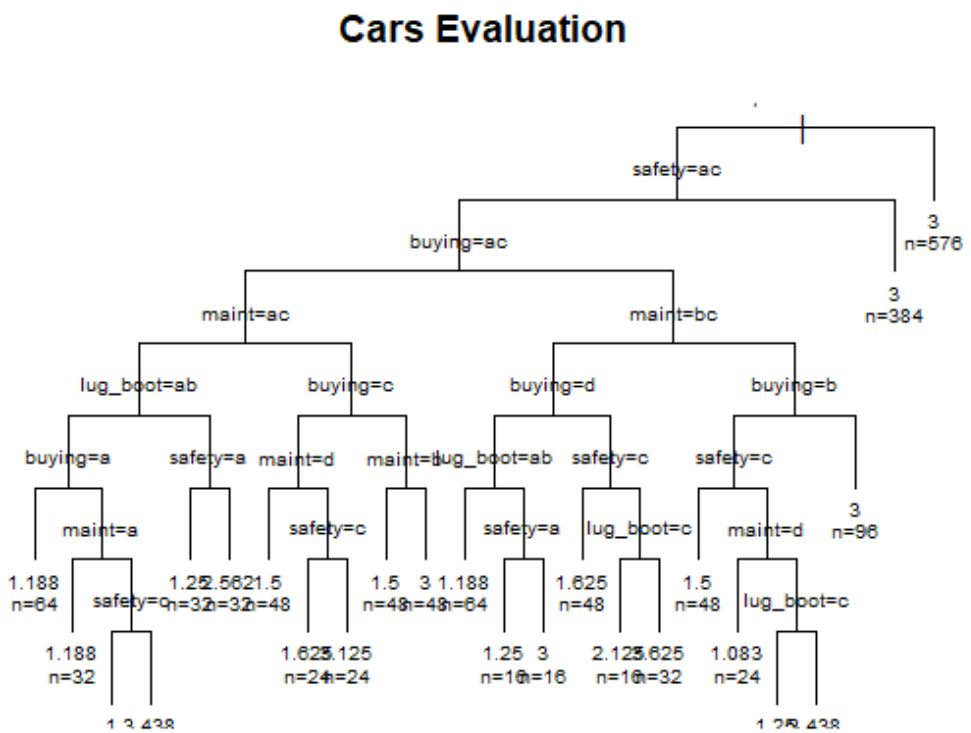
#Decision Tree

Code:

```
library(rpart)
#Build the model
model<-rpart(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval , method =
'anova')
plot(model, uniform=TRUE,main="Cars Evaluation")
text(model, use.n=TRUE,cex=.6)
printcp(model)
```

Output:

```
> #Classification and Regression Trees(CART) decision tree
> library(rpart)
> #Build the model
> model<-rpart(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval ,
method = 'anova')
> plot(model, uniform=TRUE,main="Cars Evaluation")
> text(model, use.n=TRUE,cex=.6)
> printcp(model)

Regression tree:
rpart(formula = class ~ buying + maint + doors + persons + lug_boot +
    safety, data = car_eval, method = "anova")

Variables actually used in tree construction:
[1] buying   lug_boot maint    persons  safety

Root node error: 1325.1/1728 = 0.76684

n= 1728

        CP nsplit rel error  xerror     xstd
1 0.162675      0   1.00000 1.00101 0.029822
2 0.028817      2   0.67465 0.67744 0.026490
3 0.020675      7   0.51971 0.42688 0.023983
4 0.018112     10   0.45769 0.36828 0.024915
5 0.017121     11   0.43958 0.35263 0.025713
6 0.016555     13   0.40533 0.33491 0.024881
7 0.016320     18   0.32256 0.32139 0.024957
8 0.013376     20   0.28992 0.28363 0.023593
9 0.010000     23   0.24391 0.25830 0.022847
```



Neural Network:

Code:

```
library(nnet)
#Build the model
model<-
nnet(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval,size
= 4,decay = 0.0001,maxit = 500)
model
```

Output:

```
> library(nnet)
> #Build the model
> model<-nnet(class~buying+maint+doors+persons+lug_boot+safety,data=car_ev
al,size = 4,decay = 0.0001,maxit = 500)
# weights:  84
initial  value 2876.941690
iter  10 value 664.833515
iter  20 value 351.146192
iter  30 value 196.097581
iter  40 value 127.153784
iter  50 value 102.124774
iter  60 value 88.106953
iter  70 value 84.409178
iter  80 value 79.814165
iter  90 value 76.168802
```

```
iter 100 value 71.391408
iter 110 value 69.587460
iter 120 value 68.422438
iter 130 value 67.685459
iter 140 value 65.735973
iter 150 value 63.069165
iter 160 value 55.646582
iter 170 value 50.902625
iter 180 value 49.089001
iter 190 value 47.556182
iter 200 value 47.287091
iter 210 value 46.161999
iter 220 value 45.420187
iter 230 value 44.774950
iter 240 value 43.499987
iter 250 value 41.604496
iter 260 value 39.896982
iter 270 value 39.628078
iter 280 value 39.195841
iter 290 value 36.451245
iter 300 value 34.717002
iter 310 value 33.000447
iter 320 value 31.840266
iter 330 value 31.122663
iter 340 value 30.502133
iter 350 value 30.290285
iter 360 value 30.193979
iter 370 value 30.071046
iter 380 value 29.969224
iter 390 value 29.820713
iter 400 value 29.725851
iter 410 value 29.643706
iter 420 value 29.601526
iter 430 value 29.581857
iter 440 value 29.556836
iter 450 value 29.532638
iter 460 value 29.519943
iter 470 value 29.519535
iter 480 value 29.518424
iter 490 value 29.516032
iter 500 value 29.512283
final  value 29.512283
stopped after 500 iterations
> model
a 15-4-4 network with 84 weights
inputs: buyinglow buyingmed buyingvhigh maintlow maintmed maintvhigh doors
3 doors4 doors5more persons4 personsmore lug_bootmed lug_bootsmall safetyl
ow safetymed
output(s): class
options were - softmax modelling  decay=1e-04
```

#Support Vector Machine

Code:

```
library(kernlab)
#Build the model
model<-
ksvm(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval)
model
```

Output:

```
> #Support Vector Machine
> library(kernlab)
> #Build the model
> model<-ksvm(class~buying+maint+doors+persons+lug_boot+safety,data=car_ev
al)
> model
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.180555555555556

Number of Support Vectors : 790

Objective Function Value : -86.159 -269.3608 -63.2354 -66.9172 -40.5667 -5
0.3608
Training error : 0.03125
```

#Summarize the model

```
summary(model)
```

```
> #Summarize the model
> summary(model)
Length  Class   Mode
     1   ksvm     S4
```

#NaiveBayes

Code:

```
library(e1071)
#Build the model
model<-
naiveBayes(class~buying+maint+doors+persons+lug_boot+safety,data=car_eva
l,k=5)
model
```

Output:

```
> #NaiveBayes
> library(e1071)
> #Build the model
> model<-naiveBayes(class~buying+maint+doors+persons+lug_boot+safety,data=
car_eval,k=5)
> model

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace, k = 5)

A-priori probabilities:
Y
       acc       good      unacc      vgood
0.22222222 0.03993056 0.70023148 0.03761574

Conditional probabilities:
       buying
Y            high       low       med     vhigh
  acc   0.2812500 0.2317708 0.2994792 0.1875000
  good  0.0000000 0.6666667 0.3333333 0.0000000
  unacc 0.2677686 0.2132231 0.2214876 0.2975207
  vgood 0.0000000 0.6000000 0.4000000 0.0000000

       maint
Y            high       low       med     vhigh
  acc   0.2734375 0.2395833 0.2994792 0.1875000
  good  0.0000000 0.6666667 0.3333333 0.0000000
  unacc 0.2595041 0.2214876 0.2214876 0.2975207
  vgood 0.2000000 0.4000000 0.4000000 0.0000000

       doors
Y               2         3         4     5more
  acc   0.2109375 0.2578125 0.2656250 0.2656250
  good  0.2173913 0.2608696 0.2608696 0.2608696
  unacc 0.2694215 0.2479339 0.2413223 0.2413223
  vgood 0.1538462 0.2307692 0.3076923 0.3076923

       persons
Y               2         4      more
  acc   0.0000000 0.5156250 0.4843750
  good  0.0000000 0.5217391 0.4782609
  unacc 0.4760331 0.2578512 0.2661157
```

```
  vgood 0.0000000 0.4615385 0.5384615

       lug_boot
Y             big       med     small
  acc   0.3750000 0.3515625 0.2734375
  good  0.3478261 0.3478261 0.3043478
  unacc 0.3041322 0.3239669 0.3719008
  vgood 0.6153846 0.3846154 0.0000000

       safety
Y            high       low       med
  acc   0.5312500 0.0000000 0.4687500
  good  0.4347826 0.0000000 0.5652174
  unacc 0.2289256 0.4760331 0.2950413
  vgood 1.0000000 0.0000000 0.0000000
```

#Summarize the model

summary(model)

```
> #Summarize the model
> summary(model)
          Length Class  Mode
apriori   4      table  numeric
tables    6      -none- list
levels    4      -none- character
isnumeric 6      -none- logical
call      5      -none- call

>
```

#k-Nearest Neighbors

Code:

library(caret)

#Build the model

model<-

knn3(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval,k=5)

model

Output:

```
> #k-Nearest Neighbors
> library(caret)
> #Build the model
> model<-knn3(class~buying+maint+doors+persons+lug_boot+safety,data=car_ev
al,k=5)
> model
5-nearest neighbor model
Training set outcome distribution:

  acc  good unacc vgood
  384    69  1210    65
```

#Summarize the model

summary(model)

```
> #Summarize the model
> summary(model)
        Length Class  Mode
learn   2      -none- list
k       1      -none- numeric
terms   3      terms  call
xlevels 6      -none- list
theDots 0      -none- list
```

#Bagging

Code:

library(ipred)

#Build the model

model<-

bagging(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval)

model

Output:

```
> #Bagging CART
> library(ipred)
> #Build the model
> model<-bagging(class~buying+maint+doors+persons+lug_boot+safety,data=car
_eval)
> model

Bagging classification trees with 25 bootstrap replications

Call: bagging.data.frame(formula = class ~ buying + maint + doors +
    persons + lug_boot + safety, data = car_eval)
```

#Summarize the model

summary(model)

#Random Forest

Code:

library(randomForest)

#Build the model

model<-

randomForest(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval)

model

Output:

```
> #Random Forest
> library(randomForest)
> #Build the model
> model<-randomForest(class~buying+maint+doors+persons+lug_boot+safety,dat
a=car_eval)
> model

Call:
 randomForest(formula = class ~ buying + maint + doors + persons +      lu
g_boot + safety, data = car_eval)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 3.07%
Confusion matrix:
      acc good unacc vgood class.error
acc   375    4     3     2  0.02343750
good    8   54     0     7  0.21739130
unacc  20    0  1190     0  0.01652893
vgood   9    0     0    56  0.13846154
```

#Summarize the model

summary(model)

```
> #Summarize the model
> summary(model)
                Length Class  Mode
call               3   -none- call
type               1   -none- character
predicted       1728   factor numeric
err.rate        2500   -none- numeric
confusion         20   -none- numeric
votes           6912   matrix numeric
oob.times       1728   -none- numeric
classes            4   -none- character
importance         6   -none- numeric
importanceSD       0   -none- NULL
localImportance    0   -none- NULL
proximity          0   -none- NULL
ntree              1   -none- numeric
mtry               1   -none- numeric
forest            14   -none- list
y               1728   factor numeric
test               0   -none- NULL
inbag              0   -none- NULL
terms              3   terms  call
```

#Gradient Boosted Machine

Code:

library(gbm)

#Build the model

model<-
gbm(class~buying+maint+doors+persons+lug_boot+safety,data=car_eval,distribution="multinomial")

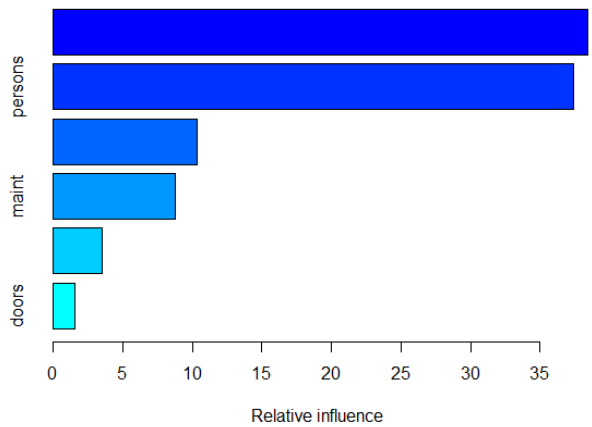
model

```
> #Gradient Boosted Machine
> library(gbm)
> #Build the model
> model<-gbm(class~buying+maint+doors+persons+lug_boot+safety,data=car_eva
l,distribution="multinomial")
> model
gbm(formula = class ~ buying + maint + doors + persons + lug_boot +
    safety, distribution = "multinomial", data = car_eval)
A gradient boosted model with multinomial loss function.
100 iterations were performed.
There were 6 predictors of which 6 had non-zero influence.
```

#Summarize the model

summary(model)

```
> #Summarize the model
> summary(model)
               var   rel.inf
safety      safety 38.430954
persons    persons 37.395453
buying      buying 10.328409
maint        maint  8.769498
lug_boot  lug_boot  3.536240
doors        doors  1.539447
```



PYTHON CODE AND RESULTS:

DATASET: CERVICAL CANCER

This dataset focuses on the prediction of indicators/diagnosis of cervical cancer. The features cover demographic information, habits, and historic medical records.

Data Set Characteristics:	Multivariate	**Number of Instances:**	858	**Area:**	Life
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	36	**Date Donated**	2017-03-03
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	76618

Attribute Information:

(int) Age
(int) Number of sexual partners
(int) First sexual intercourse (age)
(int) Num of pregnancies
(bool) Smokes
(bool) Smokes (years)
(bool) Smokes (packs/year)
(bool) Hormonal Contraceptives
(int) Hormonal Contraceptives (years)
(bool) IUD
(int) IUD (years)
(bool) STDs
(int) STDs (number)
(bool) STDs:condylomatosis
(bool) STDs:cervical condylomatosis
(bool) STDs:vaginal condylomatosis
(bool) STDs:vulvo-perineal condylomatosis
(bool) STDs:syphilis
(bool) STDs:pelvic inflammatory disease
(bool) STDs:genital herpes
(bool) STDs:molluscum contagiosum
(bool) STDs:AIDS
(bool) STDs:HIV
(bool) STDs:Hepatitis B
(bool) STDs:HPV
(int) STDs: Number of diagnosis
(int) STDs: Time since first diagnosis
(int) STDs: Time since last diagnosis
(bool) Dx:Cancer
(bool) Dx:CIN
(bool) Dx:HPV

(bool) Dx
(bool) Hinselmann: target variable
(bool) Schiller: target variable
(bool) Cytology: target variable
(bool) Biopsy: target variable

RCODE AND RESULTS:

```
> library(dplyr)
> library(readr)
> library(tidyverse)
> library(mlr)
> library(ggplot2)
> library(gridExtra)
> library(gmodels)
> library(GGally)
> library(cowplot)
> library(tidyr)
> library(magrittr)
> library(moments)
> library(purrr)
> library(data.table)
> library(latex2exp)
> library(caret)
> library(robustHD)
> library(spFSR)
> library(rjson)
> library(party)
> library(knitr)
> library(kableExtra)
> library(stringr)
> library(mlbench)
> library(e1071)
> library(MASS)
> set.seed(999)
> dataCancer <- read_csv("C:/Users/hp/Desktop/risk_factors_cervical_cancer
.csv", col_names = TRUE, na = "?", trim_ws = TRUE)
> kable(dataCancer %>% head(), caption = "Table 1. Cervical Cancer Dataset
") %>% kable_styling(bootstrap_options = c("condensed"), full_width = TRUE
, font_size = 10)
<table class="table table-condensed" style="font-size: 10px; margin-left:
auto; margin-right: auto;">
<caption style="font-size: initial !important;">Table 1. Cervical Cancer D
ataset</caption>
 <thead>
  <tr>
   <th style="text-align:right;"> Age </th>
   <th style="text-align:right;"> Number_of_sexual_partners </th>
   <th style="text-align:right;"> First_sexual_intercourse </th>
   <th style="text-align:right;"> Num_of_pregnancies </th>
   <th style="text-align:right;"> Smokes </th>
   <th style="text-align:right;"> Smokes (years) </th>
   <th style="text-align:right;"> Smokes (packs/year) </th>
   <th style="text-align:right;"> Hormonal_Contraceptives </th>
   <th style="text-align:right;"> Hormonal Contraceptives (years) </th>
   <th style="text-align:right;"> IUD </th>
   <th style="text-align:right;"> IUD (years) </th>
   <th style="text-align:right;"> STDs </th>
```

```
   <th style="text-align:right;"> STDs (number) </th>
   <th style="text-align:right;"> STDs_condylomatosis </th>
   <th style="text-align:right;"> STDs_cervical_condylomatosis </th>
   <th style="text-align:right;"> STDs_vaginal_condylomatosis </th>
   <th style="text-align:right;"> STDs_vulvo_perineal_condylomatosis </th>
   <th style="text-align:right;"> STDs_syphilis </th>
   <th style="text-align:right;"> STDs_pelvic_inflammatory_disease </th>
   <th style="text-align:right;"> STDs_genital_herpes </th>
   <th style="text-align:right;"> STDs_molluscum_contagiosum </th>
   <th style="text-align:right;"> STDs_AIDS </th>
   <th style="text-align:right;"> STDs_HIV </th>
   <th style="text-align:right;"> STDs_Hepatitis_B </th>
   <th style="text-align:right;"> STDs_HPV </th>
   <th style="text-align:right;"> STDs_Number_of_diagnosis </th>
   <th style="text-align:right;"> STDs_Time_since_first_diagnosis </th>
   <th style="text-align:right;"> STDs_Time_since_last_diagnosis </th>
   <th style="text-align:right;"> Dx_Cancer </th>
   <th style="text-align:right;"> Dx_CIN </th>
   <th style="text-align:right;"> Dx_HPV </th>
   <th style="text-align:right;"> Dx </th>
   <th style="text-align:right;"> Hinselmann </th>
   <th style="text-align:right;"> Schiller </th>
   <th style="text-align:right;"> Citology </th>
   <th style="text-align:right;"> Biopsy </th>
  </tr>
 </thead>
<tbody>
  <tr>
   <td style="text-align:right;"> 18 </td>
   <td style="text-align:right;"> 4 </td>
   <td style="text-align:right;"> 15 </td>
   <td style="text-align:right;"> 1 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> NA </td>
   <td style="text-align:right;"> NA </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
   <td style="text-align:right;"> 0 </td>
```

```
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:right;"> 15 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 14 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> NA </td>
 <td style="text-align:right;"> NA </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:right;"> 34 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> NA </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
```

```
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> NA </td>
 <td style="text-align:right;"> NA </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:right;"> 52 </td>
 <td style="text-align:right;"> 5 </td>
 <td style="text-align:right;"> 16 </td>
 <td style="text-align:right;"> 4 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 37 </td>
 <td style="text-align:right;"> 37 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 3 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> NA </td>
 <td style="text-align:right;"> NA </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:right;"> 46 </td>
 <td style="text-align:right;"> 3 </td>
 <td style="text-align:right;"> 21 </td>
 <td style="text-align:right;"> 4 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
```

```
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 15 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> NA </td>
 <td style="text-align:right;"> NA </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:right;"> 42 </td>
 <td style="text-align:right;"> 3 </td>
 <td style="text-align:right;"> 23 </td>
 <td style="text-align:right;"> 2 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> NA </td>
 <td style="text-align:right;"> NA </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
```

```
    <td style="text-align:right;"> 0 </td>
    <td style="text-align:right;"> 0 </td>
    <td style="text-align:right;"> 0 </td>
    <td style="text-align:right;"> 0 </td>
    <td style="text-align:right;"> 0 </td>
    <td style="text-align:right;"> 0 </td>
  </tr>
</tbody>
</table>
```

Table 1. Cervical Cancer Dataset

Age	Number_of_sexual_partners	First_sexual_intercourse	Num_of_pregnancies	Sn
18	4	15	1	
15	1	14	1	
34	1	NA	1	
52	5	16	4	
46	3	21	4	
42	3	23	2	

```
> dim(dataCancer)
[1] 858  36
> kable(dataCancer %>% summarizeColumns(), caption = "Table 2. Statistical
Summary") %>% kable_styling(bootstrap_options = c("condensed"), full_width
= TRUE, font_size = 10)
<table class="table table-condensed" style="font-size: 10px; margin-left:
auto; margin-right: auto;">
<caption style="font-size: initial !important;">Table 2. Statistical Summa
ry</caption>
 <thead>
  <tr>
   <th style="text-align:left;"> name </th>
   <th style="text-align:left;"> type </th>
   <th style="text-align:right;"> na </th>
   <th style="text-align:right;"> mean </th>
   <th style="text-align:right;"> disp </th>
   <th style="text-align:right;"> median </th>
   <th style="text-align:right;"> mad </th>
   <th style="text-align:right;"> min </th>
   <th style="text-align:right;"> max </th>
   <th style="text-align:right;"> nlevs </th>
  </tr>
 </thead>
<tbody>
  <tr>
   <td style="text-align:left;"> Age </td>
   <td style="text-align:left;"> numeric </td>
```

```
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 26.8205128 </td>
 <td style="text-align:right;"> 8.4979481 </td>
 <td style="text-align:right;"> 25.0 </td>
 <td style="text-align:right;"> 8.1543 </td>
 <td style="text-align:right;"> 13 </td>
 <td style="text-align:right;"> 84 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> Number_of_sexual_partners </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 26 </td>
 <td style="text-align:right;"> 2.5276442 </td>
 <td style="text-align:right;"> 1.6677605 </td>
 <td style="text-align:right;"> 2.0 </td>
 <td style="text-align:right;"> 1.4826 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 28 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> First_sexual_intercourse </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 7 </td>
 <td style="text-align:right;"> 16.9952996 </td>
 <td style="text-align:right;"> 2.8033554 </td>
 <td style="text-align:right;"> 17.0 </td>
 <td style="text-align:right;"> 2.9652 </td>
 <td style="text-align:right;"> 10 </td>
 <td style="text-align:right;"> 32 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> Num_of_pregnancies </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 56 </td>
 <td style="text-align:right;"> 2.2755611 </td>
 <td style="text-align:right;"> 1.4474141 </td>
 <td style="text-align:right;"> 2.0 </td>
 <td style="text-align:right;"> 1.4826 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 11 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> Smokes </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 13 </td>
 <td style="text-align:right;"> 0.1455621 </td>
 <td style="text-align:right;"> 0.3528756 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> Smokes (years) </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 13 </td>
 <td style="text-align:right;"> 1.2197214 </td>
```

```
 <td style="text-align:right;"> 4.0890169 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 37 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> Smokes (packs/year) </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 13 </td>
 <td style="text-align:right;"> 0.4531440 </td>
 <td style="text-align:right;"> 2.2266098 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 37 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> Hormonal_Contraceptives </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 108 </td>
 <td style="text-align:right;"> 0.6413333 </td>
 <td style="text-align:right;"> 0.4799292 </td>
 <td style="text-align:right;"> 1.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> Hormonal Contraceptives (years) </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 108 </td>
 <td style="text-align:right;"> 2.2564192 </td>
 <td style="text-align:right;"> 3.7642535 </td>
 <td style="text-align:right;"> 0.5 </td>
 <td style="text-align:right;"> 0.7413 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 30 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> IUD </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 117 </td>
 <td style="text-align:right;"> 0.1120108 </td>
 <td style="text-align:right;"> 0.3155928 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> IUD (years) </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 117 </td>
 <td style="text-align:right;"> 0.5148043 </td>
 <td style="text-align:right;"> 1.9430885 </td>
 <td style="text-align:right;"> 0.0 </td>
```

```
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 19 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.1049137 </td>
 <td style="text-align:right;"> 0.3066458 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs (number) </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.1766268 </td>
 <td style="text-align:right;"> 0.5619928 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 4 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_condylomatosis </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0584329 </td>
 <td style="text-align:right;"> 0.2347162 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_cervical_condylomatosis </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0000000 </td>
 <td style="text-align:right;"> 0.0000000 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_vaginal_condylomatosis </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0053121 </td>
 <td style="text-align:right;"> 0.0727385 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
```

```
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_vulvo_perineal_condylomatosis </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0571049 </td>
 <td style="text-align:right;"> 0.2321972 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_syphilis </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0239044 </td>
 <td style="text-align:right;"> 0.1528528 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_pelvic_inflammatory_disease </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0013280 </td>
 <td style="text-align:right;"> 0.0364420 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_genital_herpes </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0013280 </td>
 <td style="text-align:right;"> 0.0364420 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_molluscum_contagiosum </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0013280 </td>
 <td style="text-align:right;"> 0.0364420 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
```

```
</tr>
<tr>
 <td style="text-align:left;"> STDs_AIDS </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0000000 </td>
 <td style="text-align:right;"> 0.0000000 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_HIV </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0239044 </td>
 <td style="text-align:right;"> 0.1528528 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_Hepatitis_B </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0013280 </td>
 <td style="text-align:right;"> 0.0364420 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_HPV </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 105 </td>
 <td style="text-align:right;"> 0.0026560 </td>
 <td style="text-align:right;"> 0.0515025 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_Number_of_diagnosis </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0.0874126 </td>
 <td style="text-align:right;"> 0.3025447 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 3 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
```

```
 <td style="text-align:left;"> STDs_Time_since_first_diagnosis </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 787 </td>
 <td style="text-align:right;"> 6.1408451 </td>
 <td style="text-align:right;"> 5.8950240 </td>
 <td style="text-align:right;"> 4.0 </td>
 <td style="text-align:right;"> 4.4478 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 22 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_Time_since_last_diagnosis </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 787 </td>
 <td style="text-align:right;"> 5.8169014 </td>
 <td style="text-align:right;"> 5.7552705 </td>
 <td style="text-align:right;"> 3.0 </td>
 <td style="text-align:right;"> 2.9652 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 22 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> Dx_Cancer </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0.0209790 </td>
 <td style="text-align:right;"> 0.1433976 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> Dx_CIN </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0.0104895 </td>
 <td style="text-align:right;"> 0.1019392 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> Dx_HPV </td>
 <td style="text-align:left;"> numeric </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 0.0209790 </td>
 <td style="text-align:right;"> 0.1433976 </td>
 <td style="text-align:right;"> 0.0 </td>
 <td style="text-align:right;"> 0.0000 </td>
 <td style="text-align:right;"> 0 </td>
 <td style="text-align:right;"> 1 </td>
 <td style="text-align:right;"> 0 </td>
</tr>
<tr>
 <td style="text-align:left;"> Dx </td>
 <td style="text-align:left;"> numeric </td>
```

```
      <td style="text-align:right;"> 0 </td>
      <td style="text-align:right;"> 0.0279720 </td>
      <td style="text-align:right;"> 0.1649888 </td>
      <td style="text-align:right;"> 0.0 </td>
      <td style="text-align:right;"> 0.0000 </td>
      <td style="text-align:right;"> 0 </td>
      <td style="text-align:right;"> 1 </td>
      <td style="text-align:right;"> 0 </td>
    </tr>
    <tr>
      <td style="text-align:left;"> Hinselmann </td>
      <td style="text-align:left;"> numeric </td>
      <td style="text-align:right;"> 0 </td>
      <td style="text-align:right;"> 0.0407925 </td>
      <td style="text-align:right;"> 0.1979246 </td>
      <td style="text-align:right;"> 0.0 </td>
      <td style="text-align:right;"> 0.0000 </td>
      <td style="text-align:right;"> 0 </td>
      <td style="text-align:right;"> 1 </td>
      <td style="text-align:right;"> 0 </td>
    </tr>
    <tr>
      <td style="text-align:left;"> Schiller </td>
      <td style="text-align:left;"> numeric </td>
      <td style="text-align:right;"> 0 </td>
      <td style="text-align:right;"> 0.0862471 </td>
      <td style="text-align:right;"> 0.2808923 </td>
      <td style="text-align:right;"> 0.0 </td>
      <td style="text-align:right;"> 0.0000 </td>
      <td style="text-align:right;"> 0 </td>
      <td style="text-align:right;"> 1 </td>
      <td style="text-align:right;"> 0 </td>
    </tr>
    <tr>
      <td style="text-align:left;"> Citology </td>
      <td style="text-align:left;"> numeric </td>
      <td style="text-align:right;"> 0 </td>
      <td style="text-align:right;"> 0.0512821 </td>
      <td style="text-align:right;"> 0.2207011 </td>
      <td style="text-align:right;"> 0.0 </td>
      <td style="text-align:right;"> 0.0000 </td>
      <td style="text-align:right;"> 0 </td>
      <td style="text-align:right;"> 1 </td>
      <td style="text-align:right;"> 0 </td>
    </tr>
    <tr>
      <td style="text-align:left;"> Biopsy </td>
      <td style="text-align:left;"> numeric </td>
      <td style="text-align:right;"> 0 </td>
      <td style="text-align:right;"> 0.0641026 </td>
      <td style="text-align:right;"> 0.2450784 </td>
      <td style="text-align:right;"> 0.0 </td>
      <td style="text-align:right;"> 0.0000 </td>
      <td style="text-align:right;"> 0 </td>
      <td style="text-align:right;"> 1 </td>
      <td style="text-align:right;"> 0 </td>
    </tr>
  </tbody>
  </table>
```

Table 2. Statistical Summary

name	type	na	mean	disp	median
Age	numeric	0	26.8205128	8.4979481	25.
Number_of_sexual_partners	numeric	26	2.5276442	1.6677605	2.
First_sexual_intercourse	numeric	7	16.9952996	2.8033554	17.
Num_of_pregnancies	numeric	56	2.2755611	1.4474141	2.
Smokes	numeric	13	0.1455621	0.3528756	0.
Smokes (years)	numeric	13	1.2197214	4.0890169	0.
Smokes (packs/year)	numeric	13	0.4531440	2.2266098	0.
Hormonal_Contraceptives	numeric	108	0.6413333	0.4799292	1.
Hormonal Contraceptives (years)	numeric	108	2.2564192	3.7642535	0.
IUD	numeric	117	0.1120108	0.3155928	0.
IUD (years)	numeric	117	0.5148043	1.9430885	0.

```
> dataCancer <- subset(dataCancer, select = -STDs_cervical_condylomatosis)
> dataCancer <- subset(dataCancer, select = -STDs_vaginal_condylomatosis)
> dataCancer <- subset(dataCancer, select = -STDs_pelvic_inflammatory_dise
ase)
> dataCancer <- subset(dataCancer, select = -STDs_genital_herpes)
> dataCancer <- subset(dataCancer, select = -STDs_molluscum_contagiosum)
> dataCancer <- subset(dataCancer, select = -STDs_Hepatitis_B)
> dataCancer <- subset(dataCancer, select = -STDs_HPV)
> dataCancer <- subset(dataCancer, select = -STDs_AIDS)
> dataCancer <- subset(dataCancer, select = -STDs_Time_since_first_diagnos
is)
> dataCancer <- subset(dataCancer, select = -STDs_Time_since_last_diagnosi
s)
> length(which(is.na(dataCancer)))
[1] 1208
> kable(colSums(is.na(dataCancer)), caption = "Table 3. Number of Missing
Values in Each Column") %>% kable_styling(bootstrap_options = c("condensed
"), full_width = TRUE, font_size = 10)
<table class="table table-condensed" style="font-size: 10px; margin-left:
auto; margin-right: auto;">
<caption style="font-size: initial !important;">Table 3. Number of Missing
Values in Each Column</caption>
 <thead>
  <tr>
   <th style="text-align:left;">   </th>
   <th style="text-align:right;"> x </th>
  </tr>
 </thead>
<tbody>
  <tr>
   <td style="text-align:left;"> Age </td>
   <td style="text-align:right;"> 0 </td>
  </tr>
  <tr>
   <td style="text-align:left;"> Number_of_sexual_partners </td>
   <td style="text-align:right;"> 26 </td>
```

```
</tr>
<tr>
 <td style="text-align:left;"> First_sexual_intercourse </td>
 <td style="text-align:right;"> 7 </td>
</tr>
<tr>
 <td style="text-align:left;"> Num_of_pregnancies </td>
 <td style="text-align:right;"> 56 </td>
</tr>
<tr>
 <td style="text-align:left;"> Smokes </td>
 <td style="text-align:right;"> 13 </td>
</tr>
<tr>
 <td style="text-align:left;"> Smokes (years) </td>
 <td style="text-align:right;"> 13 </td>
</tr>
<tr>
 <td style="text-align:left;"> Smokes (packs/year) </td>
 <td style="text-align:right;"> 13 </td>
</tr>
<tr>
 <td style="text-align:left;"> Hormonal_Contraceptives </td>
 <td style="text-align:right;"> 108 </td>
</tr>
<tr>
 <td style="text-align:left;"> Hormonal Contraceptives (years) </td>
 <td style="text-align:right;"> 108 </td>
</tr>
<tr>
 <td style="text-align:left;"> IUD </td>
 <td style="text-align:right;"> 117 </td>
</tr>
<tr>
 <td style="text-align:left;"> IUD (years) </td>
 <td style="text-align:right;"> 117 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs </td>
 <td style="text-align:right;"> 105 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs (number) </td>
 <td style="text-align:right;"> 105 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_condylomatosis </td>
 <td style="text-align:right;"> 105 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_vulvo_perineal_condylomatosis </td>
 <td style="text-align:right;"> 105 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_syphilis </td>
 <td style="text-align:right;"> 105 </td>
</tr>
<tr>
 <td style="text-align:left;"> STDs_HIV </td>
 <td style="text-align:right;"> 105 </td>
</tr>
<tr>
```

```
    <td style="text-align:left;"> STDs_Number_of_diagnosis </td>
    <td style="text-align:right;"> 0 </td>
   </tr>
   <tr>
    <td style="text-align:left;"> Dx_Cancer </td>
    <td style="text-align:right;"> 0 </td>
   </tr>
   <tr>
    <td style="text-align:left;"> Dx_CIN </td>
    <td style="text-align:right;"> 0 </td>
   </tr>
   <tr>
    <td style="text-align:left;"> Dx_HPV </td>
    <td style="text-align:right;"> 0 </td>
   </tr>
   <tr>
    <td style="text-align:left;"> Dx </td>
    <td style="text-align:right;"> 0 </td>
   </tr>
   <tr>
    <td style="text-align:left;"> Hinselmann </td>
    <td style="text-align:right;"> 0 </td>
   </tr>
   <tr>
    <td style="text-align:left;"> Schiller </td>
    <td style="text-align:right;"> 0 </td>
   </tr>
   <tr>
    <td style="text-align:left;"> Citology </td>
    <td style="text-align:right;"> 0 </td>
   </tr>
   <tr>
    <td style="text-align:left;"> Biopsy </td>
    <td style="text-align:right;"> 0 </td>
   </tr>
  </tbody>
  </table>
```

Table 3. Number of Missing Values in Each Column

	x
Age	0
Number_of_sexual_partners	26
First_sexual_intercourse	7
Num_of_pregnancies	56
Smokes	13
Smokes (years)	13
Smokes (packs/year)	13
Hormonal_Contraceptives	108
Hormonal_Contraceptives (years)	108
IUD	117
IUD (years)	117
STDs	105

```
> dataCancer <- dataCancer[!is.na(dataCancer$Smokes),]
> dataCancer <- dataCancer[!is.na(dataCancer$First_sexual_intercourse),]
> dataCancer <- dataCancer[!is.na(dataCancer$STDs),]
> prop_NA <- function(x) { mean(is.na(x))}
> missingData <- sapply(dataCancer, prop_NA)
```

```
> missingData <- data.frame(Variables = names(missingData), Proportion = m
issingData, Completude = 1 - missingData)
>
>
> dataCancer$Num_of_pregnancies[is.na(dataCancer$Num_of_pregnancies)] <- 0
> summary(dataCancer)
      Age        Number_of_sexual_partners First_sexual_intercourse Num_of
_pregnancies     Smokes
 Min.   :13.00   Min.   : 1.000            Min.   :10.00            Min.
: 0.000     Min.   :0.0000
 1st Qu.:21.00   1st Qu.: 2.000            1st Qu.:15.00            1st Qu
.: 1.000     1st Qu.:0.0000
 Median :26.00   Median : 2.000            Median :17.00            Median
: 2.000     Median :0.0000
 Mean   :27.22   Mean   : 2.512            Mean   :17.09            Mean
: 2.182     Mean   :0.1438
 3rd Qu.:33.00   3rd Qu.: 3.000            3rd Qu.:18.00            3rd Qu
.: 3.000     3rd Qu.:0.0000
 Max.   :84.00   Max.   :28.000            Max.   :32.00            Max.
:11.000     Max.   :1.0000
                 NA's   :13
 Smokes (years)   Smokes (packs/year) Hormonal_Contraceptives Hormonal Con
traceptives (years)
 Min.   : 0.000   Min.   : 0.0000     Min.   :0.0000          Min.   : 0.0
00
 1st Qu.: 0.000   1st Qu.: 0.0000     1st Qu.:0.0000          1st Qu.: 0.0
00
 Median : 0.000   Median : 0.0000     Median :1.0000          Median : 0.5
00
 Mean   : 1.222   Mean   : 0.4606     Mean   :0.6405          Mean   : 2.2
34
 3rd Qu.: 0.000   3rd Qu.: 0.0000     3rd Qu.:1.0000          3rd Qu.: 3.0
00
 Max.   :37.000   Max.   :37.0000     Max.   :1.0000          Max.   :22.0
00
                                      NA's   :11              NA's   :11
      IUD           IUD (years)           STDs        STDs (number)    STDs
_condylomatosis
 Min.   :0.0000   Min.   : 0.0000   Min.   :0.0000   Min.   :0.0000   Min.
:0.00000
 1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st
Qu.:0.00000
 Median :0.0000   Median : 0.0000   Median :0.0000   Median :0.0000   Medi
an :0.00000
 Mean   :0.1122   Mean   : 0.5173   Mean   :0.1031   Mean   :0.1723   Mean
:0.05699
 3rd Qu.:0.0000   3rd Qu.: 0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd
Qu.:0.00000
 Max.   :1.0000   Max.   :19.0000   Max.   :1.0000   Max.   :4.0000   Max.
:1.00000
 NA's   :15       NA's   :15
 STDs_vulvo_perineal_condylomatosis STDs_syphilis        STDs_HIV       ST
Ds_Number_of_diagnosis
 Min.   :0.00000                    Min.   :0.00000   Min.   :0.00000   Mi
n.   :0.00000
 1st Qu.:0.00000                    1st Qu.:0.00000   1st Qu.:0.00000   1s
t Qu.:0.00000
 Median :0.00000                    Median :0.00000   Median :0.00000   Me
dian :0.00000
 Mean   :0.05563                    Mean   :0.02307   Mean   :0.02307   Me
an   :0.09634
```

```
 3rd Qu.:0.00000                     3rd Qu.:0.00000   3rd Qu.:0.00000   3r
d Qu.:0.00000
 Max.   :1.00000                     Max.   :1.00000   Max.   :1.00000   Ma
x.   :3.00000

   Dx_Cancer           Dx_CIN            Dx_HPV              Dx
Hinselmann
 Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   M
in.   :0.00000
 1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1
st Qu.:0.00000
 Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000   M
edian :0.00000
 Mean   :0.02307   Mean   :0.01085   Mean   :0.02307   Mean   :0.02985   M
ean   :0.04613
 3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3
rd Qu.:0.00000
 Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   M
ax.   :1.00000

    Schiller          Citology           Biopsy
 Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
 1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
 Median :0.00000   Median :0.00000   Median :0.00000
 Mean   :0.09634   Mean   :0.05427   Mean   :0.07056
 3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
 Max.   :1.00000   Max.   :1.00000   Max.   :1.00000

>
>
> dataCancer$Cancer <- dataCancer$Biopsy + dataCancer$Citology + dataCance
r$Schiller + dataCancer$Hinselmann
> dataCancer$Cancer[dataCancer$Cancer == 2] <- 1
> dataCancer$Cancer[dataCancer$Cancer == 3] <- 1
> dataCancer$Cancer[dataCancer$Cancer == 4] <- 1
> dataCancer$Cancer <- factor(dataCancer$Cancer, levels = c("0", "1"))
> dataCancer <- subset(dataCancer, select = -Biopsy)
> dataCancer <- subset(dataCancer, select = -Citology)
> dataCancer <- subset(dataCancer, select = -Schiller)
> dataCancer <- subset(dataCancer, select = -Hinselmann)


> #DECISION TREE

> library(rpart)
> model1<-rpart(Cancer~Age+First_sexual_intercourse+Smokes+Number_of_sexua
l_partners+Hormonal_Contraceptives+IUD+STDs,data=dataCancer)
> summary(model1)
Call:
rpart(formula = Cancer ~ Age + First_sexual_intercourse + Smokes +
    Number_of_sexual_partners + Hormonal_Contraceptives + IUD +
    STDs, data = dataCancer)
  n= 737

           CP nsplit rel error xerror xstd
1 0.007894737      0         1      0    0

Node number 1: 737 observations
  predicted class=0  expected loss=0.1289009  P(node) =1
    class counts:   642    95
   probabilities: 0.871 0.129
```

```
> x<-dataCancer[,1:12]
> y<-dataCancer[,23]
> dataCancer$pred_cart<-predict(model1,x,type="class")
> mtab<-table(dataCancer$pred_cart,dataCancer$Cancer)
> confusionMatrix(mtab)
Confusion Matrix and Statistics


      0   1
  0 642  95
  1   0   0

               Accuracy : 0.8711
                 95% CI : (0.8447, 0.8944)
    No Information Rate : 0.8711
    P-Value [Acc > NIR] : 0.5273

                  Kappa : 0
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.8711
         Neg Pred Value :    NaN
             Prevalence : 0.8711
         Detection Rate : 0.8711
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 0

>

> #NEURAL NETWORK

> library(nnet)
> model2<-nnet(Cancer~Age+First_sexual_intercourse+Smokes+Number_of_sexual
_partners+Hormonal_Contraceptives+IUD+STDs,data=dataCancer,size = 4,decay
= 0.0001,maxit = 500)
# weights:  37
initial  value 506.212925
iter  10 value 273.377416
iter  20 value 270.305858
iter  30 value 267.169796
iter  40 value 267.098009
iter  50 value 266.963554
iter  60 value 266.925369
iter  70 value 266.907812
iter  80 value 266.854842
final  value 266.840591
converged
> summary(model2)
a 7-4-1 network with 37 weights
options were - entropy fitting  decay=1e-04
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1
 -0.04  -3.36  -1.41  -0.08  -0.31  -0.05  -0.18  -0.04
 b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2
  0.10  -1.15  -0.64   0.16  -0.19   0.07   0.08  -0.16
 b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3
 13.18   7.89 -17.07  27.61 -28.74  22.04  -3.72  -2.16
 b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4 i6->h4 i7->h4
  0.01  -0.42  -0.35   0.12   0.09  -0.04  -0.08   0.16
```

```
  b->o  h1->o  h2->o  h3->o  h4->o
 -2.01   0.19  -2.82   1.48  -0.05
> dataCancer$pred_nnet<-predict(model2,x,type="class")
> mtab<-table(dataCancer$pred_nnet,dataCancer$Cancer)
> confusionMatrix(mtab)


> #SVM

> library(kernlab)
> model3<-ksvm(Cancer~Age+First_sexual_intercourse+Smokes+Number_of_sexual
_partners+Hormonal_Contraceptives+IUD+STDs,data=dataCancer)
> summary(model3)
Length  Class   Mode
     1   ksvm     S4
> dataCancer$pred_svm<-predict(model3,x,type="response")
> mtab<-table(dataCancer$pred_svm,dataCancer$Cancer)
> confusionMatrix(mtab)


> #LOGISTIC REGRESSION
> library(VGAM)
> model4<-vglm(Cancer~Age+First_sexual_intercourse+Smokes+Number_of_sexual
_partners+Hormonal_Contraceptives+IUD+STDs,family = "multinomial",data=dat
aCancer)
> summary(model4)

Call:
vglm(formula = Cancer ~ Age + First_sexual_intercourse + Smokes +
    Number_of_sexual_partners + Hormonal_Contraceptives + IUD +
    STDs, family = "multinomial", data = dataCancer)

Pearson residuals:
                      Min     1Q Median     3Q   Max
log(mu[,1]/mu[,2]) -3.593 0.3209 0.3445 0.3771 0.889

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)                1.362857   0.803878   1.695  0.09001 .
Age                       -0.006423   0.014219  -0.452  0.65147
First_sexual_intercourse   0.045422   0.046617   0.974  0.32987
Smokes                    -0.485906   0.303670  -1.600  0.10957
Number_of_sexual_partners  0.080258   0.087716   0.915  0.36020
Hormonal_Contraceptives    0.009675   0.236581   0.041  0.96738
IUD                       -0.538483   0.331836  -1.623  0.10465
STDs                      -0.855076   0.312771  -2.734  0.00626 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Name of linear predictor: log(mu[,1]/mu[,2])

Residual deviance: 531.6312 on 699 degrees of freedom

Log-likelihood: -265.8156 on 699 degrees of freedom

Number of Fisher scoring iterations: 5

No Hauck-Donner effect found in any of the estimates


Reference group is level  2  of the response
> probability<-predict(model4,x,type="response")
```

```
> dataCancer$pred_log_reg<-apply(probability,1,which.max)
> mtab<-table(dataCancer$pred_log_reg,dataCancer$Cancer)
> confusionMatrix(mtab)


> #NAIVE BAYES

> model5<-naiveBayes(Cancer~Age+First_sexual_intercourse+Smokes+Number_of_
sexual_partners+Hormonal_Contraceptives+IUD+STDs,data=dataCancer,k=5)
> summary(model5)
          Length Class  Mode
apriori   2      table  numeric
tables    7      -none- list
levels    2      -none- character
isnumeric 7      -none- logical
call      5      -none- call
> dataCancer$pred_naive<-predict(model5,x)
> mtab<-table(dataCancer$pred_naive,dataCancer$Cancer)
> confusionMatrix(mtab)
Confusion Matrix and Statistics


      0   1
  0 581  74
  1  61  21

               Accuracy : 0.8168
                 95% CI : (0.787, 0.8441)
    No Information Rate : 0.8711
    P-Value [Acc > NIR] : 1.0000

                  Kappa : 0.1338
 Mcnemar's Test P-Value : 0.3017

            Sensitivity : 0.9050
            Specificity : 0.2211
         Pos Pred Value : 0.8870
         Neg Pred Value : 0.2561
             Prevalence : 0.8711
         Detection Rate : 0.7883
   Detection Prevalence : 0.8887
      Balanced Accuracy : 0.5630

       'Positive' Class : 0


> #KNN

> model6<-knn3(Cancer~Age+First_sexual_intercourse+Smokes+Number_of_sexual
_partners+Hormonal_Contraceptives+IUD+STDs,data=dataCancer,k=5)
> summary(model6)
          Length Class  Mode
learn      2     -none- list
k          1     -none- numeric
terms      3     terms  call
xlevels    0     -none- list
theDots    0     -none- list
na.action 30     omit   numeric
> dataCancer$pred_knn<-predict(model6,x,type="class")
> mtab<-table(dataCancer$pred_knn,dataCancer$Cancer)
> confusionMatrix(mtab)
Confusion Matrix and Statistics
```

```
      0   1
  0 581  74
  1  61  21

               Accuracy : 0.8168
                 95% CI : (0.787, 0.8441)
    No Information Rate : 0.8711
    P-Value [Acc > NIR] : 1.0000

                  Kappa : 0.1338
 Mcnemar's Test P-Value : 0.3017

            Sensitivity : 0.9050
            Specificity : 0.2211
         Pos Pred Value : 0.8870
         Neg Pred Value : 0.2561
             Prevalence : 0.8711
         Detection Rate : 0.7883
   Detection Prevalence : 0.8887
      Balanced Accuracy : 0.5630

       'Positive' Class : 0
```

```
> library(ipred)
> model7<-bagging(Cancer~Age+First_sexual_intercourse+Smokes+Number_of_sex
ual_partners+Hormonal_Contraceptives+IUD+STDs,data=dataCancer)
> summary(model7)
attr(,"class")
   class
"sclass"


$OOB
[1] FALSE

$comb
[1] FALSE

$call
bagging.data.frame(formula = Cancer ~ Age + First_sexual_intercourse +
    Smokes + Number_of_sexual_partners + Hormonal_Contraceptives +
    IUD + STDs, data = dataCancer)

attr(,"class")
[1] "summary.bagging"
> dataCancer$pred_bagging<-predict(model7,x)
> mtab<-table(dataCancer$pred_bagging,dataCancer$Cancer)
> confusionMatrix(mtab)
Confusion Matrix and Statistics


      0   1
  0 640  19
```

```
  1   2  76

               Accuracy : 0.9715
                 95% CI : (0.9568, 0.9823)
    No Information Rate : 0.8711
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8626
 Mcnemar's Test P-Value : 0.0004803

            Sensitivity : 0.9969
            Specificity : 0.8000
         Pos Pred Value : 0.9712
         Neg Pred Value : 0.9744
             Prevalence : 0.8711
         Detection Rate : 0.8684
   Detection Prevalence : 0.8942
      Balanced Accuracy : 0.8984

       'Positive' Class : 0


> #RANDOM FOREST

> library(randomForest)
> model8<-randomForest(Cancer~Age+First_sexual_intercourse+Smokes+Number_o
f_sexual_partners+Hormonal_Contraceptives+IUD+STDs,data=dataCancer)
> summary(model8)
> dataCancer$pred_randomforest<-predict(model8,x)
> mtab<-table(dataCancer$pred_randomforest,dataCancer$Cancer)
> confusionMatrix(mtab)
Confusion Matrix and Statistics


      0   1
  0 640  19
  1   2  76

               Accuracy : 0.9715
                 95% CI : (0.9568, 0.9823)
    No Information Rate : 0.8711
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8626
 Mcnemar's Test P-Value : 0.0004803

            Sensitivity : 0.9969
            Specificity : 0.8000
         Pos Pred Value : 0.9712
         Neg Pred Value : 0.9744
             Prevalence : 0.8711
         Detection Rate : 0.8684
   Detection Prevalence : 0.8942
      Balanced Accuracy : 0.8984

       'Positive' Class : 0

> #BOOSTING

> library(gbm)
```

```
> model9<-gbm(Cancer~Age+First_sexual_intercourse+Smokes+Number_of_sexual_
partners+Hormonal_Contraceptives+IUD+STDs,data=dataCancer,distribution="mu
ltinomial")
> summary(model9)
                                                var   rel.inf
Age                                             Age 42.716009
First_sexual_intercourse   First_sexual_intercourse 15.449944
STDs                                           STDs 10.369832
Number_of_sexual_partners Number_of_sexual_partners 10.180619
Smokes                                       Smokes  8.195943
Hormonal_Contraceptives     Hormonal_Contraceptives  6.558022
IUD                                             IUD  6.529631
> probability<-predict(model9,x,n.trees=1)
> dataCancer$pred_gbm<-colnames(probability)[apply(probability,1,which.max
)]
> mtab<-table(dataCancer$pred_gbm,dataCancer$Cancer)
> mtab

      0   1
  0 642  95
> confusionMatrix(mtab)


> #ADABOOST

> library(C50)
> model10<-C5.0(Cancer~Age+First_sexual_intercourse+Smokes+Number_of_sexua
l_partners+Hormonal_Contraceptives+IUD+STDs,data=dataCancer,trials=10)
> summary(model10)

Call:
C5.0.formula(formula = Cancer ~ Age + First_sexual_intercourse + Smokes +
Number_of_sexual_partners
 + Hormonal_Contraceptives + IUD + STDs, data = dataCancer, trials = 10)


C5.0 [Release 2.07 GPL Edition]       Wed Mar 27 21:33:39 2019
-------------------------------

Class specified by attribute `outcome'

Read 737 cases (8 attributes) from undefined.data

-----  Trial 0:  -----

Decision tree:
 0 (737/95)

-----  Trial 1:  -----

Decision tree:
 0 (737/231.7)

*** boosting reduced to 1 trial since last classifier is very inaccurate

*** boosting abandoned (too few classifiers)


Evaluation on training data (737 cases):

           Decision Tree
         ----------------
```

```
          Size      Errors

             1   95(12.9%)   <<


           (a)   (b)    <-classified as
          ----  ----
           642          (a): class 0
            95          (b): class 1


Time: 0.0 secs

> dataCancer$pred_c50<-predict(model10,x)
> mtab<-table(dataCancer$pred_c50,dataCancer$Cancer)
> confusionMatrix(mtab)
Confusion Matrix and Statistics


      0   1
  0 642  95
  1   0   0

               Accuracy : 0.8711
                 95% CI : (0.8447, 0.8944)
    No Information Rate : 0.8711
    P-Value [Acc > NIR] : 0.5273

                  Kappa : 0
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 1.0000
            Specificity : 0.0000
         Pos Pred Value : 0.8711
         Neg Pred Value :    NaN
             Prevalence : 0.8711
         Detection Rate : 0.8711
   Detection Prevalence : 1.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 0
```

PYTHON CODE AND RESULTS:

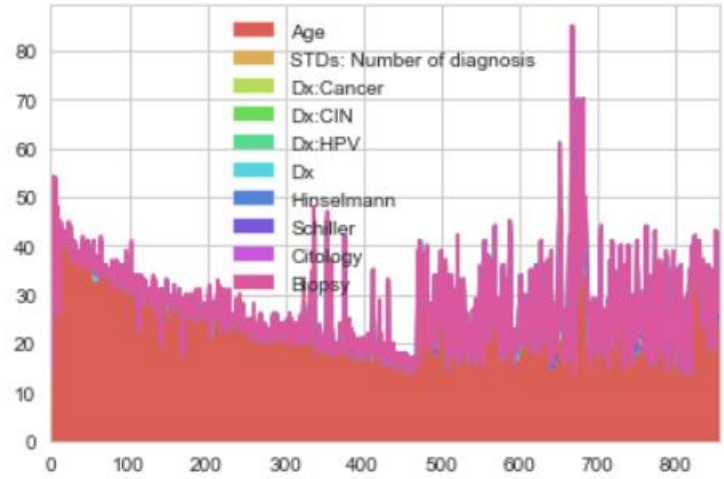
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import scikitplot as skplt

from sklearn.model_selection import train_test_split

from sklearn import preprocessing

from sklearn.preprocessing import Imputer
```

```
from pandas.plotting import scatter_matrix
from sklearn import svm

%matplotlib inline

seed = 42

df=pd.read_csv("kag_risk_factors_cervical_cancer.csv")
len(df)
```

858

```
df.plot.area()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x123cfcb70>
```




```
#Replace ?s with -1 to represent missing values
df=df.replace("?", 0)
# imp = Imputer(missing_values="?", strategy='most_frequent', axis=0)
```

```
# imp.fit(df)

# df = imp.transform(df)

#Split data into training and testing setsy=df["Biopsy"]X=df.drop(["Biopsy"], axis=1)#20% of dataset goes to test setX_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2, random_state=seed)

#Standardize the data

X_scaled = preprocessing.scale(X_train)

X_scaled.mean()
```

```
-1.4389896095932891e-17
```

```
X_scaled.std()
```

```
0.9561828874675149
```

```
#Train the random forest classifier using training data

from sklearn.ensemble import RandomForestClassifier

rfc=RandomForestClassifier(random_state=seed)

rfc.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',

            max_depth=None, max_features='auto', max_leaf_nodes=None,

            min_impurity_split=1e-07, min_samples_leaf=1,

            min_samples_split=2, min_weight_fraction_leaf=0.0,

            n_estimators=10, n_jobs=1, oob_score=False, random_state=42,

            verbose=0, warm_start=False)
```

```
#Evaluating the classifier using training set

from sklearn.metrics import accuracy_score

y_pred=rfc.predict(X_test)

accuracy_score(y_pred, y_test)
```

0.94186046511627908

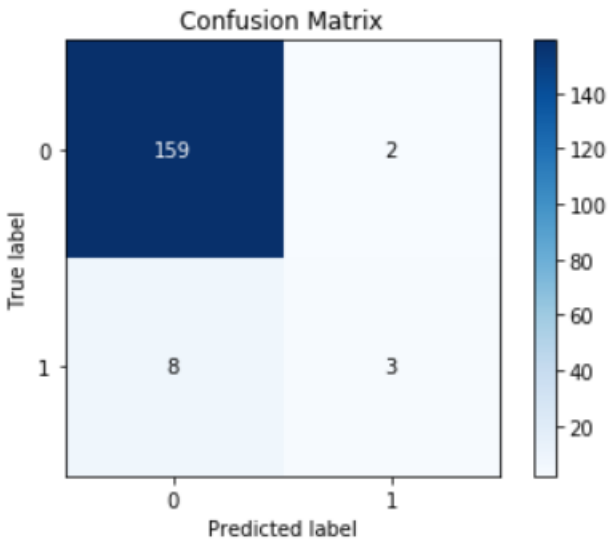
```
#Evaulating which features the classifier finds important for making its decisions
feats = rfc.feature_importances_
#Create new instance of dataframe
feat_importances=pd.DataFrame()
#set columns in datafram to features and their importances
feat_importances["feature"]=X.columns
feat_importances["rfc"]=feats
#Display data
feat_importances
```

feature	rfc	
0	Age	0.108359
1	Number of sexual partners	0.043818
2	First sexual intercourse	0.090490
3	Num of pregnancies	0.043802
4	Smokes	0.000732
5	Smokes (years)	0.008442
6	Smokes (packs/year)	0.012189
7	Hormonal Contraceptives	0.018789
8	Hormonal Contraceptives (years)	0.073255
9	IUD	0.007286
10	IUD (years)	0.014674
11	STDs	0.002097
12	STDs (number)	0.004735
13	STDs:condylomatosis	0.002229

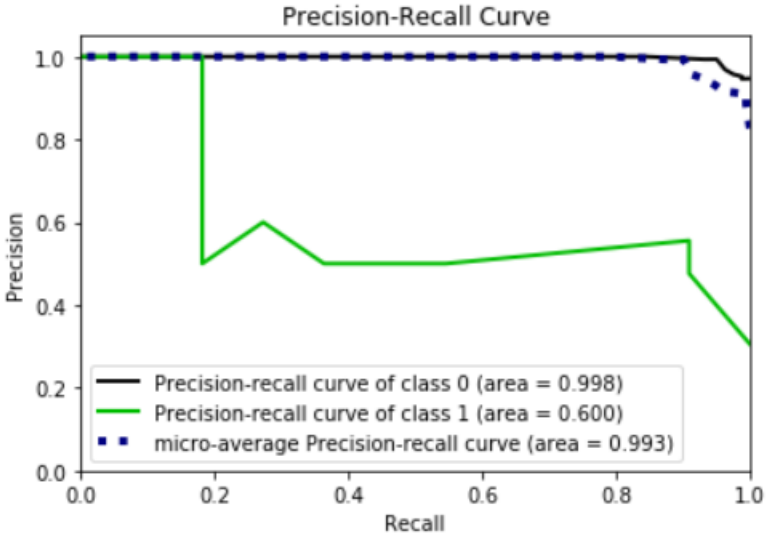
14	STDs:cervical condylomatosis	0.000000
15	STDs:vaginal condylomatosis	0.000000
16	STDs:vulvo-perineal condylomatosis	0.005680
17	STDs:syphilis	0.000113
18	STDs:pelvic inflammatory disease	0.000000
19	STDs:genital herpes	0.029498
20	STDs:molluscum contagiosum	0.000000
21	STDs:AIDS	0.000000
22	STDs:HIV	0.005334
23	STDs:Hepatitis B	0.000000
24	STDs:HPV	0.000068
25	STDs: Number of diagnosis	0.002572
26	STDs: Time since first diagnosis	0.013310
27	STDs: Time since last diagnosis	0.009971
28	Dx:Cancer	0.003834

29	Dx:CIN	0.015490
30	Dx:HPV	0.014051
31	Dx	0.015054
32	Hinselmann	0.123566
33	Schiller	0.278592
34	Citology	0.051970

```
skplt.metrics.plot_confusion_matrix(y_true=y_test, y_pred=y_pred)
plt.show()
```




```
np.sum(y_test.values==1), np.sum(y_test.values==0)
skplt.metrics.plot_precision_recall_curve(y_true=y_test, y_probas=rfc.predict_proba(X_test))
plt.show()
```




```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

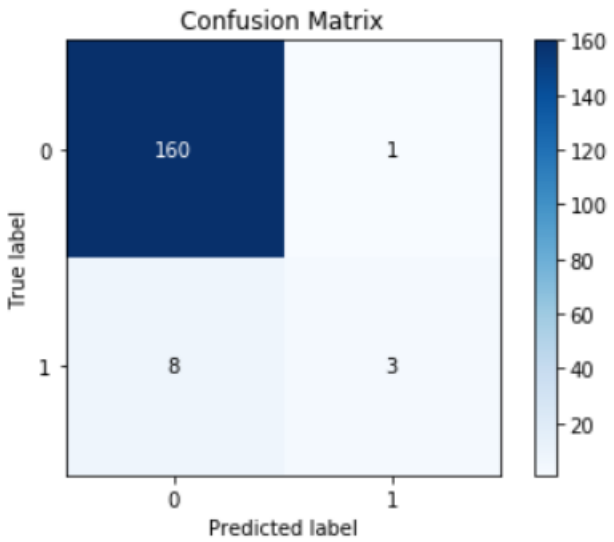
```
             precision    recall  f1-score   support

          0       0.95      0.99      0.97       161
          1       0.60      0.27      0.37        11

avg / total       0.93      0.94      0.93       172
```

```
#Train the random forest classifier using training data

from sklearn.ensemble import RandomForestClassifier

rfcw=RandomForestClassifier(random_state=seed, class_weight={0:20, 1:0.5})

rfcw.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight={0: 20, 1: 0.5},
        criterion='gini', max_depth=None, max_features='auto',
        max_leaf_nodes=None, min_impurity_split=1e-07,
        min_samples_leaf=1, min_samples_split=2,
        min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
        oob_score=False, random_state=42, verbose=0, warm_start=False)
```

```
rfcw_pred = rfcw.predict(X_test)
print(classification_report(y_test, rfcw_pred))
skplt.metrics.plot_confusion_matrix(y_true=y_test, y_pred=rfcw_pred)
plt.show()
```

```
             precision    recall  f1-score   support

          0       0.95      0.99      0.97       161
          1       0.75      0.27      0.40        11

avg / total       0.94      0.95      0.94       172
```




```
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier, VotingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
import numpy as np

classifiers = [
    KNeighborsClassifier(3),
    SVC(kernel="linear", C=0.025),
```

```
    SVC(gamma=2, C=1),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1, random_state=42),
    MLPClassifier(alpha=1),
    GradientBoostingClassifier(random_state=42)
    ]

names = ["KNC", "Linear SVC", "SVC", "RFC", "MLP", "GBC"]



clfs = list(zip(classifiers, names))

for clf, n in clfs:
    clf.fit(X_train, y_train)
    preds = np.rint(clf.predict(X_test))
    print(n, accuracy_score(y_test, preds))

vclf = VotingClassifier(estimators=clfs, voting="hard")
```

```
KNC 0.93023255814
Linear SVC 0.936046511628
SVC 0.936046511628
RFC 0.941860465116
MLP 0.953488372093
GBC 0.953488372093
```

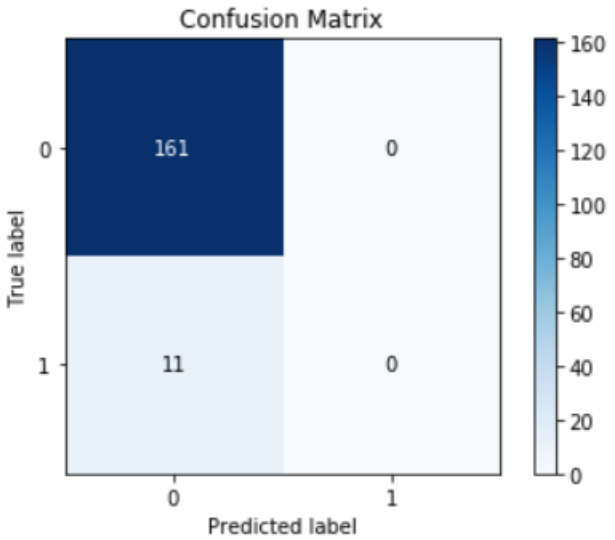
```
from sklearn.decomposition import PCA
from sklearn.gaussian_process.kernels import Product
from itertools import product

pca = PCA(n_components=2)
```

```
pca.fit(X_train)

X_train2d = pca.transform(X_train)

# Plotting decision regions
x_min, x_max = X_train2d[:, 0].min() - 1, X_train2d[:, 0].max() + 1
y_min, y_max = X_train2d[:, 1].min() - 1, X_train2d[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

f, axarr = plt.subplots(2, 2, sharex='col', sharey='row', figsize=(10, 8))

for idx, clf, tt in zip(product([0, 1], [0, 1]),
                        classifiers,
                        names):

    clf.fit(X_train2d, y)
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    axarr[idx[0], idx[1]].contourf(xx, yy, Z, alpha=0.4)
    axarr[idx[0], idx[1]].scatter(X[:, 0], X[:, 1], c=y,
                                  s=20, edgecolor='k')
    axarr[idx[0], idx[1]].set_title(tt)

plt.show()

svmc=svm.SVC()
svmc.fit(X_train, y_train)

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
```

```
 decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
 max_iter=-1, probability=False, random_state=None, shrinking=True,
 tol=0.001, verbose=False)
```

```
svmc_pred = svmc.predict(X_test)
print(classification_report(y_test, svmc_pred))
skplt.metrics.plot_confusion_matrix(y_true=y_test, y_pred=svmc_pred)
plt.show()#Evaluating the classifier using training set
from sklearn.metrics import accuracy_score
y_pred=rfc.predict(X_test)
accuracy_score(y_pred, y_test)
```

```
             precision    recall  f1-score   support

          0       0.94      1.00      0.97       161
          1       0.00      0.00      0.00        11

avg / total       0.88      0.94      0.91       172
```




```
0.94186046511627908
```

```
from sklearn.metrics import accuracy_score
y_pred=rfc.predict(X_test)
accuracy_score(y_pred, y_test)
```

0.94186046511627908

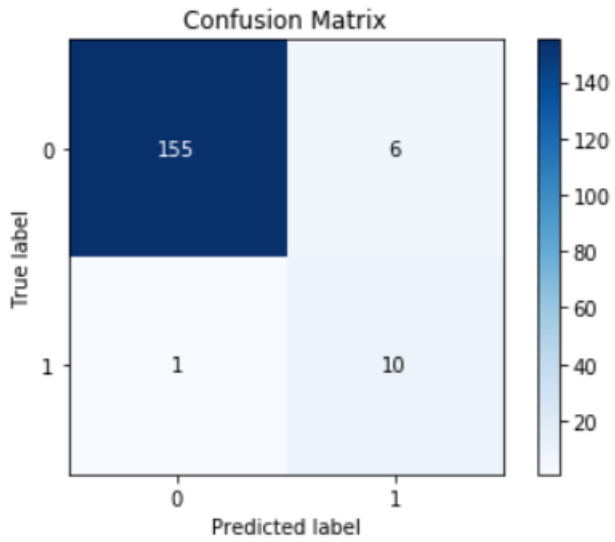
```
def make_meshgrid(x, y, h=.02):
    """Create a mesh of points to plot in

    Parameters
    ----------
    x: data to base x-axis meshgrid on
    y: data to base y-axis meshgrid on
    h: stepsize for meshgrid, optional

    Returns
    -------
    xx, yy : ndarray
    """
    x_min, x_max = x.min() - 1, x.max() + 1
    y_min, y_max = y.min() - 1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    return xx, yy


def plot_contours(ax, clf, xx, yy, **params):
    """Plot the decision boundaries for a classifier.

    Parameters
    ----------
    ax: matplotlib axes object
    clf: a classifier
    xx: meshgrid ndarray
```

```
    yy: meshgrid ndarray
    params: dictionary of params to pass to contourf, optional
    """
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    out = ax.contourf(xx, yy, Z, **params)
    return out



# we create an instance of SVM and fit out data. We do not scale our
# data since we want to plot the support vectors
C = 1.0  # SVM regularization parameter
models = (svm.SVC(kernel='linear', C=C),
          svm.LinearSVC(C=C),
          svm.SVC(kernel='rbf', gamma=0.7, C=C),
          svm.SVC(kernel='poly', degree=3, C=C))
models = (clf.fit(X_train, y_train) for clf in models)

# title for the plots
titles = ('SVC with linear kernel',
          'LinearSVC (linear kernel)',
          'SVC with RBF kernel',
          'SVC with polynomial (degree 3) kernel')

# Set-up 2x2 grid for plotting.
fig, sub = plt.subplots(2, 2)
plt.subplots_adjust(wspace=0.4, hspace=0.4)

X0, X1 = X_train[:, 0], X_train[:, 1]
xx, yy = make_meshgrid(X0, X1)
```

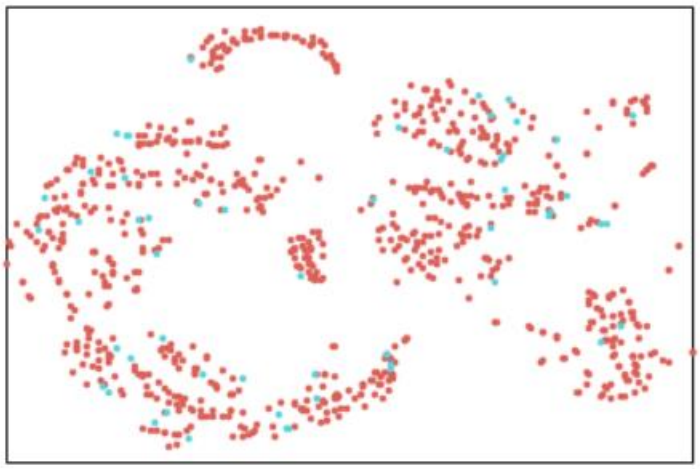
```
for clf, title, ax in zip(models, titles, sub.flatten()):
    plot_contours(ax, clf, xx, yy,
                  cmap=plt.cm.coolwarm, alpha=0.8)
    ax.scatter(X0, X1, c=y_test, cmap=plt.cm.coolwarm, s=20, edgecolors='k')
    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xlabel('Sepal length')
    ax.set_ylabel('Sepal width')
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(title)

plt.show()

from sklearn.ensemble import GradientBoostingClassifier
gbc = GradientBoostingClassifier(max_depth=1)
gbc.fit(X_train, y_train)

GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=1,
              max_features=None, max_leaf_nodes=None,
              min_impurity_split=1e-07, min_samples_leaf=1,
              min_samples_split=2, min_weight_fraction_leaf=0.0,
              n_estimators=100, presort='auto', random_state=None,
              subsample=1.0, verbose=0, warm_start=False)

from sklearn.metrics import accuracy_score
gbcy_pred=gbc.predict(X_test)
accuracy_score(gbcy_pred, y_test)
```

0.95930232558139539

```
skplt.metrics.plot_confusion_matrix(y_true=y_test, y_pred=gbcy_pred)
plt.show()
```




```
from sklearn import datasets
import hypertools as hyp

hyp.plot(X, '.', reduce='TSNE', group=y, ndims=2)
# hyp.plot(X, '.', reduce='TSNE', n_clusters=10, ndims=2)
```



DATASET: CHRONIC KIDNEY DISEASE

RCODE AND RESULTS:

PYTHON CODE AND RESULTS:

```
import pandas as pd
import numpy as np
from IPython.display import display
from scipy.stats import mode
import matplotlib.pyplot as plt

csv_file = "/content/kidney_disease.csv"
data = pd.read_csv(csv_file)

#No of people with chronic kidney disease
n_ckd = len(data[data['classification']=='ckd'])

#No of people without chronic kidney disease
n_notckd = len(data[data['classification']=='notckd'])
```

```
#Filling missing value with most frequent for nominal and median for numerical


X = pd.DataFrame(data)
fill = pd.Series([X[c].value_counts().index[0]
        if X[c].dtype == np.dtype('O') else X[c].median() for c in X],
        index=X.columns)
new_data=X.fillna(fill)
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
data=new_data.copy()
for items in data:
    if data[items].dtype == np.dtype('O'):
        data[items]=le.fit_transform(data[items])


from sklearn.preprocessing import StandardScaler, RobustScaler
target_class = data['classification']
features = data.drop('classification', axis = 1)
data_robust = pd.DataFrame(RobustScaler().fit_transform(features), columns=features.columns)
from sklearn.decomposition import PCA
pca = PCA()
data_pca = pd.DataFrame(pca.fit_transform(data_robust), columns=data_robust.columns)
pca = PCA(n_components=11)
pca.fit(data_robust)
reduced_data = pca.transform(data_robust)
reduced_data = pd.DataFrame(reduced_data, columns =
['dim1','dim2','dim3','dim4','dim5','dim6','dim7','dim8','dim9','dim10','dim11'])
from sklearn.model_selection import train_test_split
XTrain, XTest, yTrain, yTest = train_test_split(reduced_data,target_class, test_size=0.25,
random_state=42)


from sklearn.metrics import confusion_matrix, accuracy_score,r2_score,mean_squared_error
from sklearn import tree
```

```
from sklearn import linear_model
from sklearn import svm
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
```

Logistic Regression

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
r2 = r2_score(yTest,yPred)
rmse = mean_squared_error(yTest,yPred)
print"Logistic Regression :"
print"Accuracy = ", accuracy
print"R2 = ",r2
print"RMSE = ",rmse
print cm
```

Output:

```
Logistic Regression :
```

```
Accuracy = 0.99
R2 = 0.9560439560439561
RMSE = 0.04
[[64 1]
 [ 0 35]]
```

K Nearest Neighbors

```
#K Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5, p=2,
metric='minkowski')
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print"K Nearest Neighbors :"
print"Accuracy = ", accuracy
print cm
```

output:

```
K Nearest Neighbors :
Accuracy = 0.97
[[62 3]
 [ 0 35]]
```

Support Vector Machine

```
#Support Vector Machine
from sklearn.svm import SVC
classifier = SVC(kernel='linear',random_state=0)
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
```

```
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print "Support Vector Machine :"
print "Accuracy = ", accuracy
print cm
```

```
Support Vector Machine :
Accuracy = 1.0
[[65 0]
 [ 0 35]]
```

Gaussian Naive Bayes

```
#Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print"Gaussian Naive Bayes :"
print"Accuracy = ", accuracy
print cm
```

```
Output:
Gaussian Naive Bayes :
Accuracy = 0.96
[[61 4]
 [ 0 35]]
```

Decision Tree Classifier

```
#Decision Tree Classifier
```

```
from sklearn.tree import DecisionTreeClassifier as DT
classifier = DT(criterion='entropy', random_state=0)
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print "Decision Tree Classifier :"
print "Accuracy = ", accuracy
print cm
```

```
Decision Tree Classifier :
Accuracy = 0.93
[[63 2]
 [ 5 30]]
```

```
#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier as RF
classifier = RF(n_estimators=10, criterion='entropy', random_state=0)
classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print"Random Forest Classifier :"
print"Accuracy = ", accuracy
print cm
```

```
Output:
Random Forest Classifier :
Accuracy = 0.97
[[64 1]
 [ 2 33]]
```

Artificial Neural Network

```
classifier = MLPClassifier()
classifier.set_params(hidden_layer_sizes =(100,100), max_iter =
1000,alpha = 0.01, momentum = 0.7)
classifier = classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
r2 = r2_score(yTest,yPred)
rmse = mean_squared_error(yTest,yPred)
print"Artificial Neural Network :"
print"Accuracy = ", accuracy
print"R2 = ",r2
print"RMSE = ",rmse
print cm
```

Output:

```
Artificial Neural Network :
Accuracy = 1.0
R2 = 1.0
RMSE = 0.0
[[65 0]
 [ 0 35]]
```

Bagging

```
classifier = BaggingClassifier()
#classifier.set_params(n_estimators = 30,max_samples = 1000)
classifier = classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
r2 = r2_score(yTest,yPred)
rmse = mean_squared_error(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
```

```
print"Bagging :"
print"Accuracy = ", accuracy
print"R2 = ",r2
print"RMSE = ",rmse
print cm
```

Output:

```
Bagging :
Accuracy = 0.94
R2 = 0.7362637362637362
RMSE = 0.24
[[64 1]
 [ 5 30]]
```

Gradient Boosting

```
classifier = GradientBoostingClassifier()
classifier.set_params(n_estimators = 30,learning_rate = 1)
classifier = classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
r2 = r2_score(yTest,yPred)
rmse = mean_squared_error(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print"Gradient Boosting :"
print"Accuracy = ", accuracy
print"R2 = ",r2
print"RMSE = ",rmse
print cm
```

Output:

```
Gradient Boosting :
```

```
Accuracy = 0.96
R2 = 0.8241758241758241
RMSE = 0.16
[[64 1]
 [ 3 32]]
```

AdaBoost

```
classifier = AdaBoostClassifier()
classifier.set_params(n_estimators = 10, learning_rate = 1)
classifier = classifier.fit(XTrain,yTrain)
yPred = classifier.predict(XTest)
cm = confusion_matrix(yTest,yPred)
r2 = r2_score(yTest,yPred)
rmse = mean_squared_error(yTest,yPred)
accuracy = accuracy_score(yTest,yPred)
print"AdaBoost :"
print"Accuracy = ", accuracy
print"R2 = ",r2
print"RMSE = ",rmse
print cm
```

Output:

```
AdaBoost :
Accuracy = 0.82
R2 = 0.20879120879120872
RMSE = 0.72
[[65 0]
 [18 17]]
```