## Introduction:

Breast cancer is the most common cancer among women ,and one of the major causes of death among women worldwide. Every year approximately 124 out of 100,000 women are diagnosed with breast cancer, and the estimation is that 23 out of the 124 women will die of this disease. Breast Cancer occurs as a results of abnormal growth of cells in the breast tissue, commonly referred to as a tumor. A tumor does not mean cancer - tumors can be benign (not cancerous), pre-malignant (pre-cancerous), or malignant (cancerous) .Tests such as MRI, mammogram, ultrasound and biopsy are commonly used to diagnose breast cancer.

The use of machine learning and artificial intelligence techniques has revolutionized the process of diagnosis the breast cancer ,Which helped detect it early so that the chance of treatment is greater and An automated screening system might also provide greater detection accuracy by removing the inherently subjective human component from the process.

In this project we used three types of machine learning :Knn, Naïve Bayes, and Decision Tree to predict if a tumor is cancerous or not using 31 feature then we made a comparison on performance of methods to select the more accurate methods.

## Methodology:

### 1.kNN:

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

**First: pre-processing** : We started by importing data and checking that there is no missing

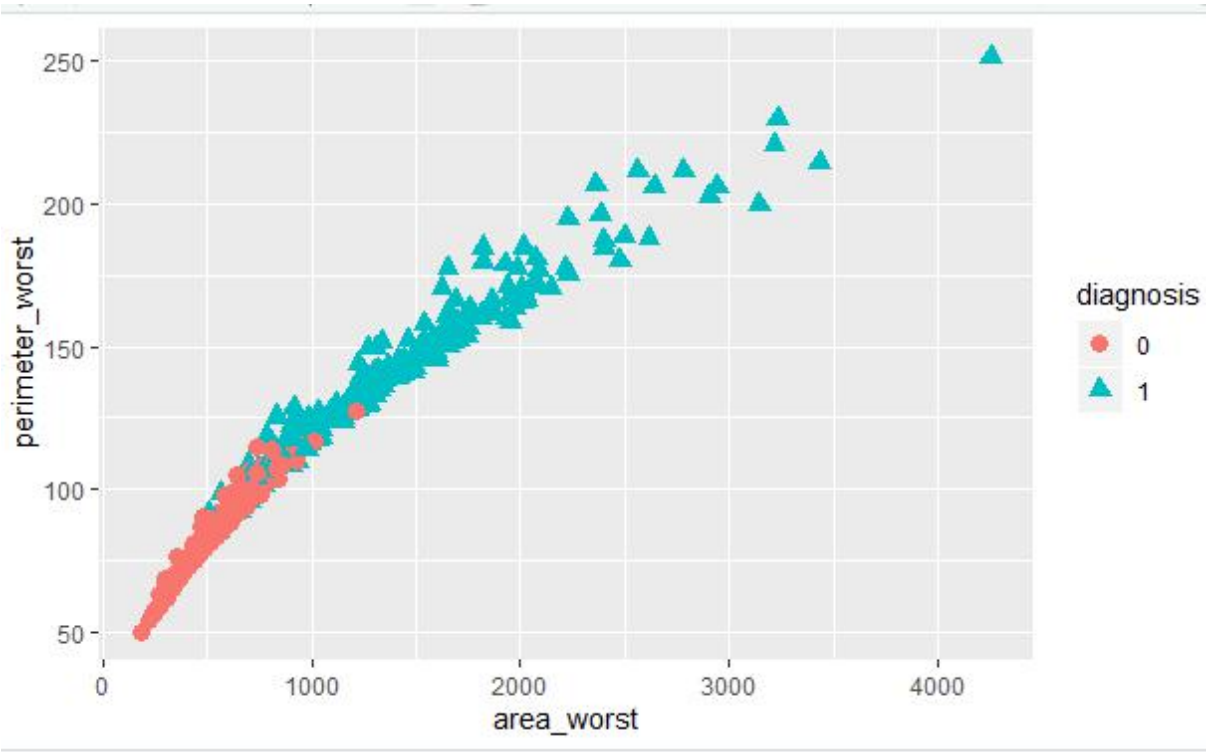
```
#importing library
library(readxl)
library(caTools)
library(class)
#importing data
data <-  read_excel("wbcd.xlsx")
str(data)
data<-data[,-1] #remove ID col
#missing data
#no missing data
#categrical data
data$diagnosis<- factor(x=data$diagnosis,levels =c("B","M"),labels = c(0,1))
```

values,then we make 0 stands for benign and 1 stands for malignant tumors

After that we made feature slection to choose most effective features  using library ”FSelector” that rank features by some criteria and select the ones that are above a defined threshold and then, search for optimum feature subsets from a space of feature

```
#feature selection
library(FSelector)
library(ggplot2)
f<-information.gain(diagnosis~., data)
print(f)
subset <- cutoff.k(f, 2)
subset
ggplot(data, aes( area_worst,perimeter_worst, col=diagnosis) ) +
  geom_point(size = 3, aes(pch = diagnosis))
```

subsets.




We used “ggplot2” to  plot of feature selection result as shown :

The last thing we did in pre-processing is scaling data(normalising data):

```
#scaling data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
data_nor<-as.data.frame(lapply(data[2:31], normalize))
data<-cbind(data[,1],data_nor)
#data[,2:31]<-scale(data[,2:31])
```

**Second:appling knn and k-folded cv:** we used for loop to randomly split the data k times to use all data for training and all data for testing .

```
set.seed(123)   #so that same sample can be reproduced in future
sample<- sample.split(data,SplitRatio = 0.8)
data_train<-subset(data,sample==TRUE)        #select 80% of data as a train sample
data_test<-subset(data,sample==FALSE)#select 20% of data as a test sample
#knn
y_pred <- knn(train = data_train[,-1],test = data_test[,-1],cl=data_train$diagnosis,k=23)
#confusion matrix
cm<-table(data_test$diagnosis,y_pred)
#evaluation
ACC<-sum(diag(cm))/sum(cm)
SP<-cm[1,1]/sum(cm[,1])
SN<-cm[2,2]/sum(cm[,2])
#k-fold cross validation
folds<-cut(seq(1,nrow(data)),breaks=10,labels = FALSE) #split the data into 10 sections (k=10)
for(i in 1:10){
  CV_test_data<-data[folds==i,]
  CV_train_data<-data[folds!=i,]
  CV_pred<-knn(train =CV_train_data[,-1] ,test =CV_test_data[,-1],cl=CV_train_data$diagnosis,k=23)
  CV_cm<-table(CV_test_data$diagnosis,CV_pred)
  print(CV_cm)
  CV_ACC[i]<-sum(diag(CV_cm))/sum(CV_cm)
  CV_SP[i]<-cm[1,1]/sum(cm[,1])
  CV_SN[i]<-cm[2,2]/sum(cm[,2])
}
mean(CV_ACC)
mean(CV_SP)
mean(CV_SN)
```

# 2.Naive Bayes:

Naive Bayes models are a special kind of classification machine learning algorithms. They are based on a statistical classification technique called ‘Bayes Theorem’. Naive Bayes model are called ‘naive’ algorithms because they make an assumption that the predictor variables are independent from each other. In other words, that the presence of a certain feature in a dataset is completely unrelated to the presence of any other feature.They provide an easy way to build accurate models with very good performance given their simplicity.It assumes that features follow a normal distribution and Gaussian carve.

**First: pre-processing** : We started by importing data and checking that there is no missing values, then we make 0 stands for benign and 1 stands for malignant tumors.

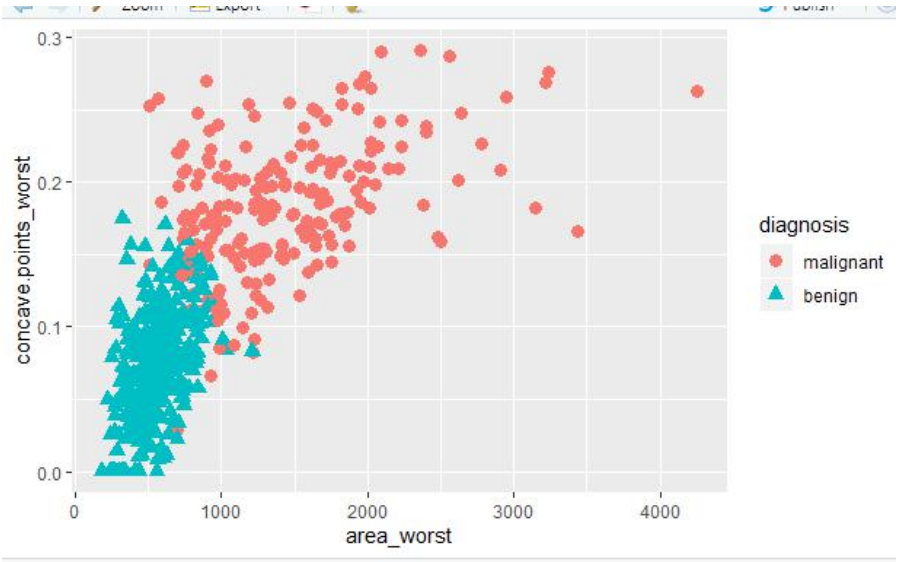
```
data<- read.csv("C:/Users/user1/Desktop/prototype/data-set.csv", sep=";")
data<-data
str(data)
x<- is.na(data)#check if there's missing value


for (i in x){
  if(i ==TRUE){
    na.omit(x)
  }
}

data <- data[,-1] #remove 1st column(id number)
data[,1] <- factor(data[,1], levels=c('M' , 'B'), labels=c("malignant", "benign"))
```

After that we made feature slection to choose most effective features  using library ”Bor uta” that do feature selection following these steps :

- firstly, extending information system
- added attributes are shuffled randomly in order to remove any corellation with the response variable
- random forest classifier is run on the whole data set and Z-scores are computed for all attributes
- out of all shadow attributes find the one with the maximum Z score and then assign a hit to every attribute that scored better than the one with maximum Z-score
- for each attribute with undetermined importance perform a two-sided test of equality with the the one obtained for shadow attribute with maximum Z-score
- mark the attributes which have importance signifcantly lower than the shadow with maximum Z-score as `unimportant’ and permanently remove them from the dat aset
- remove all shadow, artificially added attributes
- repeat the procedure until the importance is assigned for all the attributes, or the algorithm has reached the previously set limit of the random forest runs.

```
#feature selection
library(Boruta)
library(ggplot2)
set.seed(1234)
bor<-Boruta(diagnosis~.,data = data,doTrace=1)
print(bor)
plot(bor,las=2)
borr<-TentativeRoughFix(bor)
print(borr)
attStats(borr)
ggplot(data, aes(area_worst, concave.points_worst, col=diagnosis) ) +
  geom_point(size = 3, aes(pch = diagnosis))
```

We used “ggplot2” to  plot of feature selection result as shown :



**Second:appling naïve bayes and k-folded cv:** we used library "e1071" for naïve bayes .Also, we used for loop to randomly split the data k times to use all data for training and all data for testing .

```
library(caret)
k<- 10
folds <- cut(seq(1,nrow(data)), breaks = k,labels = F)
folds
library(e1071)
trainAcc <-0
testAcc <-0
SN_train<-0
SP_train<-0
SN_test<-0
SP_test<-0
for(i in 1:k){
  test<- data[folds==i,]
  train <- data[folds !=i,]
  model1<-naiveBayes(diagnosis~. ,data=train , type = c("class"))
  trainTable=table(predict(  model1,newdata = train, type = "class"),train$diagnosis)
  testTable=table(test$diagnosis,predict(model1, newdata=test, type="class"))
  trainAcc[i] <-sum(diag(trainTable))/sum(trainTable)
  SN_train[i]<-trainTable[1,1]/sum(trainTable[,1])
  SP_train[i]<-tab[2,2]/sum(tab[,2])
  testAcc[i]<-sum(diag(testTable))/sum(testTable)
  SN_test[i]<-testTable[1,1]/sum(testTable[,1])
  SP_test[i]<-testTable[2,2]/sum(testTable[,2])
}
#get average
mean( trainAcc)
mean(SN_train)
mean(SP_train)
mean(testAcc)
mean(SN_test)
mean(SP_test)
```

## 3.Decision Tree :

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. This algorithm is recursive binary Splitting in nature as the groups formed can be sub-divided using same strategy. Due to this procedure, this algorithm is also known as the greedy search, as it has an excessive desire of getting more pure data and information gain by calculating entropy before and after filtration of data. This makes the root node as best predictor/classifier. The decision tree is considered the easiest models to deal with binary classification, because it solves the problem of over fitting by making more than one model and comparing among them in the end.

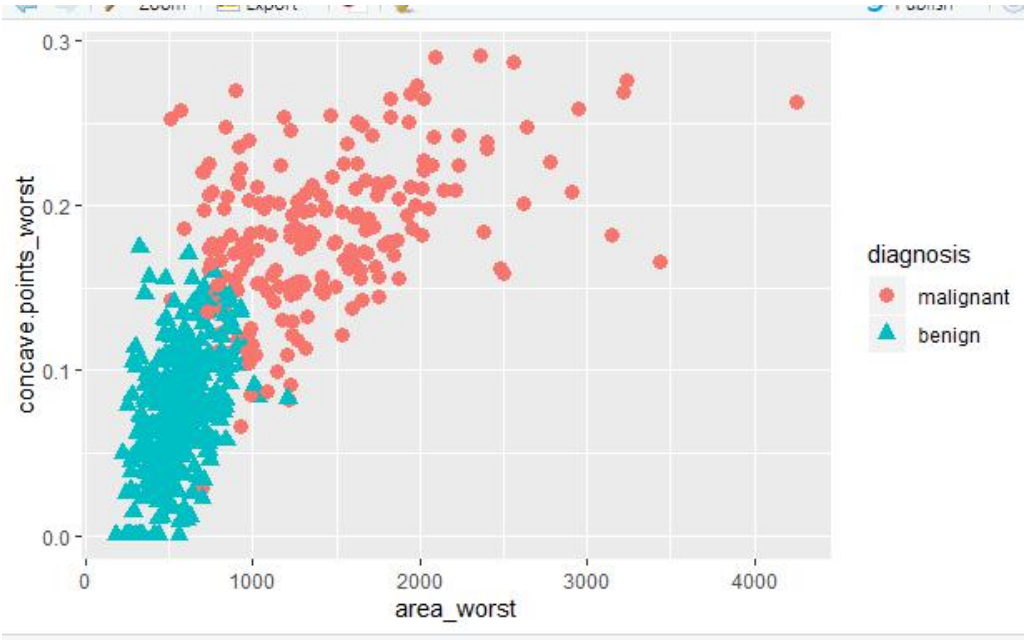
**First: pre-processing** : We started by importing data and checking that there is no missing values,then we make 0 stands for benign and 1 stands for malignant tumors.

```
data.set <- read.csv("C:/Users/user1/Desktop/prototype/data-set.csv", sep=";")
#prepare data
data<-data.set

x<- is.na(data)#check if there's missing value
for (i in x){
  if(i ==TRUE){
    na.omit(x)
  }
}

data <- data[,-1] #remove 1st column(id number)

#categorical encoding
data[,1] <- factor(data[,1], levels=c('M' , 'B'), labels=c("malignant", "benign"))
print(summary(data))
```

After that we made feature slection to choose most effective features  using library ”Bor uta” (explained above).

```
#feature selection
library(Boruta)
library(ggplot2)
set.seed(1234)
bor<-Boruta(diagnosis~.,data = data,doTrace=1)
print(bor)
plot(bor,las=2)
borr<-TentativeRoughFix(bor)
print(borr)
attStats(borr)
ggplot(data, aes(area_worst, concave.points_worst, col=diagnosis) ) +
  geom_point(size = 3, aes(pch = diagnosis))
```

We used “ggplot2” to  plot of feature selection result as shown :



**Second:appling Decision Tree and k-folded cv:** we used library “rpart” for Decision Tree .Also, we used for loop to randomly split the data  k times to use all data for training and all data for testing.

```
# Decision tree with k-fold CV :
library(rpart)
k<- 10
folds <- cut(seq(1,nrow(data)), breaks = k,labels = F)
folds
accu_train<-0
SN_train<-0
SP_train<-0
accu_test<-0
SN_test<-0
SP_test<-0
for(i in 1:k){
test<- data[folds==i,]
train <- data[folds!=i,]
model<- rpart(diagnosis~.,data=train, method = "class")
tab<- table(predict(model,type = "class"),train$diagnosis)
print(tab)
accu_train[i]<-(sum(diag(tab))/sum(tab))
SN_train[i]<-(sum(tab[1,1])/sum(tab[,1]))
SP_train[i]<-(sum(tab[2,2])/sum(tab[,2]))
tested<- predict(model,type = "class",newdata=test)
tab_test<-table(tested,test$diagnosis)
print(tab_test)
accu_test[i]<-(sum(diag(tab_test))/sum(tab_test))
SN_test[i]<-(tab_test[1,1]/sum(tab_test[,1]))
SP_test[i]<-(tab_test[2,2]/sum(tab_test[,2]))
}
#get average
mean(accu_train)
mean(SN_train)
mean(SP_train)
mean(accu_test)
mean(SN_test)
mean(SP_test)
```

## Results :

we printed the mean of k-folded C.M, accuracy ,sensitivity ,and specifity  for each method as shown:

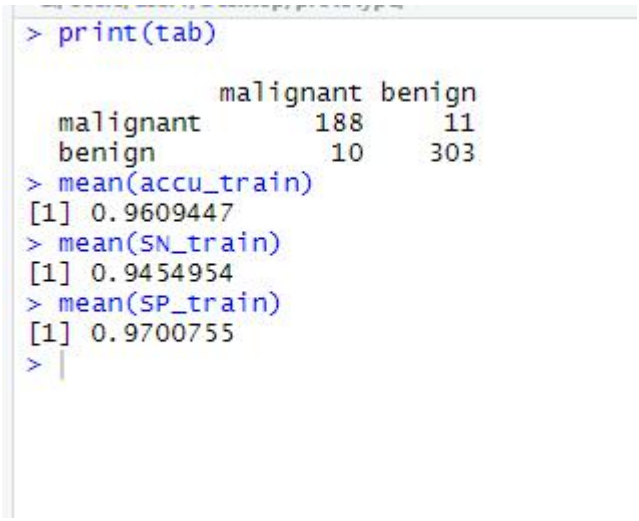
```
> print(CV_cm)
   CV_pred
     0  1
  0 43  0
  1  1 13
> mean(CV_ACC)
[1] 0.9578634
> mean(CV_SP)
[1] 0.923913
> mean(CV_SN)
[1] 0.9189189
>
```

1.kNN:

2.Naive Bayes:

```
> print(trainTable)

           malignant benign
  malignant       179     12
  benign           19    302
> mean( trainAcc)
[1] 0.93927
> mean(SN_train)
[1] 0.8988156
> mean(SP_train)
[1] 0.9649682
>
```

3.Decision Tree :




When we compare the results of the three models we found that Decision Tree did the best performance ,so Decision Tree is the most fit for our project.

# Source codes:

**https://github.com/sbme-tutorials/sbe304-fall19-project-19**

## refrences:

https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

https://s3.amazonaws.com/academia.edu.documents/42413734/Boruta_-_A_System_for_Feature_Selection20160208-14055-1dxlam0.pdf?response-content-disposition=inline%3B%20filename%3DBoruta_-_A_system_for_feature_selection.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20191204%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20191204T160223Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=95236ce302095877969ac95f89f8e4bd98c9eab40518e1edc0607537b7f7816f