Diagnosis of Breast Cancer

Saurabh Khodake
Saurabh Sharma
Piyush Narang
Gaurav Gola
Kallol Samanta
Pramit Patra

PROBLEM STATEMENT

To predict the type of tumor :

i) **Benign**(Non cancerous)

OR

ii) **Malignant**(Cancerous)

INTRODUCTION

- Breast cancer is most common form of cancer in Women
- It represents about 12% of all new cancer cases and 25% of all cancers in women
- Rates for breast cancer vary worldwide, depending on health care services in nations
- This study is based on tumor study from US state of Wisconsin

Data Obtained For STUDY

- 3-D digitized images of breast mass(lump) used for extracting information
- Real-valued features are computed for each cell nucleus
- Grouped into 3 types: Mean, Standard Error, Worst
- 32 total variables like:

  -Radius

  -Area

  -Smoothness etc.

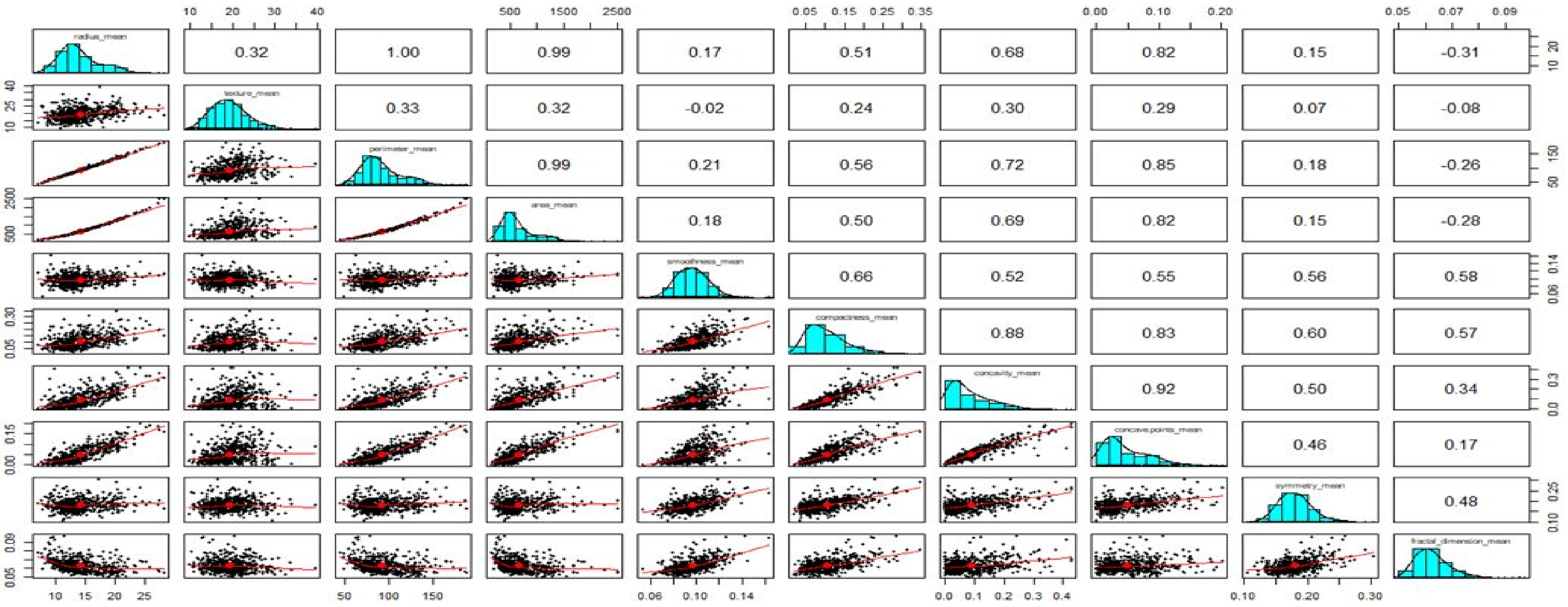
- To determine type of tumor i) Benign ii) Malignant

Exploratory Data Analysis

- 500+ records
- No missing values.
- Type of variables - Only Diagnosis Variable is “Categorical” Other 31 variables are “Numerical”.
- Data split into 70:30 ratio for training & testing resp.
- Since each variable has different variance, data is scaled

Collinearity Graph

Collinearity among 10 variables



Principal Component Analysis

We used PCA for:

- **Dimension Reduction**

a. To reduce dimensions with minimal loss of information
b. 66% reduction in features

- **To Reduce Multicollinearity**

a. To eliminate covariance among explanatory variables

PCA Implementation

- First 10 components accounts for 95% of total variance
- First 15 components accounts for 98% of total variance

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7     PC8    PC9
Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172 0.69037 0.6457
Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251 0.01589 0.0139
Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759 0.91010 0.92598 0.9399
                          PC10   PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18
Standard deviation     0.59219 0.5421 0.51104 0.49128 0.39624 0.30681 0.28260 0.24372 0.22939
Proportion of Variance 0.01169 0.0098 0.00871 0.00805 0.00523 0.00314 0.00266 0.00198 0.00175
Cumulative Proportion  0.95157 0.9614 0.97007 0.97812 0.98335 0.98649 0.98915 0.99113 0.99288
                          PC19    PC20   PC21    PC22    PC23   PC24    PC25    PC26    PC27
Standard deviation     0.22244 0.17652 0.1731 0.16565 0.15602 0.1344 0.12442 0.09043 0.08307
Proportion of Variance 0.00165 0.00104 0.0010 0.00091 0.00081 0.0006 0.00052 0.00027 0.00023
Cumulative Proportion  0.99453 0.99557 0.9966 0.99749 0.99830 0.9989 0.99942 0.99969 0.99992
                          PC28    PC29    PC30
Standard deviation     0.03987 0.02736 0.01153
Proportion of Variance 0.00005 0.00002 0.00000
Cumulative Proportion  0.99997 1.00000 1.00000
>
```

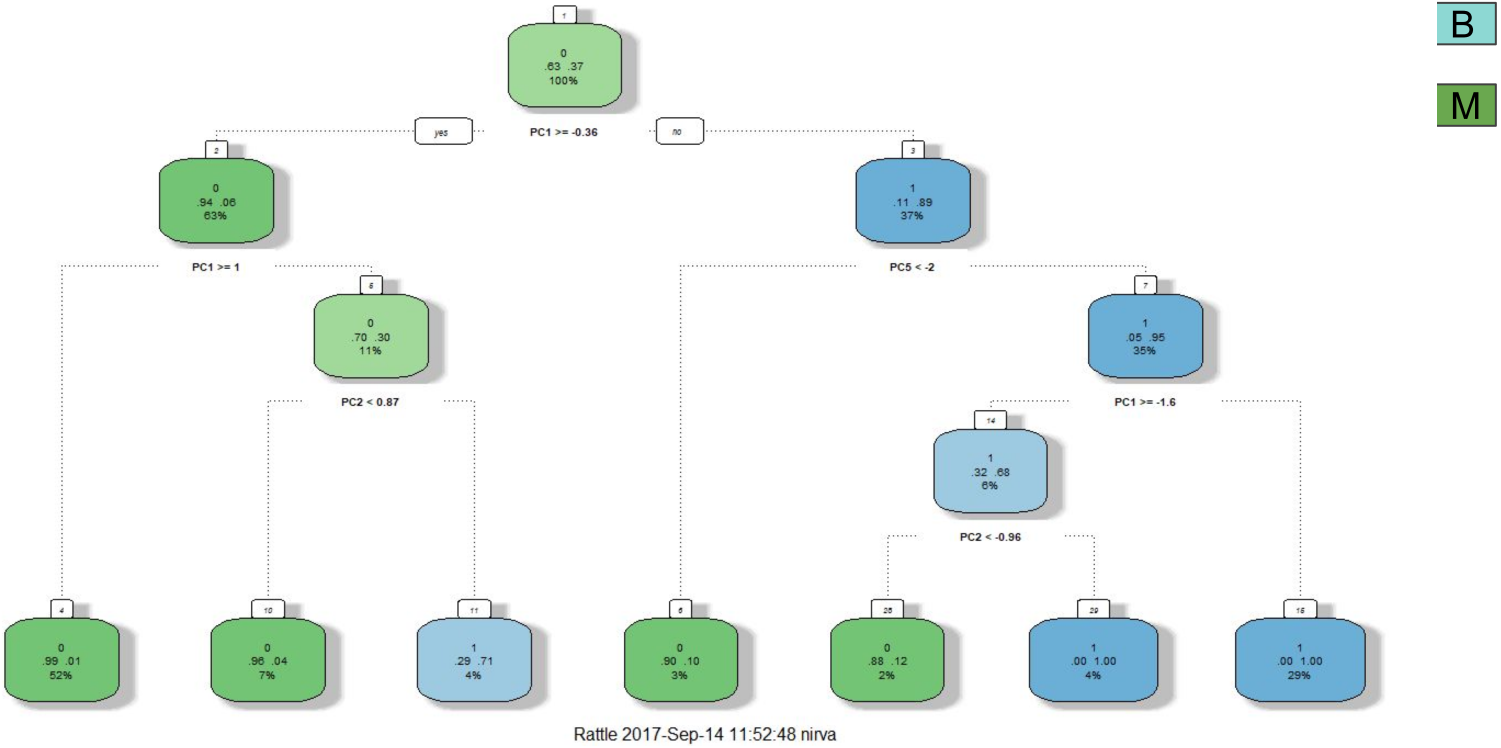
Model Building

- 8 different algorithms were trained and tested
    - Neural Network
    - Logistic Regression
    - K-nearest neighbor
    - Random Forest
    - Learning Vector Quantization
    - Linear Discriminant Analysis
    - Logistic Model Trees
    - Stochastic Gradient Boosting
- Iterated over 50 times
- Average performance was recorded

Model Building

```r
n=50

#glm,gbm,rf,knn,lvq,lda,LMT,nnet
for(j in c('glm','knn','rf','lvq','lda','LMT','gbm','nnet')){
  for(i in 1:n){
data1=data.frame(pca$x,y)
data2=data1[,c(1:10,31)]
data2$y=ifelse(data2$y=='M',yes = 1,0)
index=sample.split(Y = data2$y,SplitRatio = 0.7)
trainData=data2[index,]
trainData$y=as.factor(trainData$y)
test=data2[!index,]
test$y=as.factor(test$y)

require(caret)
model=train(y~.,data=trainData,method = j)

pred=predict(object = model,newdata = trainData)
confusionMatrix(pred,trainData$y)

predTest=predict(object = model,newdata = test)
metric=confusionMatrix(predTest,test$y)
avg_accuracy_list[i]=as.numeric(metric$overall['Accuracy'])
falseNegativeErrorRate_list[i]=1-as.numeric(metric$byClass['Sensitivity'])
  }
  avg_accuracy=sum(unlist(avg_accuracy_list))
  avg_falseNegativeErrorRate=sum(unlist(falseNegativeErrorRate_list))
  print(j)
  modelPerformance[nrow(modelPerformance)+1,]=c('ModelName'=j,'Accuracy'=round(avg_accuracy,digits = 4
}
modelPerformance$Accuracy=as.numeric(modelPerformance$Accuracy)/n
modelPerformance$False_Error_Rate=as.numeric(modelPerformance$False_Error_Rate)/n
```

Decision Tree



Performance Indicators

2 most important parameters for judgement here are

i) Accuracy (classified correctly)

ii) False Negative Error rate (Predicted an ill person healthy)

Accuracy with 10 PCA components

Neural Network is the best performing model with 97.26% accuracy.

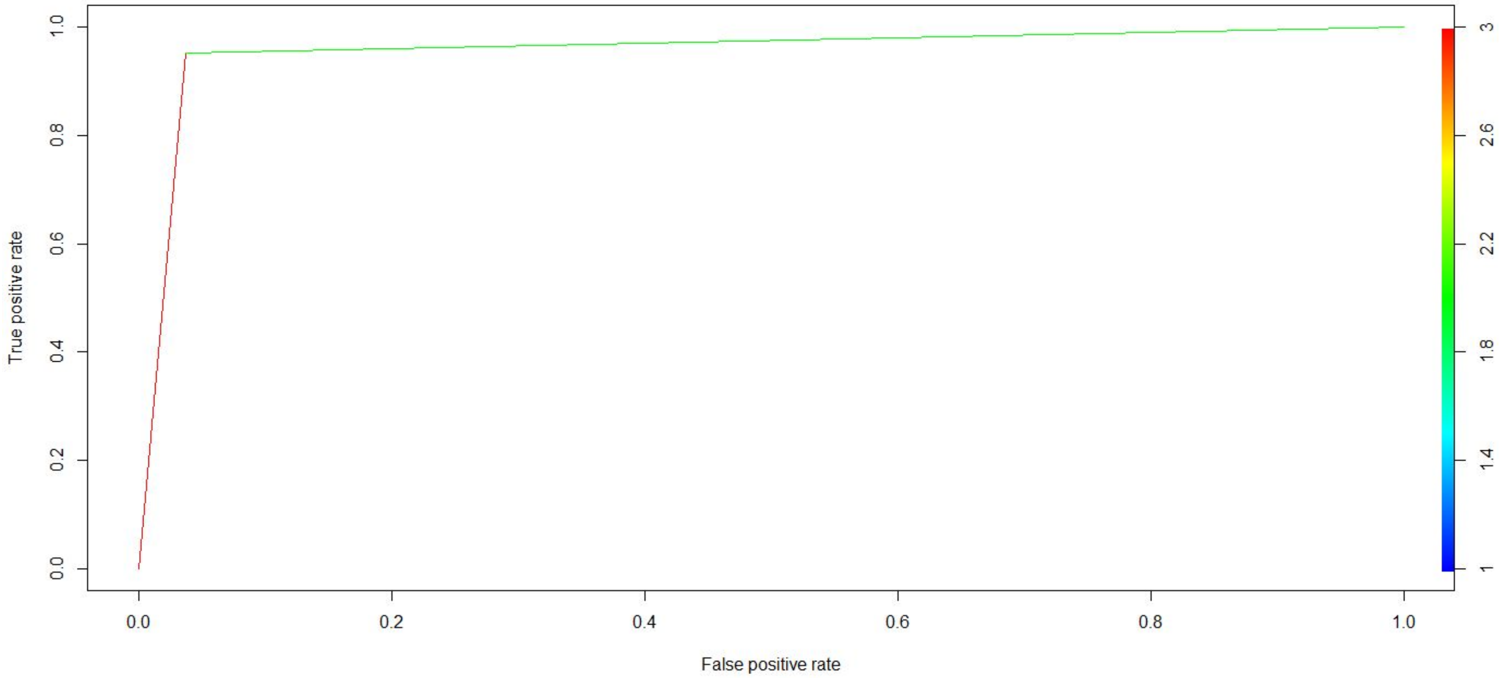
	ModelName	Accuracy
1	Logistic Regression	0.967602
2	K-nearest neighbor	0.965964
3	Random Forest	0.948772
4	Learning Vector Quantization	0.941404
5	Linear Discriminant Analysis	0.952632
6	Logistic Model Trees	0.971228
7	Stochastic Gradient Boosting	0.955906
8	Neural Network	0.971696

False Negative Error-Rate

- This calculates the error when patient with cancer is predicted as healthy
- False negative error rate of Linear Discriminant Analysis is the lowest

	ModelName	Accuracy	False_Error_Rate
1	glm	0.974502	0.017384
2	knn	0.961404	0.012336
3	rf	0.949240	0.040000
4	lvq	0.939884	0.027290
5	lda	0.955322	0.002990
6	LMT	0.975322	0.011962
7	gbm	0.960350	0.025046
8	nnet	0.970878	0.019066

ROC Curve



True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
1
1.4
1.8
2.2
2.6
3

Way Forward

- We also performed PCA with 15 components, very marginal improvement in accuracy
- Depending on priority of accuracy or False negative error rate, Neural network or Linear Discriminant Analysis can be chosen
- Ensembling of models can be done to improve accuracy
- Git Hub link: Click here


Thank you!