Data Analytics

Final Project

Breast Cancer Detection:

Classification of Malignant and Benign Tumors

Agustin Jose Olivo — ajo54
Catherine Horng — ch756
Jiahan Xie — jx353
Neil Gogri — ng378
Sirena DePue — sgd63

December 14, 2021

Contents

1 Introduction

Breast cancer is one of the leading causes of cancer deaths among women [1]. Because of this, quick and accurate detection of breast cancer is crucial for treatment. While many detection methods have been developed including studying clinical history and imaging with mammography and ultrasound, the only definitive method to detect breast cancer is with a biopsy of the tumor. The best method of obtaining a breast biopsy is with fine needle aspiration (FNA).

While manual examination of the tumor cells obtained from FNA was common, with the rise of technology and the ability to utilize these advances in the medical field, the use of automated detection of cancer may aid in the detection of breast cancers. Therefore, in order to improve on the accuracy of breast cancer diagnosis and identifying malignant versus benign tumors, the Wisconsin Diagnosis Breast Cancer dataset [2] was developed. This dataset contains characteristics of individual nuclei in the cancer mass obtained from a FNA such as radius, texture, perimeter, and other visual features of nuclei.

With this dataset, two methods of classification will be used to improve the diagnosis of breast cancer. We will use neural networks (NN) and support vector machines (SVM) to discriminate benign from malignant tumors. We found an accuracy rate of up to 95% for NN and up to 97% for SVM. This is significant as it may allow for an accurate diagnosis from just an aspirate, and therefore reduce the need for a surgical biopsy. To create classification plots and gain a more intuitive understanding of the data, we also used PCA to reduce the dimensions of the data before performing SVM. The methods, process, and interpretation of our results is described below.

2 Methodology

2.1 Dataset

This dataset contains 569 total instances, with 357 benign cases (63%) and 212 malignant cases (27%). The response variable in the dataset is the diagnosis: malignant or benign. Ten real-valued features are computed for each cell nucleus in a digitized image of a fine needle aspirate (FNA) of a breast mass. Those attributes are:

- Radius *(mean of distances from center to points on the perimeter)*
- Texture *(standard deviation of gray-scale values)*
- Perimeter
- Area
- Smoothness *(local variation in radius lengths)*
- Compactness *(*$perimeter^2/area - 1.0$*)*
- Concavity *(severity of concave portions of the contour)*
- Concave points *(number of concave portions of the contour)*
- Symmetry
- Fractal dimension *("coastline approximation" - 1)*

For each of these variables, a value for the mean, standard error, and worst is given (total of 30). We chose to only consider the mean values of the 10 variables, as the standard error and the "worst" values were not as meaningful as the mean value for each of the above features.

The data was cleaned prior to working with the selected algorithms. The diagnosis column (response variable) was changed from 'benign' and 'malignant' to binary variables (0 for benign and 1 for malignant) and were cast as factors. Finally, we normalized the data and randomly split the dataset into a training set of 400 datapoints (70%) and a testing set of 169 datapoints (30%). This split will allow to us to test how well our models generalize to real world situations.

2.2 Tools

2.2.1 Support Vector Machines

Support vector machines (SVM) are another supervised method used for classification. Using SVM, datapoints are separated into their respective classes using hyper-planes. In the case of a 2-dimensional example with 2 classes, SVM finds a line that allows us to separate one class from another, maximizing the distance from the

line to the data on for each class. After finding this line, in order to classify a new example, we look on which side of the line the datapoint falls on and classify it as such.

In higher dimensions, SVMs work by constructing a hyperplane that splits the data such that the distance between the nearest datapoint of either class is maximized, thus minimizing misclassification errors. Eventually kernel methods were developed to allow for non-linear decision boundaries which increased the interest of the use of SVMs. Kernel methods work by transforming the original dataset into another space. This allows us to again try and construct a hyperplane on the transformed data with the hope that this newly transformed data is linearly separable. A range of kernels exist that can be applied to the data and each have varying hyperparameters to set in order to transform the data in a multitude of various ways. These various kernels and their hyperparameters are explored as we construct our own SVM model.

Traditionally, regular SVMs can only separate linearly-separable data. That is, they separate data in which each side of the hyperplane correctly classifies all datapoints on each side. In order to apply SVMs to non-linearly separable data even with the use of kernel methods, the soft-margin SVM algorithm was developed. This makes it more applicable to real world datasets. Soft-margin SVMs allow us to modify the penalities for misclassifying datapoints when constructing our model. This way, we can allow for some misclassification in the hopes that this model that occasionally misclassifies some training points can still generalize well, or even better, to real world data.

2.2.2 Neural Networks and Deep Learning

Neural networks are a supervised learning algorithm used to solve a wide array of problems, including prediction, classification, image processing, pattern recognition, etc. A neural network consists of a set of interconnected "nodes" organized in multiple layers with a "feed-forward" structure (most commonly, although other structures exist). In feed-forward networks, the information "travels" from the input layer (independent variables, attributes), into the "hidden layers", and finally to the "output" layer (result of the prediction). Connections between nodes in the multiple layers are mediated by different weights, bias bodes, summation and activation functions (some of which are tuning parameters for the algorithm).

Some of the main "tuning parameters" that can be adjusted in the model to improve its robustness include the number of hidden layers, and the activation function. The activation function defines how the weighted sum of the inputs is transformed into an output from a node or nodes, in a layer of the network. Some of the most common activation functions are linear, binary step function, logistic function, hyperbolic tangent, and rectified linear unit function.

Deep learning is a variant of neural networks with an emphasis on models that have more and larger hidden layers. Deep learning models in general have more tuning parameters that can be used to increase the complexity of the model.

3 Results and Discussion

3.1 Support Vector Machines

After the pre-processing step described in the dataset section, we also tried removing variables to result in a simpler model, but this only reduced the performance. Therefore, we stuck to the 10 variables we had. While constructing our model, several kernels were trialed including linear, polynomial, and radial, with radial outperforming the others in most cases. After deciding to use the radial kernal, an automatic parameter sweep was done for the cost and gamma values, and the best one returned (cost=1.5 and gamma=0.08).

```
set.seed (1)
tune.out=tune(svm ,diagnosis~.,data=train_data ,kernel ="radial",
              ranges =list(cost=c(0.001,0.01,0.1,1,5,10,100)))
summary(tune.out)

obj = tune.svm(diagnosis ~., data=train_data,cost=seq(from=0.001, to=10,by=0.1),
    gamma = 0.08)
print(obj)
```

Depending on how the data was split, we received accuracy rates that ranged from 92% to 97%. The various accuracies are due to the random nature of how the data was split, however, these results still remain fairly high;

additionally, these results align with other sources, which received accuracy rates of 97.2% [3] and 96.09% [4] using SVM for the same dataset. The result for our SVM model and one of the training and test split is shown below:

	0 (B)	1 (M)
0 (B)	99	2
1 (M)	3	65

Figure 1: Confusion Matrix for SVM

This resulted in 97.04% accuracy, but misclassified 3 instances of malignant tumors as benign, and misclassified 2 instances of benign tumors as malignant. It might be preferable to be misdiagnosed with a malignant tumor than to be misdiagnosed with benign tumor. We do not want false reassurance of a benign tumor, as this could delay critical treatment options and effect the patient's outcome.

The effect of such misclassifications must be weighed against the overall accuracy rate when constructing the model. Currently, the cost and gamma parameters were tuned only to increase the overall accuracy. Since we are randomly splitting the dataset into a training and testing set, the best split should result in high accuracy and low misclassification of malignant tumors as benign:

	0 (B)	1 (M)
0 (B)	95	5
1 (M)	0	69

Figure 2: Confusion Matrix for SVM, Improved Misclassification

The above is an example of a split that does not sacrifice accuracy (97.04%), but whose results does not misclassify a malignant tumor as benign.

3.1.1 Support Vector Machines with Principal Component Analysis

In order to gain a more intuitive understanding of how the SVM was working, we decided to try some dimension reduction techniques.

Based on the correlation matrix (Figure 9), we noticed several of the variables were highly dependent on others, particularly perimeter, radius, and area. This aligns with our understanding of this dataset as mathematically, features such as the perimeter and area all depend on the radius. Some other factors appeared to be correlated as well such as concave points, concavity, and compactness. This makes the data a good candidate for PCA in an effort to reduce the dimensionality. We decided to reduce the dataset into two dimensions, which would allow us to create SVM classification plots to gain a more intuitive understanding of the data.

After applying principal component analysis (PCA), we were able to reduce the data to 2 principal components while maintaining a good amount of variability. As seen in Figure 13 and Figure 11, this data reduction allows us to to capture 79% of the total variability with two principal components. As suspected, variability in features such as perimeter, area, and radius can be captured together (Figure 10). Additionally, features such as compactness, concavity, and concave points also appear to be correlated with each other. Since 79% is a fairly high amount of variability captured, we tried performing SVM on the transformed data.

Using the first two principal components from PCA in a new SVM model, we reduced all the data we had to two dimensions before again splitting it into a training and test set. After some tuning of the kernel function, cost, and gamma values, we find no great difference in using different values than when we constructed our SVM model without dimension reduction. Thus, we use the same radial kernel with a cost of 1.5 and gamma value of 0.08.

As expected, this model built with the reduced dimension dataset yielded a lower accuracy rate of 92% (because we lose some variability). However, this accuracy is still fairly high, and quite high for only two out of the ten original components. The results for this is shown below:

	0 (B)	1 (M)
0 (B)	102	13
1 (M)	4	50

Figure 3: Confusion Matrix for SVM using PCA

We again see that there are more instances of misclassifying benign tumors as malignant than there are of misclassifying malignant tumors as benign, which again is preferable.

With this data of reduced dimensions, we are also able to plot our model in Figure 14 to gain a more intuitive understanding of how SVM is splitting the data. The plot shows each training datapoint, graphed against the first two principal components. The 'O's represent datapoints not used as support vectors, while the 'X's represent datapoints used as support vectors. As expected, the support vectors are mainly datapoints that lie close to the boundary.

The maroon color indicates malignant (1), while the yellow color indicates benign (0), as shown in the side bar. Misclassifications in the training set occur where the color of the data point is opposite to the color of the side it appears in. From this, we can see that the model correctly classifies most observations. Additionally, misclassified observations are mostly close to our margin, indicating that if we do not classify a point correctly, the model is still fairly close.

Looking at our plot of the decision boundary, we see that observations with a higher principal component 1 value generally correspond to malignant tumors while a lower principal component 1 value corresponds to a benign tumor. From Figure 12, we see that principal component 1 does well in capturing the variability in radius, perimeter, area, compactness, concavity, and concave points. This indicates to us that a tumor with higher values for the aforementioned features is more likely to be malignant.

Additionally, it appears that observations with a relatively low principal component 2 value (ranging from -2 to 2), often get misclassified as seen in Figure 14. From Figure 12, we see that principal component 2 does well in capturing the variability in smoothness, symmetry, and fractal dimension. This indicates to us that lower values for PCA1 paired with lower values for the aformentioned features for PCA2 may not be classified correctly using this model. If a patient has values that fall within this range, it should be taken into account and perhaps another diagnostic test (or surgical biopsy) should be performed.

In general, principal component 1 appears to more consistently distinguish between malignant and benign tumors as the decision boundary in Figure 14 appears to primarily separate observations with higher and lower principal component 1 values, with some amount of curvature. This makes sense as principal component 1 already captures 54.79% of variability, while principal component 2 only captures an additional 25.19% of variability.

3.2 Neural Networks

3.2.1 Neural Networks

The "neuralnet" command was utilized in R. Initially, prediction capabilities of the model were estimated using a "logistic" activation function, and 2 hidden layers with 3 nodes each (figure 15). The architecture of the network was also composed by 10 input layers (10 predictors) and 2 output layers (binary response variable). The bias neurons are shown in blue, which is a constant value added to each layer.

Using neural networks as described, we achieved classification with 93.57% accuracy (figure 4). On average, there is about 4.9% error in the model when using the training set. With a low error rate and a high accuracy rate, we can expect the model to generalize well to new data. The model misclassified 4 malignant tumors as benign and 7 benign tumors as malignant. Despite the high performance, as stated earlier, the misclassifications is cause for concern if this were to be used as the single diagnostic strategy for breast cancer.

	0 (B)	1 (M)
0 (B)	96	7
1 (M)	4	64

Figure 4: Confusion Matrix for Neural Networks

The hyperbolic tangent activation function was also tested in the model with 3 hidden layers, and 3 nodes each. The model yielded an accuracy of 93.57%, so no different from the previous model that used the logistic function.

3.2.2 Deep Learning

After working with the neuralnet library, we implemented a deep learning model using H2O, for further predictions on the same dataset. Predictors were standardized through the same command. The initial architecture of the network included 2 hidden layers with 3 nodes each (same as before). "Maxout" was used as activation function. The model achieved a 93.57% accuracy, equal to the previous neural network model run.

Following, a more complex architecture, including 5 hidden layers, with 5 nodes each was tested. The final accuracy of the more complex model was 95.32% (figure 5). This shows that incorporation of more hidden layers and nodes may help improve the accuracy of the prediction.

	0 (B)	1 (M)
0 (B)	102	1
1 (M)	7	61

Figure 5: Confusion matrix for deep learning model with 5 hidden layers, and 5 nodes each.

In figure 17, we can see the training history of the deep leaning model. Over the course of 25 epochs, we can see the training an validation sets start off close and parallel to each other and slowly converge. This is a good indicator that the model is performing well, and continues to improve (albeit at a slower rate), over the course of many iterations.

The H2O command was also utilized to identify the predictors with the largest impact on the response variable. We can see from figure 16 that "radius mean" is the most critical variable in our model followed by "compactness mean" and "smoothness mean" for different cell nuclei. These value are all ranked upwards of 85% in terms of relative importance, indicating that these variables contribute more to the prediction (determined by weighted connections), relative to the other variables. After these variables, the relative importance begins to drop off, down to 20% for the mean perimeter.

In comparing the two methods, we see that neural networks as well as SVM with PCA show that the radius and compactness appears to be a higher indicator as whether the tumor is benign or malignant. This gives us a good intuitive understanding about this dataset; the radius and compactness of a cell nucleus obtained by FNA can be predictive of its nature. While smoothness does not seem to be indicative of how well one can classify a tumor cell as benign or malignant according to SVM with PCA, this could be because of the lack of variability captured when doing PCA on the dataset.

4 Conclusion

For SVM, we achieved a high test accuracy of 97% accuracy with lower misclassification of benign tumors than malignant tumors. With PCA and dimension reduction, we still achieved a 92% accuracy, again with a lower misclassification of benign tumors than malignant ones. With the dimension reduction we are able to gain a better intuition about what the variables correspond to what diagnosis.

Neural networks and deep learning strategies seem to show slightly lower performance in terms of prediction accuracy, than SVM (the highest accuracy values achieved with more complex neural networks was 95%). The highest accuracy is achieved by setting 4 hidden layers with 3 nodes in each layer with logistic function as our active function. An increase of hidden layer or node will decrease the accuracy due to overfitting. We deduce that neural networks and deep learning accuracy is lower is due to certain overfitting. We can further improve the performance by adding more variables with more features and by customizing our activation function. By using deep learning API h2o, we gain more insights that running iterations of 25 times can best improve our score. And we can also conclude that some of the most relevant variables to define if a tumor is benign or malignant using neural networks are radius, compactness and smoothness (radius is of dominant importance, in particular).

Both methods analyzed indicated high accuracy for diagnosing breast cancer when using the predictors in the dataset. However, it is possible to have false negatives, what might be a concern from the medical point of view. Therefore, this strategy may need to be complemented with an additional diagnostic methods.

5 Appendix

5.1 Tables and Figures

5.1.1 Support Vector Machines & PCA Figures

predictdiagnosis	0	1
0	99	2
1	3	65

Figure 6: Confusion Matrix for SVM

predictdiagnosis	0	1
0	95	5
1	0	69

Figure 7: Confusion Matrix for SVM, Improved Misclassification

test_pca_y		
predictdiagnosis	0	1
0	102	13
1	4	50

Figure 8: Confusion Matrix for SVM using PCA

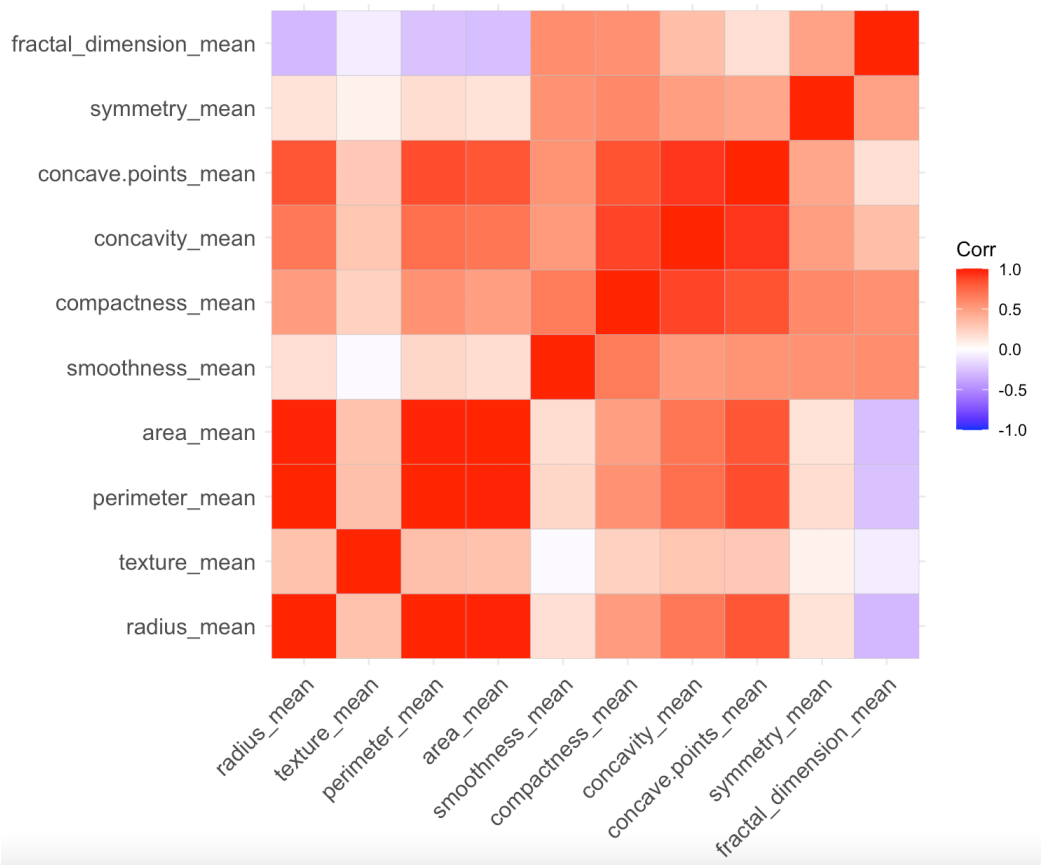

Figure 9: Correlation Matrix

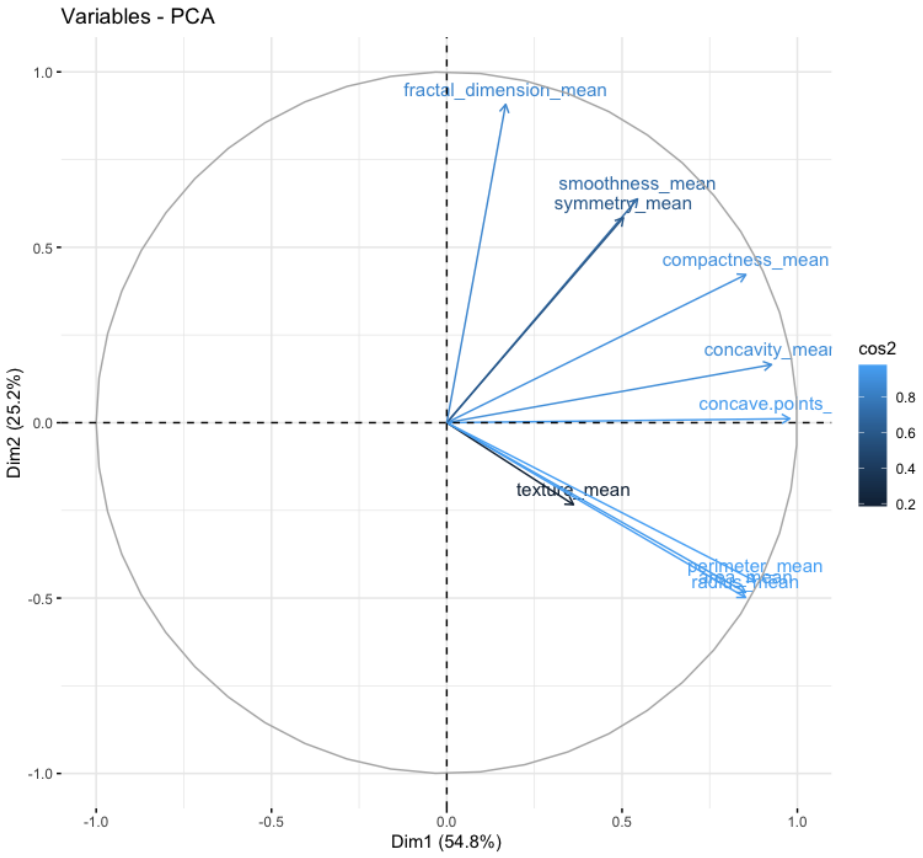

Figure 10: Biplot for PCA

	eigenvalue	percentage of variance	cumulative percentage of variance
comp 1	5.4785879917	54.785879917	54.78588
comp 2	2.5187135854	25.187135854	79.97302
comp 3	0.8806151792	8.806151792	88.77917
comp 4	0.4990094357	4.990094357	93.76926
comp 5	0.3725391897	3.725391897	97.49465
comp 6	0.1241417485	1.241417485	98.73607
comp 7	0.0800853104	0.800853104	99.53692
comp 8	0.0348897928	0.348897928	99.88582
comp 9	0.0111354606	0.111354606	99.99718
comp 10	0.0002823059	0.002823059	100.00000

Figure 11: Percentage of Each Principal Component

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
radius_mean	13.2450815	9.855146313	1.5482225	0.087372607	0.096515986
texture_mean	2.3855151	2.166222009	90.4508638	0.007949655	4.836602090
perimeter_mean	14.1409347	8.103011156	1.3015067	0.018111962	0.003534399
area_mean	13.2558504	9.292847088	1.5222095	0.018070570	0.037408287
smoothness_mean	5.4047197	16.157370963	2.7733064	1.162127842	71.190611809
compactness_mean	13.2818014	7.076299441	0.3396309	3.448464321	5.768785778
concavity_mean	15.6616866	1.087556319	0.1693033	2.777339679	9.767702880
concave.points_mean	17.4756104	0.005160418	0.4699627	0.532665703	0.008427604
symmetry_mean	4.6327384	13.564556008	0.1348626	79.744627706	1.274371588
fractal_dimension_mean	0.5160617	32.691830285	1.2901314	12.203269955	7.016039579

Figure 12: Contribution of Each Variable

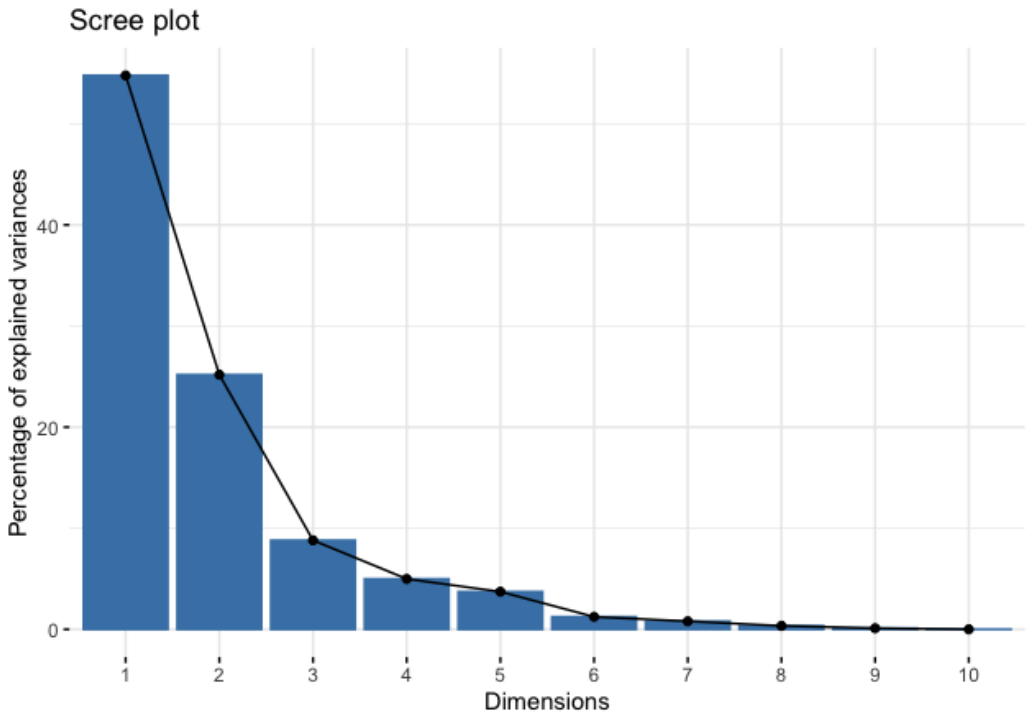

Figure 13: Scree Plot for PCA

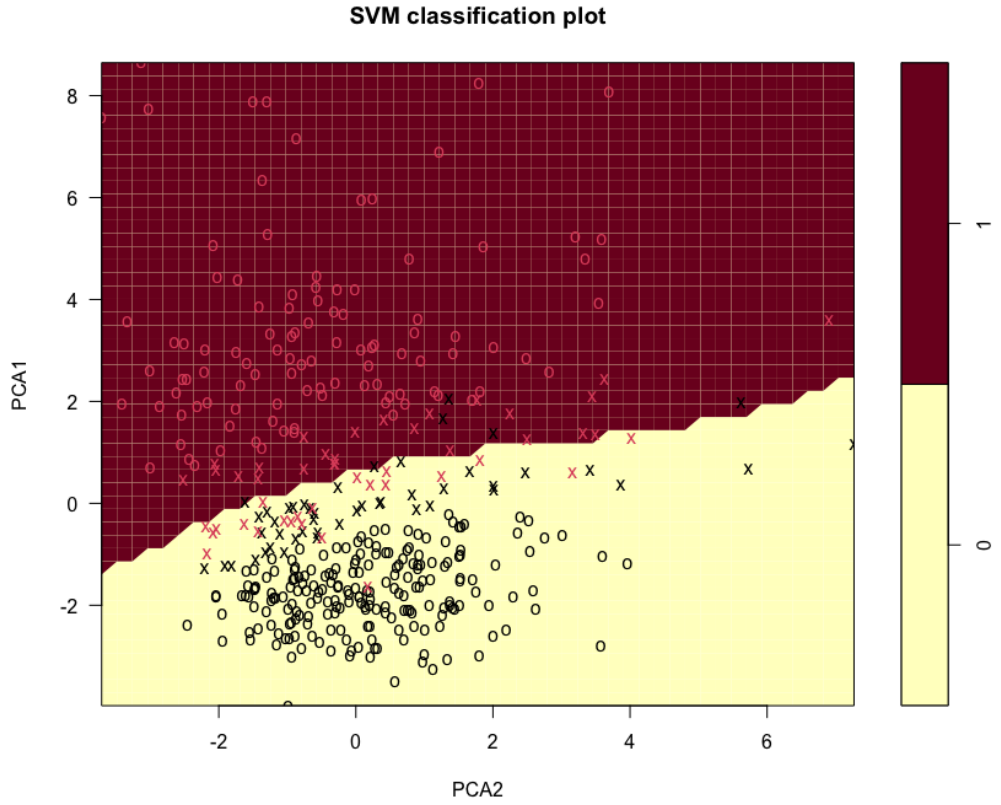

Figure 14: SVM Plot

5.1.2 Neural Networks Figures

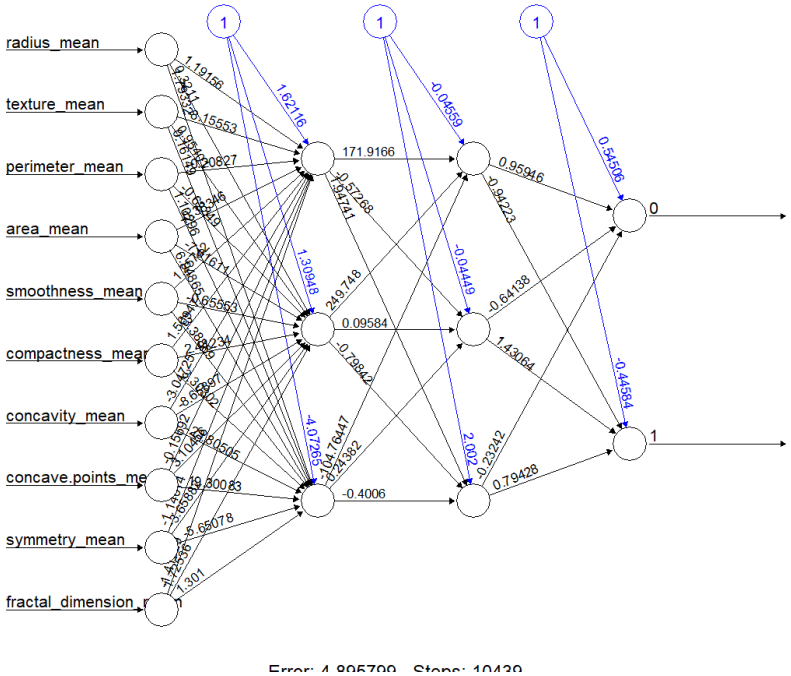

Figure 15: Graphical Representation of the Neural Network with 2 Hidden Layers.

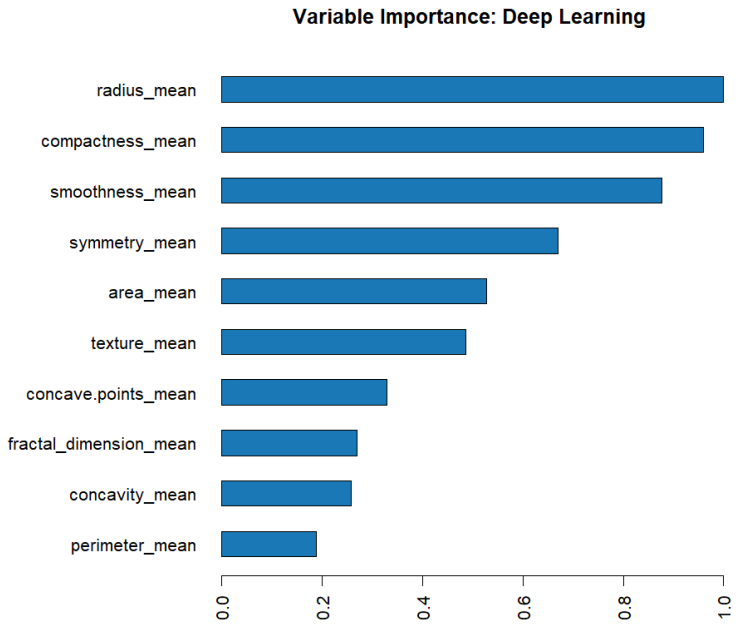

Figure 16: Importance of Individual Attributes in the Prediction of the Response Variable

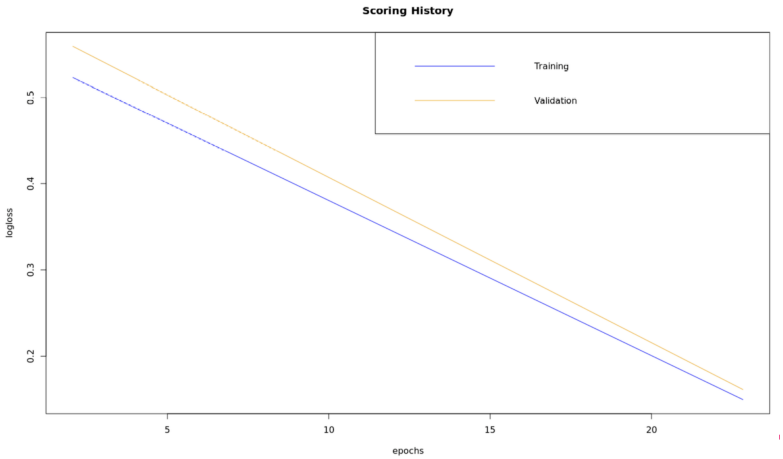

Figure 17: Training History of Deep Learning Model

5.2 Code

5.2.1 SVM Code

Original SVM Set

```
library(ggplot2)
library(ggcorrplot)
library(e1071)

getwd()
setwd("/Users/sirenadepue/Desktop/Data Analytics")
frame<-read.csv("project2data.csv", row.names=1)
frame<-subset(frame, select=-c(id,X.1))

############ Correlation ############
frame_corr<-subset(frame,select=c(2:11))
corr <- cor(frame_corr)
corr2 = as.matrix(corr)
print(corr)
ggcorrplot(corr)

############ Frame Manipulation ############
frame$diagnosis[frame$diagnosis=="B"]<-0
frame$diagnosis[frame$diagnosis=="M"]<-1
frame$diagnosis=as.factor(frame$diagnosis)
head(frame)
frame2<-subset(frame,select=-c
    (1,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31))
plot(frame2)

############ Normalization ############
normalize<-function(x) {return ((x-min(x))/(max(x)-min(x)))}
norm_train<-as.data.frame(lapply(frame2, normalize))

############ Train/Test Sets ############
frame2<-cbind(frame$diagnosis,norm_train)
ind<-sample(1:nrow(frame2), 400)
print(ind)
train_data<-frame2[ind,]
test_data<-frame2[-ind,]
print(test_data)
colnames(train_data)[1] <- "diagnosis"
colnames(test_data)[1]  <- "diagnosis"

############ Fit LDA model using training data set ############
result<-svm(diagnosis~., cost=1.5, gamma=0.08, kernel="radial", data=train_data)
print(result)

############ Predict using test data set and evaluate accuracy ############
predictdiagnosis <- predict(result, newdata=test_data)
mean(predictdiagnosis == test_data$diagnosis)
print(predictdiagnosis)

CT<-table(predictdiagnosis, test_data$diagnosis)
print(CT)

############ Search for Best Parameters ############
set.seed (1)
tune.out=tune(svm ,diagnosis~.,data=train_data ,kernel ="radial",
```

```
                ranges =list(cost=c(0.001,0.01,0.1,1,5,10,100)))
summary(tune.out)

obj = tune.svm(diagnosis ~., data=train_data,cost=seq(from=0.001, to=10,by=0.1),
    gamma = 0.05)
print(obj)
```

Reduced SVM Set

```
##############
#### PCA ####
##############
library(factoextra)

data<-frame2[c(2:11)]

### run PCA ###
Sol<-PCA(data, scale.unit=TRUE, graph=TRUE)
fviz_eig(Sol)
fviz_pca_var(Sol, col.var="cos2")
Sol$eig

### set x and y to be first two principal components ###
x<-Sol$ind$coord[,c(1)]
y<-Sol$ind$coord[,c(2)]
pca_data<-data.frame("diagnosis" = frame2[c(1)], "PCA1" = x, "PCA2" = y)

### split train and test sets ###
ind<-sample(1:nrow(data_mat), 400)

train_pca_data<-pca_data[ind,]
train_pca_y<-as.factor(unlist(train_pca_data[c(1)]))
train_pca_x<-train_pca_data[c(2,3)]
train_pca<-data.frame("diagnosis" = train_pca_y, "PCA1" = train_pca_x[1], "PCA2"
    = train_pca_x[2])

test_pca_data<-pca_data[-ind,]
test_pca_y<-as.factor(unlist(test_pca_data[c(1)]))
test_pca_x<-test_pca_data[c(2,3)]
test_pca<-data.frame("diagnosis" = test_pca_y, "PCA1" = test_pca_x[1], "PCA2" =
    test_pca_x[2])

### run SVM ###
result<-svm(diagnosis~., cost=1.5, gamma=0.08, kernel="radial", data=train_pca)
print(result)

### predict test data ###
predictdiagnosis<-predict(result, newdata=test_pca_x)
mean(predictdiagnosis == test_pca_y)

### get confusion matrix ###
CT<-table(predictdiagnosis, test_pca_y)
print(CT)

### plot SVM ###
plot(result, train_pca)
```

5.2.2 Neural Network Code

```
############################ LOAD LIBRARIES ##############################
```

```
library(MASS)
library(neuralnet)
library(forecast)
library(ggplot2)
library(lime)
library(h2o)
h2o.init()

############################ REGULAR_NEURAL_NETWORKS ################################

frame<-read.csv("C:/Users/ajo54/Box/1 - My Files/2 - PhD Program/1 - Classes/1 - 3 Semester/Group Project/Group Project 2/data.csv", row.names = 1)
head(frame)

frame$diagnosis[frame$diagnosis=="B"]<-0
frame$diagnosis[frame$diagnosis=="M"]<-1

######## Normalize the data ########

normalize<-function(x) {return ((x-min(x))/(max(x)-min(x)))}
norm_frame<-as.data.frame(lapply(frame[2:11], normalize))
nrow(norm_frame)
norm_frame$diagnosis <- frame$diagnosis

######## Define correlation among variables ########

correlation <- cor(frame[2:11])
corrplot::corrplot(cor(frame[2:11]))
pairs(~radius_mean+texture_mean+perimeter_mean+area_mean+smoothness_mean+
    compactness_mean+concavity_mean+concave.points_mean+symmetry_mean+fractal_
    dimension_mean,data=frame[2:11], main="Scatterplots for Election Data")


######## Create training and test dataset #########

set.seed(1)
ind<-sample(1:nrow(norm_frame), nrow(norm_frame)*0.7)
train_data<-norm_frame[ind,]
test_data<-norm_frame[-ind,]
head(train_data)

######## Run the model #########

Net.Est<-neuralnet(diagnosis~radius_mean+texture_mean+perimeter_mean+area_mean+
    smoothness_mean+compactness_mean+concavity_mean+concave.points_mean+symmetry_
    mean+fractal_dimension_mean, hidden=c(3,3), data=train_data, act.fct="tanh")
?neuralnet
plot(Net.Est)
Net.Est$result.matrix

######## Test the model in test dataset #########

temp_test <- subset(test_data, select = c("radius_mean","texture_mean", "
    perimeter_mean", "area_mean", "smoothness_mean", "compactness_mean", '
    concavity_mean', "concave.points_mean","symmetry_mean", "fractal_dimension_
    mean"))
head(temp_test)
nn.results <- compute(Net.Est, temp_test)
results <- data.frame(actual = test_data$diagnosis, prediction = nn.results$net.
    result[,2])
```

```
results

####### Build confusion matrix ########

roundedresults<-sapply(results$prediction,round,digits=0)
roundedresultsdf=data.frame(roundedresults)
roundedresultsdf
table(results$actual,roundedresults)


######################### DEEP LEARNING / h2o ################################

######## Load data and build training and test datasets ########

frame$diagnosis= factor(frame$diagnosis,levels = c(0, 1), labels = c(0, 1))
set.seed(1)
head(frame)
ind<-sample(1:nrow(frame), nrow(frame)*0.7)
train_data<-as.h2o(frame[ind,])
test_data<-as.h2o(frame[-ind,])
nrow(train_data)
nrow(test_data)
head(train_data)

######## Run model ########

DeepLearningModel <- h2o.deeplearning(y="diagnosis", x=c("radius_mean","texture_
    mean", "perimeter_mean", "area_mean", "smoothness_mean", "compactness_mean", '
    concavity_mean', "concave.points_mean","symmetry_mean", "fractal_dimension_
    mean"), training_frame = train_data, validation_frame=test_data, standardize =
     TRUE, hidden=c(5,5,5,5,5), nfolds=5, activation="Maxout", adaptive_rate=TRUE,
     l1=0.01, epoch=20, seed=33)
print(DeepLearningModel)

######## Visualization########

plot(DeepLearningModel)
h2o.varimp_plot(DeepLearningModel, 15)
summary(DeepLearningModel)
```

References

[1] L. Duijm, H. Groenewoud, F. Jansen, J. Fracheboud, M. Beek, and H. Koning, "Mammography screening in the netherlands: Delay in the diagnosis of breast cancer after breast cancer screening," *British journal of cancer*, vol. 91, pp. 1795–9, 11 2004.

[2] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[3] M. A. Naji, S. E. Filali, M. Bouhlal, E. H. Benlahmar, R. A. Abdelouhahid, and O. Debauche, "Breast cancer prediction and diagnosis through a new approach based on majority voting ensemble classifier," *Procedia Computer Science*, vol. 191, pp. 481–486, 2021. The 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC), The 16th International Conference on Future Networks and Communications (FNC), The 11th International Conference on Sustainable Energy Information Technology.

[4] A. F. Agarap, "On breast cancer detection: An application of machine learning algorithms on the wisconsin diagnostic dataset," *CoRR*, vol. abs/1711.07831, 2017.