The Report Committee for Mengjie Yu
Certifies that this is the approved version of the following report:

Breast Cancer Prediction
Using Machine Learning Algorithm

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Michael J. Daniels

Edward C. Theriot

Breast Cancer Prediction
Using Machine Learning Algorithm

by

Mengjie Yu, B.S.

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Statistics

The University of Texas at Austin

May 2017

Dedication

This report is dedicated to my parents, Xinyou Yu and Huiping Zhang, my husband Forest Pfeiffer, and my extended family for their endless support and love during graduate school.

Acknowledgements

Special thanks go to Dr. Michael Daniels for serving as my supervisor for this master report. I would like to express my sincere gratitude to my advisor Dr. Edward Theriot for supporting me pursuing a master degree in Statistics along with my PhD study on Genomics.

I would also like to thank my intern managers, Dr. Gene Shin from Takeda Pharmaceutical Computational Biology Group, and Dr. Amy Xiujuan Wang from Dow Agrosciences Mathematics and Statistics Group, for offering me the very opportunity to work on real world data, which further strengthened my determination of pursuing systematic training in Statistics.

At last, I would like to thank my future manager, Gregory Martinez at Intel, for offering me an opportunity to continue my applied data science journey upon graduation.

Abstract

Breast Cancer Prediction
Using Machine Learning Algorithm

Mengjie Yu, M.S. Stat.

The University of Texas at Austin, 2017

Supervisor: Michael Daniels

Breast cancer, mostly occurring in women, is the mostly frequently diagnosed cancer. Early detection based on phenotype and genotype features can greatly increases the chances for successful treatment. In this report, four different machine learning algorithms were tested for breast cancer prediction. Principal component analysis was used to reduce dimension for the original correlated dataset. The results show that KNN, SVM with linear kernel and Logistic Regression outperform Naive Bayes with very similar accuracy. KNN achieved the highest average accuracy of 0.9756 after 10 fold cross-validation when k equals to 7. The highest AUC value of 0.9944 was achieved by SVM with linear kernel. The results also show that increasing number of top eigenvectors increases the prediction accuracy, however, as the eigenvector number goes above a certain threshold, it adds more noise instead of signal.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Breast cancer, mostly occurring in women, is a major health problem throughout the world. It is the second most common cancer, after lung cancer (Ferlay *et al.*, 2010). Histological examination of tissue specimen remains the main method for breast cancer detection. The histological examination involves visual inspection of tissue specimen from a slide, then determine the probability of breast cancer. The vast amount of identified visual features learned through machine learning algorithms, will possibly open the door for the automated breast cancer identification. Early detection based on phenotype features can greatly increases the chances for successful treatment.

Machine learning methods have been applied to a broad range of areas. Machine learning can be generally divided into two major categories: supervised and unsupervised learning methods. In supervised learning methods, the outcome label is present to guide the learning process, whereas unsupervised methods learn pattern without outcome label. The intermediate between supervised and unsupervised learning is semi-supervised learning, where only a subset of data has associated labels.

The machine learning algorithms are trained on the training data, and tested on the untrained data. If the model is excessively complex, such as having too many parameters, it is likely to lead to the problem of overfitting. Likewise, if the model is excessively simple that cannot capture the underlying trend of the data, underfitting occurs. Both overfitting and underfitting lead to poor predictive performance. There are several techniques to overcome overfitting, such as cross-validation, regularization and drop out *etc*. One of the most commonly used methods is k-fold cross-validation, where the original data is randomly partitioned into k equal sized subsamples. Out of the k subsamples, one subsample is used to testing the model, and the remaining k-1

subsamples are used to train the model. The k results are then averaged to generate one single estimation. One advantage of k-fold cross validation is each testing subsample is used exactly once.

Large amount of data with features in high-dimensional spaces will lead to curse of dimensionality. Several algorithms have been used to cope with this problem, such as Principal Component Analysis, Curvilinear Component Analysis, and Self-Organizing Maps or Kohonen's maps *etc* (Verleysen and François, 2005). PCA is a useful statistical method to reduce high dimension in data. It uses orthogonal transformation to convert observations to linearly uncorrelated variables, *i.e.* principal components, then calculates the eigenvalues and corresponding eigenvectors of the covariance matrix. The number of principal components is less than or equal to the number of original variables. The proportion of the variance of each eigenvector is proportional to the corresponding eigenvalues. The first principal component has the largest possible variance, which counts for the most of the variability of the data.

Supervised learning analyzes labeled training data and produces an inferred function, which is used for testing new examples. K-nearest neighbor classifier (KNN) classifies unknown by relating the unknown to the known using distance function. KNN is a brute-force computation of all pairs of points in the dataset. If $k = 1$, it simply assigns the unknown to the class of the nearest neighbor, also called the nearest neighbor algorithm. If $k > 1$, the classification is decided by majority vote based on the k nearest neighbor prediction result, with ties broken at random.

Support vector machine (SVM), a binary classifier, searches the hyperplane leaving the largest possible fraction of points of the same class on the same side, while maximizing the distance of each class from the hyperplane. Assuming we have n training

samples and each sample is represented by xi with class labels yi, where yi $\in \{-1, +1\}$. SVM can be formulated as the following optimization problem:

$$\max \sum_{i=1}^{n} ai - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} ai\ aj\ yi\ yj\ [\phi(xi), \phi(xj)]$$

$$s.t.\ \ 0 \leq ai \leq C, i = 1, \ldots, n$$

$$\sum_{i=1}^{n} ai\ yi = 0$$

where $\phi(x)$ is the feature vectors of input x, and C is the penalty. $K(xi, xj) = \phi(xi) \cdot \phi(yi)$ is the kernel function. Additional kernel functions enable SVM to operate in a high-dimensional space by simply computing the inner products between all pairs of data in the feature space. Three most common kernel functions are:

Linear Kernel: $K(x, x') = x \cdot x'$

Polynomial Kernel: $K(x, x') = (1 + x \cdot x')^p$

Radial Basis Kernel: $K(x, x') = \exp(-\| x - x' \|^2)$

Logistic regression algorithm was developed by David Cox in 1958 (Cox, 1958). The model is used to estimate the probability of a binary outcome based on continuous or category predictor variables. The formula is below, where *P(y)* indicates the estimated probability.

$$P(y) = \frac{1}{1 + e^{-(B0 + B1\ X1 + \cdots + BnXn)}}$$

Naive Bayes classifier is a probabilistic classifier based on Bayes' Theorem with naive independence assumption between predictors. Despite the rather optimistic assumption of independence, Naive Bayes classifier often outperform more sophisticated alternatives (Hastie *et al.*, 2009).

In this study, PCA is used to pre-process the data and extract features, four different machine learning models, KNN, SVM with different kernels, Logistic Regression and Naive Bayes were implemented and their performance compared. The effect of number of top eigenvector number on prediction accuracy is also evaluated.

Chapter 2: Methods

Dataset

The Wisconsin Diagnostic Breast Cancer dataset was obtained from the UCI machine learning repository (available at: http://archive.ics.uci.edu/ml). The dataset was created by Wolberg, Street and Mangasarian by extracting 30 features from analyzing 569 tumor images from fine needle aspiration slides (Street *et al.*, 1993). The dataset contains 357 cases of benign breast cancer and 212 cases of malignant breast cancer. The dataset contains 32 columns, with the first column being the ID number, the second column being the diagnosis result (benign or malignant), and followed by the mean, standard deviation and the mean of the worst measurements of ten features. There were no missing values. The features together with description were listed in Table 2.1.

Data pre-processing

Exploratory data analyses were conducted using "dplyr", "tidyr" and "ggplot2" packages in R version 3.3.1 (R core Team). The distributions of each attribute were visualized and colored by different diagnosis category (Figure 3.1-3.3; Appendix Figure B.1-B.3). The correlation scores between features were measured using Pearson correlation. The correlation results were visualized using "corrplot" library in R. Ten fold cross-validation were used to split training and testing dataset using "caret" package. PCA was applied on training dataset to create orthogonal eigenvector spaces ranked by eigenvalues. The testing dataset were then projected to this eigenvector space. Different number of top eigenvector ranging from 2 to 30 were used to test their effect on prediction accuracy.

Radius	Mean of distances from center to points on the perimeter
Texture	Standard deviation of gray-scale values
Perimeter	The total distance between the snake points constitutes the nuclear perimeter.
Area	Number of pixel on the interior of the snake and adding one-half of the pixel in the perimeter
Smoothness	Local variation in radius length, quantified by measuring the difference between the length of a radial line and the mean length of lines surrounding it.
Compactness	Perimeter ^2 / area
Concavity	Severity of concave portions of the contour
Concave points	Number of concave portions of the contour
Symmetry	The length difference between lines perpendicular to the major axis to the cell boundary in both directions.
Fractal dimension	Coastline approximation. A higher value corresponds to a less regular contour and thus to a higher probability of malignancy.

Table 2.1: Description of ten features used in the dataset

Data modeling

KNN classifier was implemented using the "class" library in R. Prediction accuracy was calculated for a set of k values ranges from 1 to 20. The best performed k value was selected to test for the effect of different number of top eigenvectors on prediction accuracy.

SVM classifier and Naive Bayes classifier were implemented using the "e1071" library in R. Three different kernel functions (linear, polynomial and radial basis) were

tested. The best fitted cost and gamma parameters for each kernel were tested using the tune function (Table 2.2). The linear kernel performed the best, was then selected to test the effect of number top eigenvectors on accuracy.

Logistic regression was implemented using the glm function with family specified to binomial in "stats" library in R.

Kernel	Cost	Gamma
Linear	10	0.5
Polynomial	0.1	0.5
Radial basis	0.1	0.5

Table 2.2: Fitted cost and gamma parameter values for SVM kernels

Model performance evaluation

The average accuracy for 10 fold cross-validation were calculated using the confusionMat( ) function with specified positive class "M".

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

where TP indicates true positive, TN indicates true negative, FP indicates false positive, and FN indicates false negative.

The receiver operating characteristic curve, or ROC curve, was plotted for 10 fold cross-validation for each classifier to illustrate model performance (Appendix Figure B.4-B.5). The ROC curve plots true positive rate (TPR), aka sensitivity or recall, against false positive rate (FPR), aka fall-out or 1- specificity, at various threshold settings. The area under the curve, referred as AUC, is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative

instance. The AUC were measured using the performance() function in "ROCR" package. The average AUC across the 10 fold cross-validation for different classifier were compared.

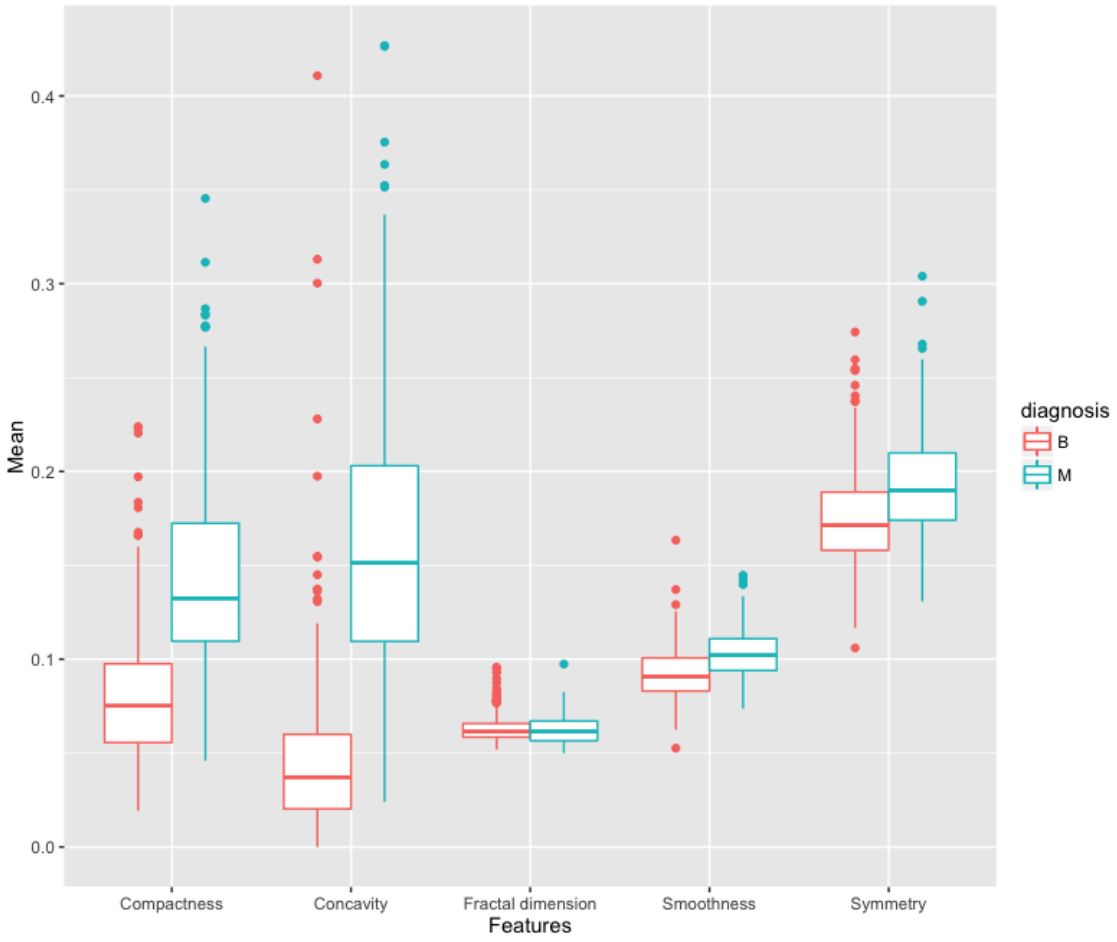


Figure 3.1: Distribution of mean values of features with small means

Chapter 3: Results

Data exploration

The distributions of the mean, standard error and worst average of the 10 features extracted from the fine needle aspiration slides show that compactnesss, concavity, factal dimension, smoothness and symmetry each have relatively small values for the measurement (Figure 3.1; Appendix Figure B.1). Perimeter, radius and texture each have relatively large values for the measurement, with areas that show the largest measurement value and amount of variation for all three measurements (Figure 3.2-3.3; Appendix Figure B.2-B.3). From the distribution visualization, we can see overall the malignant diagnosis class has relatively higher mean for all the attributes (Figure 3.1-3.3; Appendix Figure B.1-B.3).

Among the mean measurement of the 10 attributes, we can see several of them are highly correlated between each other (Figure 3.4; Appendix Table A.1). The mean features with correlation coefficient higher than or equal to 0.8 were visualized in a scatterplot in Figure 3.5. From Figure 3.5, we can see all the pairs show approximately linear relationship between each other, with the mean of the malignant class higher than the benign class.

We further explored the correlation between the mean, standard error, and worst average measurement of the same attributes. From Figure 3.6, we can see several attributes measurement between the mean and worst average were highly correlated. The attribute measurement with the standard error is less correlated with the mean and the worst mean, respectively (Figure 3.7-3.8).

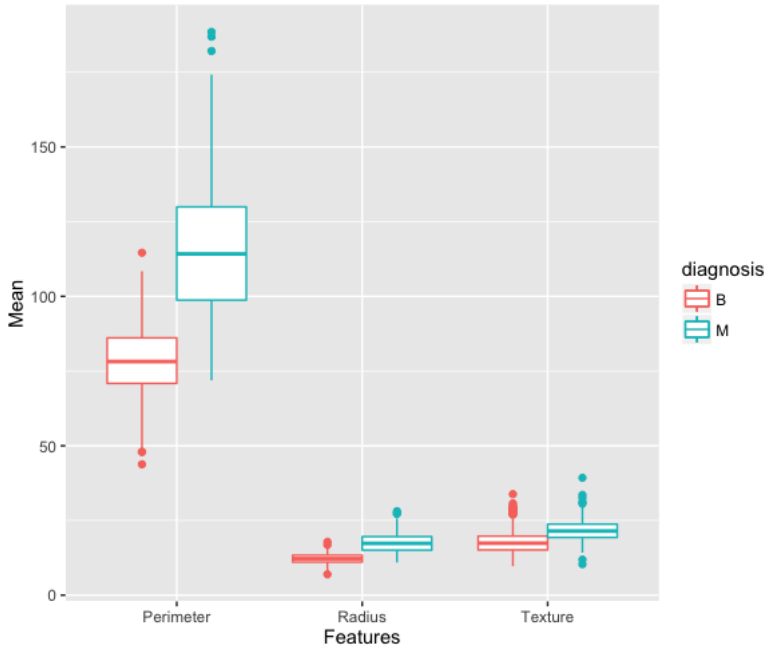


Figure 3.2: Distribution of mean values of features with large means

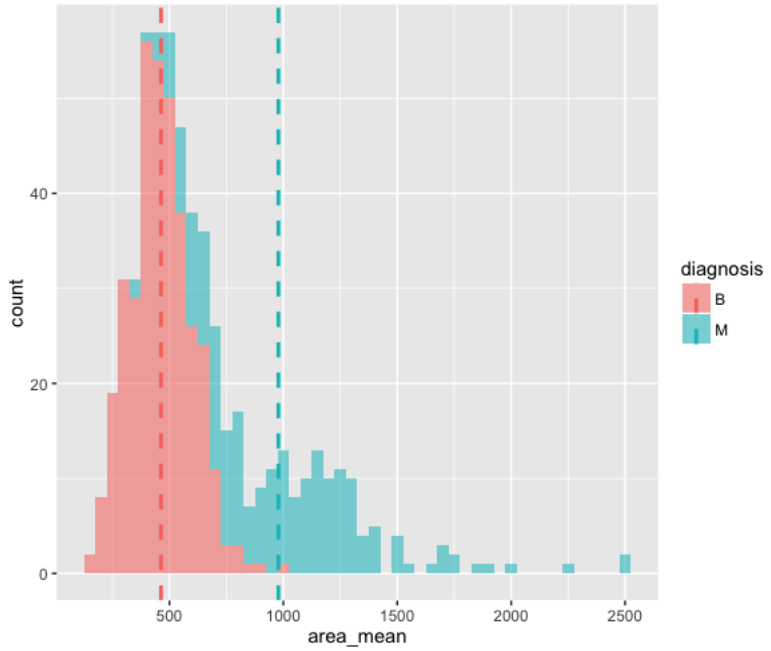


Figure 3.3: Distribution of area mean values.

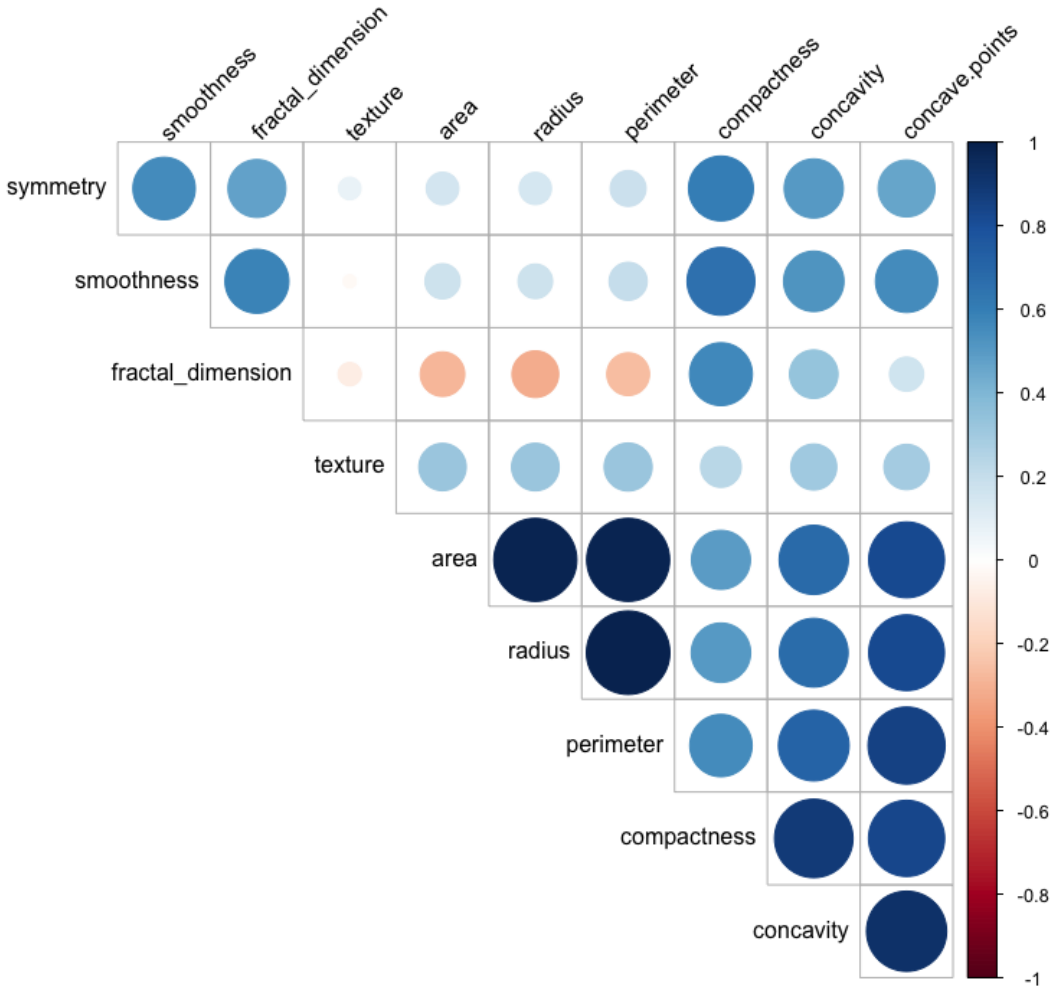


Figure 3.4: Correlation between different feature mean values. Blue indicates positive correlation, red indicates negative correlation. The intensity of the color is proportional to the correlation score.

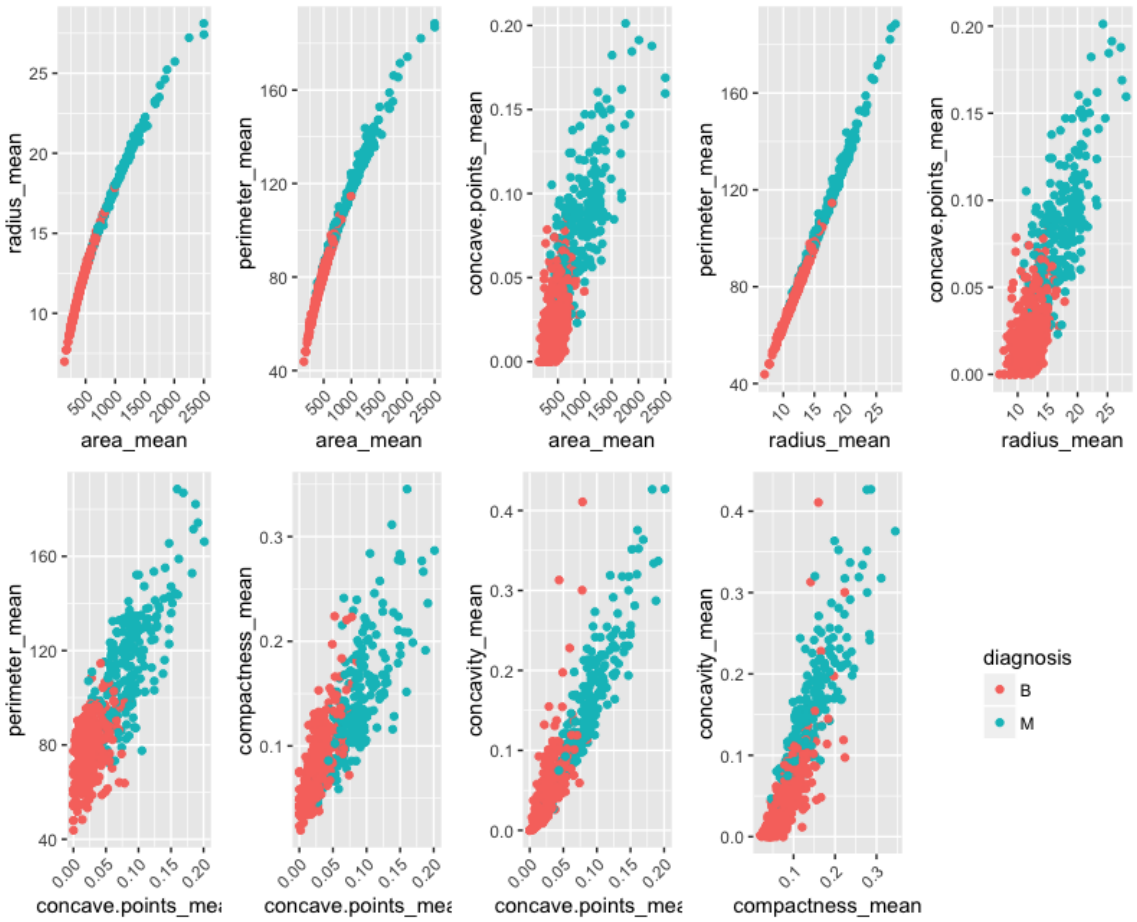


Figure 3.5: Scatterplot showing relationship between highly correlated mean features (correlation coefficient >= 0.8)

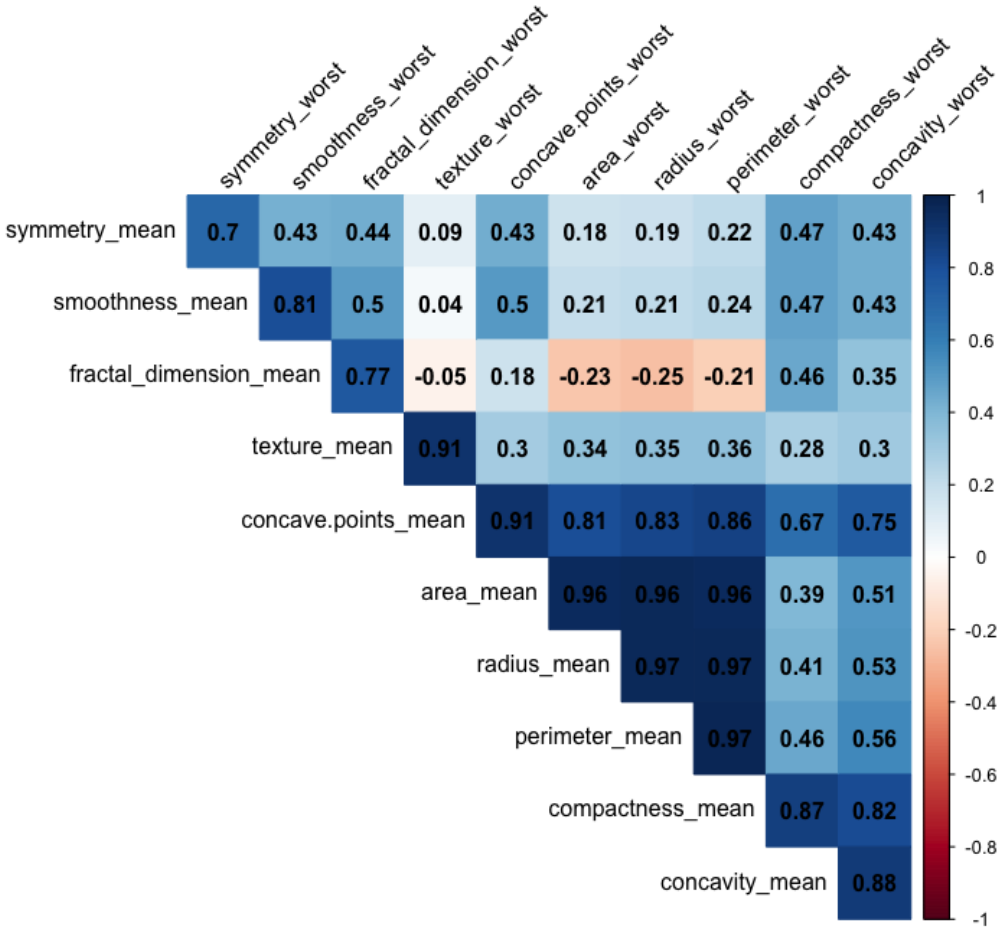


Figure 3.6: Correlation between mean and worst average among features. Blue indicates positive correlation, red indicates negative correlation. The intensity of the color is proportional to the correlation score.

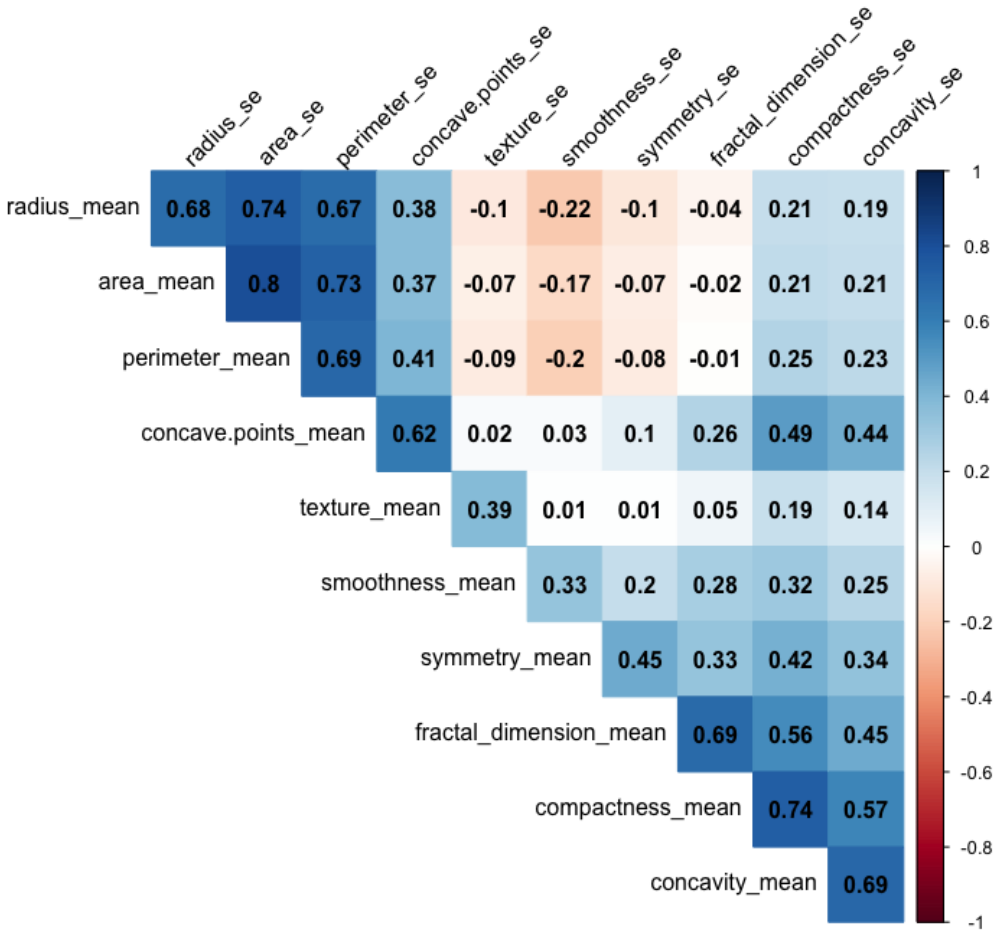


Figure 3.7: Correlation between mean and standard error among features. Blue indicates positive correlation, red indicates negative correlation. The intensity of the color is proportional to the correlation score.

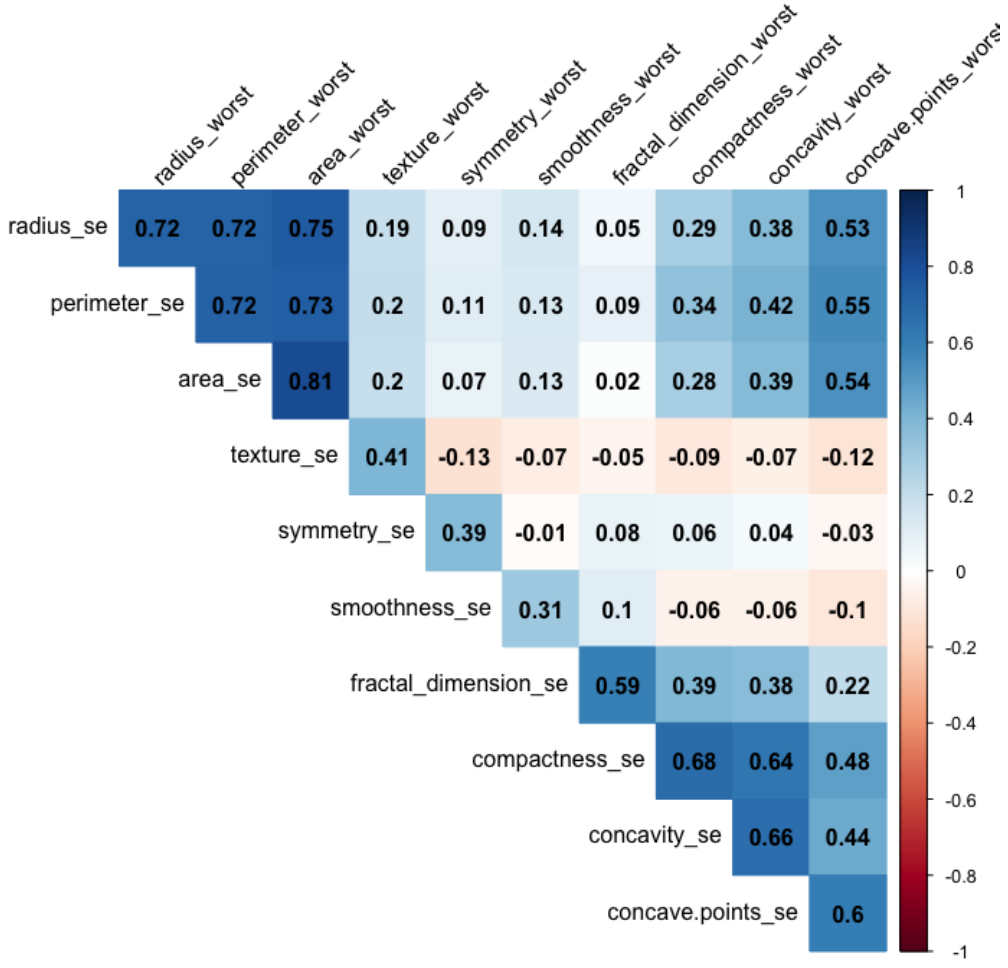


Figure 3.8: Correlation between standard error and worst average among features. Blue indicates positive correlation, red indicates negative correlation. The intensity of the color is proportional to the correlation score.

.

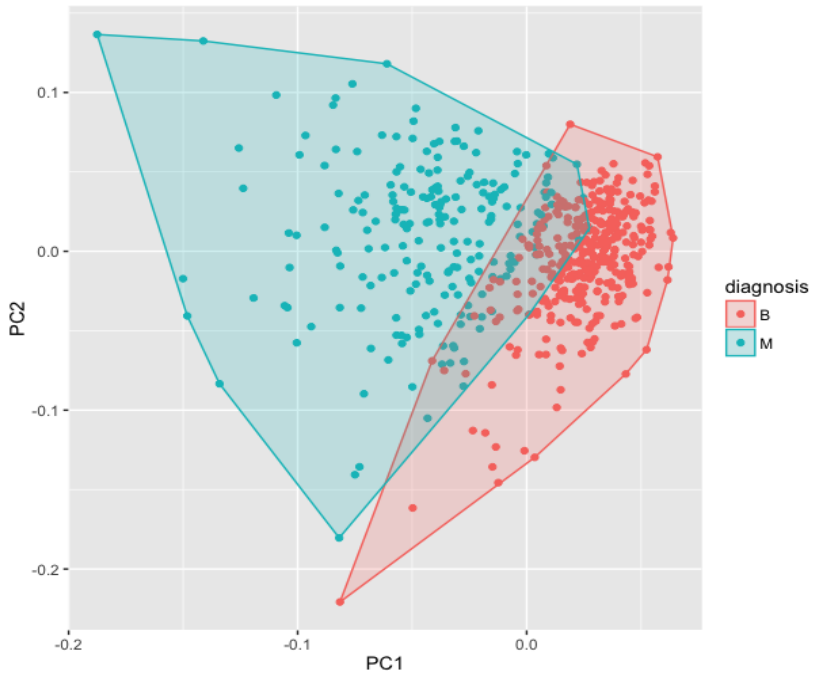


Figure 3.9: Separation of PCA transformed data on PC1 and PC2 axis

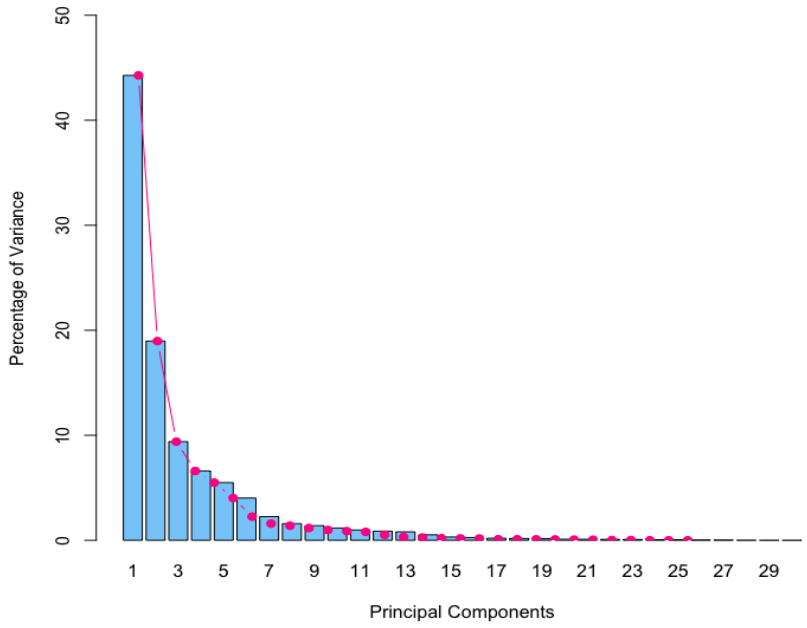


Figure 3.10: Percentage of variance by each principal component

PCA processed data

The original dataset were projected into 30 uncorrelated principal component axis planes. From Figure 3.10 we can see the 1st principal component counts for the largest amount of variation of the data, followed by the 2nd principal component *etc*. Figure 3.9 shows the separation of the data on PC1 and PC2, we can see the malignant and benign class can approximately linearly separated.

Model performance

For KNN, we can see the accuracy increase when the number of k neighbor increased from 1 to 7, then the accuracy dropped when the number of k neighbor further increased. The highest accuracy was achieved when k equals to 5 (Figure 3.11). When setting k to 5, the accuracy increased when the number of top eigenvector increases to 15, after that the accuracy dropped and stayed relatively the same when the number of top eigenvector further increased (Figure 3.12).

For SVM, three different kernels were tested, linear kernel outperformed polynomial and radial basis kernels overall (Figure 3.12). With the increasing number of top eigenvector for the data set to project on, we can see the accuracy of the linear kernel stays relative the same after a slight initial increase. While the polynomial and radial basis kernels showed a sharp decrease in accuracy after the number of top eigenvector increases after 5.

For Logistic Regression and Naive Bayes, the accuracy and the number of top eigenvector showed similar pattern. The accuracy increased in Logistic Regression and Naive Bayes when the number of top eigenvector reached to 5 and 6, respectively. Further increasing the number of top eigenvector lead to a sharp drop in accuracy (Figure 3.12).

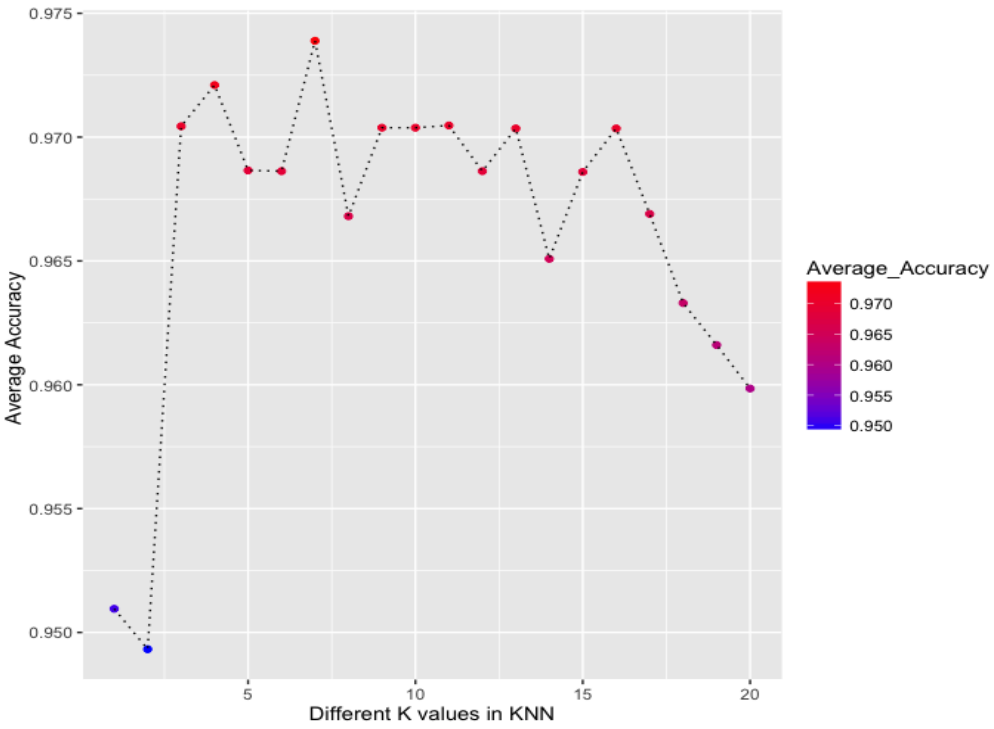


Figure 3.11: Prediction accuracy with different k values in KNN

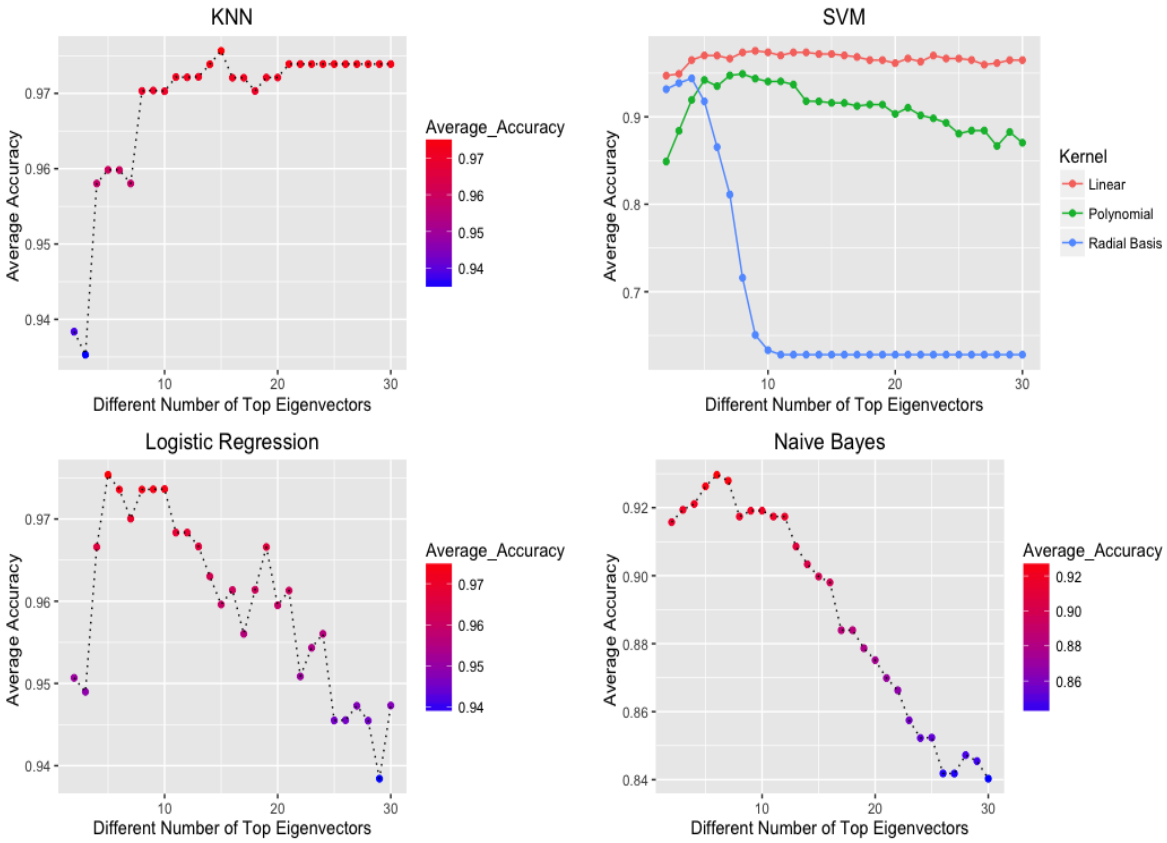


Figure 3.12: Prediction accuracy with different number of top eigenvector in four models. For KNN, the k is set to 7.

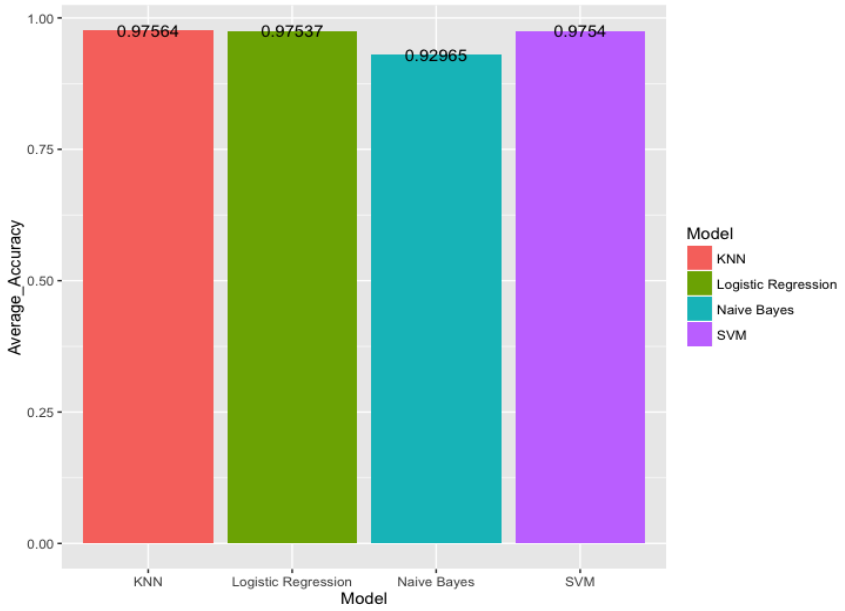


Figure 3.13: Comparison of average accuracy among four models

The highest average accuracy across the 10 fold cross-validation was achieved in KNN (accuracy = 0.97564) when k was set 5 and number of top eigenvector set to 15. SVM with linear kernel achieved the second highest accuracy (accuracy = 0.9754) when top 9 eigenvector was used. Logistic Regression also achieved a very similar average accuracy of 0.97537 when top 5 eigenvectors were included. Overall, the average accuracy of KNN, SVM with linear kernel and Logistic Regression were very similar. Naive Bayes obtained the lowest average accuracy of 0.92965 when the number of top eigenvector was set to 6.

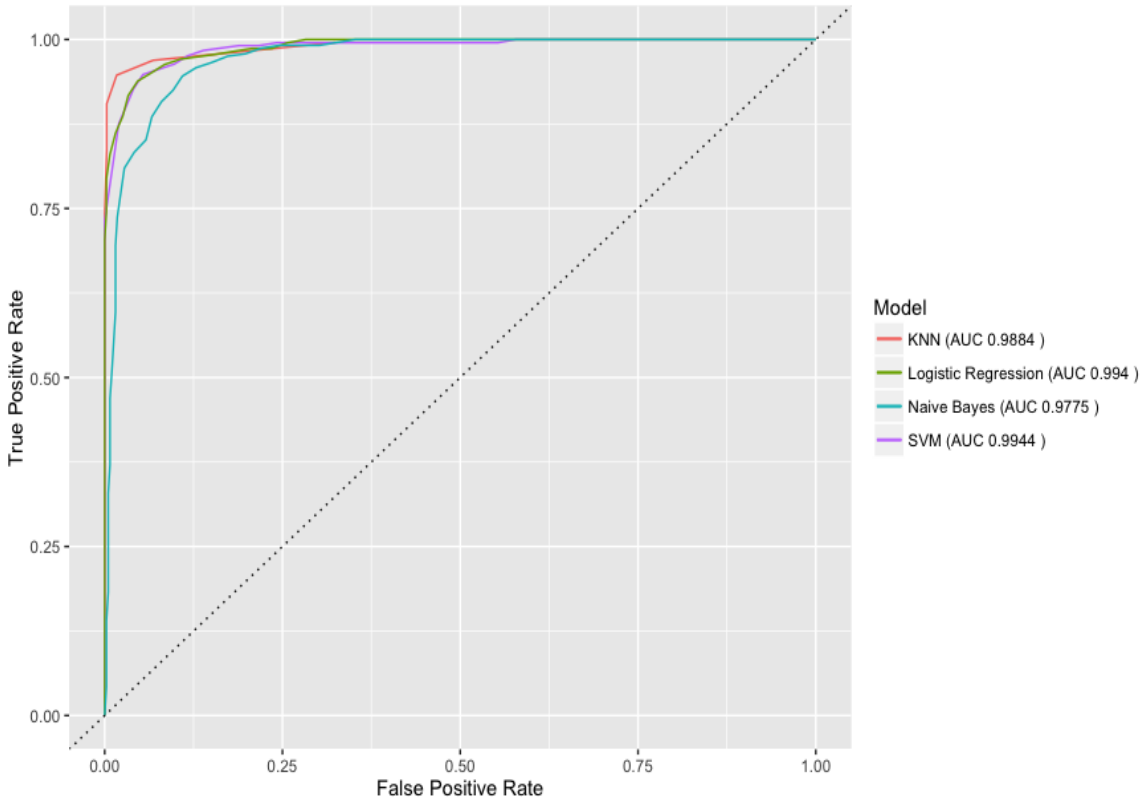


Figure 3.14: Comparison of ROC curve and average AUC of four models. TPR and FPR values are the average from 10 fold cross validation.

Among the four models, SVM with linear kernel obtained the largest AUC (AUC =0.9944), followed by Logistic Regression (AUC = 0.994) and KNN (AUC = 0.9884). Naive Bayes had the lowest AUC value of 0.9775 among the four tested models.

Chapter 4: Discussion and Conclusion

In this project, the performances of four different machine learning models on breast cancer prediction were evaluated after original data projected to PCA axis. The best performed parameter settings were explored in each model. The average accuracy of 10 fold cross-validation and AUC were compared. The effect of top eigenvector on prediction accuracy was also tested.

In PCA orthogonal linear transformation, the great variance by the projection of the data lies on the first principal component, the second greatest variance on the second coordinate, and so on. From our results, for KNN, SVM, Logistic Regression and Naive Bayes, the prediction accuracy increases as the top number of eigenvectors increases to 15, 9, 6 and 6, respectively. With the further increasing number of top eigenvectors, the accuracy showed sharp decrease, except for SVM with linear kernel where accuracy stayed relatively stable. This shows the majority of useful information is captured in first few top eigenvectors, and more eigenvectors of less importance might introduce noise, rather than signal.

K-nearest neighbor classifier (KNN), a brute-force computation of all pairs of points, classifies unknown by relating the unknown to the known using distance function. KNN is nonparametric, it has zero training time but expensive in terms of runtime. In our result, we see an initial increase in accuracy when k value increased to 7, then decrease in accuracy when k further increases. In KNN, the increase of k will reduce the effect of noise on the classification, however, large values of k makes the boundaries between classes less distinct. That is why we see a trend of decreasing accuracy when k gets large.

SVM has been largely used in breast cancer prediction (Ferreira *et al.*, 2016; Huang *et al.*, 2017). SVM requires large amount of time to learn the classifier. Our

results indicate that support vector machine with linear kernel outperforms polynomial and radial basis kernels in predicting breast cancer outcome, which is consistent with our data exploration that the data projected to PCA axis can approximately be linearly separated.

Naive Bayes has a strong assumption that the features being independent from each other. Previous studies showed that despite the conditional independence assumption, which is rarely true in real world data, Naive Bayes can still be optimal (ZHANG, 2005). In our study, we projected data to the PCA axis, and each transformed columns being independent.

Recent study has shown that SVM performs better in predicting breast cancer than Naive Bayes (Asri *et al.*, 2016). Our result also indicates that SVM is superior to Naive Bayes for this data set in terms of accuracy and AUC measurement (Figure 3.13-3.14).

Overall, comparing the four machine learning model, KNN, SVM and Logistic Regression achieved very similar accuracy, with KNN has the highest average accuracy of 0.9756 across 10 fold cross-validation. SVM with linear kernels outperforms polynomial and radial basis kernels achieving the highest AUC value of 0.9944. SVM shows as a very promising classifier for breast cancer prediction. Other classifier like neural network has also been tested in reported research. In neural network , the parameters of neural networks are optimized in discriminative supervised learning to separate patterns of different classes. However, neural network takes very long time to train the model, and the weights are hard to interpret. In future work, it might be interesting to compare neural network performance with SVM. Distributed computing tools such as OpenMP, MPI, Hadoop and Spark might also be helpful in reducing the cost of computation time.

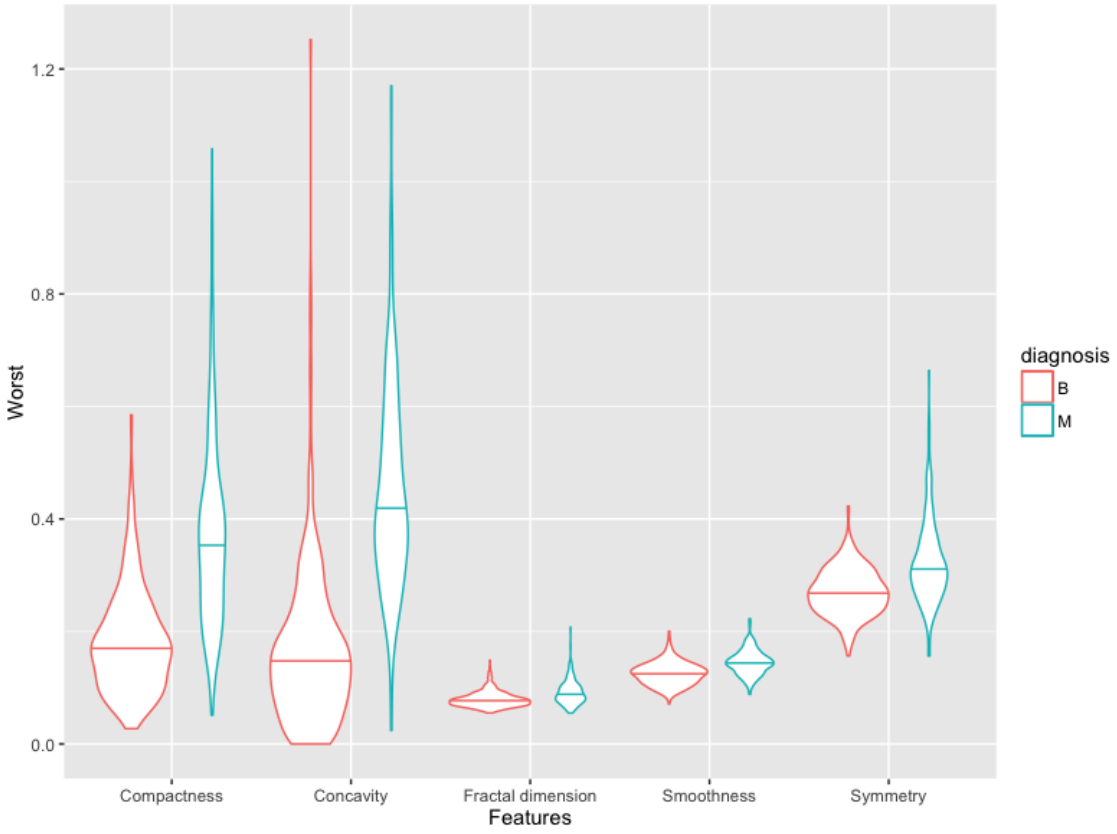
Appendices

Appendix A: Supplementary Table

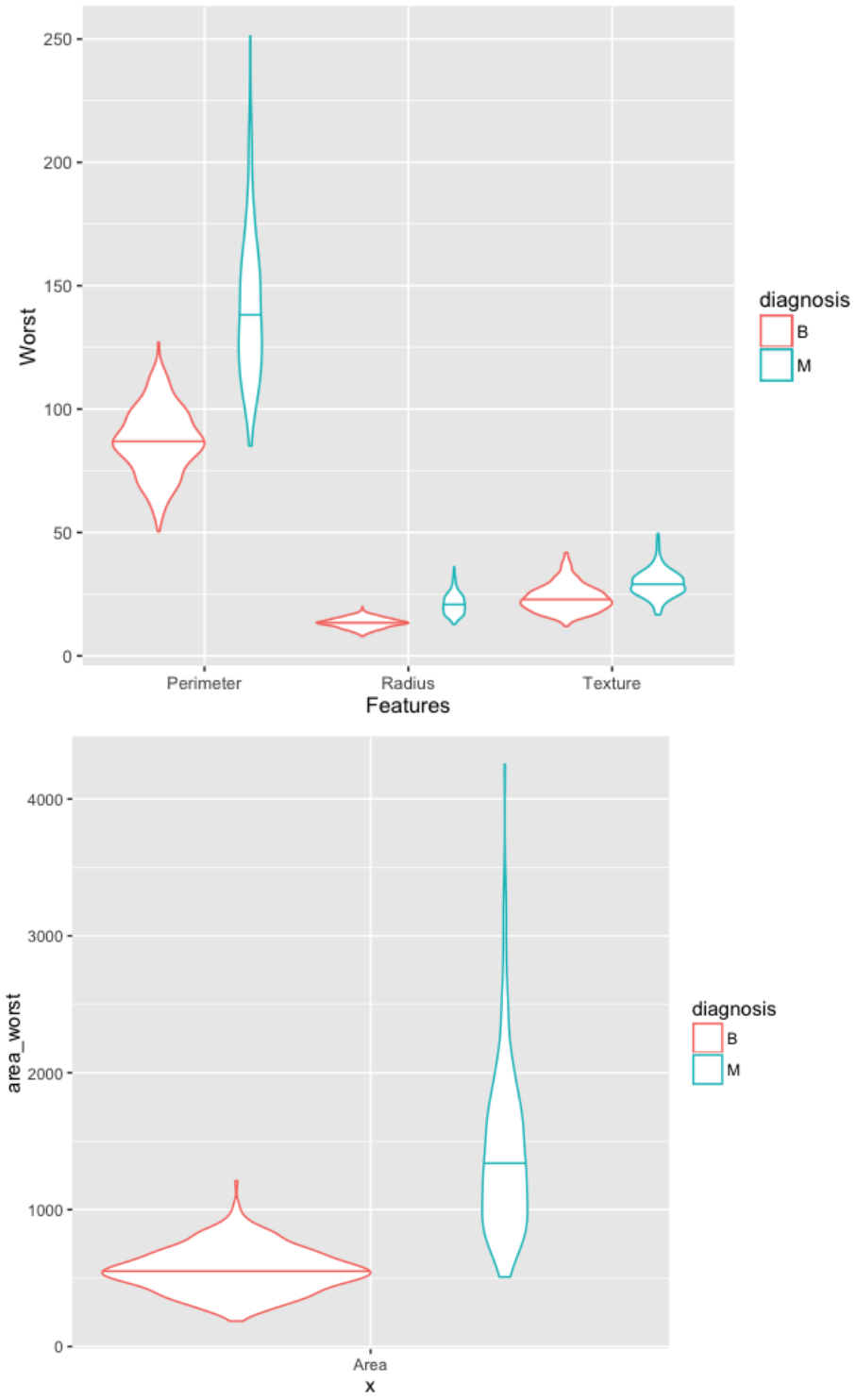
	symmetry	smoothness	fractal_dimension	texture	area	radius	perimeter	compactness	concavity	concave.points
symmetry	1	0.55777479	0.47992133	0.07140098	0.15129308	0.14774124	0.18302721	0.602641048	0.50066662	0.462497388
smoothness	0.55777479	1	0.584792002	-0.0233885	0.17702838	0.17058119	0.20727816	0.659123215	0.52198377	0.553695173
fractal_dimension	0.47992133	0.584792	1	-0.0764372	-0.2831098	-0.3116308	-0.2614769	0.565368663	0.33678336	0.166917383
texture	0.07140098	-0.0233885	-0.076437183	1	0.3210857	0.32378189	0.32953306	0.236702222	0.30241783	0.293464051
area	0.15129308	0.17702838	-0.283109812	0.3210857	1	0.98735717	0.9865068	0.498501682	0.68598283	0.823268869
radius	0.14774124	0.17058119	-0.311630826	0.32378189	0.98735717	1	0.99785528	0.506123578	0.67676355	0.822528522
perimeter	0.18302721	0.20727816	-0.261476908	0.32953306	0.9865068	0.99785528	1	0.556936211	0.71613565	0.850977041
compactness	0.60264105	0.65912322	0.565368663	0.23670222	0.49850168	0.50612358	0.55693621	1	0.88312067	0.831135043
concavity	0.50066662	0.52198377	0.336783359	0.30241783	0.68598283	0.67676355	0.71613565	0.88312067	1	0.921391026
concave.points	0.46249739	0.55369517	0.166917383	0.29346405	0.82326887	0.82252852	0.85097704	0.831135043	0.92139103	1

Appendix Table A.1: Pearson correlation score between mean features

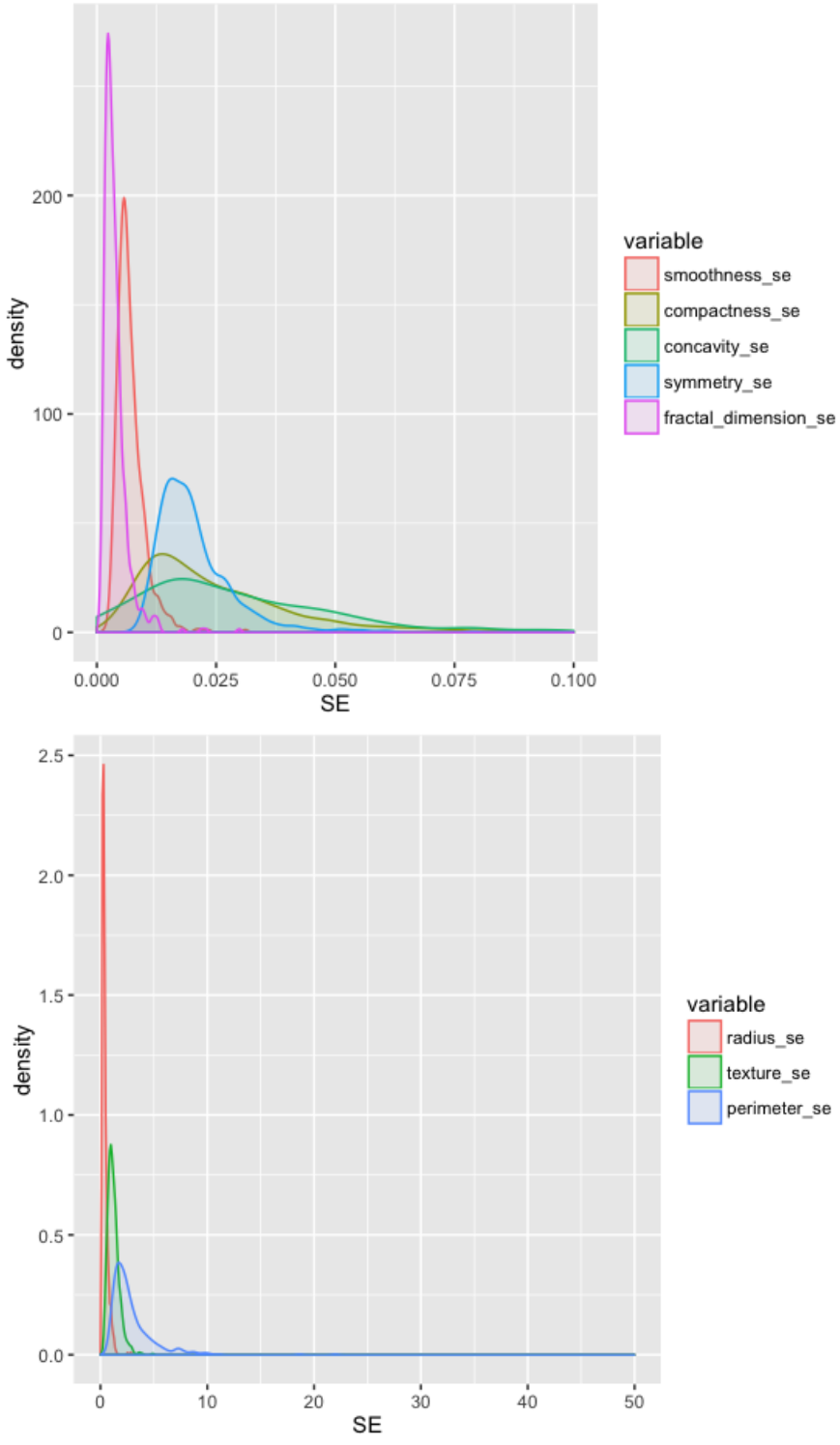
Appendix B: Supplementary Figures



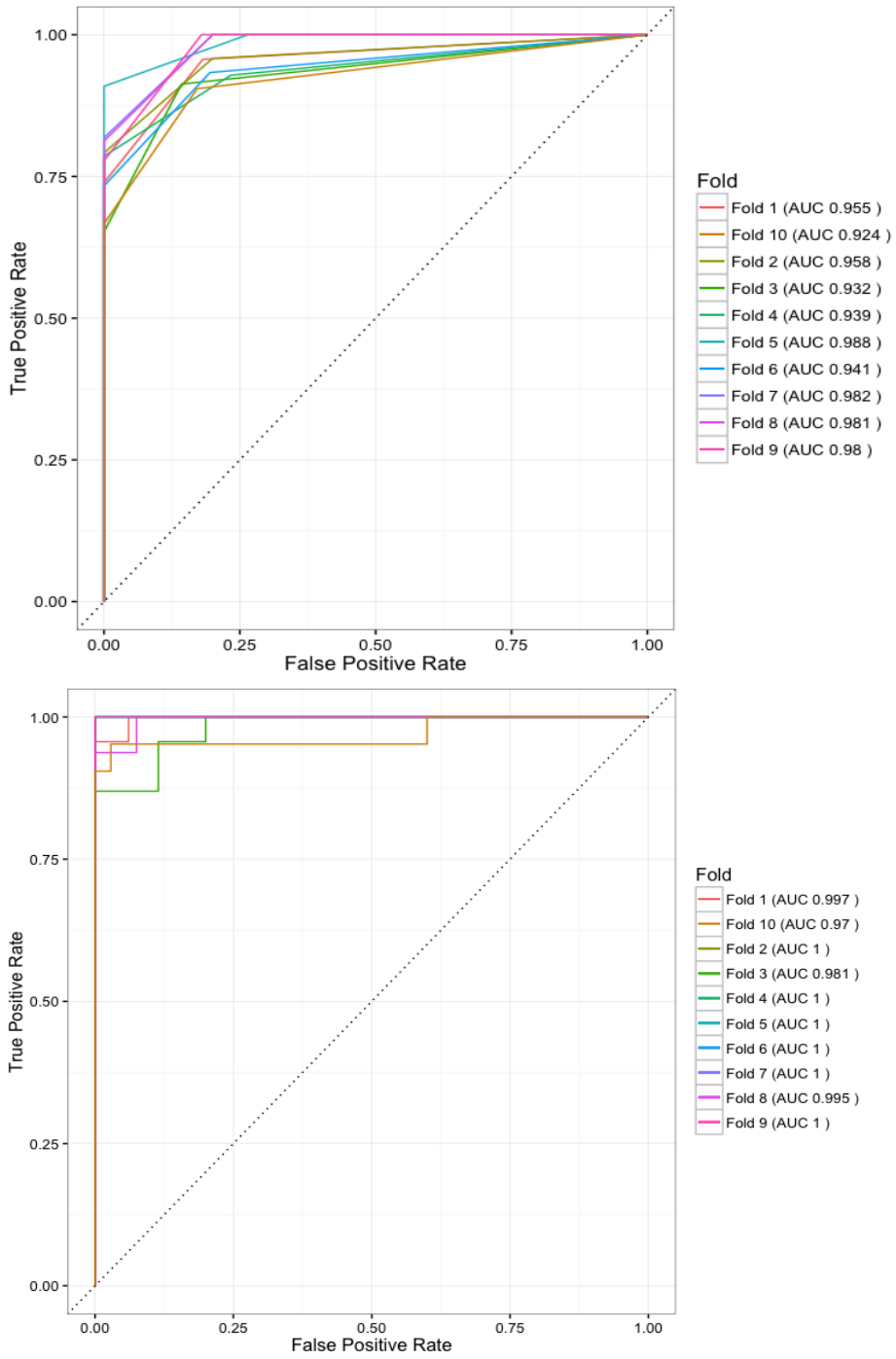
Appendix Figure B.1: Distribution of small worst average values.



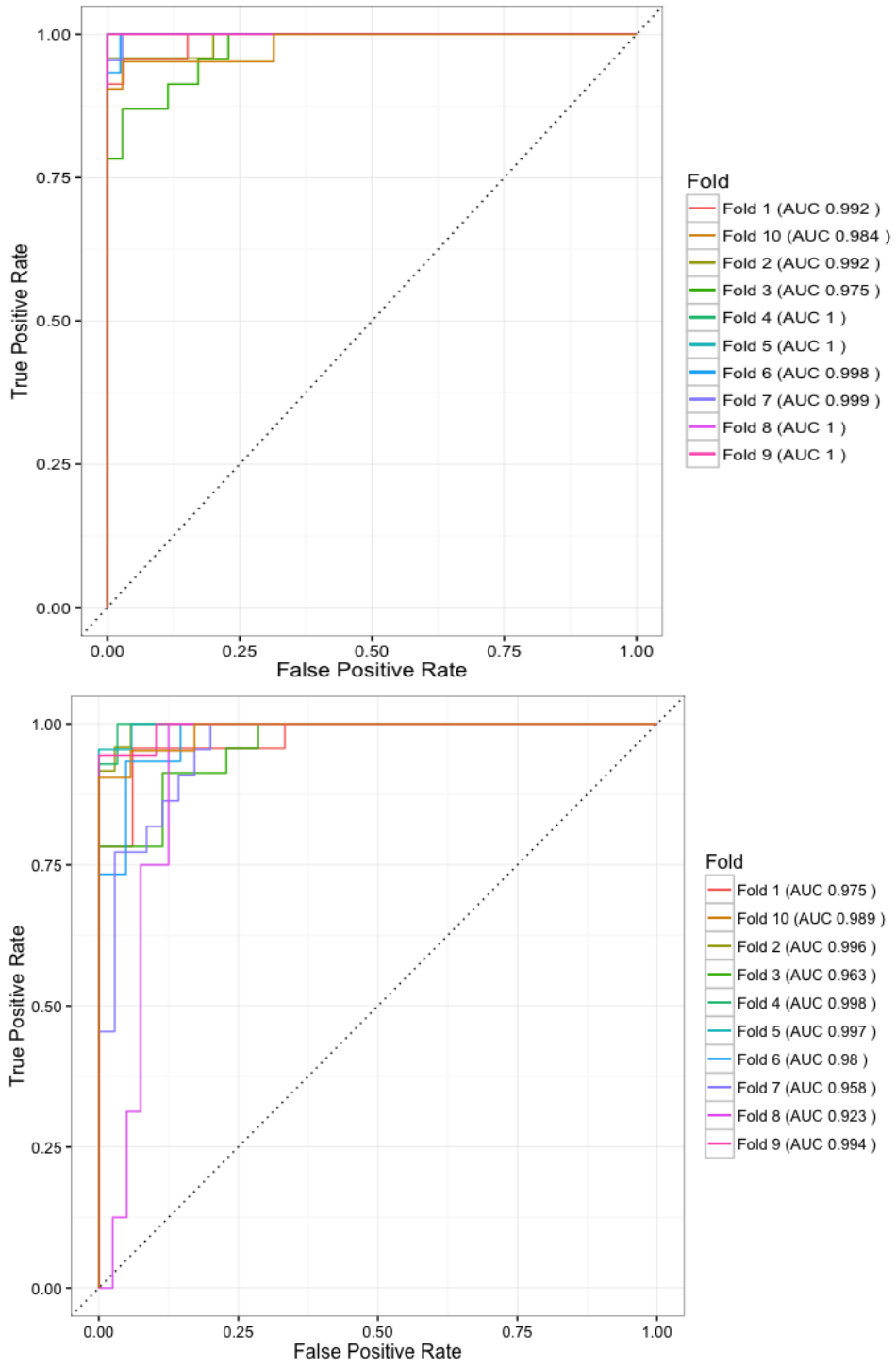
Appendix Figure B.2: Distribution of large worst average values.



Appendix Figure B.3: Distribution of standard error values across 10 features.



Appendix Figure B.4: ROC curve for 10 fold cross-validation for KNN and SVM. Top: KNN. Bottom: SVM linear kernel



Appendix Figure B.5: ROC curve for 10 fold cross-validation for Logistic Regression and Naive Bayes. Top: Logistic Regression. Bottom: Naive Bayes

Appendix C: Program in R

```
# load data
Breast_Cancer_table <- read.csv("data.csv",header = TRUE)
Breast_Cancer_table <- data.frame(Breast_Cancer_table)

# last column is empty, get rid of last column
Breast_Cancer_table <- Breast_Cancer_table[,-ncol(Breast_Cancer_table)]

# > nrow(Breast_Cancer_table)
# [1] 569
# > ncol(Breast_Cancer_table)
# [1] 32

# subset data into mean, se and worst
Breast_Cancer_mean <- Breast_Cancer_table[,2:12]
Breast_Cancer_SE <- Breast_Cancer_table[,c(2,13:22)]
Breast_Cancer_Worst <- Breast_Cancer_table[,c(2,23:32)]

# visualiza distribution of Breast_cancer_mean data
# smaller mean groups
mean_smallValue <-
 ggplot(Breast_Cancer_mean,aes(x="Smoothness",y=smoothness_mean)) +
 geom_boxplot(aes(colour=diagnosis)) +
  geom_boxplot(aes(x="Compactness",y=compactness_mean,colour=diagnosis)) +
 geom_boxplot(aes(x="Concavity",y=concavity_mean,colour=diagnosis)) +

geom_boxplot(aes(x="Symmetry",y=symmetry_mean,colour=diagnosis)) +
 geom_boxplot(aes(x="Fractaldimension",y=fractal_dimension_mean,colour=diagnosis))
 + xlab("Features") + ylab("Mean")

# larger mean groups
mean_largeValue <-ggplot(Breast_Cancer_mean,aes(x="Radius", y=radius_mean)) +
 geom_boxplot(aes(colour=diagnosis)) +
 geom_boxplot(aes(x="Texture",y=texture_mean,colour=diagnosis)) +
  geom_boxplot(aes(x="Perimeter",y=perimeter_mean,colour=diagnosis)) +
 xlab("Features") + ylab("Mean")

# separate area plot by itself due to large mean value
area_mean.Category <- ddply(Breast_Cancer_mean,"diagnosis",summarise,
            area_mean.Category = mean(area_mean))

mean_area<- ggplot(Breast_Cancer_mean,aes(x=area_mean,fill=diagnosis)) +
```

```
 geom_histogram(binwidth =50,alpha=0.55) +
 geom_vline(data=area_mean.Category,aes(xintercept=area_mean.Category,colour=diag
 nosis), linetype="dashed",size=1)

# test correlation between variables, use default Pearson correlation
mean_cor <- cor(Breast_Cancer_mean[,2:11])
colnames(mean_cor) <- gsub("_mean","",colnames(mean_cor))
rownames(mean_cor) <- gsub("_mean","",rownames(mean_cor))

corrplot(mean_cor,type="upper",order="hclust",tl.col="black",tl.srt=45,diag = FALSE)
write.csv(mean_cor_p,"correlation_meanValues.csv")

# visualize the relationship between those highly correlated  (rho >=0.8) variables
p1 <- ggplot(Breast_Cancer_mean,aes(x=area_mean,y=radius_mean)) +
 geom_point(aes(colour=diagnosis)) + theme(legend.position="none",
axis.text.x = element_text(angle = 45, hjust = 1))

p2<- ggplot(Breast_Cancer_mean,aes(x=area_mean,y=perimeter_mean)) +
 geom_point(aes(colour=diagnosis)) +
  theme(legend.position="none",axis.text.x = element_text(angle = 45, hjust = 1))

p3<- ggplot(Breast_Cancer_mean,aes(x=area_mean,y=concave.points_mean)) +
 geom_point(aes(colour=diagnosis)) +
  theme(legend.position="none",axis.text.x = element_text(angle = 45, hjust = 1))

p4<- ggplot(Breast_Cancer_mean,aes(x=radius_mean,y=perimeter_mean)) +
 geom_point(aes(colour=diagnosis)) +
  theme(legend.position="none",axis.text.x = element_text(angle = 45, hjust = 1))

p5<- ggplot(Breast_Cancer_mean,aes(x=radius_mean,y=concave.points_mean)) +
 geom_point(aes(colour=diagnosis)) +
  theme(legend.position="none",axis.text.x = element_text(angle = 45, hjust = 1))

p6<- ggplot(Breast_Cancer_mean,aes(x=concave.points_mean,y=perimeter_mean)) +
 geom_point(aes(colour=diagnosis)) +
  theme(legend.position="none",axis.text.x = element_text(angle = 45, hjust = 1))

p7<- ggplot(Breast_Cancer_mean,aes(x=concave.points_mean,y=compactness_mean)) +
 geom_point(aes(colour=diagnosis)) +
  theme(legend.position="none",axis.text.x = element_text(angle = 45, hjust = 1))


p8<- ggplot(Breast_Cancer_mean,aes(x=concave.points_mean,y=concavity_mean)) +
```

```
 geom_point(aes(colour=diagnosis)) +
 theme(legend.position="none",axis.text.x = element_text(angle = 45, hjust = 1))

p9<-ggplot(Breast_Cancer_mean,aes(x=compactness_mean,y=concavity_mean)) +
 geom_point(aes(colour=diagnosis))+
 theme(legend.position="none",axis.text.x = element_text(angle = 45, hjust = 1))

p<-ggplot(Breast_Cancer_mean,aes(x=compactness_mean,y=concavity_mean)) +
 geom_point(aes(colour=diagnosis))

# Function to only get the legend of ggplot
g_legend<-function(a.gplot){
  tmp <- ggplot_gtable(ggplot_build(a.gplot))
  leg <- which(sapply(tmp$grobs, function(x) x$name) == "guide-box")
  legend <- tmp$grobs[[leg]]
  legend
}

grid.arrange(p1,p2,p3,p4,p5,p6,p7,p8,p9,g_legend(p),ncol=5)

# visualize the worst values
# the horizontal line indicates 50% quantile
p_cancer_worst <-
 ggplot(Breast_Cancer_Worst,aes(x="Smoothness",y=smoothness_worst)) +
  geom_violin(aes(colour=diagnosis),scale = "count",draw_quantiles=0.5) +
  geom_violin(aes(x="Compactness",y=compactness_worst,colour=diagnosis),
              scale = "count",draw_quantiles=0.5) +
  geom_violin(aes(x="Concavity",y=concavity_worst,colour=diagnosis),
              scale = "count",draw_quantiles=0.5) +
  geom_violin(aes(x="Symmetry",y=symmetry_worst,colour=diagnosis),
              scale = "count",draw_quantiles=0.5) +
  geom_violin(aes(x="Fractal dimension",y=fractal_dimension_worst,colour=diagnosis),
              scale = "count",draw_quantiles=0.5) +
  xlab("Features") + ylab("Worst")

worst_largeValue <-ggplot(Breast_Cancer_Worst,aes(x="Radius", y=radius_worst)) +
  geom_violin(aes(colour=diagnosis),scale = "count",draw_quantiles=0.5) +
  geom_violin(aes(x="Texture", y=texture_worst,colour=diagnosis),
              scale = "count",draw_quantiles=0.5) +
  geom_violin(aes(x="Perimeter",y=perimeter_worst,colour=diagnosis),
              scale = "count",draw_quantiles=0.5) +
  xlab("Features") + ylab("Worst")
```

```
worst_area <- ggplot(Breast_Cancer_Worst,aes(x="Area", y=area_worst)) +
 geom_violin(aes(colour=diagnosis),scale = "count",draw_quantiles=0.5)

# visualizae SE
Breast_Cancer_SE_long <- reshape2::melt(Breast_Cancer_SE)
Breast_Cancer_SE_long_largeValue <- filter(Breast_Cancer_SE_long,variable
 %in% c("perimeter_se","radius_se","texture_se"))

Breast_Cancer_SE_long_smallValue <- filter(Breast_Cancer_SE_long,variable
 %in% c("compactness_se","concavity_se",
 "fractal_dimension_se","smoothness_se", "symmetry_se"))

Breast_Cancer_SE_long_area <- filter(Breast_Cancer_SE_long,variable == "area_se")

# correlation between mean and worst
cor_mean_worst <- cor(Breast_Cancer_mean[,2:11],Breast_Cancer_Worst[,2:11])

corrplot(cor_mean_worst,method="color",type = upper",order="hclust",addCoef.col =
 "black",tl.col = "black",tl.srt = 45)

# correlation between worst and se
cor_se_worst <- cor(Breast_Cancer_SE[,2:11],Breast_Cancer_Worst[,2:11])

corrplot(cor_se_worst,method="color",type = "upper",order="hclust",addCoef.col =
 "black",tl.col = "black",tl.srt = 45)

cor_mean_se <- cor(Breast_Cancer_mean[,2:11],Breast_Cancer_SE[,2:11])
corrplot(cor_mean_se,method="color",type = "upper",order="hclust",addCoef.col =
 "black",tl.col = "black",tl.srt = 45)

# use PCA to transform correlated data
PCA_Breast_Cancer <- prcomp(Breast_Cancer_table[,3:32],scale. = TRUE)

# Eigenvalue
eigenValue <- (PCA_Breast_Cancer$sdev)^2

#Variance in percentage
variance_percentage <- eigenValue*100/sum(eigenValue)

eig.data <-
 data.frame(eigenValue=eigenValue,variance_percentage=variance_percentage)
```

```
barplot(eig.data[,2],names.arg=1:nrow(eig.data), xlab="Principal Components",
 ylab="Percentage of Variance", col = "lightskyblue",ylim=c(0,50))

lines(x = 1:nrow(eig.data), eig.data[, 2], type="b", pch=19, col = "deeppink")

text(pca_barplot,par("usr")[3],labels =1:nrow(eig.data), srt=0,adj = c(1.1,3.1), xpd =
 TRUE, cex=.9)


pc1_pc2 <- autoplot(PCA_Breast_Cancer,data=Breast_Cancer_table[,2:32],
              colour='diagnosis',frame=TRUE)

Breast_Cancer_PCAtransformed <-
 predict(PCA_Breast_Cancer,newdata=Breast_Cancer_table[,2:32])


# create 10-fold cross validation
k_foldValue =10
set.seed(100)

TenFold_CrossValidation <-createFolds(c(1:nrow(Breast_Cancer_table)),
                            k=k_foldValue, list=TRUE, returnTrain=FALSE)


########### Classification with KNN ###################
##### Experiment with a series of k values for KNN ####

k_values <- c(1:20)
knn_accuracy_different_Ks <- matrix(,nrow=length(k_values),ncol=k_foldValue)

for (i in k_values){
  for (j in 1:k_foldValue ){

    testingDataRowIndex <- TenFold_CrossValidation[[j]]

    # get training data
    trainingDataRaw <- Breast_Cancer_table[-testingDataRowIndex,]
    training_label <- Breast_Cancer_table[-testingDataRowIndex,2]

    # get testing data
    testingDataRaw <- Breast_Cancer_table[testingDataRowIndex,]
    testing_label <- Breast_Cancer_table[testingDataRowIndex,2]
```

```
    # Project raw data to PCA plane
    training_PCA <- prcomp(Breast_Cancer_table[-testingDataRowIndex,3:32],scale. =
  TRUE)
    trainingData_PCA_transformed <- predict(training_PCA,newdata=trainingDataRaw)
    testingData_PCA_transformed <- predict(training_PCA,newdata=testingDataRaw)

    # Build KNN model
    knn_result<- knn(trainingData_PCA_transformed,testingData_PCA_transformed,
                training_label,k=i,prob = TRUE)

    xtab <- table(knn_result,testing_label)
    confusionMat <- confusionMatrix(xtab,positive = "M")
    knn_accuracy_different_Ks[i,j] <- confusionMat$overall[[1]]
  }
}

avg_fold_accuracy_Ks <- rowMeans(knn_accuracy_different_Ks)
avg_fold_accuracy_Ks <- data.frame(avg_fold_accuracy_Ks);
avg_fold_accuracy_Ks$K <- k_values
colnames(avg_fold_accuracy_Ks) <- c("Average_Accuracy","Ks")

# plot accuracy vs different Ks
knn_accuracy_k_p <-ggplot(avg_fold_accuracy_Ks,aes(x=Ks,y=Average_Accuracy)) +
  geom_point(aes(colour=Average_Accuracy)) +
  xlab("Different K values in KNN") + ylab("Average Accuracy ")+
  scale_colour_gradient(low="blue",high="red") +
  geom_line(linetype=3)

## Experiment with different number of top eigenvectors ##

top_eigenvector <- c(2:30)
knn_accuracy_different_topEigenvectors <-
  matrix(,nrow=length(top_eigenvector),ncol=k_foldValue)

for (i in top_eigenvector){
  for (j in 1:k_foldValue ){

    testingDataRowIndex <- TenFold_CrossValidation[[j]]

    # get training data
    trainingDataRaw <- Breast_Cancer_table[-testingDataRowIndex,]
    training_label <- Breast_Cancer_table[-testingDataRowIndex,2]
```

```
    # get testing data
    testingDataRaw <- Breast_Cancer_table[testingDataRowIndex,]
    testing_label <- Breast_Cancer_table[testingDataRowIndex,2]

    # Project raw data to PCA plane
    training_PCA <- prcomp(Breast_Cancer_table[-testingDataRowIndex,3:32],scale. =
  TRUE)
    trainingData_PCA_transformed <- predict(training_PCA,newdata=trainingDataRaw)
    testingData_PCA_transformed <- predict(training_PCA,newdata=testingDataRaw)

    # Build KNN model using different number of top eigenvectors at K=7
    knn_result<- knn(trainingData_PCA_transformed[,1:i],
                     testingData_PCA_transformed[,1:i],training_label,k=7,prob = TRUE)

    xtab <- table(knn_result,testing_label)
    confusionMat <- confusionMatrix(xtab,positive = "M");confusionMat
    knn_accuracy_different_topEigenvectors[i-1,j] <- confusionMat$overall[[1]]
  }
}

knn_avg_fold_accuracy_Eigs <- rowMeans(knn_accuracy_different_topEigenvectors)
knn_avg_fold_accuracy_Eigs <- data.frame(knn_avg_fold_accuracy_Eigs);
knn_avg_fold_accuracy_Eigs$Number_Eigenvector <- top_eigenvector
colnames(knn_avg_fold_accuracy_Eigs) <-
 c("Average_Accuracy","Top_Eigenvector_Number")

# plot accuracy vs different top number of eigenvectors
knn_accuracy_eig_p <-
 ggplot(knn_avg_fold_accuracy_Eigs,aes(x=Top_Eigenvector_Number,
                                       y=Average_Accuracy)) +
  geom_point(aes(colour=Average_Accuracy)) +
  xlab("Different Number of Top Eigenvectors") + ylab("Average Accuracy ")+
  scale_colour_gradient(low="blue",high="red") +
  geom_line(linetype=3) + ggtitle("KNN")

# The Best performed KNN is k=7, topEigenvector_num=15, plot ROC curve
KNN_AUC_list <- NULL # initiate a list to store AUC values in each fold
KNN_FPR_table <- matrix(,nrow = 60,ncol = 10)
KNN_TPR_table <- matrix(,nrow = 60,ncol=10)

KNN_10fold_ROC <- ggplot(data = NULL,aes(x=FPR,y=TPR)) + theme_bw() +
  xlab("False Positive Rate") + ylab("True Positive Rate")
```

```
for (j in 1:k_foldValue ){

 testingDataRowIndex <- TenFold_CrossValidation[[j]]

 # get training data
 trainingDataRaw <- Breast_Cancer_table[-testingDataRowIndex,]
 training_label <- Breast_Cancer_table[-testingDataRowIndex,2]

 # get testing data
 testingDataRaw <- Breast_Cancer_table[testingDataRowIndex,]
 testing_label <- Breast_Cancer_table[testingDataRowIndex,2]

 # Project raw data to PCA plane
 training_PCA <- prcomp(Breast_Cancer_table[-testingDataRowIndex,3:32],scale. =
TRUE)
 trainingData_PCA_transformed <- predict(training_PCA,newdata=trainingDataRaw)
 testingData_PCA_transformed <- predict(training_PCA,newdata=testingDataRaw)

 # Build KNN model using different number of top eigenvectors at K=7
 knn_result<-
knn(trainingData_PCA_transformed[,1:15],testingData_PCA_transformed[,1:15],
            training_label,k=7,prob = TRUE)

 # Calculate ROC curve table
 prob_knn <- attr(knn_result,"prob")
 prob_knn <- 2*ifelse(knn_result == "B", 1-prob_knn, prob_knn) - 1
 pred_knn <- prediction(prob_knn,testing_label)
 perf_knn <- performance(pred_knn,"tpr","fpr")

 FPR <-slot(perf_knn,"x.values")[[1]] # get False Positive Rate- x.values
 TPR <-slot(perf_knn,"y.values")[[1]] # get True Positive Rate- x.values

 ROC_table <- NULL
 ROC_table$FPR <- FPR
 KNN_FPR_table[1:length(FPR),j] = FPR

 ROC_table$TPR <- TPR
 KNN_TPR_table[1:length(TPR),j] = TPR

 # get AUC value
 auc_knn <- performance(pred_knn,measure = "auc")
 auc_value <- slot(auc_knn,"y.values")[[1]]
 KNN_AUC_list[j] <- auc_value
```

```
 ROC_table$Fold <- paste("Fold",toString(j),"(AUC",toString(round(auc_value,3)),")")
 ROC_table <- data.frame(ROC_table)

 KNN_10fold_ROC <- KNN_10fold_ROC +
 geom_line(data=ROC_table,aes(colour=Fold))
}

# plot diagnoal line
KNN_ROC <- KNN_10fold_ROC + geom_abline(slope=1,linetype=3)

############## Classification with SVM ####################
# tune parameter for linear, polynomial and radial basis kernels using
tune(svm, train.x=trainingData_PCA_transformed,
 train.y=training_label,kernel="linear",ranges=list(cost=10^(-1:2), gamma=c(.5,1,2)))

# For linear: cost =10, gamma = 0.5
# For Polynomial: cost =0.1, gamma = 0.5
# For Radial Basis: cost =0.1, gamma = 0.5

## Experiment with different number of top eigenvectors ##
svm_linear_accuracy_different_topEigenvectors <-
 matrix(,nrow=length(top_eigenvector),ncol=k_foldValue)

svm_poly_accuracy_different_topEigenvectors <-
 matrix(,nrow=length(top_eigenvector),ncol=k_foldValue)

svm_radial_accuracy_different_topEigenvectors <-
 matrix(,nrow=length(top_eigenvector),ncol=k_foldValue)

for (i in top_eigenvector){
 for (j in 1:k_foldValue ){

   testingDataRowIndex <- TenFold_CrossValidation[[j]]

   # get training data
   trainingDataRaw <- Breast_Cancer_table[-testingDataRowIndex,]
   training_label <- Breast_Cancer_table[-testingDataRowIndex,2]

   # get testing data
   testingDataRaw <- Breast_Cancer_table[testingDataRowIndex,]
   testing_label <- Breast_Cancer_table[testingDataRowIndex,2]
```

```
  # Project raw data to PCA plane
  training_PCA <- prcomp(Breast_Cancer_table[-testingDataRowIndex,3:32],scale. =
TRUE)

  trainingData_PCA_transformed <- predict(training_PCA,newdata=trainingDataRaw)
  testingData_PCA_transformed <- predict(training_PCA,newdata=testingDataRaw)

  # test linear, polynomial and radial basis models
  svm_model <- svm(training_label ~ ., data=trainingData_PCA_transformed[,1:i],
    kernel = "radial", cost=010,gamma=0.5)
  svm_result <- predict(svm_model,testingData_PCA_transformed[,1:i],
                  decision.values = TRUE)

  svm_xtab <- table(svm_result,testing_label); svm_xtab
  svm_confusionMat <- confusionMatrix(svm_xtab,positive = "M")

  svm_linear_accuracy_different_topEigenvectors[i-1,j] <-
svm_confusionMat$overall[[1]]

  #svm_poly_accuracy_different_topEigenvectors[i-1,j] <-
svm_confusionMat$overall[[1]]
  #svm_radial_accuracy_different_topEigenvectors[i-1,j] <-
svm_confusionMat$overall[[1]]
 }
}

# average accuracy in linear kernel
svm_linear_avg_fold_accuracy_Eigs <-
 rowMeans(svm_linear_accuracy_different_topEigenvectors)

svm_linear_avg_fold_accuracy_Eigs <-
 data.frame(svm_linear_avg_fold_accuracy_Eigs);

svm_linear_avg_fold_accuracy_Eigs$Number_Eigenvector <- top_eigenvector
svm_linear_avg_fold_accuracy_Eigs$Kernel <- "Linear"

colnames(svm_linear_avg_fold_accuracy_Eigs) <-
 c("Average_Accuracy","Top_Eigenvector_Number","Kernel")

# average accuracy in polynomial kernel
svm_poly_avg_fold_accuracy_Eigs <-
 rowMeans(svm_poly_accuracy_different_topEigenvectors)
```

```
svm_poly_avg_fold_accuracy_Eigs <- data.frame(svm_poly_avg_fold_accuracy_Eigs);

svm_poly_avg_fold_accuracy_Eigs$Number_Eigenvector <- top_eigenvector
svm_poly_avg_fold_accuracy_Eigs$Kernel <- "Polynomial"

colnames(svm_poly_avg_fold_accuracy_Eigs) <-
  c("Average_Accuracy","Top_Eigenvector_Number","Kernel")

# average accuracy in radial basis kernel
svm_radial_avg_fold_accuracy_Eigs <-
 rowMeans(svm_radial_accuracy_different_topEigenvectors)

svm_radial_avg_fold_accuracy_Eigs <-
 data.frame(svm_radial_avg_fold_accuracy_Eigs);

svm_radial_avg_fold_accuracy_Eigs$Number_Eigenvector <- top_eigenvector
svm_radial_avg_fold_accuracy_Eigs$Kernel <- "Radial Basis"

colnames(svm_radial_avg_fold_accuracy_Eigs) <-
  c("Average_Accuracy","Top_Eigenvector_Number","Kernel")

# combine rows
svm_acc_eig_combined <- rbind(svm_linear_avg_fold_accuracy_Eigs,
                        svm_poly_avg_fold_accuracy_Eigs,
                        svm_radial_avg_fold_accuracy_Eigs)

# plot accuracy vs different top eigenvector numbers
svm_accuracy_eig_p <-ggplot(svm_acc_eig_combined,
                      aes(x=Top_Eigenvector_Number,y=Average_Accuracy)) +
  geom_line(aes(colour=Kernel)) + geom_point(aes(colour=Kernel)) +
  xlab("Different Number of Top Eigenvectors") + ylab("Average Accuracy ") +
 ggtitle("SVM")

# The highest accuracy is achieved when top 9 eigenvectors were used and using linear
  kernel
# get corresponding average AUC list and ROC curve
SVM_AUC_list <- NULL # initiate a list to store AUC values in each fold
SVM_FPR_table <- matrix(,nrow = 60,ncol = 10)
SVM_TPR_table <- matrix(,nrow = 60,ncol=10)

SVM_10fold_ROC <- ggplot(data = NULL,aes(x=FPR,y=TPR)) + theme_bw() +
  xlab("False Positive Rate") + ylab("True Positive Rate")
```

```
for (j in 1:k_foldValue ){

 testingDataRowIndex <- TenFold_CrossValidation[[j]]

 # get training data
 trainingDataRaw <- Breast_Cancer_table[-testingDataRowIndex,]
 training_label <- Breast_Cancer_table[-testingDataRowIndex,2]

 # get testing data
 testingDataRaw <- Breast_Cancer_table[testingDataRowIndex,]
 testing_label <- Breast_Cancer_table[testingDataRowIndex,2]

 # Project raw data to PCA plane
 training_PCA <- prcomp(Breast_Cancer_table[-testingDataRowIndex,3:32],scale. =
TRUE)

 trainingData_PCA_transformed <- predict(training_PCA,newdata=trainingDataRaw)
 testingData_PCA_transformed <- predict(training_PCA,newdata=testingDataRaw)

 # test linear, polynomial and radial basis models
 svm_model <- svm(training_label ~ ., data=trainingData_PCA_transformed[,1:9],kernel
= "linear", cost=10,gamma=0.5)

 svm_result <- predict(svm_model,testingData_PCA_transformed[,1:9],decision.values
= TRUE)

 svm.pred <- prediction(attributes(svm_result)$decision.values,testing_label)

 svm.roc <- performance(svm.pred, 'tpr', 'fpr')

 FPR <-slot(svm.roc,"x.values")[[1]] # get False Positive Rate- x.values
 TPR <-slot(svm.roc,"y.values")[[1]] # get True Positive Rate- x.values

 ROC_table <- NULL
 ROC_table$FPR <- FPR
 SVM_FPR_table[1:length(FPR),j] = FPR

 ROC_table$TPR <- TPR
 SVM_TPR_table[1:length(TPR),j] = TPR

 # get AUC value
 svm.auc <- performance(svm.pred, 'auc');
```

```
 svm.auc.value <- slot(svm.auc,"y.values")[[1]]
 SVM_AUC_list[j] <- svm.auc.value

 ROC_table$Fold <-
paste("Fold",toString(j),"(AUC",toString(round(svm.auc.value,3)),")")
 ROC_table <- data.frame(ROC_table)

 SVM_10fold_ROC <- SVM_10fold_ROC +
geom_line(data=ROC_table,aes(colour=Fold))
}

# plot diagnoal line
SVM_10fold_ROC + geom_abline(slope=1,linetype=3)


### Classification with Logistic Regression ######

########Experiment with different #################
logit_accuracy_different_topEigenvectors <- matrix(,nrow=length(top_eigenvector),
                                    ncol=k_foldValue)

for (i in top_eigenvector){
 for (j in 1:k_foldValue ){

   testingDataRowIndex <- TenFold_CrossValidation[[j]]

   # get training data
   trainingDataRaw <- Breast_Cancer_table[-testingDataRowIndex,]
   training_label <- Breast_Cancer_table[-testingDataRowIndex,2]

   # get testing data
   testingDataRaw <- Breast_Cancer_table[testingDataRowIndex,]
   testing_label <- Breast_Cancer_table[testingDataRowIndex,2]

   # Project raw data to PCA plane
   training_PCA <- prcomp(Breast_Cancer_table[-testingDataRowIndex,3:32],scale. =
TRUE)

   trainingData_PCA_transformed <- predict(training_PCA,newdata=trainingDataRaw)
   testingData_PCA_transformed <- predict(training_PCA,newdata=testingDataRaw)

   # build logistic regression model with training data
   logit_model <-glm(training_label ~., data.frame(trainingData_PCA_transformed)[,1:i],
```

```
                family = binomial())

  # apply model to testing adata
  logit_result <-
 predict(logit_model,data.frame(testingData_PCA_transformed)[,1:i],type = "response")

  cutoff <- 0.5
  logit_predict_label <- ifelse(logit_result >= cutoff, "M","B")
  logit_xtab <- table(logit_predict_label,testing_label)

  logit_confusionMat <- confusionMatrix(logit_xtab,positive = "M")

  # store accuracy value
  logit_accuracy_different_topEigenvectors[i-1,j] <- logit_confusionMat$overall[[1]]
 }
}

# average accuracy in logistic regression
logit_avg_fold_accuracy_Eigs <- rowMeans(logit_accuracy_different_topEigenvectors)

logit_avg_fold_accuracy_Eigs <- data.frame(logit_avg_fold_accuracy_Eigs);

logit_avg_fold_accuracy_Eigs$Number_Eigenvector <- top_eigenvector

colnames(logit_avg_fold_accuracy_Eigs) <-
 c("Average_Accuracy","Top_Eigenvector_Number")

# plot
logit_accuracy_eig_p <-ggplot(logit_avg_fold_accuracy_Eigs,
 aes(x=Top_Eigenvector_Number,y=Average_Accuracy)) +
 geom_point(aes(colour=Average_Accuracy)) +
       xlab("Different Number of Top Eigenvectors") +     ylab("Average Accuracy ")+
       scale_colour_gradient(low="blue",high="red") +
 geom_line(linetype=3) + ggtitle("Logistic Regression")

# The highest accuracy is achieved when top 5 eigenvectors
# get corresponding average AUC list and ROC curve
Logit_AUC_list <- NULL
# initiate a list to store AUC values in each fold

Logit_FPR_table <- matrix(,nrow = 60,ncol = 10)
Logit_TPR_table <- matrix(,nrow = 60,ncol=10)
```

```
logit_10fold_ROC <- ggplot(data = NULL,aes(x=FPR,y=TPR)) + theme_bw() +
  xlab("False Positive Rate") + ylab("True Positive Rate")

for (j in 1:k_foldValue ){

  testingDataRowIndex <- TenFold_CrossValidation[[j]]

  # get training data
  trainingDataRaw <- Breast_Cancer_table[-testingDataRowIndex,]
  training_label <- Breast_Cancer_table[-testingDataRowIndex,2]

  # get testing data
  testingDataRaw <- Breast_Cancer_table[testingDataRowIndex,]
  testing_label <- Breast_Cancer_table[testingDataRowIndex,2]

  # Project raw data to PCA plane
  training_PCA <- prcomp(Breast_Cancer_table[-testingDataRowIndex,3:32],scale. =
 TRUE)

  trainingData_PCA_transformed <- predict(training_PCA,newdata=trainingDataRaw)
  testingData_PCA_transformed <- predict(training_PCA,newdata=testingDataRaw)

  # build logistic regression model with training data
  logit_model <-glm(training_label ~., data.frame(trainingData_PCA_transformed)[,1:5],
                    family = binomial())

  # apply model to testing data
  logit_result <- predict(logit_model,data.frame(testingData_PCA_transformed)[,1:5],
                          type = "response")

  logit_pred <- prediction(logit_result,testing_label)

  # AUC
  logit_auc <- performance(logit_pred,measure = "auc")
  logit_auc_value <- slot(logit_auc,"y.values")[[1]]
  Logit_AUC_list[j] <- logit_auc_value

  # ROC
  logit_roc <- performance(logit_pred,"tpr","fpr")

  FPR <-slot(logit_roc,"x.values")[[1]] # get False Positive Rate- x.values
  TPR <-slot(logit_roc,"y.values")[[1]] # get True Positive Rate- x.values
```

```
 ROC_table <- NULL
 ROC_table$FPR <- FPR
 Logit_FPR_table[1:length(FPR),j] = FPR

 ROC_table$TPR <- TPR
 Logit_TPR_table[1:length(TPR),j] = TPR

 ROC_table$Fold <-
paste("Fold",toString(j),"(AUC",toString(round(logit_auc_value,3)),")")
 ROC_table <- data.frame(ROC_table)

 logit_10fold_ROC <- logit_10fold_ROC +
geom_line(data=ROC_table,aes(colour=Fold))
}
# plot diagnoal line
logit_10fold_ROC + geom_abline(slope=1,linetype=3)

######## Classification with Naive Bayes ############
## Experiment with different number of top eigenvectors ##
nb_accuracy_different_topEigenvectors <-
 matrix(,nrow=length(top_eigenvector),ncol=k_foldValue)

for (i in top_eigenvector){
 for (j in 1:k_foldValue ){

   testingDataRowIndex <- TenFold_CrossValidation[[j]]

   # get training data
   trainingDataRaw <- Breast_Cancer_table[-testingDataRowIndex,]
   training_label <- Breast_Cancer_table[-testingDataRowIndex,2]

   # get testing data
   testingDataRaw <- Breast_Cancer_table[testingDataRowIndex,]
   testing_label <- Breast_Cancer_table[testingDataRowIndex,2]

   # Project raw data to PCA plane
   training_PCA <- prcomp(Breast_Cancer_table[-testingDataRowIndex,3:32],scale. =
TRUE)

   trainingData_PCA_transformed <- predict(training_PCA,newdata=trainingDataRaw)
   testingData_PCA_transformed <- predict(training_PCA,newdata=testingDataRaw)

   # Build Naive Bayes model using different number of top eigenvectors
```

```
    nb_model <- naiveBayes(training_label ~ .,
      data= data.frame(trainingData_PCA_transformed)[,1:i])

    nb_result <- predict(nb_model, data.frame(testingData_PCA_transformed)[,1:i])

    nb_xtab <- table(nb_result,testing_label)

    nb_confusionMat <- confusionMatrix(nb_xtab,positive = "M");

    nb_accuracy_different_topEigenvectors[i-1,j] <- nb_confusionMat$overall[[1]]
  }
}

# average accuracy in 10 fold
nb_avg_fold_accuracy_Eigs <- rowMeans(nb_accuracy_different_topEigenvectors)

nb_avg_fold_accuracy_Eigs <- data.frame(nb_avg_fold_accuracy_Eigs);
nb_avg_fold_accuracy_Eigs$Number_Eigenvector <- top_eigenvector

colnames(nb_avg_fold_accuracy_Eigs) <-
 c("Average_Accuracy","Top_Eigenvector_Number")

# plot accuracy vs different Ks
nb_accuracy_eig_p <-ggplot(nb_avg_fold_accuracy_Eigs,
 aes(x=Top_Eigenvector_Number,y=Average_Accuracy)) +
  geom_point(aes(colour=Average_Accuracy)) +
  xlab("Different Number of Top Eigenvectors") + ylab("Average Accuracy ")+
  scale_colour_gradient(low="blue",high="red") +
  geom_line(linetype=3) + ggtitle("Naive Bayes")

# The highest accuracy is achieved when top 6 eigenvectors
# get corresponding average AUC list and ROC curve
NB_AUC_list <- NULL # initiate a list to store AUC values in each fold
NB_FPR_table <- matrix(,nrow = 60,ncol = 10)
NB_TPR_table <- matrix(,nrow = 60,ncol=10)

NB_10fold_ROC <- ggplot(data = NULL,aes(x=FPR,y=TPR)) + theme_bw() +
 xlab("False Positive Rate") + ylab("True Positive Rate")

for (j in 1:k_foldValue ){

  testingDataRowIndex <- TenFold_CrossValidation[[j]]
```

```
  # get training data
  trainingDataRaw <- Breast_Cancer_table[-testingDataRowIndex,]

  training_label <- Breast_Cancer_table[-testingDataRowIndex,2]

  # get testing data
  testingDataRaw <- Breast_Cancer_table[testingDataRowIndex,]
  testing_label <- Breast_Cancer_table[testingDataRowIndex,2]

  # Project raw data to PCA plane
  training_PCA <- prcomp(Breast_Cancer_table[-testingDataRowIndex,3:32],scale. =
 TRUE)

  trainingData_PCA_transformed <- predict(training_PCA,newdata=trainingDataRaw)
  testingData_PCA_transformed <- predict(training_PCA,newdata=testingDataRaw)

  # Build Naive Bayes model using different number of top eigenvectors
  nb_model <- naiveBayes(training_label ~ .,
                         data= data.frame(trainingData_PCA_transformed)[,1:6])
  nb_result <- predict(nb_model,data.frame(testingData_PCA_transformed)[,1:6],type =
 "raw")

  nb_pred <- prediction(nb_result[,2],testing_label)

  # AUC
  nb_auc <- performance(nb_pred,measure = "auc")
  nb_auc_value <- slot(nb_auc,"y.values")[[1]]
  NB_AUC_list[j] <- nb_auc_value

  # ROC
  nb_roc <- performance(nb_pred,"tpr","fpr")

  FPR <-slot(nb_roc,"x.values")[[1]]
 # get False Positive Rate- x.values
  TPR <-slot(nb_roc,"y.values")[[1]]
 # get True Positive Rate- x.values

  ROC_table <- NULL
  ROC_table$FPR <- FPR
  NB_FPR_table[1:length(FPR),j] = FPR

  ROC_table$TPR <- TPR
  NB_TPR_table[1:length(TPR),j] = TPR
```

```
 ROC_table$Fold <-
paste("Fold",toString(j),"(AUC",toString(round(nb_auc_value,3)),")")
 ROC_table <- data.frame(ROC_table)


NB_10fold_ROC <- NB_10fold_ROC + geom_line(data=ROC_table,aes(colour=Fold))
}
# plot diagnoal line
# NB_10fold_ROC + geom_abline(slope=1,linetype=3)


################## Model Comparison ####################

# Accuracy comparison
model_accuracy_cmp <- data.frame(c(knn_max_avg_accuracy,svm_max_avg_accuracy,
 logit_max_avg_accuracy,nb_max_avg_accuracy))

model_accuracy_cmp$Model <- c("KNN","SVM","Logistic Regression","Naive Bayes")
model_accuracy_cmp <- data.frame(model_accuracy_cmp)
colnames(model_accuracy_cmp) <- c("Average_Accuracy","Model")

p_accuracy<- ggplot(model_accuracy_cmp,
                aes(x=Model,y=Average_Accuracy,label=round(Average_Accuracy,5)))
 +geom_bar(stat = "identity",aes(fill=Model)) +
 geom_text()+ theme(legend.position="none",axis.text.x = element_text(angle = 45,
 hjust = 1))

# rearrange accuracy vs top eigenvector number plot
grid.arrange(knn_accuracy_eig_p,svm_accuracy_eig_p,logit_accuracy_eig_p,nb_accurac
 y_eig_p,ncol=2)

# combined average ROC plot
KNN_avg_ROC <- rowMeans(KNN_FPR_table,na.rm = TRUE);
KNN_avg_ROC <- data.frame(KNN_avg_ROC)
KNN_avg_ROC$TPR <- rowMeans(KNN_TPR_table);
KNN_avg_ROC$Model <-
 paste("KNN","(AUC",toString(round(mean(KNN_AUC_list),4)),")")
colnames(KNN_avg_ROC) <- c("False_Positive_Rate","True_Positive_Rate","Model")

SVM_avg_ROC <- rowMeans(SVM_FPR_table,na.rm = TRUE);
SVM_avg_ROC <- data.frame(SVM_avg_ROC)
SVM_avg_ROC$TPR <- rowMeans(SVM_TPR_table,na.rm = TRUE);
SVM_avg_ROC$Model <-
```

```
 paste("SVM","(AUC",toString(round(mean(SVM_AUC_list),4)),")")
colnames(SVM_avg_ROC) <-c("False_Positive_Rate","True_Positive_Rate","Model")

Logit_avg_ROC <- rowMeans(Logit_FPR_table,na.rm = TRUE);
Logit_avg_ROC <- data.frame(Logit_avg_ROC)
Logit_avg_ROC$TPR <- rowMeans(Logit_TPR_table,na.rm = TRUE);
Logit_avg_ROC$Model <-paste("Logistic Regression",
 "(AUC",toString(round(mean(Logit_AUC_list),4)),")")
colnames(Logit_avg_ROC) <- c("False_Positive_Rate","True_Positive_Rate","Model")

NB_avg_ROC <- rowMeans(NB_FPR_table,na.rm = TRUE);
NB_avg_ROC <- data.frame(NB_avg_ROC)
NB_avg_ROC$TPR <- rowMeans(NB_TPR_table,na.rm = TRUE);
NB_avg_ROC$Model <-paste("Naive Bayes",
 "(AUC",toString(round(mean(NB_AUC_list),4)),")")
colnames(NB_avg_ROC) <- c("False_Positive_Rate","True_Positive_Rate","Model")

Model_ROC<- ggplot(data=NULL,aes(x=False_Positive_Rate,y=True_Positive_Rate)) +
  geom_line(data=KNN_avg_ROC,aes(color=Model)) +
  geom_line(data=SVM_avg_ROC,aes(color=Model)) +
  geom_line(data=Logit_avg_ROC,aes(color=Model)) +
  geom_line(data=NB_avg_ROC,aes(color=Model)) +
  xlab("False Positive Rate") + ylab("True Positive Rate") +
 geom_abline(slope=1,linetype=3)
```

References

Asri, H., Mousannif, H., Moatassime, H.A., and Noel, T. (2016). Using Machine Learning Algorithms for Breast Cancer Risk Prediction and Diagnosis. Procedia Computer Science *83*, 1064-1069.

Cox, D.R. (1958). The Regression Analysis of Binary Sequences. Journal of the Royal Statistical Society Series B (Methodological) *20*, 215-242.

Ferlay, J., Héry, C., Autier, P., and Sankaranarayanan, R. (2010). Global Burden of Breast Cancer. In Breast Cancer Epidemiology, C. Li, ed. (New York, NY, Springer New York), pp. 1-19.

Ferreira, P., Dutra, I., Salvini, R., and Burnside, E. (2016). Interpretable models to predict Breast Cancer. Paper presented at: 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM).

Hastie, T., Tibshirani, R., and Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Springer).

Huang, M.-W., Chen, C.-W., Lin, W.-C., Ke, S.-W., and Tsai, C.-F. (2017). SVM and SVM Ensembles in Breast Cancer Prediction. PLoS ONE *12*, e0161501.

Street, W.N., Wolberg, W.H., and Mangasarian, O.L. (1993). Nuclear feature extraction for breast tumor diagnosis.

Verleysen, M., and François, D. (2005). The Curse of Dimensionality in Data Mining and Time Series Prediction. In Computational Intelligence and Bioinspired Systems: 8th International Work-Conference on Artificial Neural Networks, IWANN 2005, Vilanova i la Geltrú, Barcelona, Spain, June 8-10, 2005 Proceedings, J. Cabestany, A. Prieto, and F. Sandoval, eds. (Berlin, Heidelberg, Springer Berlin Heidelberg), pp. 758-770.

ZHANG, H. (2005). EXPLORING CONDITIONS FOR THE OPTIMALITY OF NAÏVE BAYES. International Journal of Pattern Recognition and Artificial Intelligence *19*, 183-198.