04-12-2022

FALL 2022

STAT 441 Final Project

GROUP 17

Breast Cancer Classification - Benign and Malignant Tumors

Suhaas Vadigi

Cameron Hinton

Introduction

Breast cancer is a disease that occurs when cells in the breast reproduce uncontrollably, and primarily affects women (although some cases have been observed in male patients as well). In 2022, breast cancer has accounted for approximately 14% of all cancer-related deaths in women, while constituting 25% of all cancer diagnoses.

Tumors do not immediately mean that an individual has cancer, however. For a diagnosis of cancer, you need to have a tumor that has been identified as *malignant*.

Tumors typically come in two types: Malignant, and Benign. According to Splane (2022), Benign tumors are those which have been identified as unnatural cell mass growth, but may not be cancerous - that is, they do not grow rapidly and spread. These types of tumors may still be removed or treated for other purposes, but do not typically pose a great threat to the health of the individual unless they are causing damage somehow (such as pushing against tissues, such as the trachea).

Malignant tumors, however, are those that have been identified as cancerous, i.e. they have uncontrollable growth and spread to nearby tissues. For example, breast cancer can spread to the rest of the body if it reaches the lymph nodes.

Often, a sample of tissue is taken from the suspected area of the patient, and various tests are done. One such test is to view the cells under a microscope, and the characteristics of cell growth and shape/size may indicate malignancy.

Problem of Interest

Often, Pathologists (Tissue analysis specialists) will look at the structure of the cells in tissue samples taken from patients to determine the malignancy of the cell (to determine if the tissue sample contains cancerous cells or not). There are specific dimensional characteristics of the cells that pathologists use to determine if a cell is cancerous, and these characteristics can be represented numerically. Using these numerical measurements, the goal of this project is to build a model that can classify tumors based on tissue samples and statistics about the cells in the tumors for each patient.

With such a machine learning model, if we were to have enough data and successfully classify with a high degree of accuracy (and furthermore a low false-negative rate), we could automate the detection of cancerous cells without human intervention using a microscope, and thus both reduce the workload and potential for human error for pathologists. A machine learning classification model can thus reduce wait times on test results significantly, and contribute to hospitals and clinics increasing their overall productivity.

Methodology & Analysis

Dataset

The dataset used to create the classification models is the “Breast Cancer Wisconsin” dataset which can be found online here www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data. The dataset contains 569 data entries (357 benign, 212 malignant) with 30 possible predictors. The predictors are measurements of a digital image obtained of the cell nuclei of a breast mass. It is explained what each of the 30 predictors is a measure of in the kaggle page linked above.

Criteria for Analysis

Area under the curve (AUC), specificity (true negative rate), and sensitivity (true positive rate) were used to rank the effectiveness of each model. AUC is measured by plotting sensitivity and specificity over different threshold values and taking the area under the resulting curve. It is an indicator of the overall performance of a classifier. Sensitivity is one of the most important metric for this dataset since a higher sensitivity means less false negatives. A false negative could create a medical risk for patients as it creates a false belief of safety and would delay treatment. Specificity is how often a tumor that is benign is labeled as benign. Higher specificity indicates less people being misdiagnosed with a malignant tumor. This is something we aim to get as high as possible after first ensuring high sensitivity.

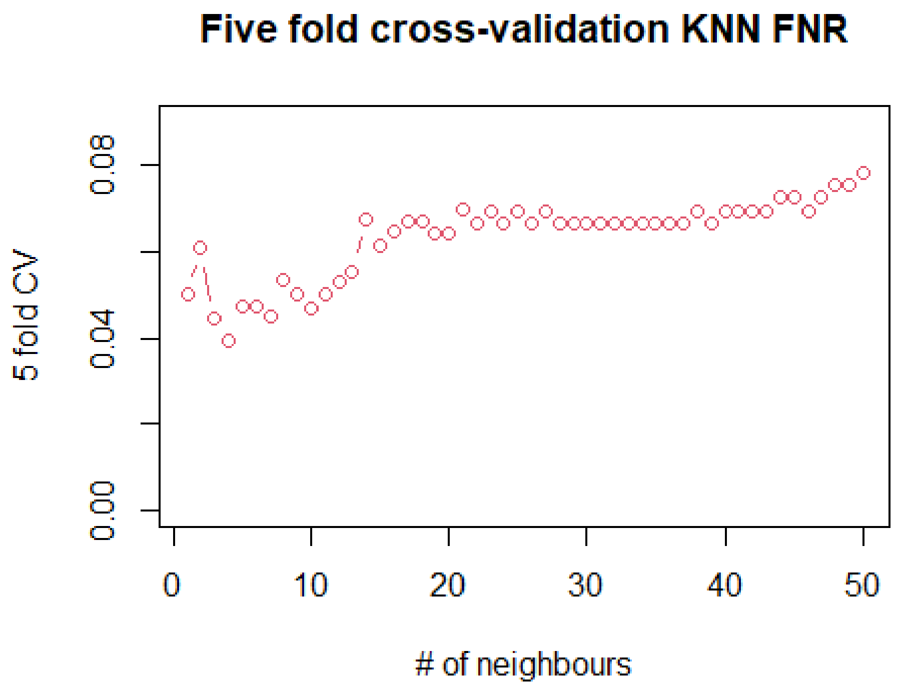
K-Nearest Neighbours (KNN)

KNN is a method by which the “K” closest neighbours to a given observation (often in terms of Euclidean distance, although other measures of distance may be used) determine the classification of the observation. It is non-parametric (i.e. does not have a set model) and is memory based, so it becomes more accurate as the number of observations increases. However, it may sometimes fail to identify outlier classifications, which are rare but definitely exist for breast cancer malignancies - sometimes, a seemingly benign tumor may actually be cancerous, while only exhibiting very slow growth, for example. This puts KNN at risk of an error rate increase, but this is not a significant concern due to the fact that the error rate increase from these outliers may not be very high, given enough training observations, and considering how rare misdiagnoses actually are in the training set.

KNN is affected strongly by the scale of the variables, and due to the difference in scale for the predictors in the data, all variables were standardized using the dplyr package in R to have a mean of 0, and variance of 1. KNN was applied to the data set based on all variables, and the variable “isMalignant” as the classes. K values ranging from 1 to 50 were used, and the initial application of KNN using a 70-30 training/testing data split showed that K=3 was the best number of neighbors for predictions on the testing data, with a staggering accuracy of ~99.4%.

However, suspecting that this accuracy is not very reproducible, 5-fold cross-validation was used to better determine the accuracy of KNN on the data set, which produced different results: K = 4 was now the desirable number of neighboring observations, and the accuracy was significantly different, now reduced to ~96.5%, which was still a respectable accuracy. However, while we only have very low false negative rates in our initial model (due to the 99% overall accuracy), when using 5 fold CV, we see a marked increase in FNR, as we now see approximately 3.93% false-negative rates. While this may mean that the model is statistically sound, since this is a medical application, it may not be quite as viable due to legal liabilities.

Both the test error rate and the false negative rate of the 5-fold Cross validation KNN models were plotted (see appendix). The false negative rate plot is shown below.



These plots show that anywhere between 3 and 10 variables shows a lot of variance, and is likely the sweet spot (The result varied depending on the random seed used at the time of the 5 fold CV) range for the number of neighbours, based on the limited dataset.

Results at threshold = 0.5:

Sensitivity: 94.77% (**1st**) **Specificity:** 97.20% (4th) **AUC:** 0.9854 (5th)

Logistic Regression

A logistic regression model predicts the probability that an observation belongs to a category. The model is an extension of linear regression designed to handle categorical output. It is a simple model with minimal assumptions. This means it still performs well on smaller sample sizes. Because of these properties, logistic regression is very well suited for the breast cancer dataset where there is a relatively small sample size and where binary classification is the goal. We used many different methods of variable selection and model fitting and then compared their results on a five-fold cross validation set to find the best logistic regression model. The methods used were ridge regression, LASSO, elastic net, and a manually selected logistic regression model.

For the methods that use tuning parameter λ, this was chosen using the one standard error method which selects the largest λ within one standard error of the λ value that minimizes the cross-validated error [4]. This value of λ is used in practice to avoid overfitting by creating the simplest possible model that still has low cross-validated error.

LASSO Regression

LASSO is a regularization model that can be generalized to logistic regression when we fit a logistic regression model by minimizing maximum likelihood and an additional penalty term which is the sum of coefficient magnitudes multiplied by tuning parameter λ. This penalty term penalizes complex models that have many different variables with a large influence on prediction. It has the ability to shrink parameter coefficients to zero. These properties make it very good at handling data with a large number of irrelevant parameters as it is able to find the relevant parameters and shrink the others to zero. However, when dealing with highly correlated parameters (which seem to be present in this dataset) LASSO is dominated by Ridge regression. LASSO was still tested despite this for comparison.

Results at threshold = 0.5:

Sensitivity: 92.93% (4th) **Specificity:** 98.94% (3rd) **AUC:** 0.9920 (3rd)

Ridge Regression

With ridge regression we add a penalty like in LASSO except we use the penalty term of the sum of squared coefficients multiplied by tuning parameter λ. Ridge cannot shrink coefficients to zero like in LASSO but it handles highly correlated parameters much better. We expect ridge to outperform LASSO on this dataset since our pair plots (see appendix) showed a high number of significant predictors that we also observed to have high pairwise correlation. The results we got supported this idea showing that ridge outperforms LASSO on this dataset.

Results at threshold = 0.5:

Sensitivity: 94.39% (2nd) **Specificity:** 99.43% (**1st**) **AUC:** 0.9950 (**1st**)

Elastic Net

Elastic net takes a weighted average of the penalty terms from LASSO and ridge (multiplied by λ) as the additional penalty term for model fitting. For our model, we chose an equal weighting of both penalty terms. This approach allows for shrinking parameters to zero like in LASSO and better handling of pairwise correlation like in ridge. The results show that elastic net only slightly underperforms ridge. The elastic net model is simpler than ridge (since it reduces coefficients to zero) while nearly performing as well on our metrics. This makes elastic net a possible choice over ridge since a simpler model is often preferred in the case of similar performance (Occam's Razor).

Results at threshold = 0.5:

Sensitivity: 93.88% (3rd) **Specificity:** 99.25% (2nd) **AUC:** 0.9939 (2nd)

Logistic Regression with naive coefficient elimination

While LASSO, Ridge and Elastic net based penalties are highly effective, it is sometimes also useful to eliminate coefficients/predictors from the model which have very low correlation to the outcome variable. However, this method is not usually recommended. This was done to see the effect on accuracy when variables that are less correlated than 0.5 (threshold used in this example) are removed, and to compare with more traditional methods. As expected, this method performs worse than the others almost universally across the three criteria used in this report. This may be due to the threshold being too high. The sensitivity is also lower, which indicates higher false negative rates, which as discussed before are dangerous in medical applications.

Results at threshold = 0.5:

Sensitivity: 92.69% (5th) **Specificity:** 96.08% (5th) **AUC:** 0.9888 (4th)

Observations

KNN was able to achieve the highest sensitivity but had poor relative performance to the other models in specificity and AUC. A high AUC indicates that the classifier maintains high sensitivity and specificity across different choices of thresholds. This means that the most probabilities are near 0 or 1 to indicate strong evidence against or evidence for a malignant tumor respectfully. In these classifiers, the threshold can be adjusted to better handle outliers and make a tradeoff between sensitivity and specificity. For this dataset we want as high sensitivity as possible while maintaining a reasonable specificity because the health of people is at risk. Since ridge regression has the highest AUC and second highest sensitivity, let us check how close we can get to 100% sensitivity while still maintaining a reasonable specificity.

After trial and error on the same seed and cross validation as the previous results it was found that a threshold of 0.14 yields the best tradeoff between sensitivity and specificity. Any lower and the sensitivity does not change while the specificity drops drastically.

Results of ridge logistic regression at threshold = 0.14:

Sensitivity: 99.41% **Specificity:** 87.76% **AUC:** 0.9950

These results are far superior to any results seen at a threshold of 0.5. Since ridge has the highest AUC it is likely that the above classifier is the best possible of the models tested.

Conclusion

With the dataset analyzed being relatively small, logistic regression outperformed KNN. This can be explained by the simplicity of logistic regression which makes very few assumptions allowing it to perform well on smaller sample sizes. KNN is non-parametric, so it makes no assumptions on the data but it still requires large amounts of data to perform optimally and accurately group observations by their nearest neighbors.

Of the logistic regression classifiers, the ridge logistic regression performed the best on the metric of AUC and specificity while being a close second to KNN in sensitivity. Ridge was a near perfect classifier obtaining an AUC of 0.995 out of a possible 1 obtained by a classifier that is perfect for any threshold from 0.05 to 0.95 on the five-fold cross validation set. Elastic net performed only slightly worse than ridge but produced a simpler model making it another potential choice for best model.

Maximizing the sensitivity at the cost of specificity (threshold set to 0.14), ridge was able to fit a logistic regression classifier with a sensitivity of 99.4% and specificity of 87.8% on the five-fold cross validation set. This means that if these results held true in the real world, 99.4% percent of people with a malignant tumor would get the correct diagnosis from this classifier. The coefficients of ridge logistic regression fit on the entire dataset is attached at the end of the appendix. The model contains too many significant variables to properly discuss in this conclusion.

The results found from this dataset could be affected by the size of the dataset. It is likely that another type of classifier like KNN would start to outperform logistic regression when the sample size becomes large. Larger datasets would also allow for more accurate testing of the three metrics utilized in this report by lowering variance in data between cross validation sets.

Contribution of each group member

Suhass

- Found the dataset
- Created and analyzed KNN, and naive logistic regression
- Wrote introduction
- Wrote problem of interest
- Wrote analysis of KNN, and naive logistic regression
- Talked about sections worked on in the video

Cameron

- Created and analyzed Ridge, LASSO, and elastic net approaches
- Ran final cv results of each model
- Wrote remaining parts of Methodology & Analysis section (Not KNN, and naive logistic regression)
- Wrote conclusion
- Talked about sections worked on in the video

References

1. Lee, S. (n.d.). *Breast cancer statistics*. Canadian Cancer Society. Retrieved December 3, 2022, from https://cancer.ca/en/cancer-information/cancer-types/breast/statistics

2. Splane, B. (n.d.). *Differences between a malignant and benign tumor*. Verywell Health. Retrieved December 4, 2022, from https://www.verywellhealth.com/what-does-malignant-and-benign-mean-514240

3. Centers for Disease Control and Prevention. (2022, September 26). *What is breast cancer?* Centers for Disease Control and Prevention. Retrieved December 3, 2022, from https://www.cdc.gov/cancer/breast/basic_info/what-is-breast-cancer.htm

4. *An introduction to `glmnet`*. An Introduction to `glmnet` • glmnet. (n.d.). Retrieved December 2, 2022, from https://glmnet.stanford.edu/articles/glmnet.html

5. Dataset - www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

Appendix

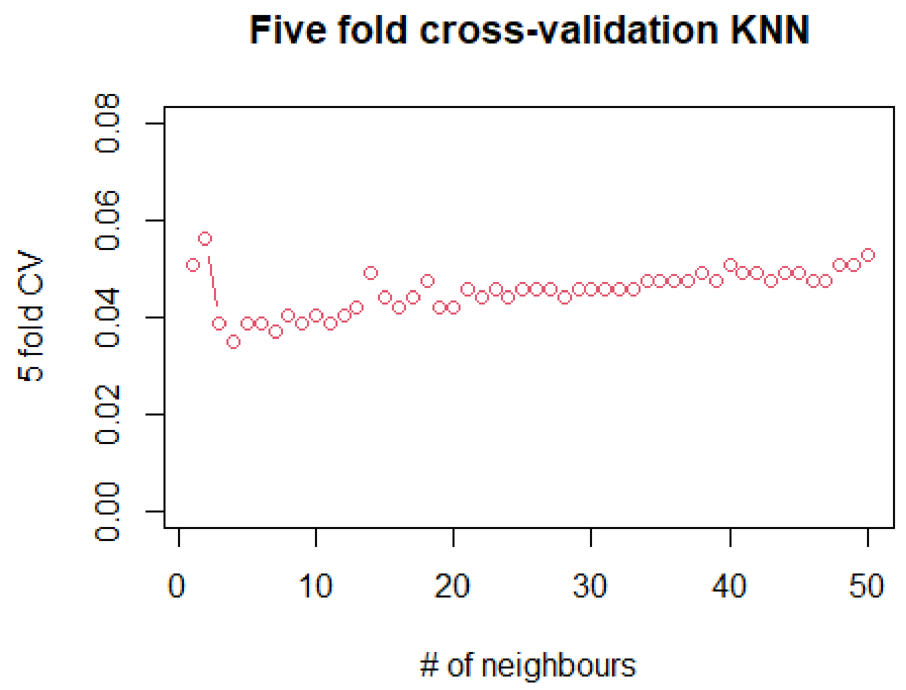
Data processing

```
dat = read.csv("breast-cancer.csv", header=TRUE)
dat = dat[,-1]
dat$diagnosis = factor(dat$diagnosis)

x = model.matrix(diagnosis ~ ., dat)[,-1]
y = dat$diagnosis
```

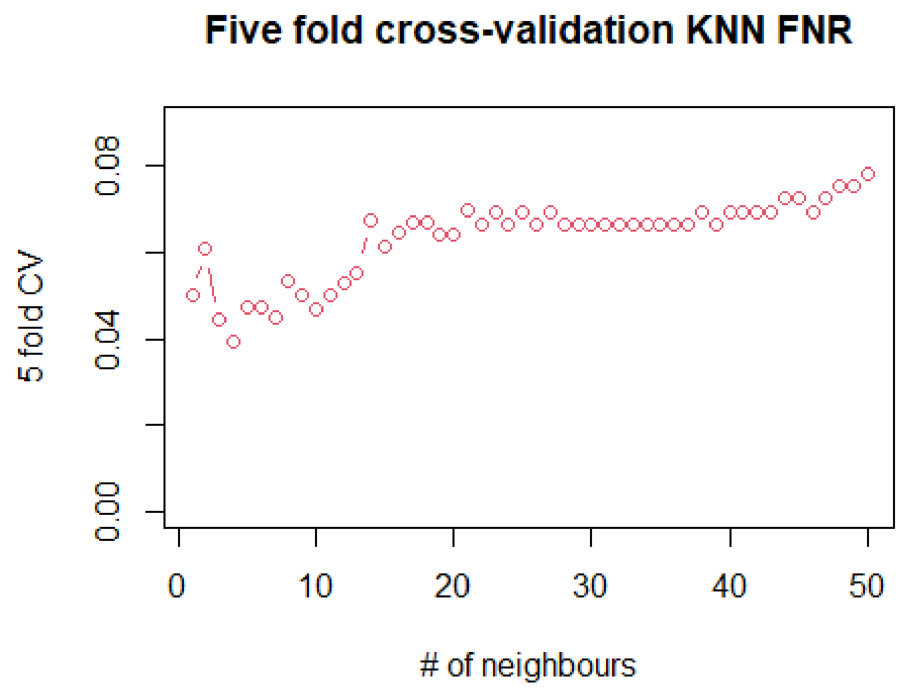
KNN

```
library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(class)
set.seed(777) #2.8% error @ 250 size obs, #3.
train <- sample(569, 400)

#pairs(data[,3:12], col=data$diagnosis, pch=3)

## Standardize data to mean 0, var 1 using dplyr library

standardized.data <- dat[,2:31] %>% mutate_all(~(scale(.) %>% as.vector))

standardized.data$isMalignant <- as.numeric(dat$diagnosis) - 1
standardized.data$diagnosis <- dat$diagnosis

train.data <- standardized.data[train,]
test.data <- standardized.data[-train,]
```

```
X.train <- cbind(train.data[1:30])
X.test <- cbind(test.data[1:30])
train.diag <- as.numeric(train.data$diagnosis) - 1
test.diag <- as.numeric(test.data$diagnosis) - 1

testerror <- rep(NA,50)
tables <- rep(NA, 50)

for(j in 1:50){
  knn.pred <- knn(X.train, X.test, train.diag, k=j)
  table(knn.pred,test.diag)
  testerror[j] <- mean(knn.pred!=test.diag)
}

testerror

##  [1] 0.04142012 0.03550296 0.00591716 0.00591716 0.00591716 0.00591716
##  [7] 0.00591716 0.01183432 0.00591716 0.00591716 0.00591716 0.01183432
## [13] 0.01183432 0.01183432 0.00591716 0.01775148 0.01775148 0.02366864
## [19] 0.02366864 0.00591716 0.02958580 0.02958580 0.02958580 0.02366864
## [25] 0.02366864 0.04733728 0.02958580 0.03550296 0.03550296 0.03550296
## [31] 0.03550296 0.04142012 0.03550296 0.03550296 0.03550296 0.03550296
## [37] 0.03550296 0.03550296 0.03550296 0.03550296 0.03550296 0.03550296
## [43] 0.04142012 0.04142012 0.04142012 0.04733728 0.04733728 0.04733728
## [49] 0.04733728 0.05325444

which.min(testerror)

## [1] 3

fivefoldcv <- matrix(NA, 5,50)
randomseq <- c(1:569)[order(runif(569))]
fold <- rep(1:5, 569/5)
FNR.matrix <- matrix(NA, 5,50)
for(i in 1:5){
  train=randomseq[fold!=i]
  train.data <- standardized.data[train,]
  test.data <- standardized.data[-train,]

  train.X <- cbind(train.data[1:30])
  test.X <- cbind(test.data[1:30])
  train.diag <- as.numeric(train.data$isMalignant)
  test.diag <- as.numeric(test.data$isMalignant)

  for(j in 1:50) {
    knn.pred <- knn(train.X, test.X, train.diag, k=j)
    fivefoldcv[i, j] <- mean(knn.pred!=test.diag)
    FNR.matrix[i, j] <- sum(knn.pred==0 & test.diag==1)/sum(test.diag==0)
  }
}
fivefold <- apply(fivefoldcv, MARGIN=2, FUN=mean)
ffFNR <- apply(FNR.matrix, MARGIN=2, FUN=mean)
plot(1:50, fivefold, type='b', main="Five fold cross-validation KNN", xlab="#
of neighbours", ylab="5 fold CV", col=2, ylim=c(0,0.08))
```



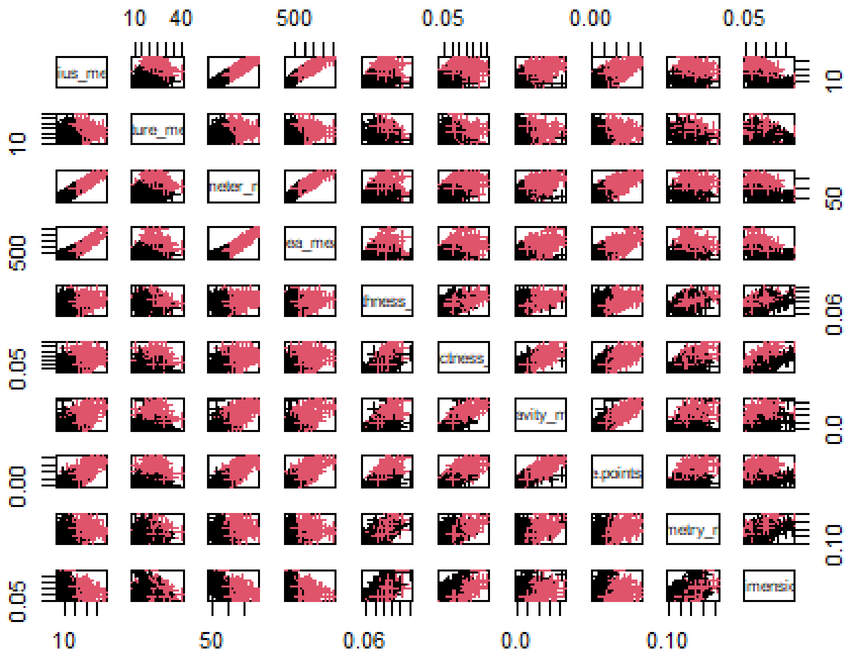
```
plot(1:50, ffFNR, type='b', main="Five fold cross-validation KNN FNR",
xlab="# of neighbours", ylab="5 fold CV", col=2, ylim=c(0,0.09))
```



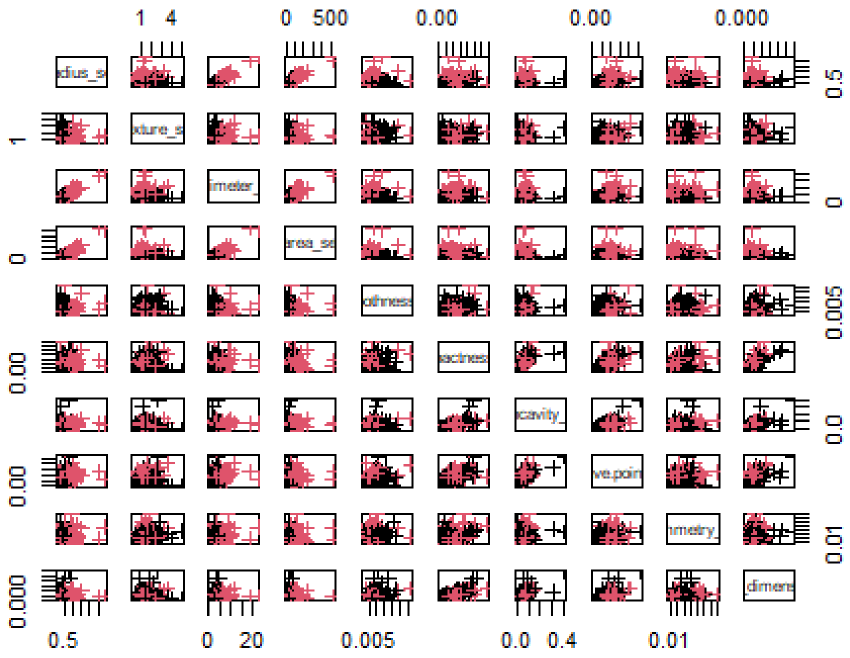
```r
# Results at k=4 for five fold cv
sens.knn <- rep(NA, 5)
spec.knn <- rep(NA, 5)
auc.knn <- rep(NA, 5)
for(i in 1:5){
  train=randomseq[fold!=i]
  train.data <- standardized.data[train,]
  test.data <- standardized.data[-train,]

  train.X <- cbind(train.data[1:30])
  test.X <- cbind(test.data[1:30])
  train.diag <- as.numeric(train.data$isMalignant)
  test.diag <- as.numeric(test.data$isMalignant)
  knn.pred <- knn(train.X, test.X, train.diag, k=20, prob=TRUE)
  knn.pred.prob <- ifelse(as.numeric(knn.pred) - 1,
attributes(knn.pred)$prob,
                          1 - attributes(knn.pred)$prob)
  knn.pred.class <- ifelse(knn.pred.prob >= 0.5, 1, 0)

  ## confusion matrix
  conf.matrix <- table(knn.pred.class, test.diag)
  #conf.matrix

  ## Sensitivity
  sens.knn[i] <- conf.matrix[4] / (conf.matrix[3]+conf.matrix[4])
  ## Specificity
  spec.knn[i] <- conf.matrix[1] / (conf.matrix[1]+conf.matrix[2])

  ## Area under curve
  auc.knn[i] <- roc(test.diag~knn.pred.prob, levels=c(0,1),
direction="<")$auc
}

# average sensitivity across 5 folds
mean(sens.knn)

## [1] 0.8914029

# average specificity across 5 folds
mean(spec.knn)

## [1] 0.9940179

# average AUC across 5 folds
mean(auc.knn)

## [1] 0.990159
```

Logistic Regression

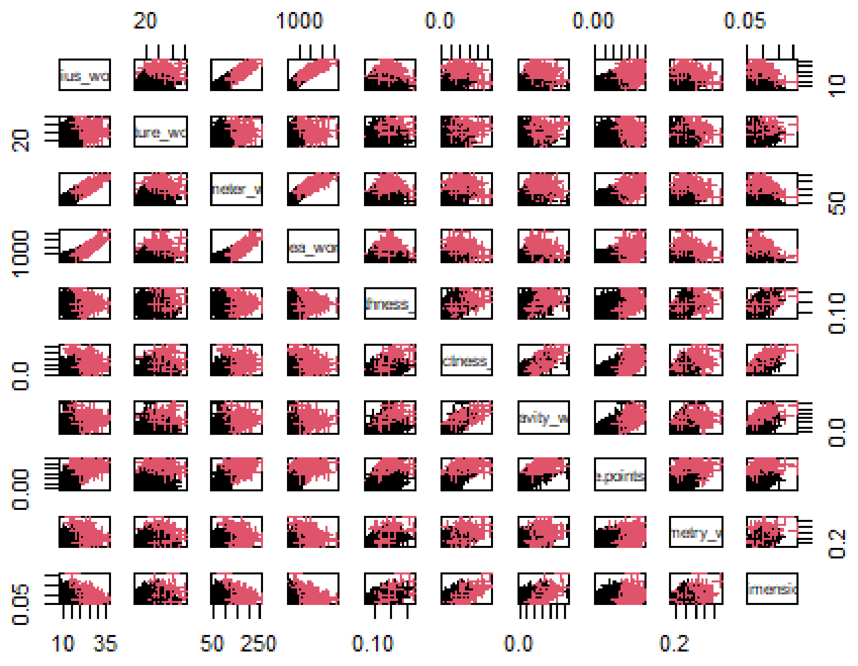
```
set.seed(777)
# Since logisitc regression is parametric, there is no need to standardize
data
# (unlike with KNN)

# First, let's see if there is seperation between features for each class, to
# See if logistic regression is a viable option. (particularly, we can see if
# the linear boundary decision from logistic regression may be problematic)
pairs(dat[,2:11], col=dat$diagnosis, pch=3)
```



```
pairs(dat[,12:21], col=dat$diagnosis, pch=3)
```



```
pairs(dat[,22:31], col=dat$diagnosis, pch=3)
```



```r
# from the pairs graphs, we can see some good seperation in most graphs, but
not
# all. Let's eliminate the coefficients which have the lowest correlation
with
# our binary outcome.

dat$isMalignant <- as.numeric(dat$diagnosis) - 1

# Create correlation matrix between all variables
cor.matrix <- cor(dat[,2:32])

# We just want to ensure correlation with malignancy binary variable, so just
# grab that column (or the row, doesn't matter since this is symm matrix)

cor.mal <- abs(cor.matrix["isMalignant",])

# Let us use a naive measure to remove variables:
# now if the abs. correlation value isn't greater than some threshold, we
# consider the variable "irrelevant". (threshold = 0.3 for this example)

irrelevantvars <- cor.mal[cor.mal<=0.5]
relevantvars <- cor.mal[cor.mal > 0.5]

irrelevantvars
```

```
##            texture_mean         smoothness_mean            symmetry_mean
##             0.415185300             0.358559965              0.330498554
##  fractal_dimension_mean              texture_se            smoothness_se
##             0.012837603             0.008303333              0.067016011
##          compactness_se            concavity_se        concave.points_se
##             0.292999244             0.253729766              0.408042333
##             symmetry_se    fractal_dimension_se            texture_worst
##             0.006521756             0.077972417              0.456902821
##        smoothness_worst          symmetry_worst  fractal_dimension_worst
##             0.421464861             0.416294311              0.323872189

rel.data <- dat[names(relevantvars)]

# We'll use a 70-30 split for training and testing data
train <- sample(569, 400)
train.data <- rel.data[train,]
test.data <- rel.data[-train,]

logitreg <- glm(isMalignant~., family=binomial(link ="logit"), data=
train.data)
pred.linear <- predict(logitreg, newdata=test.data)
pred.prob <- exp(pred.linear)/(1+exp(pred.linear))
pred.class <- ifelse(pred.prob>0.5, 1, 0)
conf.matrix <- table(pred.class, true.class=test.data$isMalignant)
mean(pred.class!=test.data$isMalignant)

## [1] 0.0295858

Sensitivity <- conf.matrix[4] / (conf.matrix[3]+conf.matrix[4])
Specificity <- conf.matrix[1] / (conf.matrix[1]+conf.matrix[2])
Sensitivity

## [1] 0.9830508

Specificity

## [1] 0.9636364

roc(test.data$isMalignant~pred.prob, levels=c(0,1), direction="<")$auc

## Area under the curve: 0.9921

## 5-fold cv
sens.log <- rep(NA, 5)
spec.log <- rep(NA, 5)
auc.log <- rep(NA, 5)
for(i in 1:5){
  train=randomseq[fold!=i]
  train.data <- rel.data[train,]
  test.data <- rel.data[-train,]
  logitreg <- glm(isMalignant~., family=binomial(link ="logit"),
data=train.data)
```

```r
  pred.linear <- predict(logitreg, newdata=test.data)
  pred.prob <- exp(pred.linear)/(1+exp(pred.linear))
  pred.class <- ifelse(pred.prob>=0.5, 1, 0)

  ## confusion matrix
  conf.matrix <- table(pred.class, true.class=test.data$isMalignant)
  #conf.matrix

  ## Sensitivity
  sens.log[i] <- conf.matrix[4] / (conf.matrix[3]+conf.matrix[4])

  ## Specificity
  spec.log[i] <- conf.matrix[1] / (conf.matrix[1]+conf.matrix[2])

  ## Area under curve
  auc.log[i] <- roc(test.data$isMalignant~pred.prob, levels=c(0,1),
direction="<")$auc
}

# average sensitivity across 5 folds
mean(sens.log)

## [1] 0.9268978

# average specificity across 5 folds
mean(spec.log)

## [1] 0.9607657

# average AUC
mean(auc.log)

## [1] 0.9887989
```

Ridge

```r
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-4

set.seed(777)

grid = 10^seq(-2, 10, length = 100)

sens.ridge <- rep(NA, 5)
spec.ridge <- rep(NA, 5)
auc.ridge <- rep(NA, 5)
for(i in 1:5){
  train=randomseq[fold!=i]
  ridge.mod = cv.glmnet(x[train,], y[train], alpha = 0, lambda = grid,
                        family = binomial)

  coef(ridge.mod, s = "lambda.1se")

  ridge.pred <- predict(ridge.mod, s="lambda.1se", newx=x[-train,],
                        type="response")
  ridge.pred.class <- ifelse(ridge.pred >= 0.5, "M", "B")
  y.test = y[-train]

  ## confusion matrix
  conf.matrix <- table(ridge.pred.class, y.test)
  #conf.matrix

  ## Sensitivity
  sens.ridge[i] <- conf.matrix[4] / (conf.matrix[3]+conf.matrix[4])

  ## Specificity
  spec.ridge[i] <- conf.matrix[1] / (conf.matrix[1]+conf.matrix[2])

  ## Area under curve
  auc.ridge <- roc(y.test~ridge.pred, levels=c("B","M"), direction="<")$auc
}

# average sensitivity across 5 folds
mean(sens.ridge)

## [1] 0.9438624

# average specificity across 5 folds
mean(spec.ridge)

## [1] 0.995

# average AUC
mean(auc.ridge)

## [1] 0.9990033
```

Lasso logistic

```
set.seed(777)

sens.lasso <- rep(NA, 5)
spec.lasso <- rep(NA, 5)
auc.lasso <- rep(NA, 5)
for(i in 1:5){
  train=randomseq[fold!=i]
  lasso.mod = cv.glmnet(x[train,], y[train], alpha = 1, lambda = grid,
                        family = binomial)

  #coef(lasso.mod, s = "lambda.1se")
  lasso.pred <- predict(lasso.mod, s="lambda.1se", newx=x[-train,],
                        type="response")
  lasso.pred.class <- ifelse(lasso.pred >= 0.5, "M", "B")
  y.test = y[-train]

  ## confusion matrix
  conf.matrix.lasso <- table(lasso.pred.class, y.test)
  #conf.matrix.lasso

  ## Sensitivity
  sens.lasso[i] <- conf.matrix.lasso[4] /
(conf.matrix.lasso[3]+conf.matrix.lasso[4])

  ## Specificity
  spec.lasso[i] <- conf.matrix.lasso[1] /
(conf.matrix.lasso[1]+conf.matrix.lasso[2])

  ## Area under curve
  auc.lasso <- roc(y.test~lasso.pred, levels=c("B","M"), direction="<")$auc
}

# average sensitivity across 5 folds
mean(sens.lasso)

## [1] 0.929334

# average specificity across 5 folds
mean(spec.lasso)

## [1] 0.989375

# average AUC
mean(auc.lasso)

## [1] 0.9877076
```

Elastic Net

```
set.seed(777)

sens.elastic <- rep(NA, 5)
spec.elastic <- rep(NA, 5)
auc.elastic <- rep(NA, 5)
for(i in 1:5){
  train=randomseq[fold!=i]
  elastic.mod = cv.glmnet(x[train,], y[train], alpha = 0.5, lambda = grid,
                          family = binomial)

  #coef(elastic.mod, s = "lambda.1se")

  elastic.pred <- predict(elastic.mod, s="lambda.1se", newx=x[-train,],
                          type="response")
  elastic.pred.class <- ifelse(elastic.pred >= 0.5, "M", "B")
  y.test = y[-train]

  ## confusion matrix
  conf.matrix.elastic <- table(elastic.pred.class, y.test)

  ## Sensitivity
  sens.elastic[i] <- conf.matrix.elastic[4] /
(conf.matrix.elastic[3]+conf.matrix.elastic[4])

  ## Specificity
  spec.elastic[i] <- conf.matrix.elastic[1] /
(conf.matrix.elastic[1]+conf.matrix.elastic[2])

  ## Area under curve
  auc.elastic <- roc(y.test~elastic.pred, levels=c("B","M"),
direction="<")$auc
}

# average sensitivity across 5 folds
mean(sens.elastic)

## [1] 0.9387575

# average specificity across 5 folds
mean(spec.elastic)

## [1] 0.9925

# average AUC across 5 folds
mean(auc.elastic)

## [1] 0.9953488
```

Final Ridge model specifications

```
final.ridge.mod = cv.glmnet(x, y, alpha = 0, lambda = grid,
                        family = binomial)

coef(final.ridge.mod, s = "lambda.1se")

## 31 x 1 sparse Matrix of class "dgCMatrix"
##                                    s1
## (Intercept)             -1.802867e+01
## radius_mean              1.010864e-01
## texture_mean             8.521722e-02
## perimeter_mean           1.430914e-02
## area_mean                9.641639e-04
## smoothness_mean          1.002531e+01
## compactness_mean         6.520941e-01
## concavity_mean           3.919036e+00
## concave.points_mean      1.010290e+01
## symmetry_mean            2.615922e+00
## fractal_dimension_mean  -3.015593e+01
## radius_se                1.405223e+00
## texture_se              -5.437349e-02
## perimeter_se             1.465313e-01
## area_se                  6.809680e-03
## smoothness_se            1.056456e+01
## compactness_se          -1.035231e+01
## concavity_se            -1.500400e+00
## concave.points_se        1.809700e+01
## symmetry_se             -1.391796e+01
## fractal_dimension_se    -7.839096e+01
## radius_worst             9.510945e-02
## texture_worst            8.036534e-02
## perimeter_worst          1.272853e-02
## area_worst               7.197626e-04
## smoothness_worst         1.568710e+01
## compactness_worst        9.867830e-01
## concavity_worst          1.646757e+00
## concave.points_worst     6.831682e+00
## symmetry_worst           5.696060e+00
## fractal_dimension_worst  6.771514e+00
```