HarvardX Capstone 2 Predicting Breast Cancer Diagnostic

Raphael Oliveira Abreu

2023-04-05

Introduction

This paper will analyze and use machine learning methods to predict cancer diagnostics for the Wisconsin Diagnostic Breast Cancer (WDBC) dataset from 1995. The data set is a collaboration between two Computer Sciences Professors and on Doctor from the General Surgery Department.

The first section will analyse the dataset and its variables. The following section will detail the methods of machine learning and how to evaluate their outcomes. The third and final section will present the conclusions to this project.

Analysis

The dataset contains 32 variables which are the following, according to data description:

- ID number
- Diagnosis (M = malignant, B = benign)
- And the mean, standard error and worst or largest (mean of the three largest values) for the measures:
  a) radius (mean of distances from center to points on the perimeter)
  b) texture (standard deviation of gray-scale values)
  c) perimeter
  d) area
  e) smoothness (local variation in radius lengths)
  f) compactness (perimeter^2 / area - 1.0)
  g) concavity (severity of concave portions of the contour)
  h) concave points (number of concave portions of the contour)
  i) symmetry
  j) fractal dimension ("coastline approximation" - 1)

Here are the 12 first variables of the 5 first observations in table format:

id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
842517	M	20.57	17.77	132.90	1326.0	0.08474
84300903	M	19.69	21.25	130.00	1203.0	0.10960
84348301	M	11.42	20.38	77.58	386.1	0.14250
84358402	M	20.29	14.34	135.10	1297.0	0.10030
843786	M	12.45	15.70	82.57	477.1	0.12780
844359	M	18.25	19.98	119.60	1040.0	0.09463

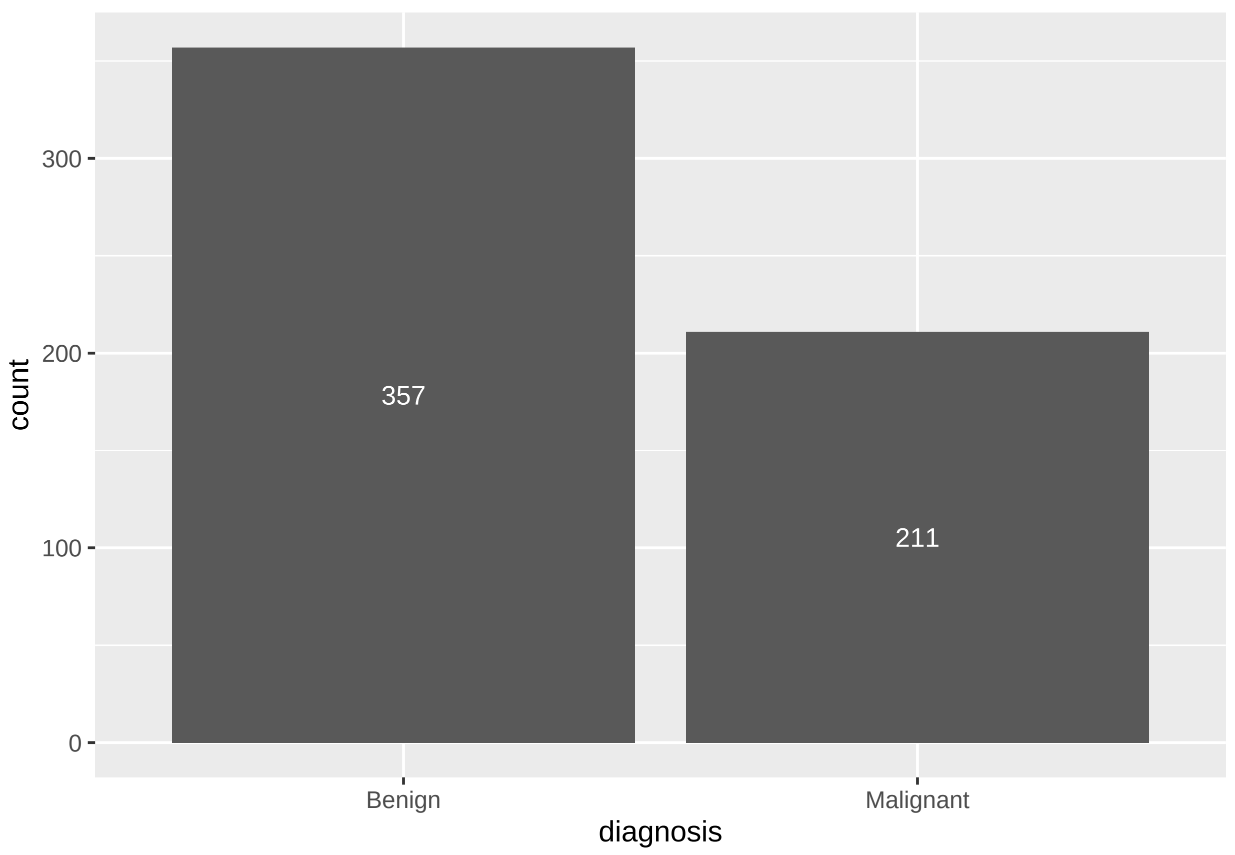
For illustrative purposes on how the whole 32 variables behave here is the print of using function glimpse on the dataset.

```
## Rows: 568
## Columns: 32
## $ id                      <int> 842517, 84300903, 84348301, 84358402, 843786, ~
## $ diagnosis               <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "~
## $ radius_mean             <dbl> 20.570, 19.690, 11.420, 20.290, 12.450, 18.250~
## $ texture_mean            <dbl> 17.77, 21.25, 20.38, 14.34, 15.70, 19.98, 20.8~
## $ perimeter_mean          <dbl> 132.90, 130.00, 77.58, 135.10, 82.57, 119.60, ~
## $ area_mean               <dbl> 1326.0, 1203.0, 386.1, 1297.0, 477.1, 1040.0, ~
## $ smoothness_mean         <dbl> 0.08474, 0.10960, 0.14250, 0.10030, 0.12780, 0~
## $ compactness_mean        <dbl> 0.07864, 0.15990, 0.28390, 0.13280, 0.17000, 0~
## $ concavity_mean          <dbl> 0.08690, 0.19740, 0.24140, 0.19800, 0.15780, 0~
## $ concave_points_mean     <dbl> 0.07017, 0.12790, 0.10520, 0.10430, 0.08089, 0~
## $ symmetry_mean           <dbl> 0.1812, 0.2069, 0.2597, 0.1809, 0.2087, 0.1794~
## $ fractal_dimension_mean  <dbl> 0.05667, 0.05999, 0.09744, 0.05883, 0.07613, 0~
## $ radius_se               <dbl> 0.5435, 0.7456, 0.4956, 0.7572, 0.3345, 0.4467~
## $ texture_se              <dbl> 0.7339, 0.7869, 1.1560, 0.7813, 0.8902, 0.7732~
## $ perimeter_se            <dbl> 3.398, 4.585, 3.445, 5.438, 2.217, 3.180, 3.85~
## $ area_se                 <dbl> 74.08, 94.03, 27.23, 94.44, 27.19, 53.91, 50.9~
## $ smoothness_se           <dbl> 0.005225, 0.006150, 0.009110, 0.011490, 0.0075~
## $ compactness_se          <dbl> 0.013080, 0.040060, 0.074580, 0.024610, 0.0334~
## $ concavity_se            <dbl> 0.01860, 0.03832, 0.05661, 0.05688, 0.03672, 0~
## $ concave_points_se       <dbl> 0.013400, 0.020580, 0.018670, 0.018850, 0.0113~
## $ symmetry_se             <dbl> 0.01389, 0.02250, 0.05963, 0.01756, 0.02165, 0~
## $ fractal_dimension_se    <dbl> 0.003532, 0.004571, 0.009208, 0.005115, 0.0050~
## $ radius_worst            <dbl> 24.99, 23.57, 14.91, 22.54, 15.47, 22.88, 17.0~
## $ texture_worst           <dbl> 23.41, 25.53, 26.50, 16.67, 23.75, 27.66, 28.1~
## $ perimeter_worst         <dbl> 158.80, 152.50, 98.87, 152.20, 103.40, 153.20,~
## $ area_worst              <dbl> 1956.0, 1709.0, 567.7, 1575.0, 741.6, 1606.0, ~
## $ smoothness_worst        <dbl> 0.1238, 0.1444, 0.2098, 0.1374, 0.1791, 0.1442~
## $ compactness_worst       <dbl> 0.1866, 0.4245, 0.8663, 0.2050, 0.5249, 0.2576~
## $ concavity_worst         <dbl> 0.24160, 0.45040, 0.68690, 0.40000, 0.53550, 0~
## $ concave_points_worst    <dbl> 0.18600, 0.24300, 0.25750, 0.16250, 0.17410, 0~
## $ symmetry_worst          <dbl> 0.2750, 0.3613, 0.6638, 0.2364, 0.3985, 0.3063~
## $ fractal_dimension_worst <dbl> 0.08902, 0.08758, 0.17300, 0.07678, 0.12440, 0~
```

The dataset contains 0 duplicated rows and no missing values.

The most meaningful variable is the one this project will try to predict: cancer diagnostic. The distribution of the outcomes to this variable can be seen in both the table and bar chart bellow.

Diagnosis	Proportion
Benign	62.85%
Malignant	37.15%



Method and Results

Accuracy, Sensitivity and Specificity

Three metrics will be analyzed in this project to verify how good such prediction method is. They are: Accuracy, sensitivity and specificity. They can be better understood with the assistance of prediction against actual results table.

	Actually Positive	Actually Negative
Predicted Positive	True Positives (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

Accuracy can be defined as the numerical value of accuracy represents the proportion of true positive results (both true positive and true negative) in the selected population.

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

Sensitivity is typically quantified by true positives divided by the sum of true positives plus false negatives, or the proportion of actual positives. This can be also referred to as true positive rate or recall.

$$Sensitivity = \frac{TP}{TP + FN}$$

Specificity is typically quantified as the true negatives divided by the sum of the two negatives plus the false positives, or the proportions of negatives, This quantity is also called the true negative rate.

$$Specificity = \frac{TN}{TN + FP}$$

Specificity can also be written as the proportion of true positives divided by the sum of the true positives plus false positives, or the proportion of outcomes called positives.

$$Specificity = \frac{TP}{TP + FP}$$

This metric is also referred to as precision, and also as the positive predictive value, PPV.

Prediction Models

Three models will be shown here: Logistic Regression, Classification (decision) Tree and K-Nearest Neighbors.

Logistic Regression

Logistic regression an extension of linear regression that assures us the estimate of the conditional probability is, in fact, between 0 and 1. This is great when applied to categorical data.

```r
#logistic regression for prediction
glm_fit <- glm(diagnosis ~ ., train_set, family = "binomial")
p_hat_logit <- predict(glm_fit, test_set, type = "response")
y_hat_logit <- ifelse(p_hat_logit > 0.5, 1, 0) %>% factor()
test_set$diagnosis <- as.factor(test_set$diagnosis)
#confusion matrix of the lm method
cm_glm <- confusionMatrix(y_hat_logit, test_set$diagnosis)
#extracting accuracy, sensitivity and specificity
accuracy_glm <- cm_glm$overall[["Accuracy"]]
sensitivity_glm <- cm_glm$byClass[["Sensitivity"]]
specificity_glm <- cm_glm$byClass[["Specificity"]]
```

```r
#table for results
results_table <- data.frame(Method = "Logistic Regression",
                            Accuracy = accuracy_glm,
                            Sensitivity = sensitivity_glm,
                            Specificity = specificity_glm)
results_table %>% knitr::kable()
```

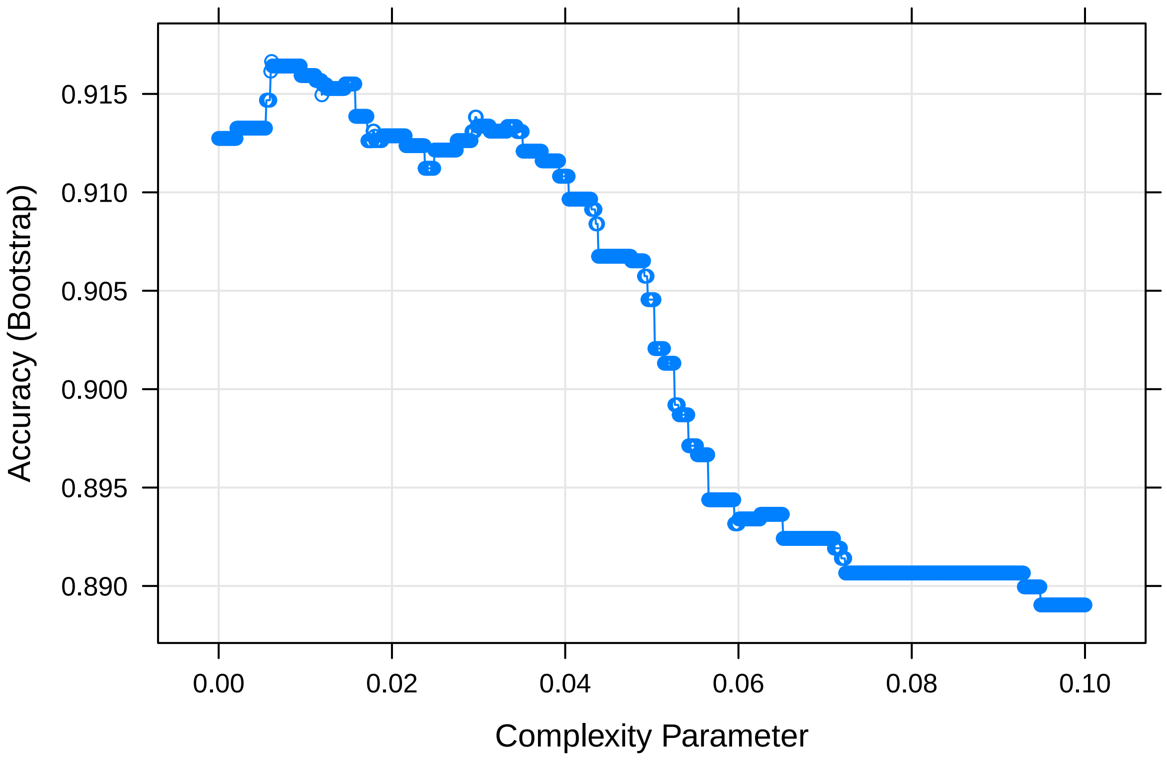
Method	Accuracy	Sensitivity	Specificity
Logistic Regression	0.973913	0.9583333	1

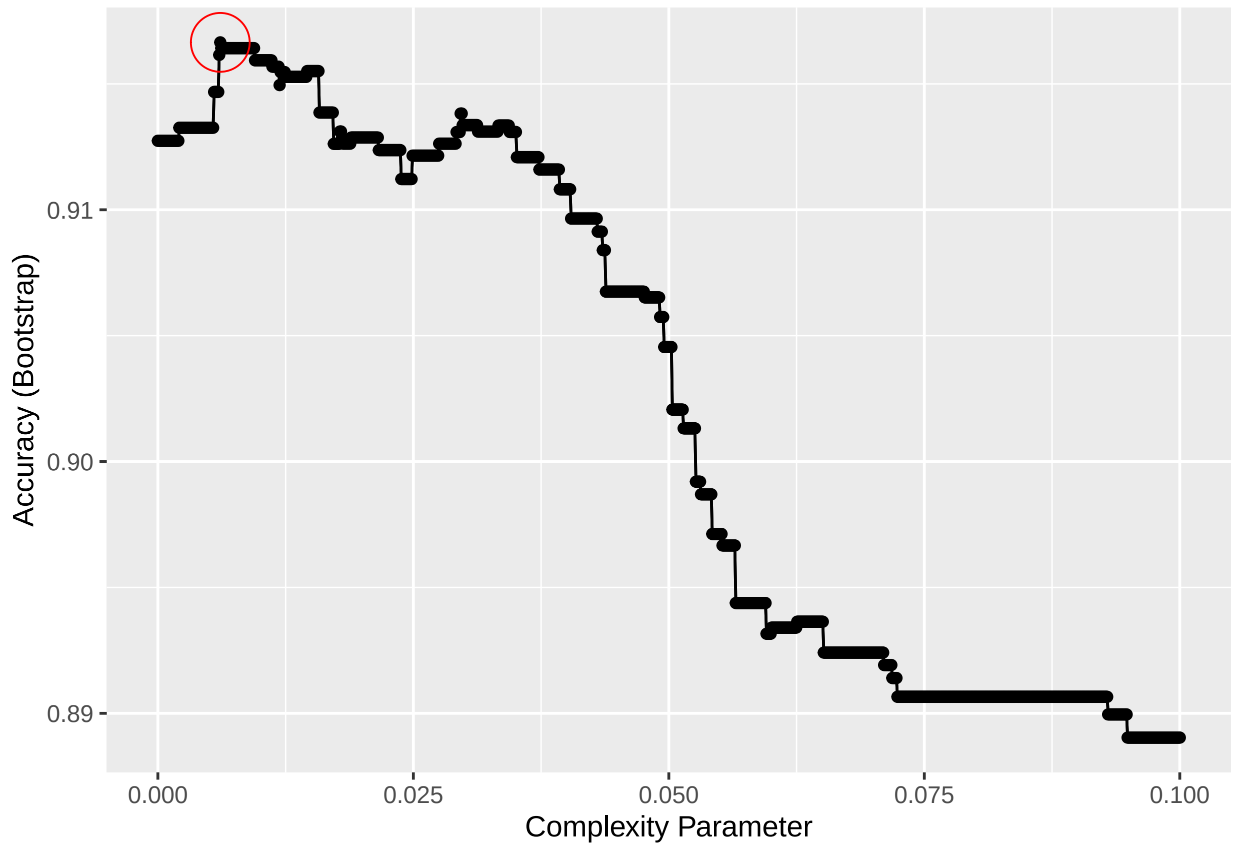
For comparison, when the data is inserted into a linear regression, the line format is too strict to the behavior of the many variables present in the dataset. It produces an accuracy of just 0.3826087, a low sensitivity of 0.0138889 and a specificity of 1.

Classification (Decision) Trees

A Classification tree, also known as Decision tree, is built through a process known as binary recursive partitioning. This is an iterative process of splitting the data into partitions, and then splitting it up further on each of the branches. This approach is excellent when data does not have too many variables and the variables have a somewhat defined frontier between then.

```r
set.seed(1, sample.kind = "Rounding")
train_rpart<- train(diagnosis ~ .,
                    method = "rpart",
                    data = train_set,
                    tuneGrid = data.frame(cp = seq(0, 0.1, len = 1000)))
```





```
y_hat <- predict(train_rpart, test_set)
cm_ct <- confusionMatrix(y_hat, test_set$diagnosis)
accuracy_ct <- cm_ct$overall[["Accuracy"]]
sensitivity_ct <-cm_ct$byClass[["Sensitivity"]]
specificity_ct <-cm_ct$byClass[["Specificity"]]
```

Method	Accuracy	Sensitivity	Specificity
Logistic Regression	0.9739130	0.9583333	1.0000000
classification (decision) tree	0.9565217	0.9861111	0.9069767

K-Nearest Neighbors

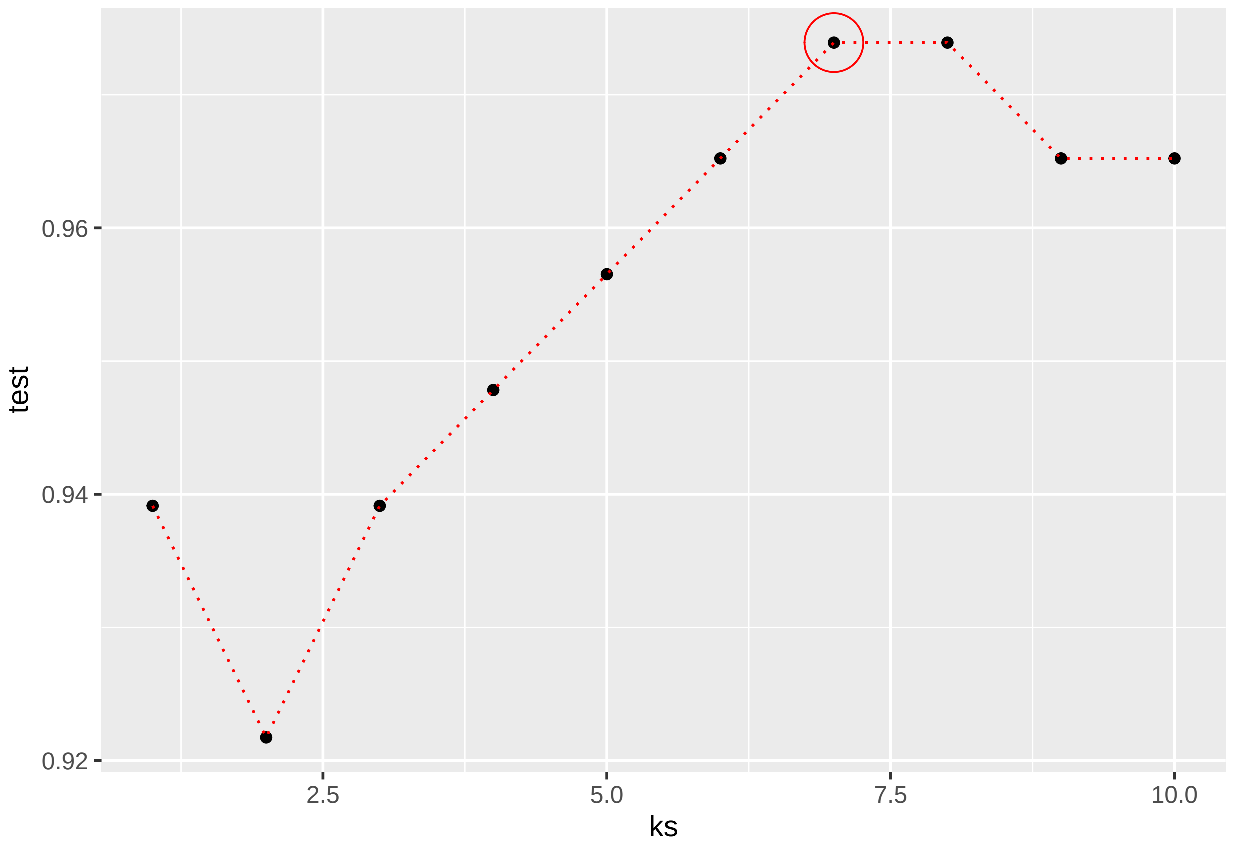
In short, KNN attempts to classify each sample in a dataset by evaluating its distance from its nearest neighbors. If the nearest neighbors are mostly of one class, the sample in question will be classified in this category. The number of nearest neighbors are called k. But how do we pick the best k? The best number of nearest neighbors that maximize our accuracy? We test with a range of values and pick the best.

Here is how:

```
ks <- seq(1, 10)
library(purrr)
accuracy <- map_df(ks, function(k){
  fit <- knn3(diagnosis ~ ., data = train_set, k = k)
  y_hat <- predict(fit, train_set, type = "class")
  cm_train <- confusionMatrix(data = y_hat, reference = train_set$diagnosis)
  train_error <- cm_train$overall["Accuracy"]

  y_hat <- predict(fit, test_set, type = "class")
  cm_test <- confusionMatrix(data = y_hat, reference = test_set$diagnosis)
  test_error <- cm_test$overall["Accuracy"]

  tibble(train = train_error, test = test_error)
})
```



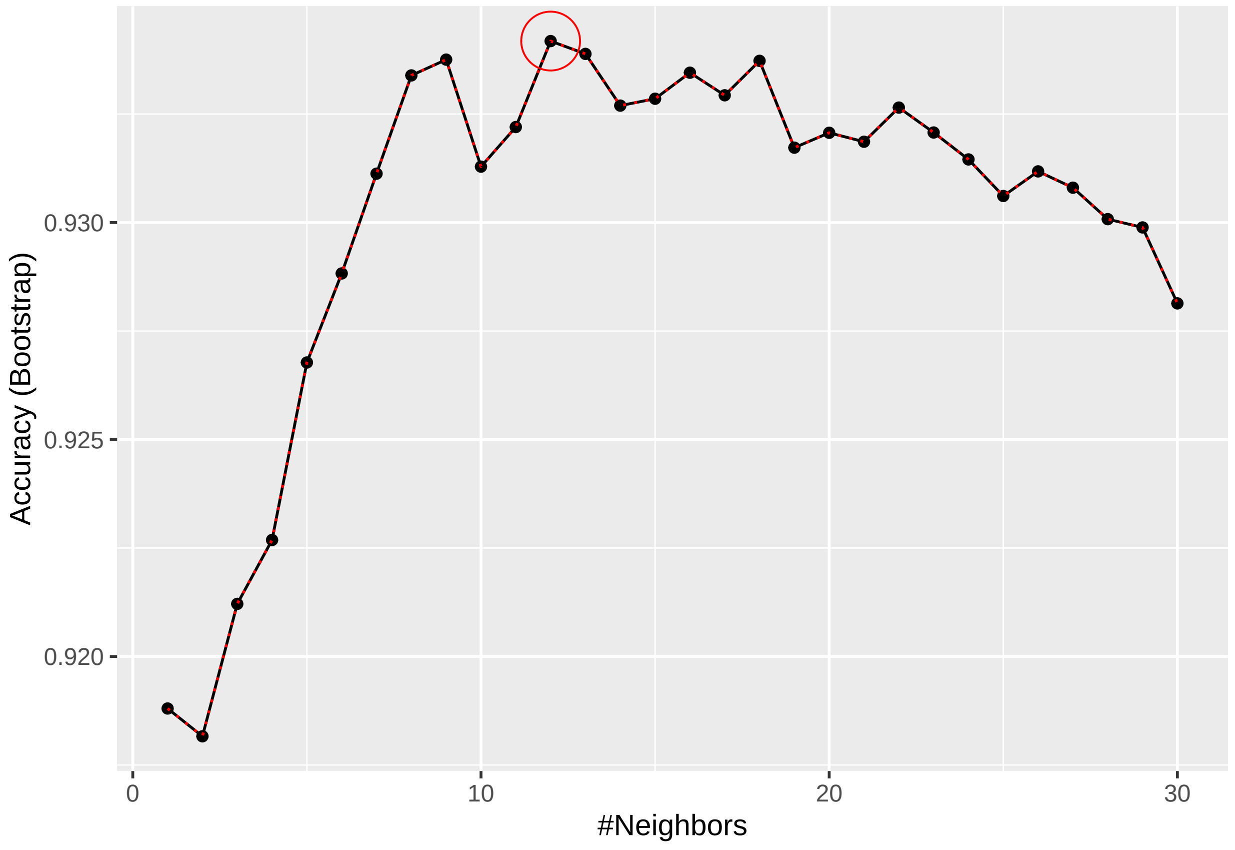
This shows that the number of neighbors that maximizes our accuracy is 7. When applied to the model the following results are produced.

```
knn_fit <-knn3(diagnosis ~., train_set, k = ks[which.max(accuracy$test)])
y_hat_knn <- predict(knn_fit, test_set, type = "class")
cm_knn <- confusionMatrix(y_hat_knn, test_set$diagnosis)
accuracy_knn <- cm_knn$overall[["Accuracy"]]
sensitivity_knn <- cm_knn$byClass[["Sensitivity"]]
specificity_knn <- cm_knn$byClass[["Specificity"]]
```

Method	Accuracy	Sensitivity	Specificity
Logistic Regression	0.9739130	0.9583333	1.0000000
classification (decision) tree	0.9565217	0.9861111	0.9069767

Method	Accuracy	Sensitivity	Specificity
K nearest neighbors	0.9739130	0.9861111	0.9534884

Again, for comparison, when using the bootstrap approach (for reference visit here) for K-nearest neighbors an lower accuracy of 0.9341815 is produced.



Conclusion

This project used three prediction models on a breast cancer diagnostic dataset by the University of Wisconsin. Surprisingly, all three models presented high accuracy, sensitivity and specificity. Two models best fitted the data with impressively the same accuracy, but k nearest neighbors had a higher sensitivity but a lower specificity. The models used in this project are based and do not required much computational power knowing the dataset contains 30 independent variables. More complex models could be applied to this project but are not viable to be produced in a home office machine.

References

- http://rafalab.dfci.harvard.edu/dsbook/large-datasets.html
- Wen Zhu, Nancy Zeng, Ning Wang. 2010. Sensitivity, Specificity, Accuracy, Associated Confidence Interval and ROC Analysis with Practical SAS Implementations https://lexjansen.com/nesug/nesug10/hl/hl07.pdf

- https://select-statistics.co.uk/blog/analysing-categorical-data-using-logistic-regression-models/#:~:text=Logistic%20regression%20models%20are%20a,and%20plan%20for%20future%20scenarios.
- https://www.solver.com/classification-tree