Breast Cancer Statistical Analysis

Jas Kainth

02/10/2020

```r
## GET THE DATA AND SPLIT IT INTO TRAINING GET AND TEST SET
# Split the data into a training and test set so we can test how good our model is
# on some real data
# We will use a 80/20 split
set.seed(1998)
split <- sample.split(data$class_logistic, SplitRatio = 0.80)
training_set <- subset(data, split == TRUE) %>%
  # Get rid of the class column since thats a recoded version of class_logistic
  select(-class)
test_set <- subset(data, split == FALSE) %>%
  select(-class)

# Note: we will set the seed in every chunk so if there is any randomness in
# any of the models, it will be reproducible
```

Model Selection

Logistical Classifier

The first model we will create is a logistic classifier. We will perform backward elimination, removing covariates based on the goodness of fit, using the AIC statistic, and looking at p-values. The basic form of a logistic model is the following;

$$Y_i \sim \text{Binomial}(\mu_i)$$
$$\ln \frac{\mu_i}{1-\mu_i} = \mathrm{X}_i\beta$$

where Y_i is the i^{th} person. In this model, the $\mathrm{X}_i\beta$ represents the log-odds and the μ_i represents the probability. To get from odds to probability, use probability $= \frac{\text{odds}}{1+\text{odds}}$ or $\mu_i = \frac{\exp \mathrm{X}_i\beta}{1+\exp \mathrm{X}_i\beta}$.

```r
## CREATE THE LOGISTIC MODELS, USING BACKWARD ELIMINATION TO PICK THE OPTIMAL MODEL
## CREATE TABLES SHOWING THE SUMMARY OUTPUT OF THE MODELS
set.seed(1998)
# We will start with a logistic regression; there are more classifcation methods
# which may give us better results but we will get to those later. The nice thing
# about logistic regression is we can see how to covariates influence the dependent
# variable via the summary() function which isn't the case for all the other
# classification methods we will use
# We will use all the predictors as covariates because from our EDA it seems they
# all follow a general trend with the dependent variables
# Some of them may result in not being statistically siginificant because of the
# high correlation among predictors
```

```r
logistic_classifier1 <- glm(formula = class_logistic ~ .,
                 family = binomial,
                 data = training_set)
logistic_classifier1 %>%
  tidy() %>%
  mutate(term = case_when(term == "clump_thickness" ~ "Clump Thickness",
                          term == "unif_of_cell_size" ~ "Uniformity of Cell Size",
                          term == "unif_of_cell_shape" ~ "Uniformity of Cell Shape",
                          term == "marginal_adhesion" ~ "Marginal Adhesion",
                          term == "single_epithelial_cell_size" ~ "Single Epithelial Cell Size",
                          term == "bare_nuclei" ~ "Bare Nuclei",
                          term == "bland_chromatin" ~ "Bland Chromatin",
                          term == "normal_nucleoli" ~ "Normal Nucleoli",
                          term == "mitoses" ~ "Mitoses",
                          TRUE ~ "Baseline")) %>%
  knitr::kable(caption = "Logistic Model 1 Summary Output")
```

Table 1: Logistic Model 1 Summary Output

term	estimate	std.error	statistic	p.value
Baseline	-10.2562297	1.3530124	-7.5802926	0.0000000
Clump Thickness	0.6997706	0.1867931	3.7462342	0.0001795
Uniformity of Cell Size	-0.1213563	0.2209680	-0.5492029	0.5828662
Uniformity of Cell Shape	0.3657786	0.2440610	1.4987182	0.1339468
Marginal Adhesion	0.3669377	0.1541567	2.3802901	0.0172990
Single Epithelial Cell Size	-0.1232997	0.2000401	-0.6163747	0.5376473
Bare Nuclei	0.3992782	0.1112554	3.5888440	0.0003321
Bland Chromatin	0.4653600	0.1774797	2.6220466	0.0087403
Normal Nucleoli	0.3005112	0.1265104	2.3753873	0.0175305
Mitoses	0.2938408	0.3299858	0.8904652	0.3732161

```r
# AIC: 101.93; AIC is a goodness of fit statistic. We want this to be as low as
# possible
# We will remove some of the covariates which have the highest p-values and see
# if we can get a better model
# Remove single_epithelial_cell_size & unif_of_cell_size


logistic_classifier2 <- glm(formula = class_logistic ~ clump_thickness +
                              unif_of_cell_shape + marginal_adhesion +
                              bare_nuclei + bland_chromatin + normal_nucleoli +
                              mitoses,
                 family = binomial,
                 data = training_set)
# AIC: 98.675; This is better, let's see if we can do better
# Remove mitoses
logistic_classifier2 %>%
  tidy() %>%
  mutate(term = case_when(term == "clump_thickness" ~ "Clump Thickness",
                          term == "unif_of_cell_shape" ~ "Uniformity of Cell Shape",
                          term == "marginal_adhesion" ~ "Marginal Adhesion",
                          term == "bare_nuclei" ~ "Bare Nuclei",
                          term == "bland_chromatin" ~ "Bland Chromatin",
                          term == "normal_nucleoli" ~ "Normal Nucleoli",
                          term == "mitoses" ~ "Mitoses",
                          TRUE ~ "Baseline")) %>%
  knitr::kable(caption = "Logistic Model 2 Summary Output")
```

Table 2: Logistic Model 2 Summary Output

term	estimate	std.error	statistic	p.value
Baseline	-10.0524710	1.2677702	-7.9292534	0.0000000
Clump Thickness	0.6580567	0.1774214	3.7090044	0.0002081
Uniformity of Cell Shape	0.2610548	0.1789558	1.4587671	0.1446292
Marginal Adhesion	0.3314126	0.1508444	2.1970489	0.0280170
Bare Nuclei	0.3815363	0.1035154	3.6857926	0.0002280
Bland Chromatin	0.4179196	0.1691267	2.4710447	0.0134719
Normal Nucleoli	0.2640726	0.1171389	2.2543543	0.0241739
Mitoses	0.2867237	0.3280821	0.8739389	0.3821516

```
logistic_classifier3 <- glm(formula = class_logistic ~ clump_thickness +
                              unif_of_cell_shape + marginal_adhesion +
                              bare_nuclei + bland_chromatin + normal_nucleoli,
                family = binomial,
                data = training_set)
# AIC: 97.766; unif_of_cell_shape is still not statistically significant so let's
# see if we can do better by removing that
logistic_classifier3 %>%
  tidy() %>%
  mutate(term = case_when(term == "clump_thickness" ~ "Clump Thickness",
                         term == "unif_of_cell_shape" ~ "Uniformity of Cell Shape",
                         term == "marginal_adhesion" ~ "Marginal Adhesion",
                         term == "bare_nuclei" ~ "Bare Nuclei",
                         term == "bland_chromatin" ~ "Bland Chromatin",
                         term == "normal_nucleoli" ~ "Normal Nucleoli",
                         TRUE ~ "Baseline")) %>%
  knitr::kable(caption = "Logistic Model 3 Summary Output")
```

Table 3: Logistic Model 3 Summary Output

term	estimate	std.error	statistic	p.value
Baseline	-10.1189029	1.2871284	-7.861611	0.0000000
Clump Thickness	0.7353629	0.1669259	4.405324	0.0000106
Uniformity of Cell Shape	0.2710022	0.1746600	1.551598	0.1207584
Marginal Adhesion	0.3480460	0.1501143	2.318540	0.0204200
Bare Nuclei	0.3707254	0.1033853	3.585861	0.0003360
Bland Chromatin	0.4225517	0.1680030	2.515144	0.0118984
Normal Nucleoli	0.2669270	0.1173013	2.275568	0.0228719

```
logistic_classifier4 <- glm(formula = class_logistic ~ clump_thickness +
                              marginal_adhesion + bare_nuclei + bland_chromatin +
                              normal_nucleoli,
                family = binomial,
                 data = training_set)
# AIC: 98.494; so it increases a little bit which is not what we would want
# Let's perform a likelihood ratio test to see which model is better between
# logistic_classifier3 and logistic_classifier4
logistic_classifier4 %>%
  tidy() %>%
  mutate(term = case_when(term == "clump_thickness" ~ "Clump Thickness",
                          term == "marginal_adhesion" ~ "Marginal Adhesion",
                          term == "bare_nuclei" ~ "Bare Nuclei",
                          term == "bland_chromatin" ~ "Bland Chromatin",
                          term == "normal_nucleoli" ~ "Normal Nucleoli",
                          TRUE ~ "Baseline")) %>%
  knitr::kable(caption = "Logistic Model 4 Summary Output")
```

Table 4: Logistic Model 4 Summary Output

term	estimate	std.error	statistic	p.value
Baseline	-10.4953530	1.3144100	-7.984839	0.0000000
Clump Thickness	0.8258562	0.1631345	5.062426	0.0000004
Marginal Adhesion	0.3972091	0.1476173	2.690804	0.0071280
Bare Nuclei	0.4338568	0.0946070	4.585888	0.0000045
Bland Chromatin	0.4994856	0.1536062	3.251729	0.0011471
Normal Nucleoli	0.3345007	0.1069814	3.126719	0.0017677

It should be noted that the point estimates have not yet been exponentiated. We will take a look at the exponentiated point estimates and their interpretation once we have a final model. Taking a look at Table 1, we see that a lot of the p-values are greater than 0.05. This would suggest that they are not important in predicting the type of cancer but when we were exploring the data in the EDA it seemed that when any of the predictors increased, the cancer was more likely to be malignant. Therefore, the high p-values are probably due to the high correlation among predictors. Also, the AIC of this model is 101.93. AIC is a goodness of fit statistic. By itself, it does not mean anything but when we are comparing models we generally want to pick the model with the lowest AIC.

To get to model 2, we remove 2 predictors with the highest p-value which are 'Uniformity of Cell Size' & 'Single Epithelial Cell Size'. For this model, we again get non-significant p-values. But, we note that the AIC for this model decreases to 98.675 which means this model is better than the previous model. However, let's see if we can do better by removing the predictor with the greatest p-value, which is 'Mitoses'.

For model 3, there is only one predictor with a p-value greater than 0.05, which is 'Uniformity of Cell Shape'. The AIC for this model is 97.766 which is lower than the previous model, so that's a good sign. Let's see if we can remove the non-significant predictor and get a better model.

For model 4, we see that there are no predictors with p-values higher than 0.05. However, the AIC for this model is 98.494 which is higher for this model than the previous model which would suggest that model 3 fits our data better than model 4. Let's explore this further using the likelihood ratio test and using cross-validation.

```
## PERFORM A LIKELIHOOD RATIO TEST OF THE MODELS TO SEE WHICH ONE IS "BETTER"
## AND CREATE CONFUSION MATRICIES TO TEST THEIR ACCURACY
# Likelihood ratio test
lmtest::lrtest(logistic_classifier3, logistic_classifier4) %>%
  knitr::kable(caption = "Likelihood Ratio Test of Model 3 vs Model 4")
```

Table 5: Likelihood Ratio Test of Model 3 vs Model 4

#Df	LogLik	Df	Chisq	Pr(>Chisq)
7	-41.88281	NA	NA	NA
6	-43.24714	-1	2.728667	0.0985615

```r
# The p-value for this is greater than 0.05 which means this test is telling us that
# model 3 is better
# At this point we have to make a decision which model to use since we are getting
# conflicting results; since the p-value for the covariate was not significant and
# also there was a high correlation among the predictors, I am choosing to include
# the covariate in our model (only looking at p-values is generally not a good idea)
# But we will use both these models to predict the results from the test set and we
# will see which one gives us better results

y_pred_model3 <- predict(logistic_classifier3, newdata = test_set,
                         type = "response")
# The predict function will give us values ranging from 0 to 1 so turn those values
# into either 0 or 1 to test the model
y_logistic_model3 <- ifelse(y_pred_model3 > 0.5, 1, 0)
y_pred_model4 <- predict(logistic_classifier4, newdata = test_set,
                         type = "response")
y_logistic_model4 <- ifelse(y_pred_model4 > 0.5, 1, 0)

# Create a confusion matrix
cm3 <- table(y_logistic_model3, test_set$class_logistic)
cm4 <- table(y_logistic_model4, test_set$class_logistic)
# Accuracy of model 3
(cm3[1] + cm3[4]) / length(test_set$class_logistic)
```

```
[1] 0.9562044
```

```r
# 0.9562044

# Accuracy of model 4
(cm4[1] + cm4[4]) / length(test_set$class_logistic)
```

```
[1] 0.9562044
```

```r
# 0.9562044

# They are the same; in fact the confusion matrix is the same for both of them
knitr::kable(cm3, caption = "Confusion Matrix of Model 3")
```

Table 6: Confusion Matrix of Model 3

	0	1
0	87	4
1	2	44

```r
knitr::kable(cm4, caption = "Confusion Matrix of Model 4")
```

Table 7: Confusion Matrix of Model 4

	0	1
0	87	4
1	2	44

Table 5 shows the results of the likelihood ratio test. The p-value of this test is greater than 0.05, which would lead us to believe that Model 3 is better than Model 4. Tables 6 & 7 are confusion matrices of Model 3 and 4, respectively. As we can see, they are the same, and both have an accuracy of 0.9562044.

```r
## USE CROSS VALIDATION TO SEE WHICH MODEL PERFORMS BETTER
set.seed(1998)
# Perform k-fold cross validation to better indicate our model's performance
train_control <- trainControl(method = "cv", number = 10)

# We will test both models; model 3 first (with 6 predictors)
# The train function doesn't like it is we use integers as Y if it is classification
# so we turn that vector into factors
log_cv_model3 <- train(as.factor(class_logistic) ~ clump_thickness +
                              unif_of_cell_shape + marginal_adhesion +
                              bare_nuclei + bland_chromatin + normal_nucleoli,
               data = training_set,
               trControl = train_control,
               method = "glm",
               family=binomial())
knitr::kable(log_cv_model3$results,
             caption = "Results of Cross-Validation for Model 3")
```

Table 8: Results of Cross-Validation for Model 3

parameter	Accuracy	Kappa	AccuracySD	KappaSD
none	0.9688552	0.9318115	0.0173743	0.0376174

```r
log_cv_model4 <- train(as.factor(class_logistic) ~ clump_thickness +
                              marginal_adhesion + bare_nuclei + bland_chromatin +
                              normal_nucleoli,
               data = training_set,
               trControl = train_control,
               method = "glm",
               family=binomial())

knitr::kable(log_cv_model4$results,
             caption = "Results of Cross-Validation for Model 4")
```

Table 9: Results of Cross-Validation for Model 4

parameter	Accuracy	Kappa	AccuracySD	KappaSD
none	0.9652525	0.923103	0.0201006	0.0447867

```
# The difference between accuracy and kappa - Accuracy is the percentage of
# correctly classified instances out of all instances. Kappa takes into account the
# possibility of the correct result occurring by chance.
# If you're interested, you can read more here
# https://machinelearningmastery.com/machine-learning-evaluation-metrics-in-r/
# Anyway, it looks like Kappa is more useful to use but let's take a look at both
# Note: Model 3 is where we kept the one predictor that was not statistically
# significant because the AIC for model 3 is lower than the AIC of model 4 whereas
# the likelihood ratio test tells us that model 4 is "better"
# Accuracy: Model 3 has a 0.3732 % higher accuracy and the SD of the accuracy is
# 13.56328% lower for model 3
# Kappa: Model 3 has a 0.9434% % higher Kappa and the SD of kappa is 16.00765 %
# higher for model 3
# This means that model 3 fits better (has higher accuracy and a higher kappa)
# and also the SD of the accuracy and kappa is lower, which means it's more
# consistent; therefore, we will use model 3 which is where we predict the cancer
# using clump_thickness, marginal_adhesion, bare_nuclei, bland_chromatin,
# normal_nucleoli & unif_of_cell_shape
# Final Accuracy: 0.9688552; Kappa: 0.9318115
```

We see that the (k-fold) cross-validation results show us accuracy and kappa. The difference between accuracy and kappa is accuracy is the percentage of correctly classified instances out of all the instances. Kappa, however, takes into account the possibility of the correct result occurring by chance. Table 8 & 9 looks at the results of the cross-validation of models 3 & 4. We see that the accuracy and kappa are higher for model 3 and the SD (standard deviation) of accuracy and kappa are lower for model 3. This means model 3 predicts the results more accurately and also the accuracy deviates a less from one test to another.

This is why we will choose model 3, where we predict the cancer type using 'Clump Thickness', 'Uniformity of Cell Shape', 'Marginal Adhesion', 'Bare Nuclei', 'Bland Chromatin' & 'Normal Nucleoli', as the optimal model. We will interpret the results of that model and also create a plot to see how the predictors affect the odds of having malignant cancer. Also, moving forward, when we create models using different classification techniques, we will be using these predictors for those rather than performing backwards elimination again. The reasoning behind this is we don't get a summary output for those models as we do for logistic regression so we would have to completely base our predictors on the accuracy of the test set rather than doing more tests as we performed for the logistic regression.

Table 8 illustrates the final accuracy statistics of our model.

```
## CREATE A SUMMARY OUTPUT OF OUR MODEL AND VISUALIZE THE RESULTS
set.seed(1998)
## What do the point estimates of our model mean?
# First, we note that the base output that we get is log-odds; to make this more
# interpretable we will switch this to odds but taking e^{point_estimate}
# This means, for instance, if e^{point_estimate} is 1.04, that increasing one
# unit of that variable will increase the odds by 1.04 - 1 = 0.04 (or 4%)
# Note, however, that this is increasing the odds, if we want probability then
# we get that by using the following formula: probability = odds/(odds + 1)
tidy_model <- logistic_classifier3 %>%
  tidy(conf.int = TRUE, exponentiate = TRUE)
tidy_model %>%
  select(term, estimate, conf.low, conf.high, p.value) %>%
  mutate(term = case_when(term == "clump_thickness" ~ "Clump Thickness",
                          term == "unif_of_cell_shape" ~ "Uniformity of Cell Shape",
                          term == "marginal_adhesion" ~ "Marginal Adhesion",
                          term == "bare_nuclei" ~ "Bare Nuclei",
                          term == "bland_chromatin" ~ "Bland Chromatin",
                          term == "normal_nucleoli" ~ "Normal Nucleoli",
                          TRUE ~ "Baseline")) %>%
  knitr::kable(caption = "Effects on the Odds of Having Malignant Cancer")
```

Table 10: Effects on the Odds of Having Malignant Cancer

term	estimate	conf.low	conf.high	p.value
Baseline	0.0000403	0.0000021	0.0003547	0.0000000
Clump Thickness	2.0862389	1.5506523	3.0184372	0.0000106
Uniformity of Cell Shape	1.3112779	0.9534260	1.9101739	0.1207584
Marginal Adhesion	1.4162974	1.0744012	1.9479781	0.0204200
Bare Nuclei	1.4487852	1.1899850	1.7957194	0.0003360
Bland Chromatin	1.5258501	1.1131882	2.1686469	0.0118984
Normal Nucleoli	1.3059451	1.0454146	1.6649755	0.0228719

```
# What do we see?
# First, we can see that the baseline (all the predictors have a value of 1)
# probability of having malignant cancer is very low 0.0000403/ (0.0000403 + 1)
# = 0.0000403
# Also, we see that increasing the value of any of the predictors will increase the
# odds (and therefore also the probability) of having malignant cancer
# The largest effect is of clump thickness which increases the odds by almost
# 109% (2.09 - 1 = 1.09 or 109%) for each increase of 1 unit!
```

Table 10 shows the final summary output of our model. The coeffcients have been exponentiated here, which means the point estimates represents odds rather than log-odds. The 'Baseline' represents the odds of having malignant cancer given the remainded of the predictors have a value of 1. The probability is then $\frac{0.0000403}{1+0.0000403} \approx 0.0000403 = 4.03 \times 10^{-5}$. For the remainder of the point estimates, the are multiplicative factors. This means, looking at 'Clump Thickiness' for example, if we increase the value of the predictor by 1, all else the same, the odds would increase by 108.6% (2.086 - 1 = 1.086 or 109%). Then, to get to probability from odds, we use probability $= \frac{\text{odds}}{1+\text{odds}}$.

```
tidy_model %>%
  filter(term != "(Intercept)") %>%
  mutate(term = case_when(term == "clump_thickness" ~ "Clump Thickness",
                        term == "marginal_adhesion" ~ "Marginal Adhesion",
                        term == "bare_nuclei" ~ "Bare Nuclei",
                        term == "bland_chromatin" ~ "Bland Chromatin",
                        term == "unif_of_cell_shape" ~ "Uniformity of Cell Shape",
                        term == "normal_nucleoli" ~ "Normal Nucleoli")) %>%
  mutate(term = fct_reorder(term, estimate)) %>%
  ggplot(aes(x = estimate, y = term)) +
  geom_errorbar(aes(xmin = conf.low, xmax = conf.high), color = "grey55") +
  geom_point(color = "black") +
  theme_minimal() +
  labs(
    title = "Point Estimates with 95% Confidence Interval",
    subtitle = "Baseline Probability = 0.0000403",
    x = " ",
    y = " ") +
  expand_limits(x = 0) +
  theme(axis.ticks = element_blank())
```

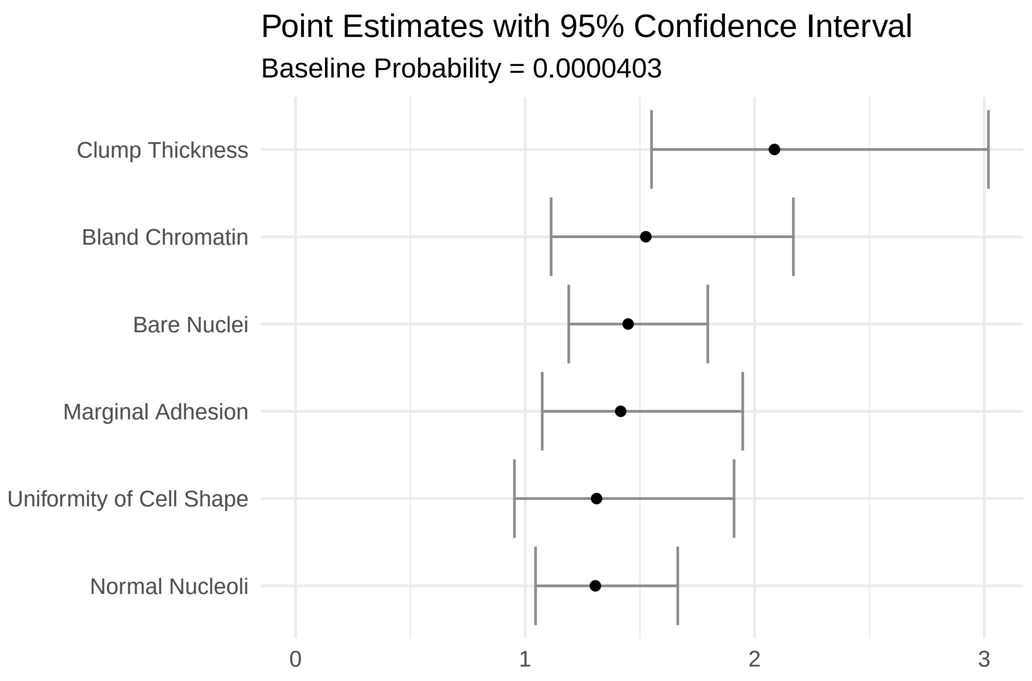

Figure 1: Effects by Predictors to the Odds of Having Malignant Breast Cancer

Figure 1 is a visualization that shows how predictors affect the odds of having malignant cancer. We note that only ‘Uniformity of Cell Shape’ has an error bar that crosses 1. This error bar is a 95% confidence interval and we expected this since the p-value is greater than 0.05. It should be noted that it crosses 1, not 0, since the terms have been exponentiated ($\exp(0) = 1$). Based on the point estimates, it seems that ‘Clump Thickness’ has the largest effect on the odds, however, the error bar is quite large. The effects from ‘Bare Nuclei’ and ‘Normal Nucleoli’ aren’t nearly as large but the error bars are more narrow for these estimates. But, we do note that all of them have point estimates greater than 1, which means that increasing the values of these predictors results in an increased odds (and therefore probability) of having malignant cancer.

K-th Nearest Neighbor (KNN)

The next model which we will create is a k-th nearest neighbor (KNN) model. This model checks the k nearest points (neighbors) to the new point and then places it in the group that has more neighbors from the corresponding group. For this model (and all future models), we will use the same covariates as the 3rd logistic model (Clump Thickness, Marginal Adhesion, Bare Nuclei, Bland Chromatin, Normal Nucleoli, Uniformity of Cell Shape). First, we will perform cross validation on the model, using these predictors, to pick the optimal k (the number of neighbors the model takes into consideration).

```r
set.seed(1998)
# We will use the caret library again to create this model to cross-validate the
# value for K
# We will use the same predictors as model 3; this model helps us not overfit the
# data due to the correlation among predictors
train_control <- trainControl(method = "cv", number = 10)
knn_cv <- train(as.factor(class_logistic) ~ clump_thickness +
                              marginal_adhesion + bare_nuclei + bland_chromatin +
                              normal_nucleoli + unif_of_cell_shape,
             method = "knn",
             # Test different values of k
             tuneGrid = expand.grid(k = 1:15),
             trControl = train_control,
             metric = "Accuracy",
             data = training_set)

knn_cv$results %>%
  knitr::kable(caption = "Result of cross-validation for knn model")
```

Table 11: Result of cross-validation for knn model

k	Accuracy	Kappa	AccuracySD	KappaSD
1	0.9524242	0.8953108	0.0286778	0.0619196
2	0.9651852	0.9237776	0.0327257	0.0713646
3	0.9671044	0.9282512	0.0340702	0.0743881
4	0.9743771	0.9442909	0.0246006	0.0532361
5	0.9725589	0.9403198	0.0231240	0.0500016
6	0.9688889	0.9323388	0.0244268	0.0528260
7	0.9707071	0.9362136	0.0246679	0.0533643
8	0.9707071	0.9361210	0.0300392	0.0655363
9	0.9725253	0.9400921	0.0313098	0.0683080
10	0.9725253	0.9400921	0.0313098	0.0683080
11	0.9743434	0.9438658	0.0275147	0.0603795
12	0.9725253	0.9397208	0.0301138	0.0663159
13	0.9725253	0.9397208	0.0301138	0.0663159
14	0.9707071	0.9354840	0.0335073	0.0743080
15	0.9688889	0.9314129	0.0322093	0.0714406

```r
# Using k = 4 we get the best model; with this we get
# Accuracy: 0.9743771; Kappa: 0.9442909
# This means that this model is slightly better than the logistic model
```

Table 11 shows the results of the cross-validation. We want to pick the k which corresponds to the highest accuracy. In case there was a tie, we would pick the larger value of k, but that is not the case for this model. The optimal k for this model is 4 (which also happens to have the largest value for kappa). As we can see, the accuracy of this model is 0.9743771. For comparison, the accuracy for the optimal logistic model was 0.9688552 which means this model performs slightly better than the logistic model.

```r
set.seed(1998)
# Using the k = 4, how well does this predict our test set results?
# The knn function gives us predicted results
# But we need dataframes that only have the columns that we need for the model
training_set_knn <- training_set %>%
  select(clump_thickness, marginal_adhesion, bare_nuclei, bland_chromatin,
         normal_nucleoli, unif_of_cell_shape)
test_set_knn <- test_set %>%
  select(clump_thickness, marginal_adhesion, bare_nuclei, bland_chromatin,
         normal_nucleoli, unif_of_cell_shape)
cl_knn <- training_set$class_logistic
y_pred_knn <- knn(train = training_set_knn,
             test = test_set_knn,
             cl = cl_knn,
             # Set k = 4
             k = 4,
             prob = TRUE)
y_actual <- test_set$class_logistic
cm_knn <- table(y_pred_knn, y_actual)

# What is the accuracy
(cm_knn[1] + cm_knn[4]) / length(test_set$class_logistic)
```

```
## [1] 0.9635036
```

```
# 0.9635036

knitr::kable(cm_knn, caption = "Confusion Matrix for KNN model")
```

Table 12: Confusion Matrix for KNN model

	0	1
0	86	2
1	3	46

```
# The accuracy of the logistic regression classifier was 0.9562044, so the
# accuracy of this model is slightly higher on the test set and using the
# cross-validation for this and using the k-fold cross-validation for the
# logistic model, the accuracy and kappa is higher for this model than the
# logistic model. However, the SD for accuracy and kappa is also higher for
# this model relative to the logistic model
```

Table 12 is the confusion matrix of this model, comparing the predicted results to the actual results from the test set. We cannot see the effect of the covariates on the response to this model. Therefore, we cannot examine if increasing the value of covariates increases or decreases the probability of malignant cancer. So, when comparing this model to the logistic regression, this model gives slightly more accurate results but the other model has more interpretable results.

Support Vector Machine (SVM)

The next model which we will create is a support vector machine (SVM). This model uses a line, plane or hyperplane (depending on the number of dimensions) to differentiate the two groups. The shape of the hyperplane can take many shapes, through the parameter of the kernel, so first, we will find the kernel which has the highest accuracy and further tune that model.

```
set.seed(1998)
# We will use the same predictors as the 4th logistic model
svm_model_linear <- svm(formula = class_logistic ~ clump_thickness +
                              marginal_adhesion + bare_nuclei + bland_chromatin +
                              normal_nucleoli + unif_of_cell_shape,
                 data = training_set,
                 type = "C-classification",
                 # We will test different kernels to see which gives us the best
                 # result; First we will try linear
                 kernel = "linear")

# Polynomail kernel
svm_model_polynomial <- svm(formula = class_logistic ~ clump_thickness +
```

```r
                              marginal_adhesion + bare_nuclei + bland_chromatin +
                              normal_nucleoli + unif_of_cell_shape,
                 data = training_set,
                 type = "C-classification",
                 kernel = "polynomial")

# Radial basis
svm_model_radial <- svm(formula = class_logistic ~ clump_thickness +
                              marginal_adhesion + bare_nuclei + bland_chromatin +
                              normal_nucleoli + unif_of_cell_shape,
                 data = training_set,
                 type = "C-classification",
                 kernel = "radial")

# Sigmoid
svm_model_sigmoid <- svm(formula = class_logistic ~ clump_thickness +
                              marginal_adhesion + bare_nuclei + bland_chromatin +
                              normal_nucleoli + unif_of_cell_shape,
                 data = training_set,
                 type = "C-classification",
                 kernel = "sigmoid")

# How do the perdictions vary when comparing to test set?
y_pred_linear <- predict(svm_model_linear, newdata = test_set)
y_pred_polynomail <- predict(svm_model_polynomial, newdata = test_set)
y_pred_radial <- predict(svm_model_radial, newdata = test_set)
y_pred_sigmoid <- predict(svm_model_sigmoid, newdata = test_set)
y_actual <- test_set$class_logistic
cm_linear <- table(y_pred_linear, y_actual)
cm_polynomail <- table(y_pred_polynomail, y_actual)
cm_radial <- table(y_pred_radial, y_actual)
cm_sigmoid <- table(y_pred_sigmoid, y_actual)

knitr::kable(cm_linear,
             caption = "Confusion matrix for SVM model using a linear kernel")
```

Table 13: Confusion matrix for SVM model using a linear kernel

	0	1
0	86	3
1	3	45

```r
(cm_linear[1] + cm_linear[4])/(length(y_actual)) # Accuracy of 0.9562044
```

```
## [1] 0.9562044
```

```r
knitr::kable(cm_polynomail,
             caption = "Confusion matrix for SVM model using a polynomial kernel")
```

Table 14: Confusion matrix for SVM model using a polynomial kernel

	0	1
0	88	6
1	1	42

```
(cm_polynomail[1] + cm_polynomail[4])/(length(y_actual)) # Accuracy of 0.9489051
```

```
## [1] 0.9489051
```

```
knitr::kable(cm_radial,
             caption = "Confusion matrix for SVM model using a radial kernel")
```

Table 15: Confusion matrix for SVM model using a radial kernel

	0	1
0	85	2
1	4	46

```
(cm_radial[1] + cm_radial[4])/(length(y_actual)) # Accuracy of 0.9562044
```

```
## [1] 0.9562044
```

```
knitr::kable(cm_sigmoid,
             caption = "Confusion matrix for SVM model using a sigmoid kernel")
```

Table 16: Confusion matrix for SVM model using a sigmoid kernel

	0	1
0	85	1
1	4	47

```
(cm_sigmoid[1] + cm_sigmoid[4])/(length(y_actual)) # Accuracy of 0.9635036
```

```
## [1] 0.9635036
```

```
# For comparison, the logistic model had an accuracy, when comparing to the test
# set, of 0.9562044 but it had an accuracy of 0.9725253 with the k-fold cross
# validation which is higher than the highest accuracy of the svm (which is with
# the sigmoid kernel)
```

Using the confusion matrices, we find that the accuracy of the SVM model is the following

```
# Summarize the accuracies of the different kernels
tibble(Kernel = c("Linear", "Polynomial", "Radial", "Sigmoid"),
       # These values correspond to the confusion matricies
       Accuracy = c(0.9562044, 0.9489051, 0.9562044, 0.9635036)) %>%
  knitr::kable(caption = "Accuracy of SVM under different kernels")
```

Table 17: Accuracy of SVM under different kernels

Kernel	Accuracy
Linear	0.9562044
Polynomial	0.9489051
Radial	0.9562044
Sigmoid	0.9635036

Table 17 shows the highest accuracy is with the sigmoid kernel, however, the train function we use the apply cross-validation doesn't have a method for the sigmoid kernel so we will use the next highest accuracy (which is linear and radial).

```r
set.seed(1998)
# We will try to cross validate the svm model under different cost of constraints
# However, in the train() function, there is no setting for the sigmoid kernel
# so we will use the radial kernel
train_control <- trainControl(method = "cv", number = 10)
svm_radial_cv <- train(as.factor(class_logistic) ~ clump_thickness +
                              marginal_adhesion + bare_nuclei + bland_chromatin +
                              normal_nucleoli + unif_of_cell_shape,
             data = training_set,
             # There is no method for the sigmoid
             method = "svmRadial",
             trControl = train_control)

svm_radial_cv$results %>%
  knitr::kable(caption = "Results from cross-validation with radial kernel")
```

Table 18: Results from cross-validation with radial kernel

sigma	C	Accuracy	Kappa	AccuracySD	KappaSD
1.022693	0.25	0.9507071	0.8965516	0.0437129	0.0899571
1.022693	0.50	0.9671380	0.9299289	0.0318150	0.0669896
1.022693	1.00	0.9689226	0.9331312	0.0271855	0.0579107

```r
# The best accuracy which we get is with C = 1; Accuracy = 0.9707407,
# Kappa = 0.9371979, Accuracy SD = 0.02597202, Kappa SD = 0.05519531

svm_linear_cv <- train(as.factor(class_logistic) ~ clump_thickness +
                              marginal_adhesion + bare_nuclei + bland_chromatin +
                              normal_nucleoli + unif_of_cell_shape,
             data = training_set,
             # There is no method for the sigmoid
             method = "svmLinear",
             trControl = train_control)

svm_linear_cv$results %>%
  knitr::kable(caption = "Results from cross-validation with linear kernel")
```

Table 19: Results from cross-validation with linear kernel

C	Accuracy	Kappa	AccuracySD	KappaSD
1	0.9725241	0.9397818	0.0154981	0.0340788

```r
# Note: We don't need to redo the knn prediction part because C = 1 is the default
# value.
# This model is better than our logistic model but the logistic model is very close
# to this model and one major advantage of that model is we can directly see the
# effects of the predictors for that model, which we cannot for this model
# So, if we want a model that just predicts the results given a matrix of
# predictors, then we would use this model but since the effectiveness of the
# logistic model is still high, we will use that model for our report to explain
# the effect of the predictors
```

With the radial kernel, the C represents the cost of constraints violation, which is what we are attempting to maximize the accuracy with respect to. As we can see, the highest accuracy is with C = 1 (which is the default with the svm function which is what we used to create the model initially so we don't need to tune that model). With the linear kernel, we note that the accuracy is higher than the accuracy of the radial kernel. Again, with this model, we cannot directly see the effects of the predictors as we could with the logistic regression classifier.

Random Forest Classifier

The next model which we will use is the random forest classifier. A random forest classifier is a type of ensemble learning algorithm, which is an algorithm where you take multiple machine learning algorithms and put them together to get another machine learning algorithm. In this case, we are taking multiple decision trees and putting them together. For the model, we use the default of 500 as the number of (decision) trees which is a relatively large number. When the number of trees increases beyond a point the performance plateaus, which means the efficiency almost peaks.

```r
set.seed(1998)
# Create an X & Y matrix
training_set_X <- training_set %>%
  select(clump_thickness, marginal_adhesion, bare_nuclei, bland_chromatin,
         normal_nucleoli, unif_of_cell_shape)
training_set_Y <- training_set %>%
  pull(class_logistic) %>%
  as.factor()

# Create the random forest algorithm
# The default ntree is 500 so we don't need to change that
rf_classifier <- randomForest(x = training_set_X, y = training_set_Y)

y_pred <- predict(rf_classifier, newdata = test_set)
cm_rf <- table(y_pred, test_set$class_logistic)
# Accuracy of 0.9708029
knitr::kable(cm_rf,
             caption = "Confusion Matrix for Random Forest Regression")
```

Table 20: Confusion Matrix for Random Forest Regression

	0	1
0	87	2
1	2	46

Table 20 shows the confusion matrix of our random forest regression model. As we see, our model got 132 correct results out of 137 (an accuracy of 0.9708029). However, one parameter which we can tune is the number of variables the function tries at each split. By default, this value is 2, so let's use cross-validation to see if we can increase the efficiency of our model.

```r
set.seed(1998)
# Create the random forest model using cross validation
rf_cv <- train(x = training_set_X, y = training_set_Y,
               # Method is naive bayes
               method = "rf",
               trControl = train_control)
rf_cv$results %>%
  knitr::kable(caption = "Results of cross-validation for the random forest model")
```

Table 21: Results of cross-validation for the random forest model

mtry	Accuracy	Kappa	AccuracySD	KappaSD
2	0.9743434	0.9439497	0.0215224	0.0468592
4	0.9706734	0.9357884	0.0262561	0.0575193
6	0.9707071	0.9359223	0.0214844	0.0468037

Table 21 shows the results of the cross-validation where mtry is the parameter that we are tuning. As we see the highest accuracy is achieved with mtry = 2, which was the default value.

Naive Bayes Classifier

For the naive bayes classifier, we use bayes theorem

$$P(\text{Malignant Cancer} \mid \text{X}) = \frac{P(\text{X} \mid \text{Malignant Cancer}) \cdot P(\text{Malignant Cancer})}{P(\text{X})}$$
$$\text{Posterior Probability} = \frac{\text{Likelihood} \cdot \text{Prior Probability}}{\text{Marginal Likelihood}}$$

where X is our data (i.e. the matrix of predictors).

```r
set.seed(1998)
# Create an X & Y matrix
training_set_X <- training_set %>%
  select(clump_thickness, marginal_adhesion, bare_nuclei, bland_chromatin,
         normal_nucleoli, unif_of_cell_shape)
training_set_Y <- training_set %>%
  pull(class_logistic) %>%
  as.factor()
# Create the model
bayes_classifier <- naiveBayes(x = training_set_X,
                    y = training_set_Y)
y_pred <- predict(bayes_classifier, newdata = test_set)

cm_bayes <- table(y_pred, test_set$class_logistic)
knitr::kable(cm_bayes,
             caption = "Confusion matrix of the naive bayes classifier")
```

Table 22: Confusion matrix of the naive bayes classifier

	0	1
0	85	1
1	4	47

```
# Accuracy of 0.9635036
# Pretty good, let's see what accuracy we can get using cross validation
```

Table 22 is the confusion matrix of naive bayes classifier. Our model gets 132 correct predictions out of 137 (accuracy of 0.9635036). This is pretty good, let's try tuning some parameters to get the best accuracy we can

```
set.seed(1998)
# Make a grid with different values of fl and adjust so we can see which gives the
# highest accuracy
grid <- data.frame(fL=rep(c(1, 1.5, 2, 2.5), 4),
                   usekernel = TRUE,
                   adjust=rep(c(1, 1.5, 2, 2.5), each = 4))
train_control <- trainControl(method = "cv", number = 10)
bayes_cv <- train(x = training_set_X, y = training_set_Y,
                  # Method is naive bayes
                  method = "nb",
                  trControl = train_control,
                  tuneGrid = grid)
bayes_cv$results %>%
  knitr::kable(caption = "Results of cross-validation for the naive bayes model")
```

Table 23: Results of cross-validation for the naive bayes model

fL	usekernel	adjust	Accuracy	Kappa	AccuracySD	KappaSD
1.0	TRUE	1.0	0.9743434	0.9442379	0.0213944	0.0462635
1.0	TRUE	1.5	0.9743434	0.9442379	0.0213944	0.0462635
1.0	TRUE	2.0	0.9761279	0.9481726	0.0244851	0.0529450
1.0	TRUE	2.5	0.9724916	0.9400305	0.0215728	0.0464766
1.5	TRUE	1.0	0.9743434	0.9442379	0.0213944	0.0462635
1.5	TRUE	1.5	0.9743434	0.9442379	0.0213944	0.0462635
1.5	TRUE	2.0	0.9761279	0.9481726	0.0244851	0.0529450
1.5	TRUE	2.5	0.9724916	0.9400305	0.0215728	0.0464766
2.0	TRUE	1.0	0.9743434	0.9442379	0.0213944	0.0462635
2.0	TRUE	1.5	0.9743434	0.9442379	0.0213944	0.0462635
2.0	TRUE	2.0	0.9761279	0.9481726	0.0244851	0.0529450
2.0	TRUE	2.5	0.9724916	0.9400305	0.0215728	0.0464766
2.5	TRUE	1.0	0.9743434	0.9442379	0.0213944	0.0462635
2.5	TRUE	1.5	0.9743434	0.9442379	0.0213944	0.0462635
2.5	TRUE	2.0	0.9761279	0.9481726	0.0244851	0.0529450

fL	usekernel	adjust	Accuracy	Kappa	AccuracySD	KappaSD
2.5	TRUE	2.5	0.9724916	0.9400305	0.0215728	0.0464766

Tuning our parameters, we see that the accuracy peaks when fL and adjust = 2. The accuracy for our model is 0.9761279 along with a standard deviation of the accuracy of 0.0232131.

Summary

```r
tibble(Model = c("Logistic", "KNN", "SVM", "Random Forest", "Naive Bayes"),
       Accuracy = c(0.9688552, 0.9743771, 0.9725241,0.9743434, 0.9761279),
       Kappa = c(0.9318115, 0.9442909, 0.9397818, 0.9439497, 0.9481726),
       AccuracySD = c(0.01737427, 0.02460061, 0.01549812, 0.02152238, 0.02448514),
       KappaSD = c(0.03761736, 0.05323610, 0.03407885, 0.04685919, 0.05294501),
       Notes = c(" ", "K = 4", "Linear Kernel", "Variables at Split = 2",
                 "fL & adjust = 2")) %>%
  knitr::kable(caption = "Summary Statistics of Models Presented")
```

Table 24: Summary Statistics of Models Presented

Model	Accuracy	Kappa	AccuracySD	KappaSD	Notes
Logistic	0.9688552	0.9318115	0.0173743	0.0376174	
KNN	0.9743771	0.9442909	0.0246006	0.0532361	K = 4
SVM	0.9725241	0.9397818	0.0154981	0.0340788	Linear Kernel
Random Forest	0.9743434	0.9439497	0.0215224	0.0468592	Variables at Split = 2
Naive Bayes	0.9761279	0.9481726	0.0244851	0.0529450	fL & adjust = 2

Table 24 summarizes all of our models, in the order they were presented where we pick the model with the highest accuracy from each model type. We see the best model in terms of accuracy is the Naive Bayes model. But it should be noted that all of the accuracies are quite similar. The percent difference between the highest accuracy and lowest accuracy is 0.747842%. However, the standard deviation of the logistic and SVM classifiers are much lower than the others (especially the SVM). This means that their accuracies deviate less when compared to the other models. For comparison, the percent difference between our most accurate model (Naive Bayes) and our model with the lowest standard deviation (SVM) is 44.95388%! This means that when we pick our "best" model, there are many factors to consider.
The most important factor to consider is interpretability. This, of course, is only true when our interpretable models perform well which in this case they do. Only with the logistic model can we see how the predictors affect the odds (or probability) of having malignant cancer. The other models may predict the results better than this model but we can interpret the results and understand the results a lot better with this model (see Figure 1 & Table 10). Therefore, if we just want to predict the results when given information, we might choose the SVM model or the naive Bayes, but if we want to report on our findings, the logistic classifier is definitely the most optimal model.