Final Project Jamie Ki

2022-04-23

Introudction

What is breast cancer?

Breast cancer is a type of cancer that forms in the cells of the breasts. After skin cancer, breast cancer is the most common cancer diagnosed in women in the United States. This cancer causes a huge number of deaths every year. Therefore, in this machine learning project, we will use classification and data mining methods to classify which type of cancer the patient has. This will make it easier for doctors to provide the correct treatment and treat patients on time for their survival.

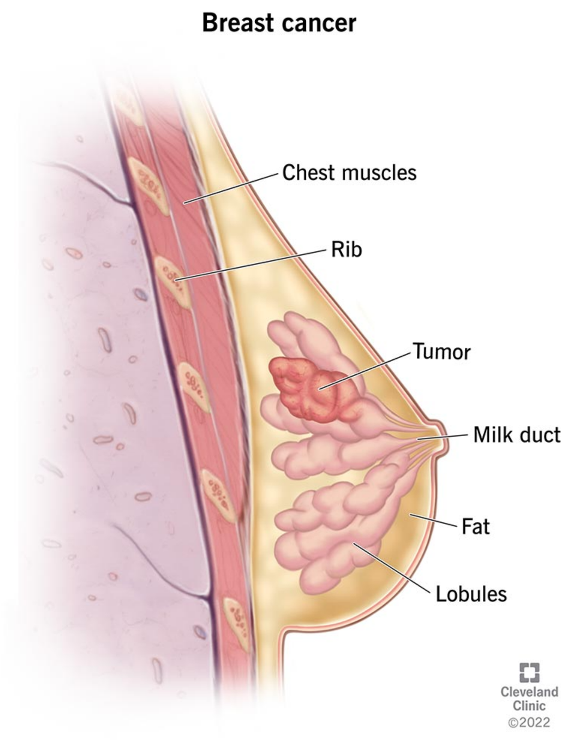


Figure 1: Fig. 1: Breast Cancer

Loading Data and Packages

In this machine learning project, we will analyze and classify the Breast Cancer Data set from University of California, Irvine, to determine which breast cancer belongs to which category.

Basically, there are two types of breast cancer we will be considering in this project : malignant type breast cancer and benign type breast cancer. Malignant indicates the presence of cancer cells whereas benign indicates the absence of cancer cells. In medical analysis, malignant would represent a positive result whereas benign would represent a negative result.

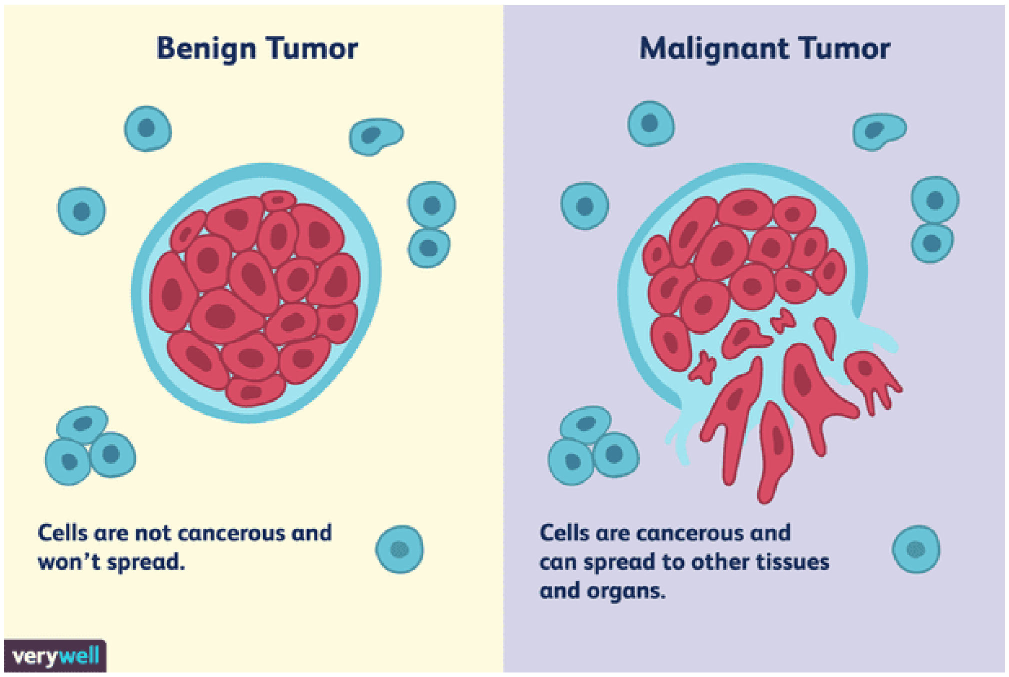

Benign Tumor
Malignant Tumor
Cells are not cancerous and won't spread.
Cells are cancerous and can spread to other tissues and organs.
verywell

Figure 2: Fig. 2: Malignant vs. Benign

So the main goal for our project is to create a model to correctly classify whether the Breast Cancer is a malignant or benign type with the help of a data set.

There are 10 columns of our predictors. We will define these terms for clarification.

-`ID` : Sample code number

-`clump_thickness` : grouping of cancer cells in multilayer

-`uniformity_size`: metastasis to lymph nodes

-`uniformity_shape`: cancerous cells of varying size

-`marginal_adhesion` : a sign of malignancy but the cancerous cells lose this property so this retention of adhesion is an indication of malignancy

-`single_epithelial_cell_sizer` : also called SECS, if the SECS become larger, then it may be a malignant cell

-`bare_nuclei` : without cytoplasm coating, this is found in benign cells

-`bland_chromatin` : usually found in benign cells

-`normal_nucleoli` : generally very small and in benign cells

-`mitoses` : the process in cell division by which the nucleus divides

-`diagnosis`: two types in our machine learning project, malignant cancer cell and benign cancer cell

Now, we will download our data set using tidy. We will name our data set `bcancer`. We can see that the data set includes cytological characteristics of fluid samples from 699 patients.

```
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-

bcancer <- read.csv(url)

str(bcancer)
```

```
## 'data.frame':    698 obs. of  11 variables:
##  $ X1000025: int  1002945 1015425 1016277 1017023 1017122 1018099 1018561 1033078 1033078 1035283 ..
##  $ X5      : int  5 3 6 4 8 1 2 2 4 1 ...
##  $ X1      : int  4 1 8 1 10 1 1 1 2 1 ...
##  $ X1.1    : int  4 1 8 1 10 1 2 1 1 1 ...
##  $ X1.2    : int  5 1 1 3 8 1 1 1 1 1 ...
##  $ X2      : int  7 2 3 2 7 2 2 2 2 1 ...
##  $ X1.3    : chr  "10" "2" "4" "1" ...
##  $ X3      : int  3 3 3 3 9 3 3 1 2 3 ...
##  $ X1.4    : int  2 1 7 1 7 1 1 1 1 1 ...
##  $ X1.5    : int  1 1 1 1 1 1 1 5 1 1 ...
##  $ X2.1    : int  2 2 2 2 4 2 2 2 2 2 ...
```

```
head(bcancer)
```

```
##   X1000025 X5 X1 X1.1 X1.2 X2 X1.3 X3 X1.4 X1.5 X2.1
## 1  1002945  5  4    4    5  7   10  3    2    1    2
## 2  1015425  3  1    1    1  2    2  3    1    1    2
## 3  1016277  6  8    8    1  3    4  3    7    1    2
## 4  1017023  4  1    1    3  2    1  3    1    1    2
## 5  1017122  8 10   10    8  7   10  9    7    1    4
## 6  1018099  1  1    1    1  2   10  3    1    1    2
```

Data Cleaning

Now, we will clean our data. We will first rename our columns into `ID`, `clump_thickness`, `uniformity_size`, `uniformity_shape`, `marginal_adhesion`, `single_epithelial_cell_size`, `bare_nuclei`, `bland_chromatin`, `normal_nucleoli`, `mitoses`, and `diagnosis`. The chart below represents our data with the cleared data and changed predictor names.

```
library(dplyr)
library(tidyverse)


bcancer <- read.csv(file = url, header = FALSE,
                 col.names = c("ID","clump_thickness", "uniformity_size", "uniformity_shape", "marginal_

head(bcancer)
```

```
##        ID clump_thickness uniformity_size uniformity_shape marginal_adhesion
## 1 1000025               5               1                1                 1
## 2 1002945               5               4                4                 5
## 3 1015425               3               1                1                 1
## 4 1016277               6               8                8                 1
## 5 1017023               4               1                1                 3
## 6 1017122               8              10               10                 8
##   single_epithelial_cell_size bare_nuclei bland_chromatin normal_nucleoli
## 1                           2           1               3               1
## 2                           7          10               3               2
## 3                           2           2               3               1
## 4                           3           4               3               7
## 5                           2           1               3               1
## 6                           7          10               9               7
##   mitoses diagnosis
## 1       1         2
## 2       1         2
## 3       1         2
## 4       1         2
## 5       1         2
## 6       1         4
```

Now, we will look for missing values. We can see from our previous chart that we have 16 missing values in bare_nuclei. We will try to exclude this value from our project to avoid errors and problems.

```
sum(bcancer$bare_nuclei == "?")
```

```
## [1] 16
```

We will also exclude the ID number with the 16 missing values in `bare_nuclei`. We are excluding the `ID` number since these are just ID numbers that would not be helpful for our model. We will be naming our new data set `bcancer2`.

```
bcancer2 <- bcancer %>%
  select(-ID, -bare_nuclei)
```

Now, our dependent variable `diagnosis` will be denoted as 2 that represents "benign" and 4 that represents "malignant". We will convert these values into a binary variable, 0 and 1 respectively, for our convenience for our model.

```
bcancer2 <- bcancer2 %>%
  mutate(diagnosis = as.factor(ifelse(diagnosis == 2, 0, 1)))
summary(bcancer2)
```

```
##  clump_thickness  uniformity_size  uniformity_shape marginal_adhesion
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 4.000   Median : 1.000   Median : 1.000   Median : 1.000
##  Mean   : 4.418   Mean   : 3.134   Mean   : 3.207   Mean   : 2.807
##  3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##  single_epithelial_cell_size bland_chromatin  normal_nucleoli     mitoses
##  Min.   : 1.000              Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 2.000              1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 2.000              Median : 3.000   Median : 1.000   Median : 1.000
##  Mean   : 3.216              Mean   : 3.438   Mean   : 2.867   Mean   : 1.589
##  3rd Qu.: 4.000              3rd Qu.: 5.000   3rd Qu.: 4.000   3rd Qu.: 1.000
##  Max.   :10.000              Max.   :10.000   Max.   :10.000   Max.   :10.000
##  diagnosis
##  0:458
##  1:241
##
##
##
##
```

Plots

Now that we have completed our data cleaning procedure, we will look more in depth into our data. We will first start by visualizing different types of plots to explore the relationships among variables. First, we will find the frequency of cancer diagnosis using a pie chart.

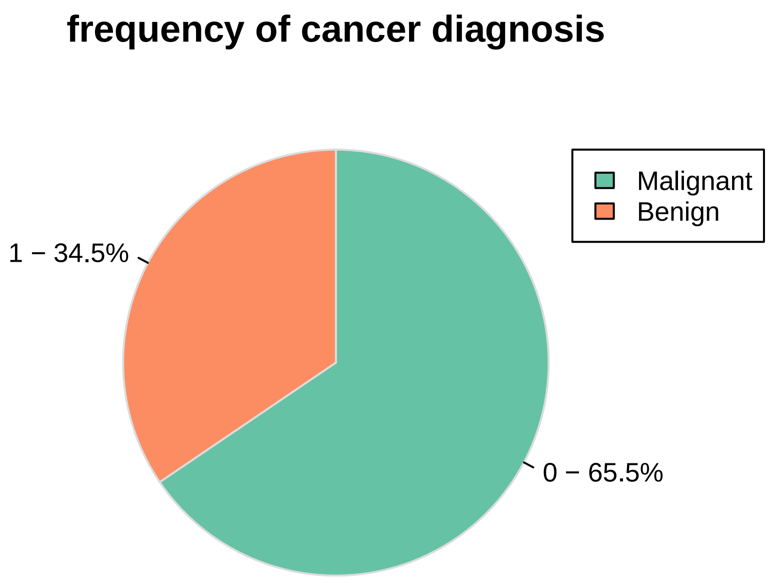
```
# Frequency table for cancer diagnosis
diagnosis.table <- table(bcancer2$diagnosis)
diagnosis.table
```

```
##
##   0   1
## 458 241
```

```
#pie chart for cancer diagnosis
library(RColorBrewer)
color <- brewer.pal(2, "Set2")
diagnosis.prop.table <- prop.table(diagnosis.table)*100
diagnosis.prop.df <- as.data.frame(diagnosis.prop.table)
pielabels <- sprintf("%s - %3.1f%s", diagnosis.prop.df[,1], diagnosis.prop.table, "%")

class(diagnosis.prop.table)
```

```
## [1] "table"
```

```
pie(diagnosis.prop.table,
  labels= pielabels,
  clockwise=TRUE,
  col=color,
  border="gainsboro",
  radius=0.9,
  cex=0.8,
  main="frequency of cancer diagnosis")
legend(1, 0.9,  c("Malignant", "Benign"), cex = 0.8, fill = color)
```



```
color
```

```
## [1] "#66C2A5" "#FC8D62" "#8DA0CB"
```

Recall that malignant indicates the presence of cancer cells and benign indicates the absence of cancer cells. We can see that there are 357 observations in malignant, the presence of cancer cells, which is 62.7% and 212 observations for benign, the absence of cancer cells, which is 37.3%. In other words, malignant would represent a positive result and benign would represent a negative result. This is an interesting frequency because it is more typical to have more negative results than positive results in a medical analysis. However, in our case, since we have more malignant, which is a positive result, our data is different than other typical medical analysis.

Data Split

Before looking more into our data, especially our predictors, we will split our data. We will split our data set into training sample and testing sample. We will name them each `bcancer_train` and `bcancer_test`. Here, we will use a 50/50 split. Also, we will fold our training data using cross-validation.

```
sample_size = floor(0.5 * nrow(bcancer2))

set.seed(1729)
bcancer_new = sample(seq_len(nrow(bcancer2)), size = sample_size)

bcancer_train = bcancer2[bcancer_new, ]
```

```r
bcancer_test = bcancer2[-bcancer_new, ]

head(bcancer_train)
```

```
##     clump_thickness uniformity_size uniformity_shape marginal_adhesion
## 467              10               6                6                 2
## 682               5              10               10                10
## 621               3               1                1                 1
## 125               5               4                6                 7
## 299               8               2                1                 1
## 499               4               1                1                 1
##     single_epithelial_cell_size bland_chromatin normal_nucleoli mitoses
## 467                           4               9               7       1
## 682                           4               5               6       3
## 621                           2               2               1       1
## 125                           9               8              10       1
## 299                           5               1               1       1
## 499                           2               2               1       1
##     diagnosis
## 467         1
## 682         1
## 621         0
## 125         1
## 299         0
## 499         0
```

```r
head(bcancer_test)
```

```
##   clump_thickness uniformity_size uniformity_shape marginal_adhesion
## 1               5               1                1                 1
## 2               5               4                4                 5
## 3               3               1                1                 1
## 4               6               8                8                 1
## 5               4               1                1                 3
## 6               8              10               10                 8
##   single_epithelial_cell_size bland_chromatin normal_nucleoli mitoses diagnosis
## 1                           2               3               1       1         0
## 2                           7               3               2       1         0
## 3                           2               3               1       1         0
## 4                           3               3               7       1         0
## 5                           2               3               1       1         0
## 6                           7               9               7       1         1
```

```r
bcancer_fold <- vfold_cv(bcancer_train, v = 3)
bcancer_fold
```

```
## #  3-fold cross-validation
## # A tibble: 3 x 2
##   splits            id
##   <list>            <chr>
## 1 <split [232/117]> Fold1
## 2 <split [233/116]> Fold2
## 3 <split [233/116]> Fold3
```

```
prop.table(table(bcancer_train$diagnosis))
```

```
##
##         0         1
## 0.6618911 0.3381089
```

```
prop.table(table(bcancer_test$diagnosis))
```

```
##
##         0         1
## 0.6485714 0.3514286
```

Recipe

Now we will create a recipe for our data set. We are including steps to fit our models at the end of our project. We will name the recipe of our project `bcancer_recipe`'.

```
bcancer_recipe <- recipe(diagnosis ~ clump_thickness + uniformity_size + uniformity_shape + marginal_adh
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_predictors())
```

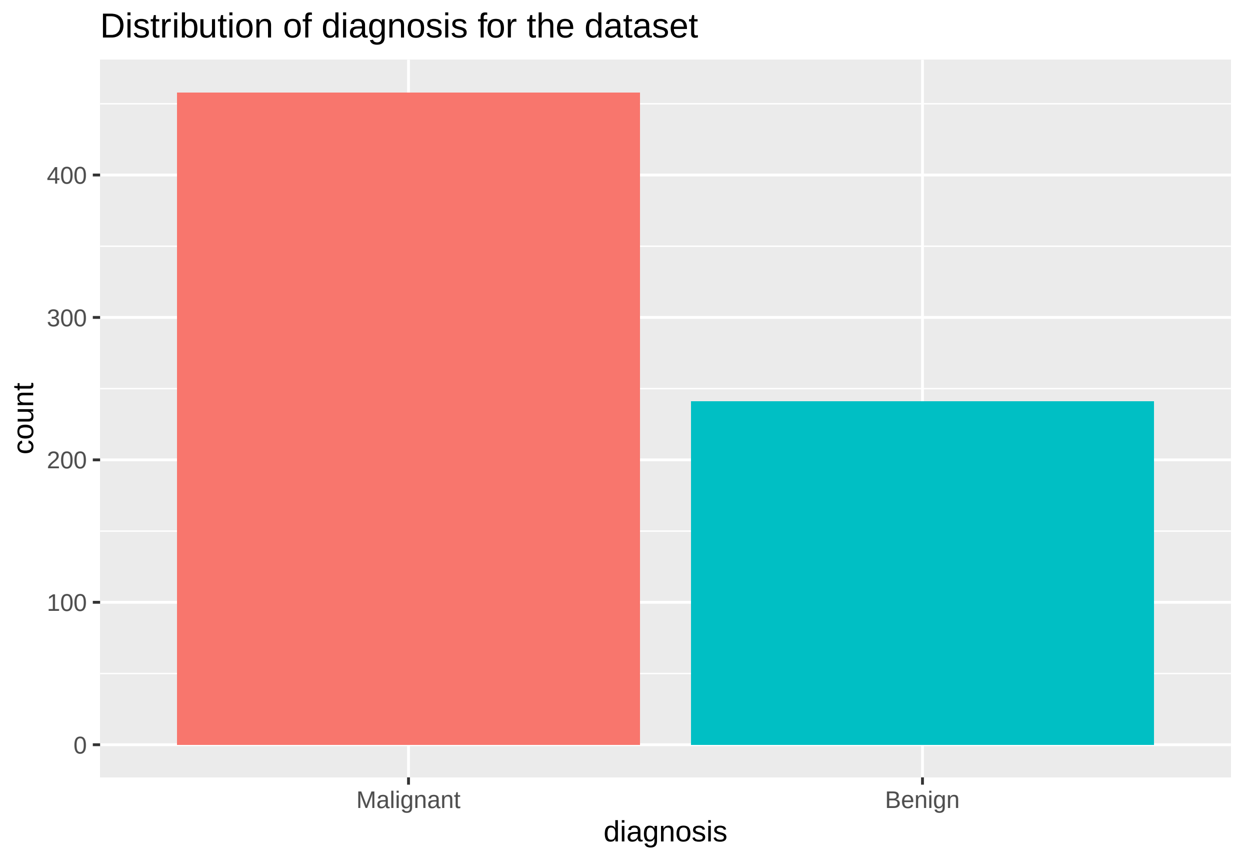
Exploratory Data Analysis

Now that we split our data and created a recipe, we will explore our entire data set, training data set, and testing data set.

We explored our entire data set using a pie chart, but for an easier comparison with our training and testing data set, we will look into the distribution of diagnosis of our entire data set using a bar chart this time.

We will first explore our entire data set with a bar chart.

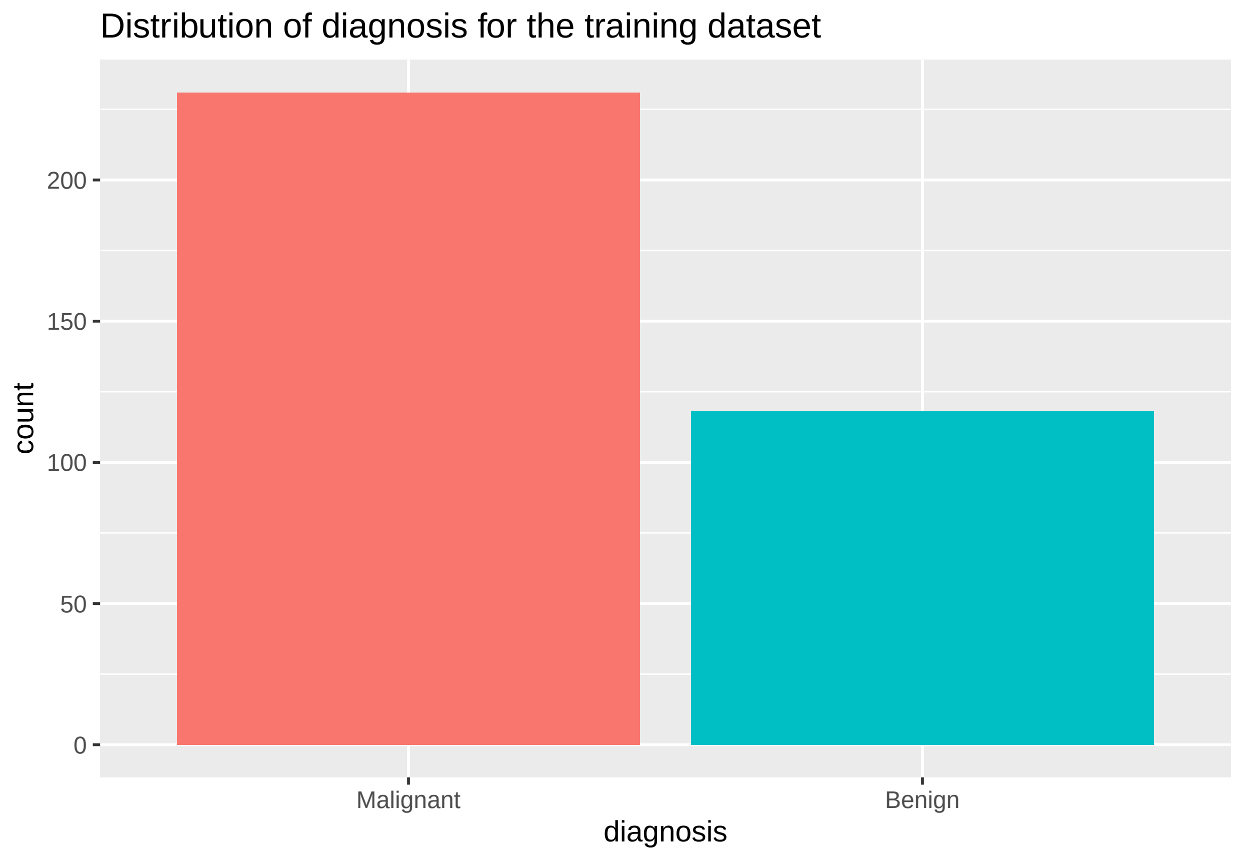
```
ggplot(bcancer2, aes(x=diagnosis, fill=diagnosis )) +
  geom_bar( ) +
  ggtitle("Distribution of diagnosis for the dataset") +
  scale_x_discrete(labels=c("Malignant", "Benign")) +
  theme(legend.position="none")
```



This is a similar result to the pie chart we used above. We have more malignant than benign.

Now, we will use a bar chart to explore our training data set. We will look into the distribution of diagnosis.

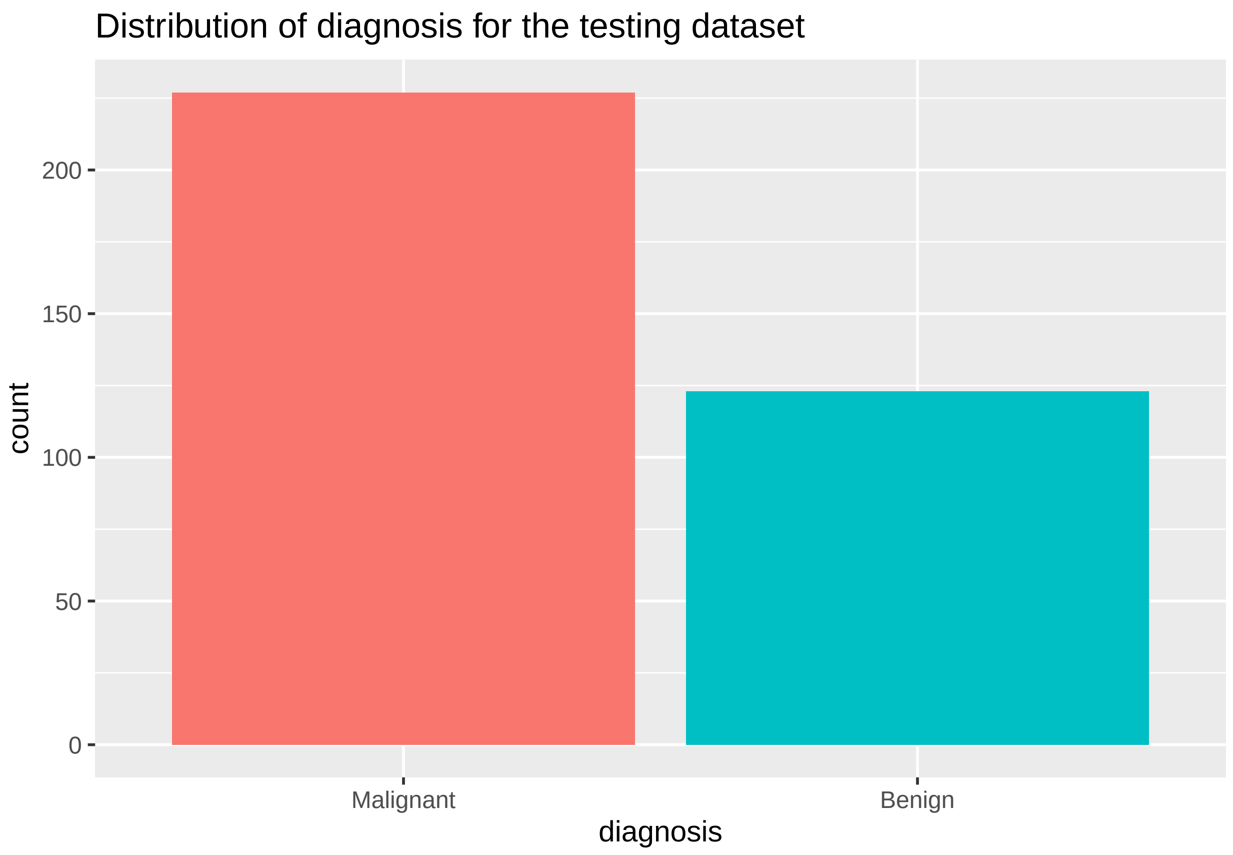
```r
ggplot(bcancer_train, aes(x=diagnosis, fill=diagnosis )) +
  geom_bar( ) +
  ggtitle("Distribution of diagnosis for the training dataset") +
  scale_x_discrete(labels=c("Malignant", "Benign")) +
  theme(legend.position="none")
```



We can see that there is more malignant than benign as well, and malignant is approximately 230 observations and benign is approximately 125 observations.

Lastly, we will look into the distribution of diagnosis of our testing data set.

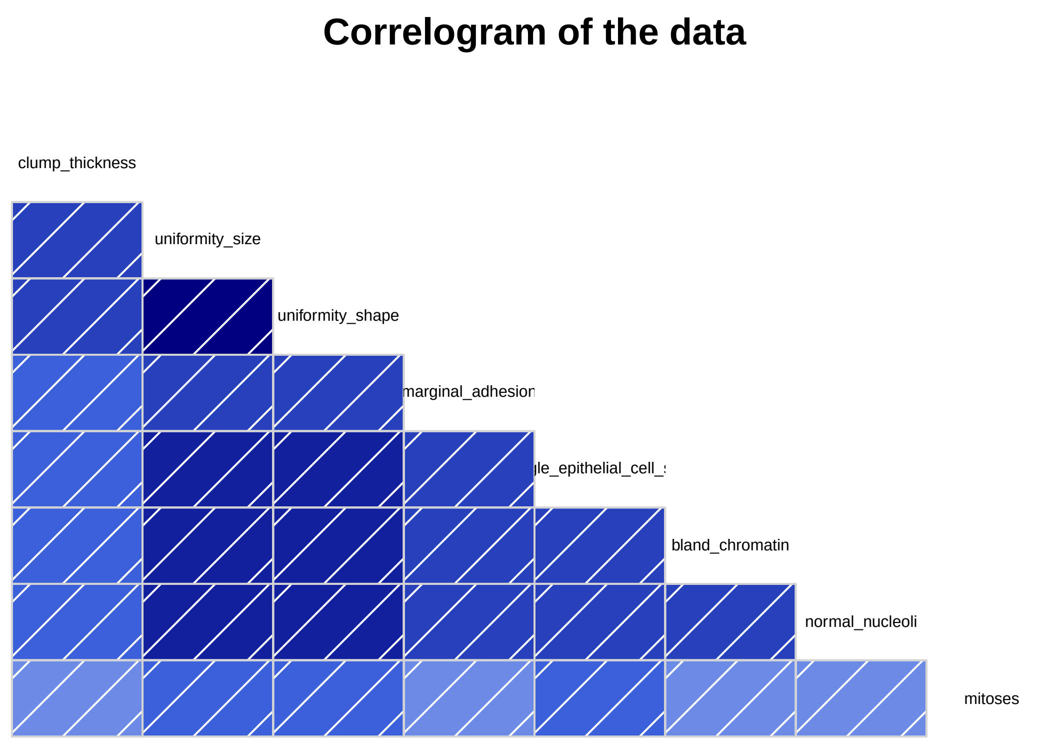
```
ggplot(bcancer_test, aes(x=diagnosis, fill=diagnosis )) +
  geom_bar( ) +
  ggtitle("Distribution of diagnosis for the testing dataset") +
  scale_x_discrete(labels=c("Malignant", "Benign")) +
  theme(legend.position="none")
```



We can see that the distribution of our testing data set is similar to our training data set with approximately 230 malignant and 125 benign observations.

Now, we will look more into our predictors. In order to see the relationship between predictors, we will use a Correlogram. By using a Correlogram, we will be able to visualize the interaction between each predictor.

```
# Corrgram of the entire data set
corrgram(bcancer2, order=NULL, lower.panel=panel.shade, upper.panel=NULL, text.panel=panel.txt, main="C
```



Note that a darker color represents a strong relationship between predictors. Since we mostly see blue colors, most of the predictors do have an interaction with each other. We can see that `uniformity_size` and `uniformity_shape` have the strongest relationship with each predictor. On the other hand, `mitoses` have the least interaction with other predictors. Other predictors such as `clump_thickness`, `marginal adhesion`, `size_epithelial_cell`, `bland_chromatin`, and `normal_nucleoli` have a decently strong relationship with other predictors. Since `uniformity_size` and `uniformity_shape` have the strongest relationship with other predictors, we can conclude that they would be the significant predictors in our project.

Fit model

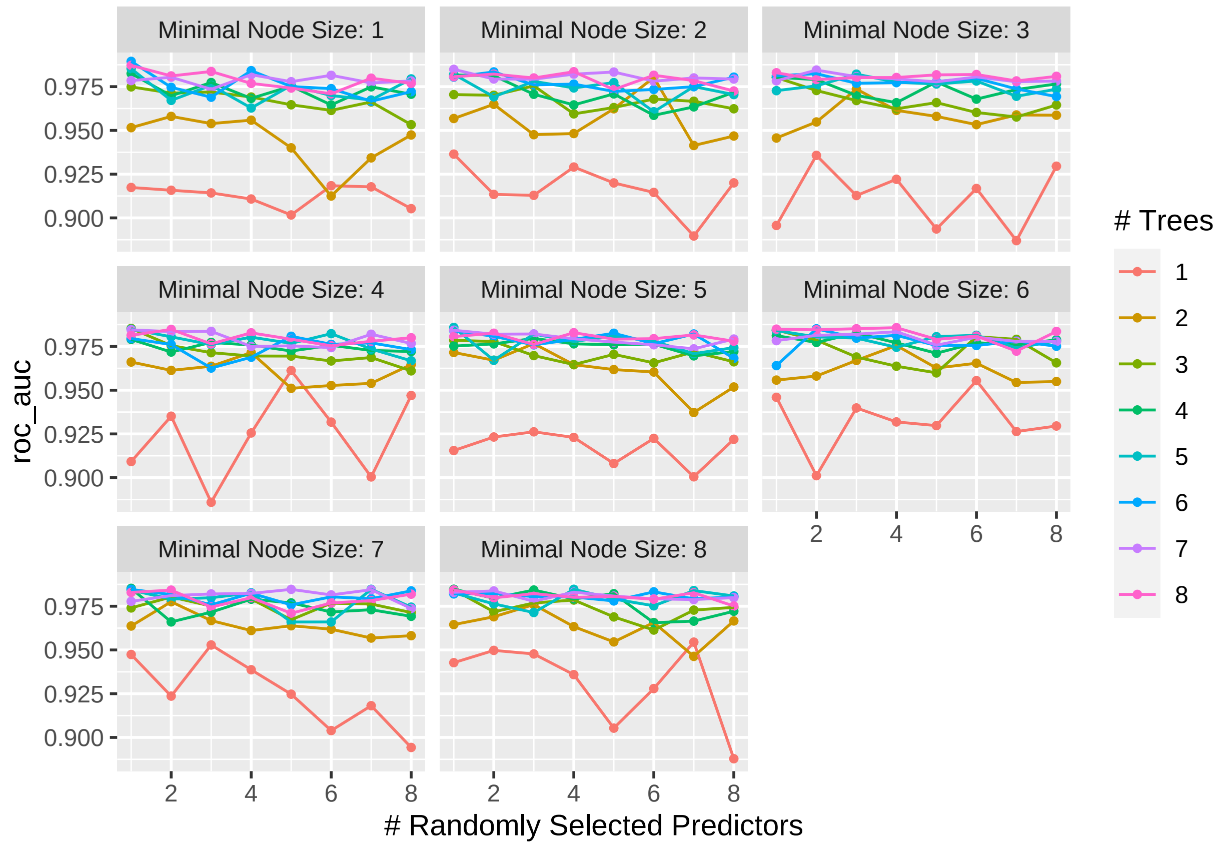
Random Forest

Now, we will fit our model into different model classes. By doing this, we will be able to study our data set more, by learning more about the relationship between the predictors and about the data set itself. In order to do this, we will be using four machine learning methods : Random Forest, Boosted Trees, Logistic Regression, and Quadratic Discriminant Analysis (QDA).

We will first start with the Random Forest model. We will start by setting the mode and engine, create a grid, create our workflow, and tune our model. Afterwards, we will find the `best_complexity` to find our best fit model.

```r
class_forest_spec <- rand_forest() %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("classification")
```

```
param_grid2 <- grid_regular(mtry(range = c(1, 8)), trees(range = c(1,8)), min_n(range = c(1,8)),  level

forest_workflow <- workflow() %>%
  add_model(class_forest_spec %>% set_args(mtry = tune(), trees = tune(), min_n = tune())) %>%
  add_recipe(bcancer_recipe)

tune_res_forest <- tune_grid(
  forest_workflow,
  resamples = bcancer_fold,
  grid = param_grid2,
  metrics = metric_set(roc_auc)
)

autoplot(tune_res_forest)
```



```
collect_metrics(tune_res_forest) %>%
  arrange(desc(mean))
```

```
## # A tibble: 512 x 9
##     mtry trees min_n .metric .estimator  mean     n std_err .config
##    <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1     1     6     1 roc_auc binary     0.989     3 0.00611 Preprocessor1_Model~
##  2     1     8     1 roc_auc binary     0.987     3 0.00545 Preprocessor1_Model~
##  3     1     5     5 roc_auc binary     0.986     3 0.00915 Preprocessor1_Model~
##  4     4     8     6 roc_auc binary     0.986     3 0.00921 Preprocessor1_Model~
##  5     1     3     4 roc_auc binary     0.985     3 0.00661 Preprocessor1_Model~
##  6     1     5     1 roc_auc binary     0.985     3 0.00738 Preprocessor1_Model~
##  7     1     4     7 roc_auc binary     0.985     3 0.00847 Preprocessor1_Model~
##  8     3     8     6 roc_auc binary     0.985     3 0.00798 Preprocessor1_Model~
##  9     2     6     6 roc_auc binary     0.985     3 0.00925 Preprocessor1_Model~
## 10     1     7     2 roc_auc binary     0.985     3 0.00792 Preprocessor1_Model~
## # ... with 502 more rows
```

```r
best_complexity <- select_best(tune_res_forest)
best_complexity
```

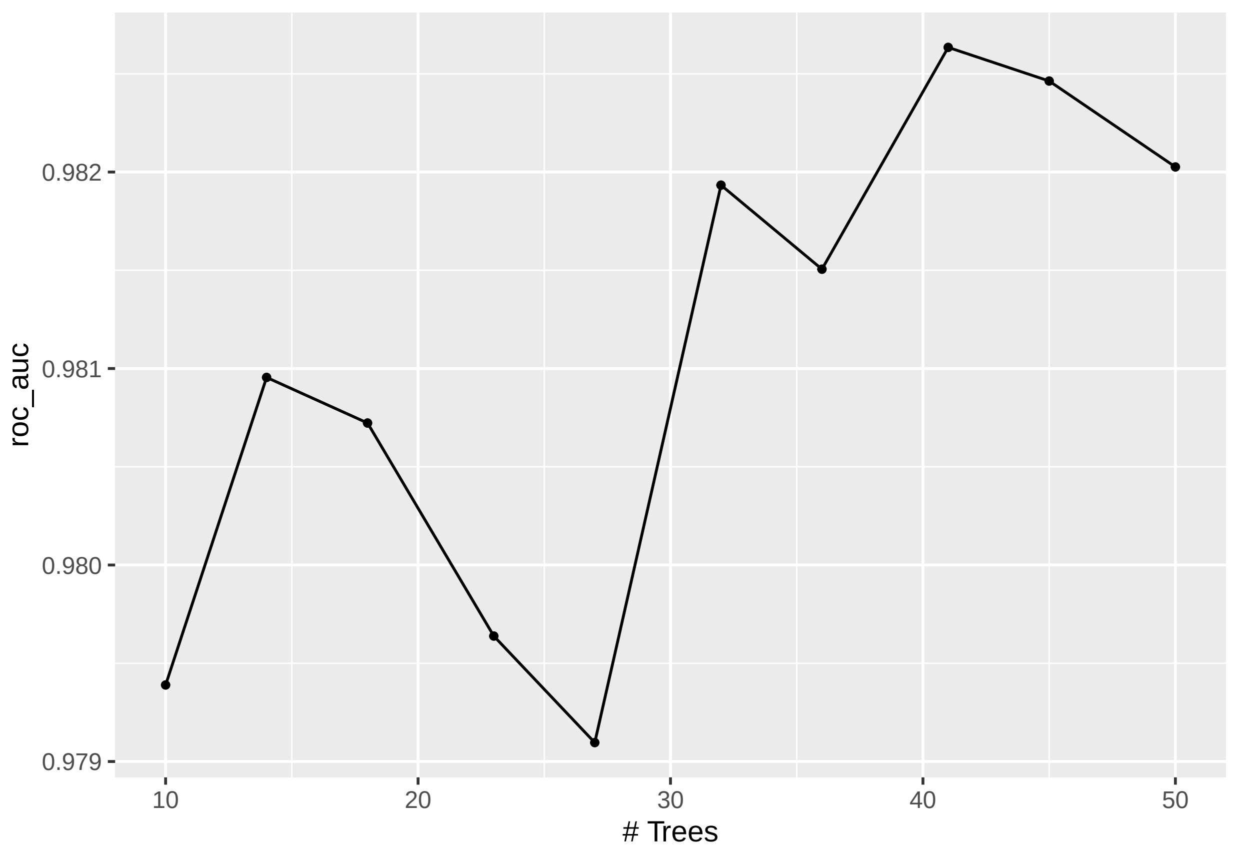
```
## # A tibble: 1 x 4
##    mtry trees min_n .config
##   <int> <int> <int> <chr>
## 1     1     6     1 Preprocessor1_Model041
```

Note that approximately 85~90 percent is a good `roc_auc` value. In our random forest model, we can see that for each Minimal Node Size, the `roc_auc` values are all above 85 percent and the best `roc_auc` value is 0.9894460. Therefore, this is a very good model overall.

Boosted Trees

For our second model, we will use boosted trees for our data. We will set the engine, set the mode, create a workflow, tune our model, and use `best_complexity3` to find the best fit model for our data.

```r
boost_spec <- boost_tree(trees = 100, tree_depth = 4) %>%
  set_engine("xgboost") %>%
  set_mode("classification")


param_grid_boost <- grid_regular(trees(range = c(10, 50)),  levels = 10)

boost_workflow <- workflow() %>%
  add_model(boost_spec %>% set_args(trees = tune())) %>%
  add_recipe(bcancer_recipe)

tune_res_boost <- tune_grid(
  boost_workflow,
  resamples = bcancer_fold,
  grid = param_grid_boost,
  metrics = metric_set(roc_auc)
)

autoplot(tune_res_boost)
```



```
collect_metrics(tune_res_boost) %>%
  arrange(desc(mean))
```

```
## # A tibble: 10 x 7
##    trees .metric .estimator  mean     n std_err .config
##    <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1    41 roc_auc binary     0.983     3 0.00593 Preprocessor1_Model08
##  2    45 roc_auc binary     0.982     3 0.00604 Preprocessor1_Model09
##  3    50 roc_auc binary     0.982     3 0.00663 Preprocessor1_Model10
##  4    32 roc_auc binary     0.982     3 0.00589 Preprocessor1_Model06
##  5    36 roc_auc binary     0.982     3 0.00565 Preprocessor1_Model07
##  6    14 roc_auc binary     0.981     3 0.00882 Preprocessor1_Model02
##  7    18 roc_auc binary     0.981     3 0.00776 Preprocessor1_Model03
##  8    23 roc_auc binary     0.980     3 0.00821 Preprocessor1_Model04
##  9    10 roc_auc binary     0.979     3 0.00978 Preprocessor1_Model01
## 10    27 roc_auc binary     0.979     3 0.00813 Preprocessor1_Model05
```

```
best_complexity3 <- select_best(tune_res_boost)
best_complexity3
```

```
## # A tibble: 1 x 2
##   trees .config
##   <int> <chr>
## 1    41 Preprocessor1_Model08
```

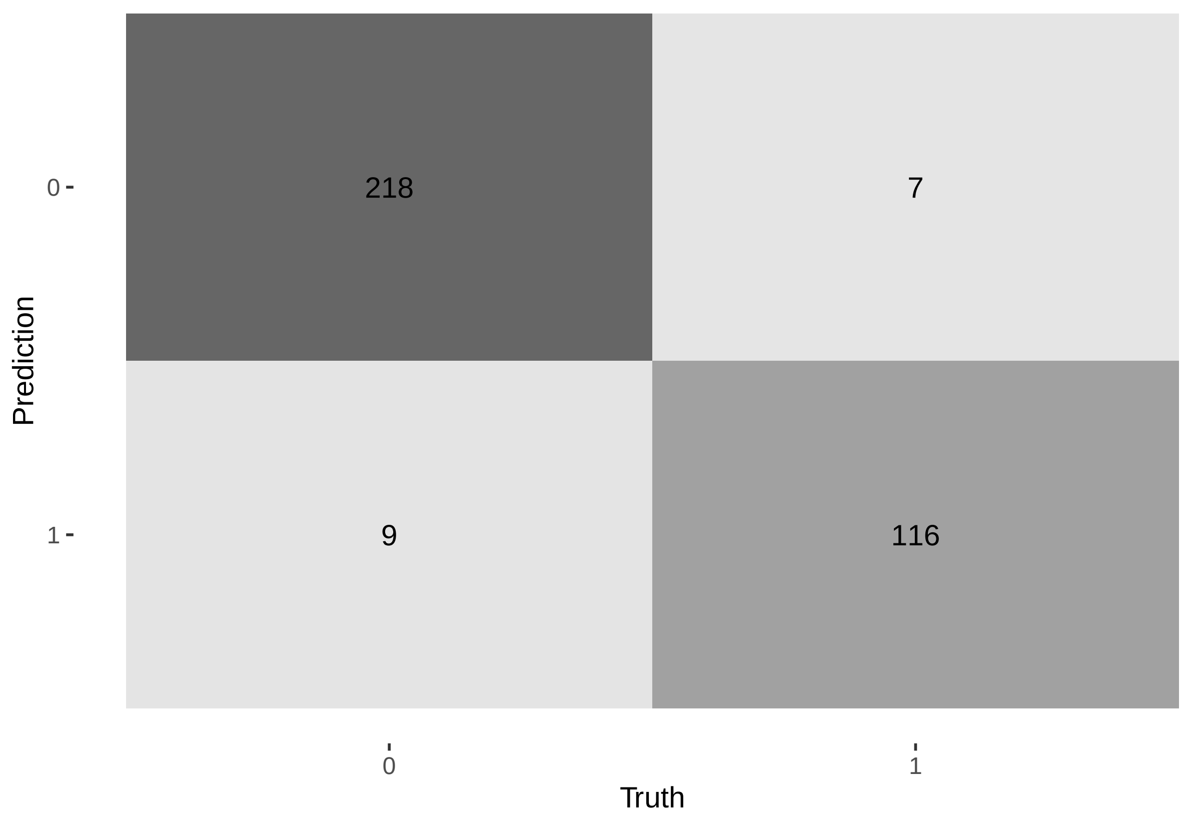
We can see that the boosted trees model is also very good. Since most of the `roc_auc` values in our boosted trees is above 0.979, this means that this is also a very good model for our data set. Additionally, the best `roc_auc` value is 0.9826345 which is extremely good.

Now since our two models are very good, we will finalize both of our models by creating a final fit and a confusion matrix for further interpretation. We will first start with our random forest model.

```
class_tree_final <- finalize_workflow(forest_workflow, best_complexity)

class_tree_final
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 2 Recipe Steps
##
## * step_dummy()
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = 1
##   trees = 6
##   min_n = 1
##
## Engine-Specific Arguments:
##   importance = impurity
##
## Computational engine: ranger
```

```
class_tree_final_fit <- fit(class_tree_final, data = bcancer_train)


#confusion matrix heat map
augment(class_tree_final_fit, new_data = bcancer_test) %>%
  conf_mat(truth = diagnosis, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



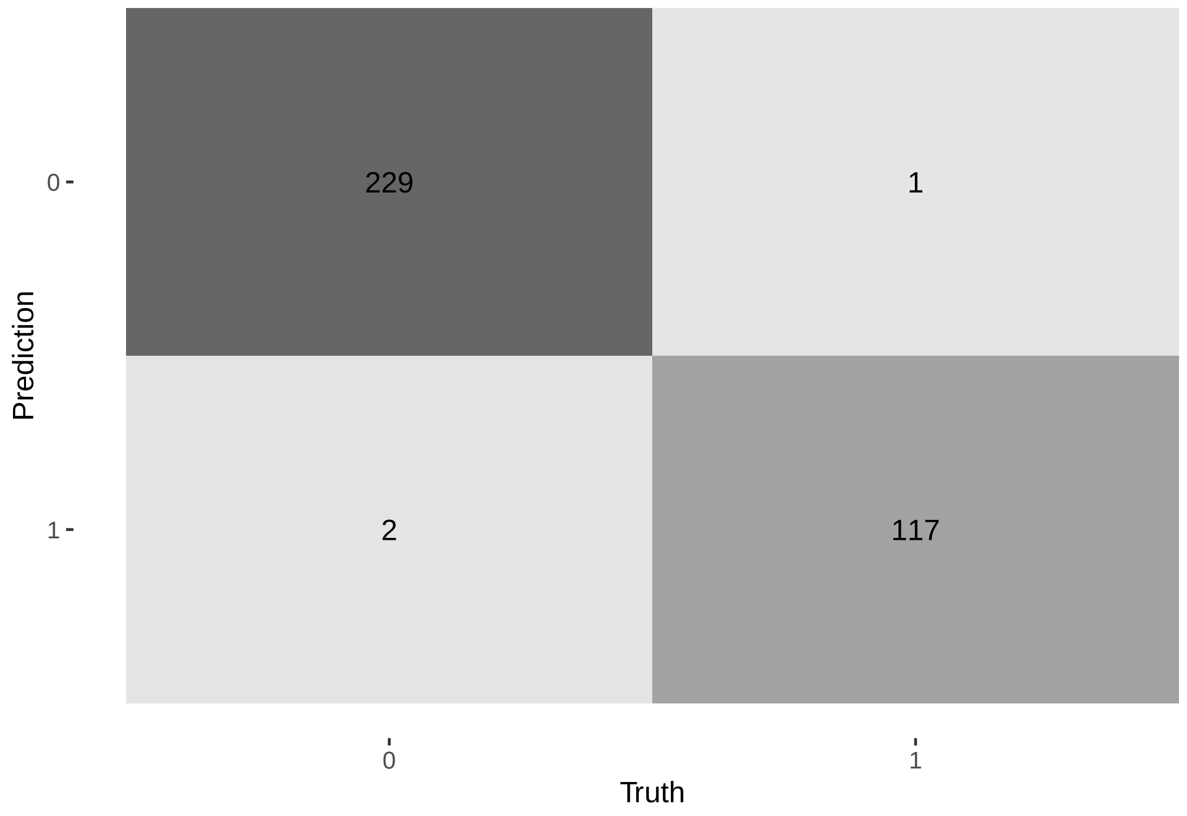
We will try to interpret the confusion matrix here. Recall that 0 represents benign and 1 represents malignant. Thus, 218 implies the number of incidents where benign patients are determined correctly as benign patients. On the other hand, 7 would represent the number of malignant patients considered to be benign patients and 9 would represent the number of benign patients as malignant patients. Lastly, 116 would represent the number of patients where they are correctly diagnosed as malignant patients as malignant patients. Recall that the case where malignant, which is positive, is diagnosed as benign which is negative, so 1 and 0 respectively, which is 7 in our case, is called a false negative, type 2 error. On the other hand, where benign patients are diagnosed as malignant patients are called a false positive, a type 1 error.

Now, we will do the same thing for our boosted trees model.

```
class_tree_final2 <- finalize_workflow(boost_workflow, best_complexity3)

class_tree_final2
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: boost_tree()
##
## -- Preprocessor ----------------------------------------------------------------
## 2 Recipe Steps
##
## * step_dummy()
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Boosted Tree Model Specification (classification)
##
## Main Arguments:
##   trees = 41
##   tree_depth = 4
##
## Computational engine: xgboost
```

```
class_tree_final_fit2 <- fit(class_tree_final2, data = bcancer_train)


#confusion matrix heat map
augment(class_tree_final_fit2, new_data = bcancer_train) %>%
  conf_mat(truth = diagnosis, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



We can see that similar to our random forest model, our boosted trees model is also a very good model for our data set. We have less type 1 and type 2 errors. Our type 1 error here has 2 cases and our type 2 error here is has only 1 case. Also, 229 benign patients were correctly observed as benign patients, and 117 malignant patients were correctly observed as malignant patients.

Logistic Regression

Now, we will use a logistic regression for our third model. This is the following code for our logistic regression :

```r
rec_poly <- recipe(diagnosis ~ ., data = bcancer_train) %>%
  step_poly( degree = 4)

lr_spec <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

lr_poly_wf <- workflow() %>%
  add_model(lr_spec) %>%
  add_recipe(rec_poly)

lr_poly_fit <- fit(lr_poly_wf, data = bcancer_train)

predict(lr_poly_fit, new_data = bcancer_train)
```

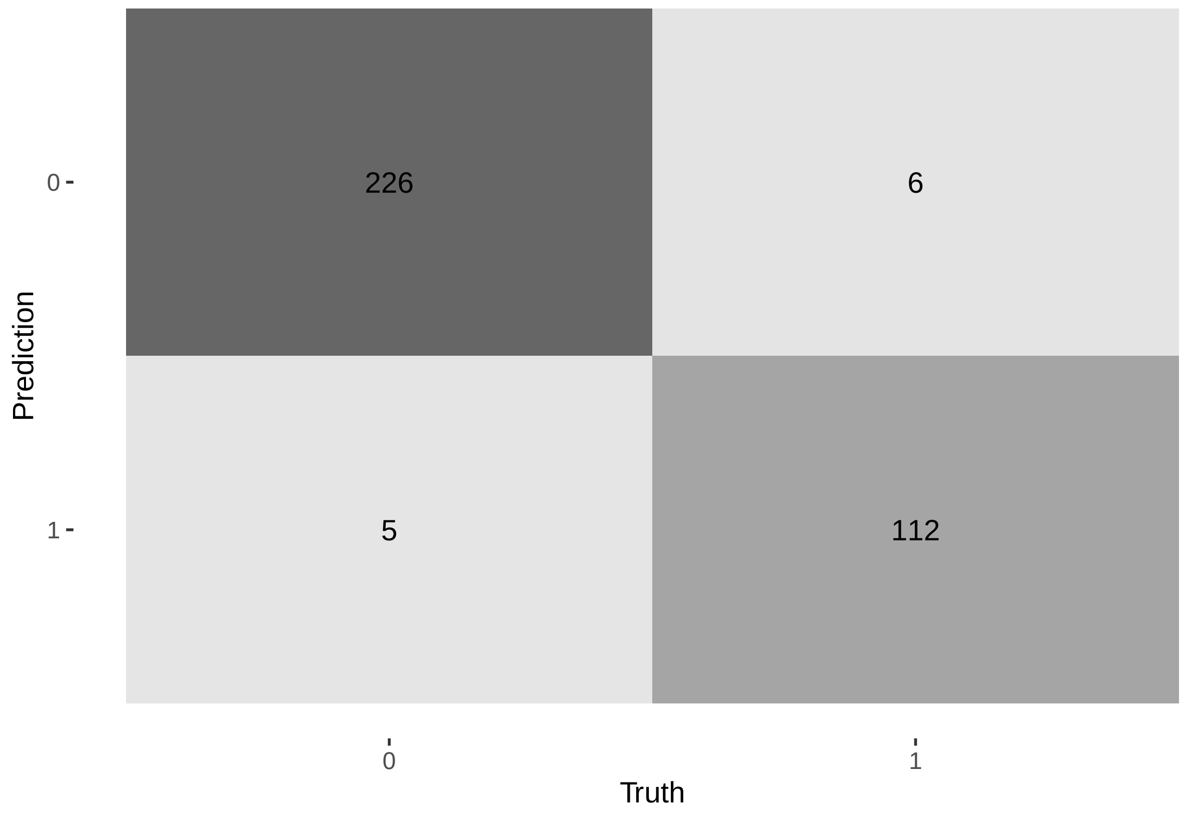
```
## # A tibble: 349 x 1
##    .pred_class
##    <fct>
##  1 1
##  2 1
##  3 0
##  4 1
##  5 0
##  6 0
##  7 0
##  8 0
##  9 1
## 10 0
## # ... with 339 more rows
```

Now, we will create a confusion matrix to see our type 1 and type 2 errors.

```r
augment(lr_poly_fit, new_data = bcancer_train) %>%
  conf_mat(truth = diagnosis, estimate = .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0 226   6
##          1   5 112
```

```r
augment(lr_poly_fit, new_data = bcancer_train) %>%
  conf_mat(truth = diagnosis, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



Here, we can see that we have a very low type 1 and type 2 error. Therefore, we can assume here that this model is also pretty good.

In order to make sure that our model is a good fit, we will look at the accuracies and roc_auc values for this.

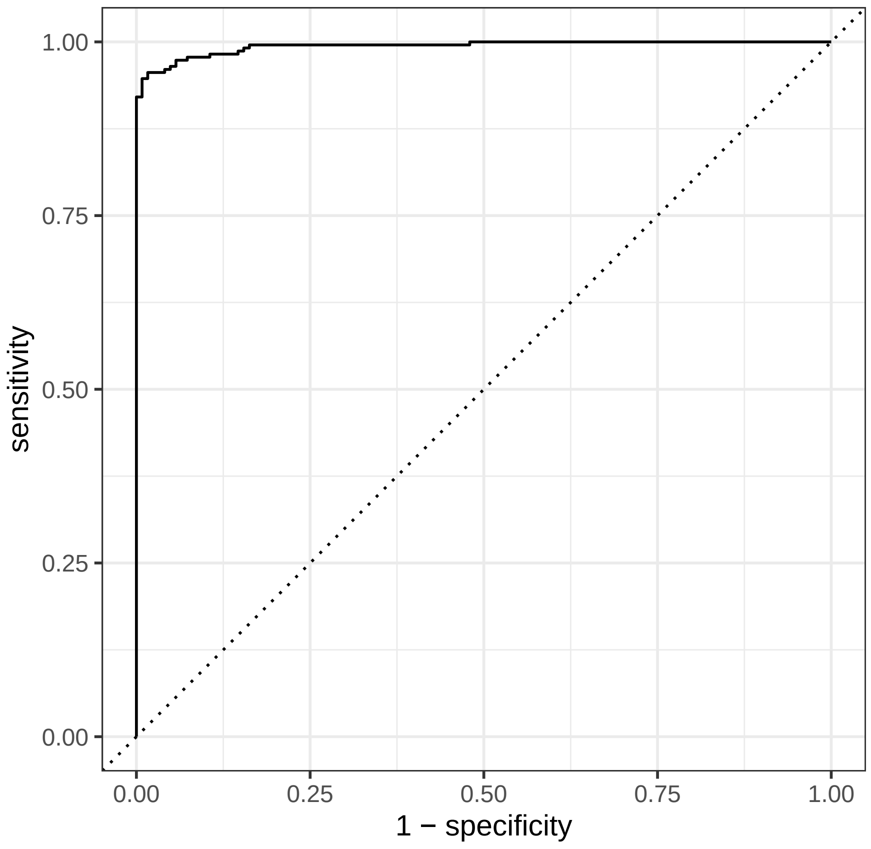
```
log_acc <- predict(lr_poly_fit, new_data = bcancer_train, type = "class") %>%
  bind_cols(bcancer_train %>% select(diagnosis)) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
log_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.968
```

```
log_test <- fit(lr_poly_wf, bcancer_test)
predict(log_test, new_data = bcancer_test, type = "class") %>%
  bind_cols(bcancer_test %>% select(diagnosis)) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary          0.96
```

```r
augment(log_test, new_data = bcancer_test) %>%
  roc_curve(diagnosis, .pred_0) %>%
  autoplot()
```



```r
augment(log_test, new_data = bcancer_test) %>%
  roc_auc(diagnosis, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.994
```

We can clearly see that this is also a very good model for our data set. By comparing the specificity and sensitivity, it is very obvious that our model is very well behaving. Our accuracy estimate is 0.9684814, which is very high. Additionally, our roc_auc value is 0.9938039 which is also extremely high. Therefore, the logistic regression model is a good fir for our data set.

QDA

Now, we will look into our last model, qda model. We wil first set the mode, engine, workflow, and recipe.

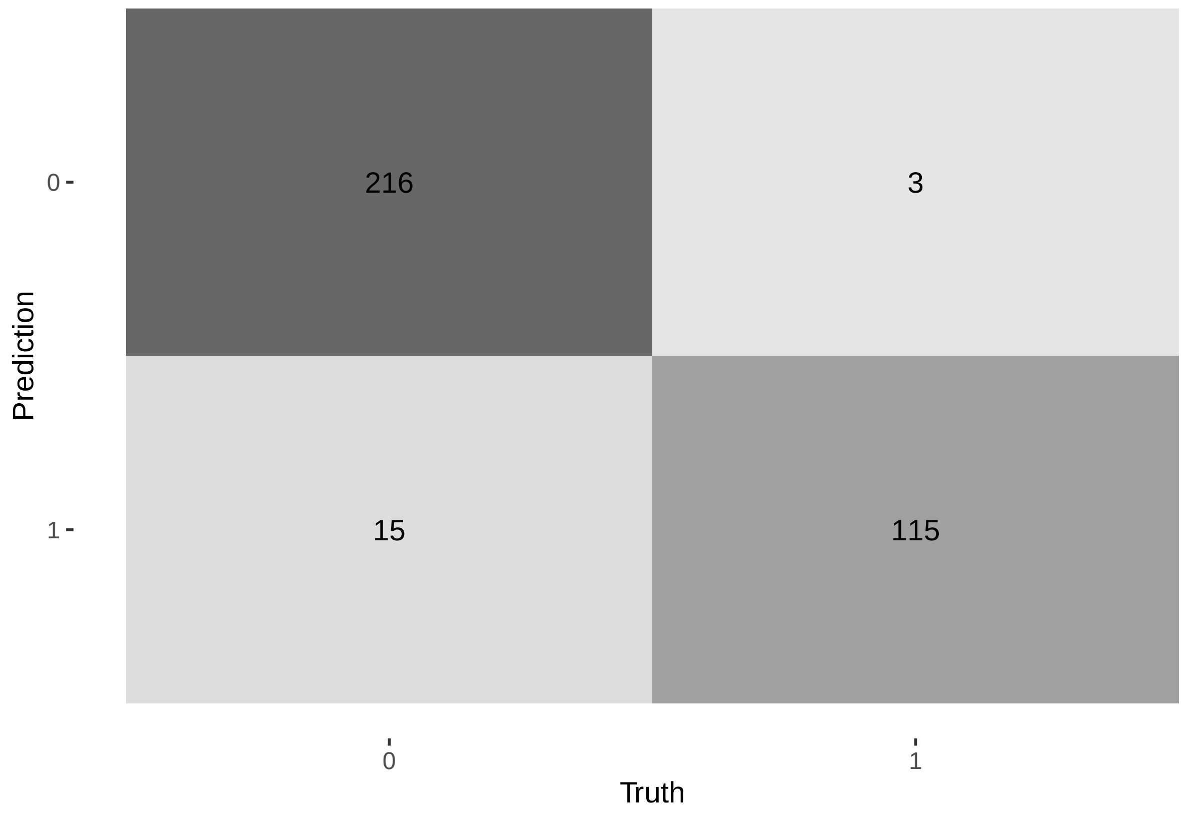
```r
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

qda_wkflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(bcancer_recipe)

qda_fit <- fit(qda_wkflow, bcancer_train)

predict(qda_fit, new_data = bcancer_train, type = "prob")
```

```
## # A tibble: 349 x 2
##      .pred_0    .pred_1
##        <dbl>      <dbl>
##  1 6.73e-12 1.00
##  2 1.17e-29 1
##  3 1.00e+ 0 0.00000176
##  4 1.17e-18 1
##  5 3.69e- 1 0.631
##  6 1.00e+ 0 0.00000460
##  7 1.00e+ 0 0.00000191
##  8 1.00e+ 0 0.00000116
##  9 1.55e- 4 1.00
## 10 9.70e- 1 0.0297
## # ... with 339 more rows
```

```r
augment(qda_fit, new_data = bcancer_train) %>%
  conf_mat(truth = diagnosis, estimate = .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0 216   3
##          1  15 115
```

```r
augment(qda_fit, new_data = bcancer_train) %>%
  conf_mat(truth = diagnosis, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



The confusion matrix here tells us that we have 3 observations of type 2 error and 15 observations of type 1 error. This is also a pretty low error for each case.

In order to see how good our model is fitting, we will find the accuracies and roc_auc value for our qda model.

```r
qda_acc <- augment(qda_fit, new_data = bcancer_train) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
qda_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.948
```

```r
predict(qda_fit, new_data = bcancer_test, type = "prob")
```

```
## # A tibble: 350 x 2
##       .pred_0     .pred_1
##         <dbl>       <dbl>
##  1 1.00e+ 0 0.0000327
##  2 1.02e- 3 0.999
##  3 1.00e+ 0 0.00000389
##  4 1.45e-19 1
##  5 1.00e+ 0 0.0000833
##  6 9.86e-26 1
##  7 1.00e+ 0 0.00000629
##  8 1.00e+ 0 0.00000156
##  9 1.00e+ 0 0.000000845
## 10 1.00e+ 0 0.00000116
## # ... with 340 more rows
```

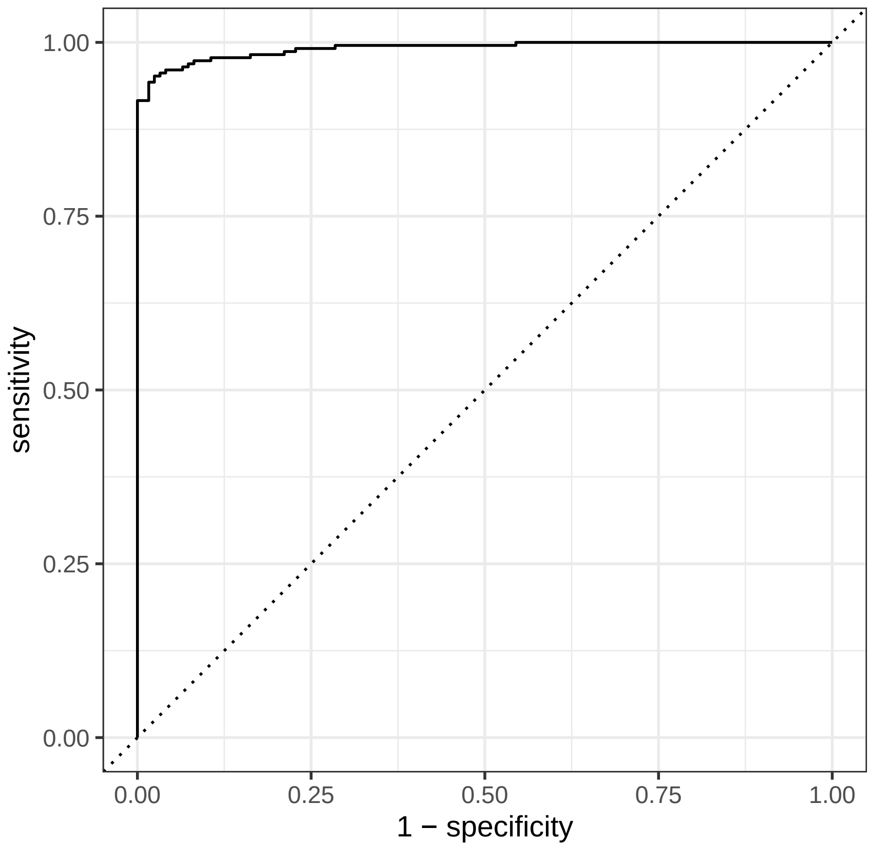
```
augment(qda_fit, new_data = bcancer_test) %>%
  conf_mat(truth = diagnosis, estimate = .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0 216   3
##          1  11 120
```

```
multi_metric <- metric_set(accuracy)

augment(qda_fit, new_data = bcancer_test) %>%
  multi_metric(truth = diagnosis, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary          0.96
```

```
augment(qda_fit, new_data = bcancer_test) %>%
  roc_curve(diagnosis, .pred_0) %>%
  autoplot()
```



```
augment(qda_fit, new_data = bcancer_test) %>%
  roc_auc(diagnosis, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.991
```

The accuracy estimate for our qda model is 0.9484241 which is pretty high. By looking at our roc_curve, we can see that our model is a really good fit for our data set. Additionally, the roc_auc value estimate is 0.9912969 which is extremely high. Therefore, we conclude that the qda model is also a very good model for our data set.

Finding our best model

Now, we will find the best model for our data set. Clearly, all of our models performed pretty well, but we will find the best one to lower our error. Finding the best fit model is very significant for our project, because it is very important to diagnose the right cancer cell to our patient for correct treatment.

```
collect_metrics(tune_res_boost) %>%
  arrange(desc(mean))
```

```
## # A tibble: 10 x 7
##    trees .metric .estimator  mean     n std_err .config
##    <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1    41 roc_auc binary     0.983     3 0.00593 Preprocessor1_Model08
##  2    45 roc_auc binary     0.982     3 0.00604 Preprocessor1_Model09
##  3    50 roc_auc binary     0.982     3 0.00663 Preprocessor1_Model10
##  4    32 roc_auc binary     0.982     3 0.00589 Preprocessor1_Model06
##  5    36 roc_auc binary     0.982     3 0.00565 Preprocessor1_Model07
##  6    14 roc_auc binary     0.981     3 0.00882 Preprocessor1_Model02
##  7    18 roc_auc binary     0.981     3 0.00776 Preprocessor1_Model03
##  8    23 roc_auc binary     0.980     3 0.00821 Preprocessor1_Model04
##  9    10 roc_auc binary     0.979     3 0.00978 Preprocessor1_Model01
## 10    27 roc_auc binary     0.979     3 0.00813 Preprocessor1_Model05
```

```
collect_metrics(tune_res_forest) %>%
  arrange(desc(mean))
```

```
## # A tibble: 512 x 9
##     mtry trees min_n .metric .estimator  mean     n std_err .config
##    <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1     1     6     1 roc_auc binary     0.989     3 0.00611 Preprocessor1_Model~
##  2     1     8     1 roc_auc binary     0.987     3 0.00545 Preprocessor1_Model~
##  3     1     5     5 roc_auc binary     0.986     3 0.00915 Preprocessor1_Model~
##  4     4     8     6 roc_auc binary     0.986     3 0.00921 Preprocessor1_Model~
##  5     1     3     4 roc_auc binary     0.985     3 0.00661 Preprocessor1_Model~
##  6     1     5     1 roc_auc binary     0.985     3 0.00738 Preprocessor1_Model~
##  7     1     4     7 roc_auc binary     0.985     3 0.00847 Preprocessor1_Model~
##  8     3     8     6 roc_auc binary     0.985     3 0.00798 Preprocessor1_Model~
##  9     2     6     6 roc_auc binary     0.985     3 0.00925 Preprocessor1_Model~
## 10     1     7     2 roc_auc binary     0.985     3 0.00792 Preprocessor1_Model~
## # ... with 502 more rows
```

```
augment(log_test, new_data = bcancer_test) %>%
  roc_auc(diagnosis, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.994
```

```
augment(qda_fit, new_data = bcancer_test) %>%
  roc_auc(diagnosis, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.991
```

```
roc_auc_values <- c(0.9826345, 0.9894460, 0.9938039, 0.9912969  )
models <- c("Random Forest", "Boosted Trees", "Logistic Regression","QDA")
results <- tibble(roc_auc_values = roc_auc_values, models = models)
results
```

```
## # A tibble: 4 x 2
##   roc_auc_values models
##            <dbl> <chr>
## 1          0.983 Random Forest
## 2          0.989 Boosted Trees
## 3          0.994 Logistic Regression
## 4          0.991 QDA
```

The following table is the roc_auc values for all four models. We can see that the logistic regression model has the highest roc_auc values. Therefore, we conclude that the logistic regression model is the best fit for our model.

Now, we will try to fit the logistic regression model into our testing data set to see how it does.

```
predict(lr_poly_fit, new_data = bcancer_test, type = "prob")
```

```
## # A tibble: 350 x 2
##      .pred_0 .pred_1
##        <dbl>   <dbl>
##  1 0.971     0.0289
##  2 0.316     0.684
##  3 0.991     0.00912
##  4 0.0702    0.930
##  5 0.977     0.0227
##  6 0.0000239 1.00
##  7 0.991     0.00936
##  8 0.998     0.00211
##  9 0.997     0.00302
## 10 0.997     0.00284
## # ... with 340 more rows
```

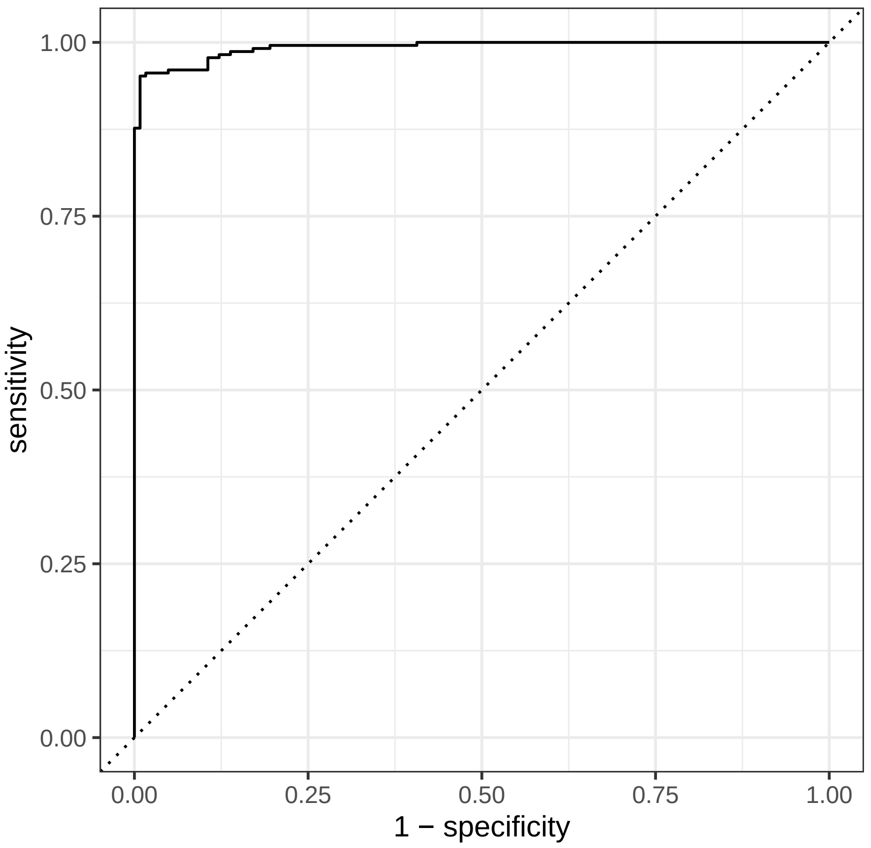
```
augment(lr_poly_fit, new_data = bcancer_test) %>%
  conf_mat(truth = diagnosis, estimate = .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0 218   7
##          1   9 116
```

```
multi_metric <- metric_set(accuracy)

augment(lr_poly_fit, new_data = bcancer_test) %>%
  multi_metric(truth = diagnosis, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.954
```

```
augment(lr_poly_fit, new_data = bcancer_test) %>%
  roc_curve(diagnosis, .pred_0) %>%
  autoplot()
```



We can see that we only have a total of 16 cases that were not predicted correctly. Additionally, the accuracy estimate is 0.9542857, which is pretty high. The roc_curve seems to look very good as well, showing a good performance.

Overall, we conclude that the logistic regression model is the best fit for our data set.

Conclusion

Our goal of this project was to create a model to correctly classify whether the breast cancer is a malignant or benign type with the help of our data set. For a breast cancer patient, the accuracy of determining the right diagnosis is very significant for their treatment. Therefore, finding the lowest error and highest accuracy, in other words, finding the best fit model is very critical in this field.

Before we started fitting our models, we cleaned our data and split our data for the process. This helped reduce any type of error for our project.

In this project, the following four models were used : random forest, boosted trees, logistic regression, and QDA. The four models were all very successful models with very low error. But recall that having the highest accuracy in our model is significant for our data set because receiving the right treatment is very important for our patients. Diagnosing the patients with wrong information is critically bad in the medical field because it could cause the patient life or death, especially since this data set covers cancer patients. Therefore, minimizing even a subtle accuracy seems important in our project. Therefore, even though all our four models performed pretty well, we found the best fitting model. We found the roc_auc values for the four models and compared the highest value. We found out that the logistic regression had the highest roc_auc value. Afterwards, we fitted the logistic regression model to our testing data set. We also got a good fit model with a high accuracy and roc_auc value. Thus, we conclude that our logistic regression model is the best model for our data set.

Further research could be done by including other potentially significant predictors to our project, so that we could get more information than just malignant or benign cancer cells. In other words, we can go more in depth by including more information to gain more information about the patient about breast cancer. Or other models could be fitted to find a better model than the logistic regression model, if possibly exists.

Overall, the research project with the breast cancer data set was very successful, determining which cancer cell, malignant or benign, a patient has. With high roc_auc values and accuracies, the models used would successfully determine whether the patient has a malignant or benign type.