Breast Cancer Image Classification

cristian razo

6/8/2022

Contents

Goal

The goal of this project is to make a model to predict whether a patient will be diagnosed malignant (cancer) or benign (not cancer). Based on the breast cancer wisconsin data set that describes the characteristics of the patients cell nuclei.

Overview of the dataset

The dataset that I will be working on is known as "Breast Cancer Wisconsin Data Set".The data set is a description of the characteristics of the cell nuclei present in an image. I will be accessing this data set from the website kaggle where they allow people to download the breast cancer csv. The dataset consist of the 32 columns with 569 instances/observations and none of the rows have any missing values.This data set has 31 predictor variables and 1 response variable which is diagnosis of the breast tissue.

Website : https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

Why should we construct a diagnosis classification

I decided to construct this classifciation model for inferential and preditcive purposes. If I am able to construct a classification model that has a high accuracy of predicting which patient has cancer. This will benefit in cutting man hours and also to be used in hospital as a second pair of doctor eyes . With a high accuracy rate we can also look in great depth at our features and understand the relationship they have with the response variable.The inferential part will help doctors understand the important factors and its

relationship with the response variable . This can help doctors look at certain features to make assumptions and correlations of what might cause a patient to get cancer.Finding the root cause can help alert cancer patient in the early stages and save more lives.

Key variables and Packages used

The packages we used in loading our data , creating our model , and data visualization

Packages *Tidyverse* : opinionated collection of R packages designed for data science

Tidymodels: packages for modeling and machine learning using tidyverse principles

MASS: engine used for classification

dplyr : a structure of data manipulation

*ggplot*2: open-source data visualization package

Key Variable The key variables I used to construct my classification model.I will explain why I used these features later in my report.

Response Variable *diagnosis* :The diagnosis of breast tissues (M = malignant, B = benign) M=cancer B=no cancer

Predictor Variable *texture_mean* : standard deviation of gray-scale values

radius_mean : mean of distances from center to points on the perimeter

perimeter_mean : mean size of the core tumor

area_mean : The mean area of nuclei space

compactness_mean : mean of perimeterˆ2 / area - 1.0

concavity_mean : mean of severity of concave portions of the contour

**Code book provided in files*

Data Loading and Cleaning

Loading in our data and factoring the response variable diagnosis into its two levels .

```
dt=read.csv("~/Desktop/Final_project_131/Data/unprocessed/data.csv") %>%
                                        mutate(diagnosis = factor(diagnosis, levels = c("M", "B")))
```

We checked our columns variable type .There is only one factor variable and the rest are numerical. Also cleaned our column names so that the syntax can run more smoothly.

```
str(dt)
```

```
## 'data.frame':    569 obs. of  33 variables:
##  $ id                     : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844
##  $ diagnosis              : Factor w/ 2 levels "M","B": 1 1 1 1 1 1 1 1 1 1 ...
##  $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
```

```
##  $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
##  $ area_mean              : num  1001 1326 1203 386 1297 ...
##  $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ concave.points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
##  $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
##  $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
##  $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
##  $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ concave.points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
##  $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
##  $ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
##  $ area_worst             : num  2019 1956 1709 568 1575 ...
##  $ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
##  $ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
##  $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
##  $ concave.points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
##  $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
##  $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
##  $ X                      : logi  NA NA NA NA NA NA ...
```

```
dt <- dt %>%
  clean_names()
```

The data set is from kaggle and it has no missing values . There was not much cleaning to do to our dataset because it was given to us clean already.

Data Splitting

The dataset is splitted into a training set and testing set . 80% of our data is in the training set leaving the remainder to be in the testing set. We stratified the sampling to resemble the population space of diagnosis and its levels.

I didnt approach EDA until I am able to establish the training data set. I want find patterns and trend based on the observation that are in the training set .

```
set.seed(69)
split <- initial_split(dt, prop = 0.80, strata = diagnosis)
train <- training(split)
test <- testing(split)

K_folds <- vfold_cv(train, strata = diagnosis,
                           v = 10)
```

The training data set has 454 observation while the testing data set has 115 observation. A K fold of 10 is also processed into a variable to be used later.

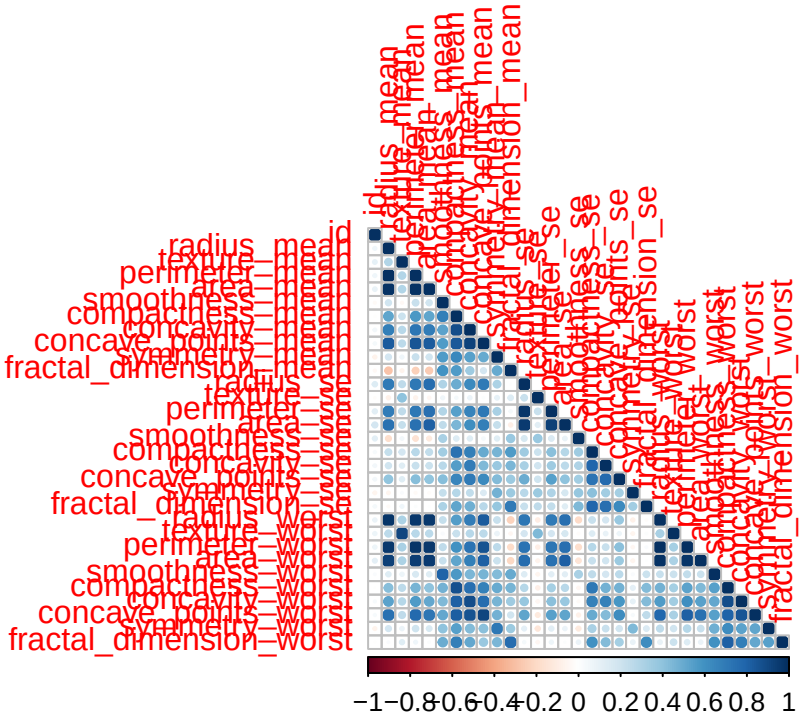
The K fold is used for cross validation . Cross validation is a resampling method that uses different portons of the data to test and train a model on different iteration . Its a way to keep our testing not to be spoiled when trying to fit our models.

Exploratory Data Analysis

Is there any relationship that will prove to be useful ? How does the cancer behave in terms of the cell nuclei ?

I will answer these questions in my EDA to understand my data better .

```
train_nu=select_if(train,is.numeric)

## train correlation

train_nu %>%
  cor() %>%
  corrplot(type = "lower")
```



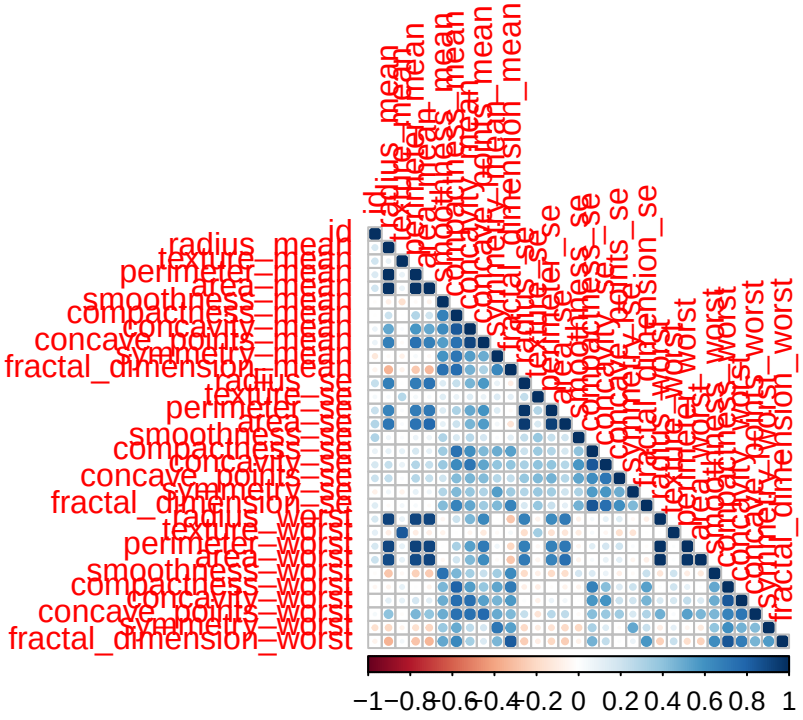
Correlation heatmap on the training set

The heat map in detail tells us which columns are correlated with each other.We can see that there alot of significaly positive correlation. This tells us that many of the features are associated with each other. This can help us in reducing our dimension in our data frame and make it less complicated.

Correlation heat map on the training set based on cancer type Seperated the training data set based on cancer type to see how the relationship founded differ from each other.

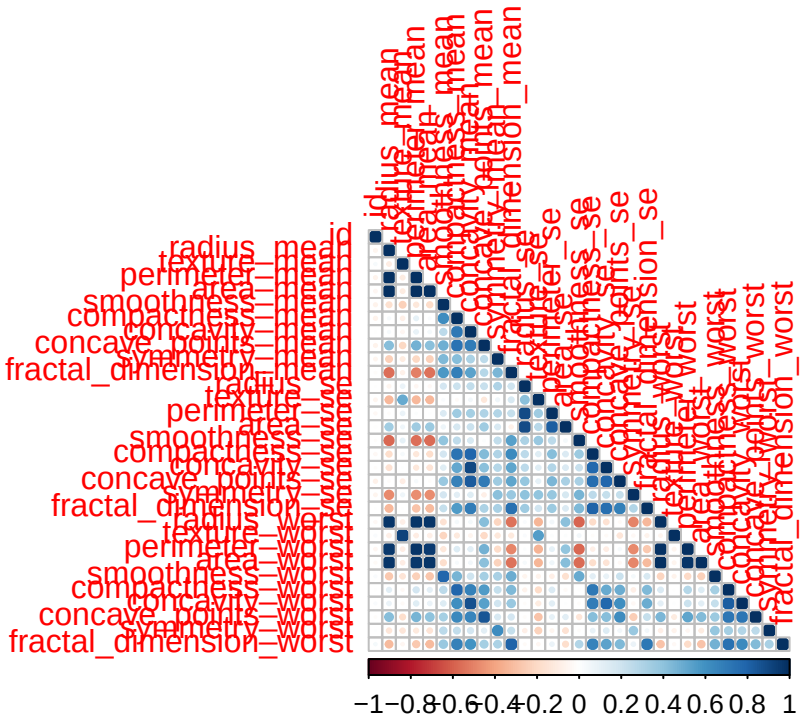
```
M_nu=train_nu[train[2]=='M',]

B_nu=train_nu[train[2]=='B',]
```

```
M_nu %>%
  cor() %>%
  corrplot(type = "lower")
```



Correlation heatmap on Malignant patients

```
B_nu %>%
  cor() %>%
  corrplot(type = "lower")
```



Correlation heatmap on Benign patients

The purpose of making a heat map for each cancer type is to check if they share the same relationships. The visualization shows us that Malignant has more positive significant correlation than Benign but Benign shows more negative significant correlations. Its best to look at these correlation in more depth so we can reduce the dimension in our data frame.

```
## see relationship look for reduce demnsion

M_c=M_nu%>%
  cor()

B_c=B_nu%>%
  cor()
```

```
corr_M <- as.data.frame(as.table(M_c))
#corr=corr[lower.tri(corr,diag=TRUE)] <- NA
corr_M=corr_M[!duplicated(corr_M$Freq), ]

corr_B<- as.data.frame(as.table(B_c))
#corr=corr[lower.tri(corr,diag=TRUE)] <- NA
corr_B=corr_B[!duplicated(corr_B$Freq), ]


sig_m=corr_M %>%
   filter(Freq>.7 )

sig_b=corr_B %>%
  filter(Freq>.7 )
```

```
sig_M2 = sig_m%>% group_by(Var2) %>% summarise(n=n())
sig_B2 = sig_b%>% group_by(Var2) %>% summarise(n=n())
```

Correlation in depth

```
sig_M2 %>%
  filter(n>1)
```

Significant correlation founded in Malignant by columns

```
## # A tibble: 14 x 2
##    Var2                      n
##    <fct>                 <int>
##  1 radius_mean               6
##  2 perimeter_mean            6
##  3 area_mean                 6
##  4 smoothness_mean           2
##  5 compactness_mean          8
##  6 concavity_mean            4
##  7 concave_points_mean       2
##  8 fractal_dimension_mean    2
##  9 radius_se                 3
## 10 area_se                   3
## 11 compactness_se            2
## 12 concavity_se              2
## 13 radius_worst              2
## 14 compactness_worst         2
```

```
sig_B2 %>%
  filter(n>1)
```

Significant correlation founded in Benign by columns

```
## # A tibble: 12 x 2
##    Var2                      n
##    <fct>                 <int>
##  1 radius_mean               5
##  2 perimeter_mean            4
##  3 area_mean                 3
##  4 compactness_mean          4
##  5 concavity_mean            4
##  6 concave_points_mean       2
##  7 fractal_dimension_mean    2
##  8 radius_se                 2
##  9 compactness_se            4
```

```
## 10 concavity_se               3
## 11 radius_worst               2
## 12 compactness_worst          3
```

Based on the tibble presented above I will reduce my model only to include radius mean ,perimeter mean , area mean ,compactness mean, concavity mean. I chose these variables because they positively represent other columns and information will not be lost when using them.

```
c1=corr_M %>% filter(Freq<.20)

c2=corr_B %>% filter(Freq<.20)

sig_M2 = c1%>% group_by(Var2) %>% summarise(n=n())
sig_B2 = c2%>% group_by(Var2) %>% summarise(n=n())
```

```
sig_M2 %>%
  filter(n>1)
```

Insignificant correlation founded in Malignant by columns

```
## # A tibble: 25 x 2
##    Var2                    n
##    <fct>               <int>
##  1 id                     24
##  2 radius_mean            15
##  3 texture_mean           26
##  4 perimeter_mean         14
##  5 area_mean              14
##  6 smoothness_mean         8
##  7 compactness_mean        4
##  8 concavity_mean          2
##  9 concave_points_mean     5
## 10 symmetry_mean           9
## # ... with 15 more rows
```

```
sig_B2 %>%
  filter(n>1)
```

Insignificant correlation founded in Benign by columns

```
## # A tibble: 25 x 2
##    Var2                    n
##    <fct>               <int>
##  1 id                     30
##  2 radius_mean            21
##  3 texture_mean           26
##  4 perimeter_mean         20
```

```
##  5 area_mean              20
##  6 smoothness_mean        12
##  7 compactness_mean        7
##  8 concavity_mean          8
##  9 concave_points_mean     5
## 10 symmetry_mean           8
## # ... with 15 more rows
```

I dont want to shy away from that share no correlation as these features can still prove to be informative. Based on the results founded I will include texture mean to be induced into my model along with the other features I named recently .

In all together I am able to reduce my dimension from 32 columns to only 7 columns. This will help our model to be less complicated.

- Dataframe reduced to ; diagnosis ,texture_mean, radius mean ,perimeter mean , area mean ,compactness mean concavity mean.

Scatter plots of high correlated pairs Malignant

```
sig_m %>% filter(Freq>.98)
```

```
##              Var1           Var2      Freq
## 1              id             id 1.0000000
## 2  perimeter_mean    radius_mean 0.9952619
## 3       area_mean    radius_mean 0.9896203
## 4       area_mean perimeter_mean 0.9866424
## 5 perimeter_worst   radius_worst 0.9837732
## 6      area_worst   radius_worst 0.9852090
```

I found 5 relationship that have a really high correlation value. This means that they are very close in the way they act.

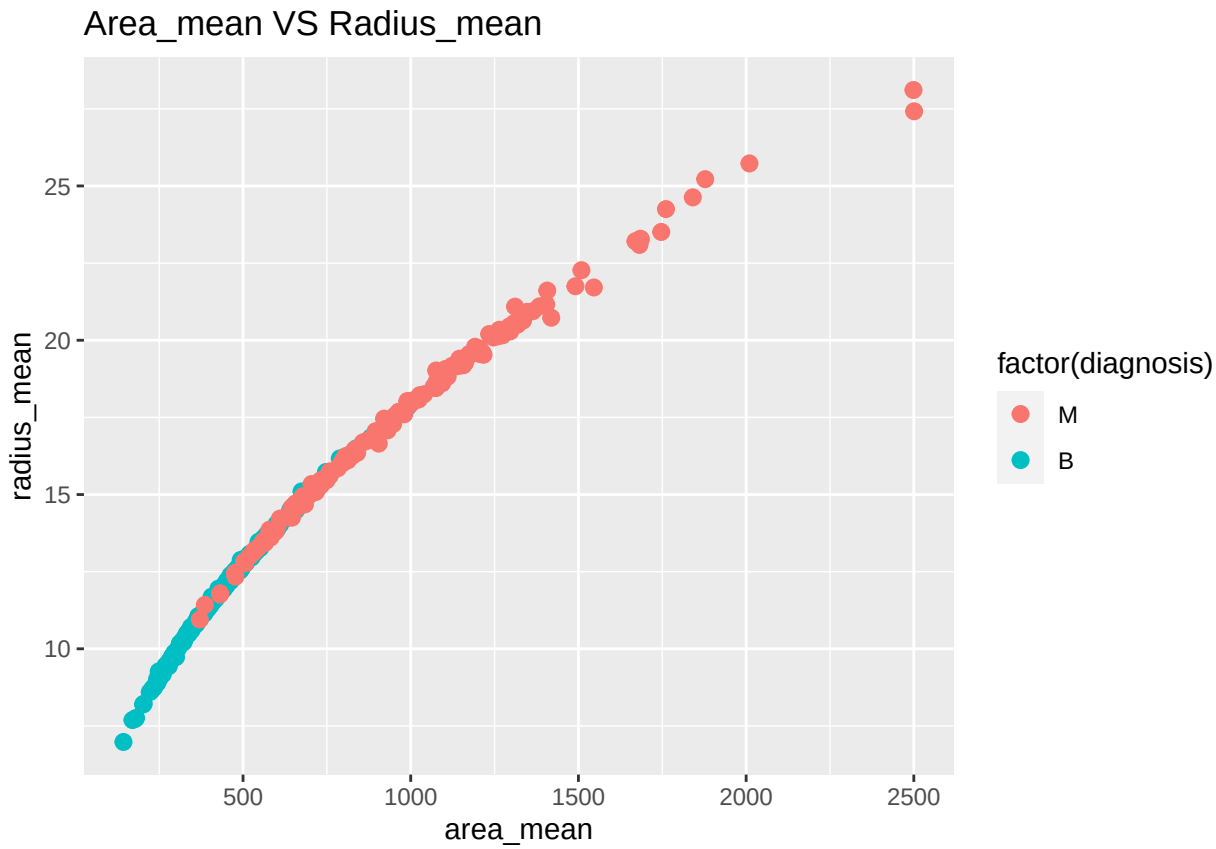
Benign

```
sig_b %>% filter(Freq>.98)
```

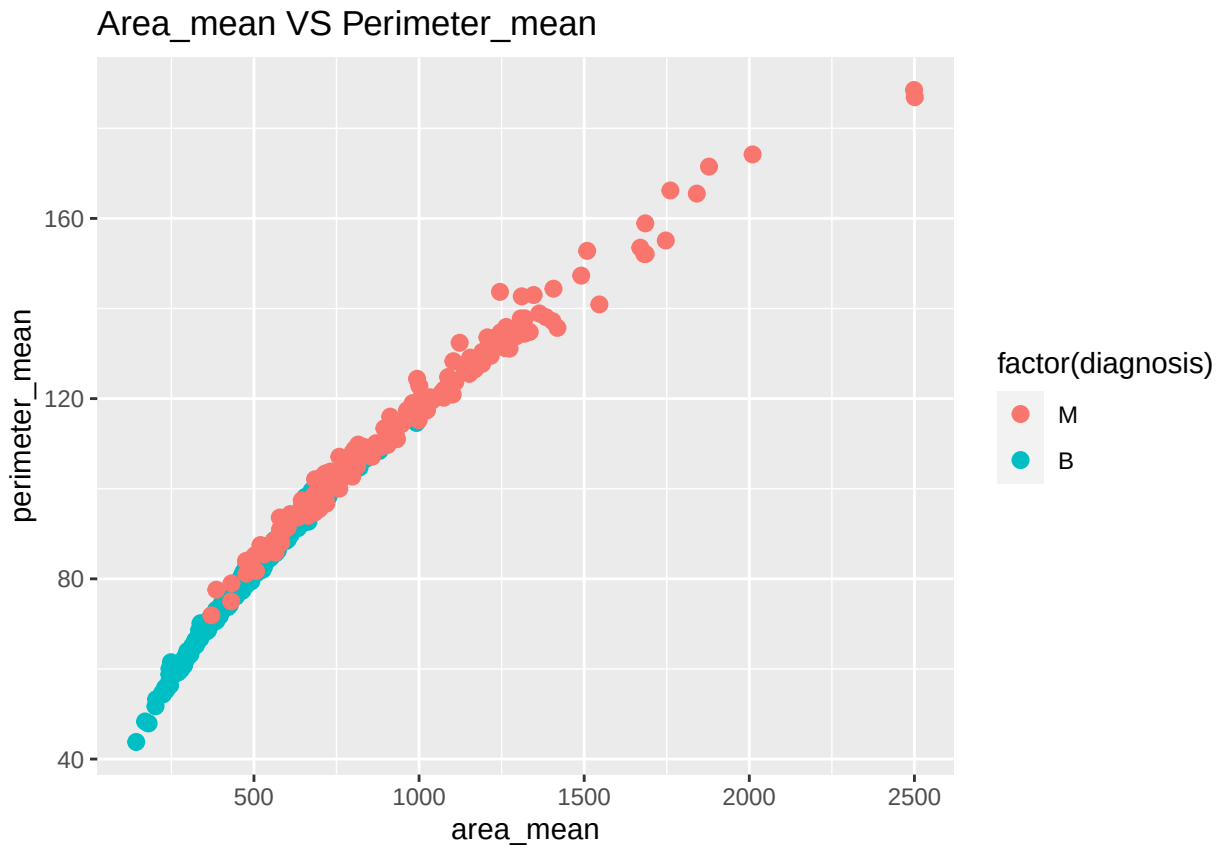
```
##              Var1           Var2      Freq
## 1              id             id 1.0000000
## 2  perimeter_mean    radius_mean 0.9971616
## 3       area_mean    radius_mean 0.9940867
## 4       area_mean perimeter_mean 0.9904797
## 5 perimeter_worst   radius_worst 0.9850737
## 6      area_worst   radius_worst 0.9939828
```

Same goes for the Benign there are 5 relationship that are high correlated . It be best to see their linear relationship of these features

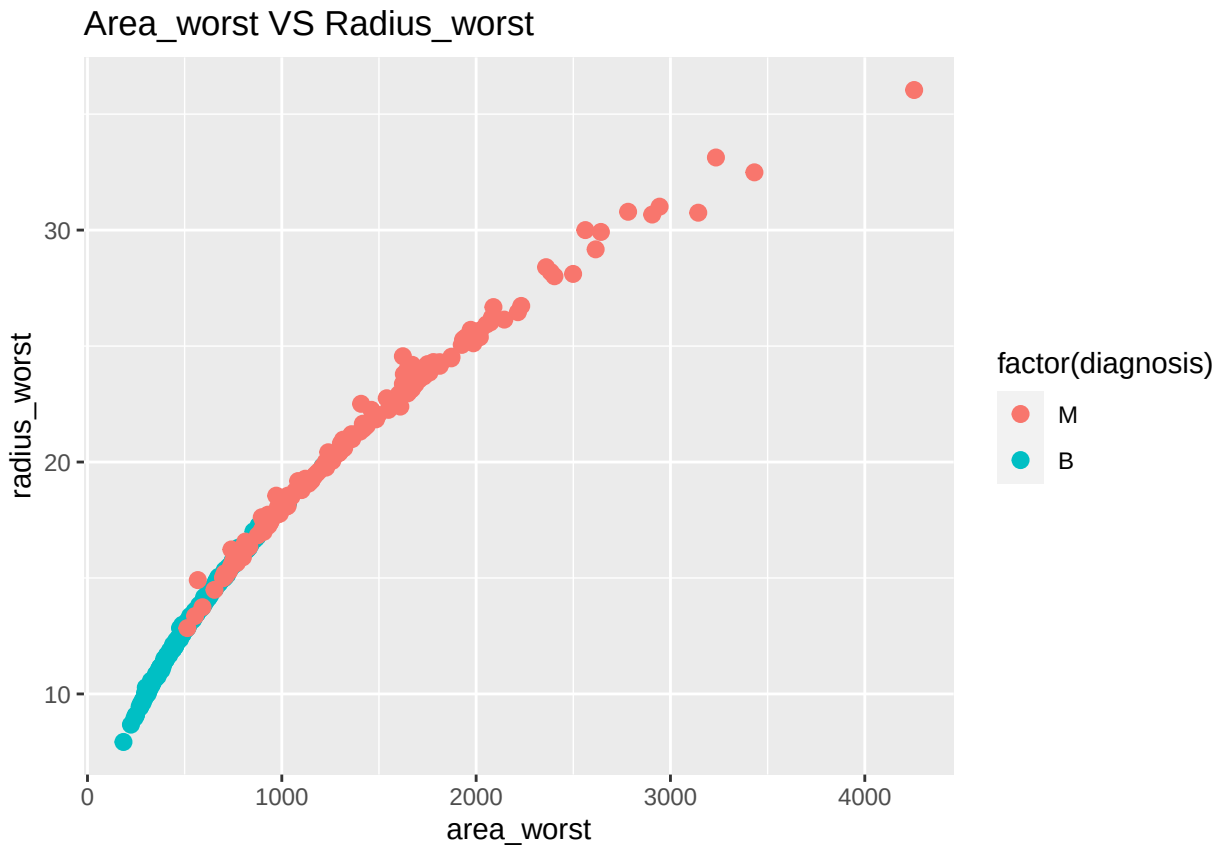
```
ggplot(train, aes(x =area_mean , y = radius_mean, colour = factor(diagnosis)))+
  geom_point(size=2.5)+ ggtitle("Area_mean VS Radius_mean")
```



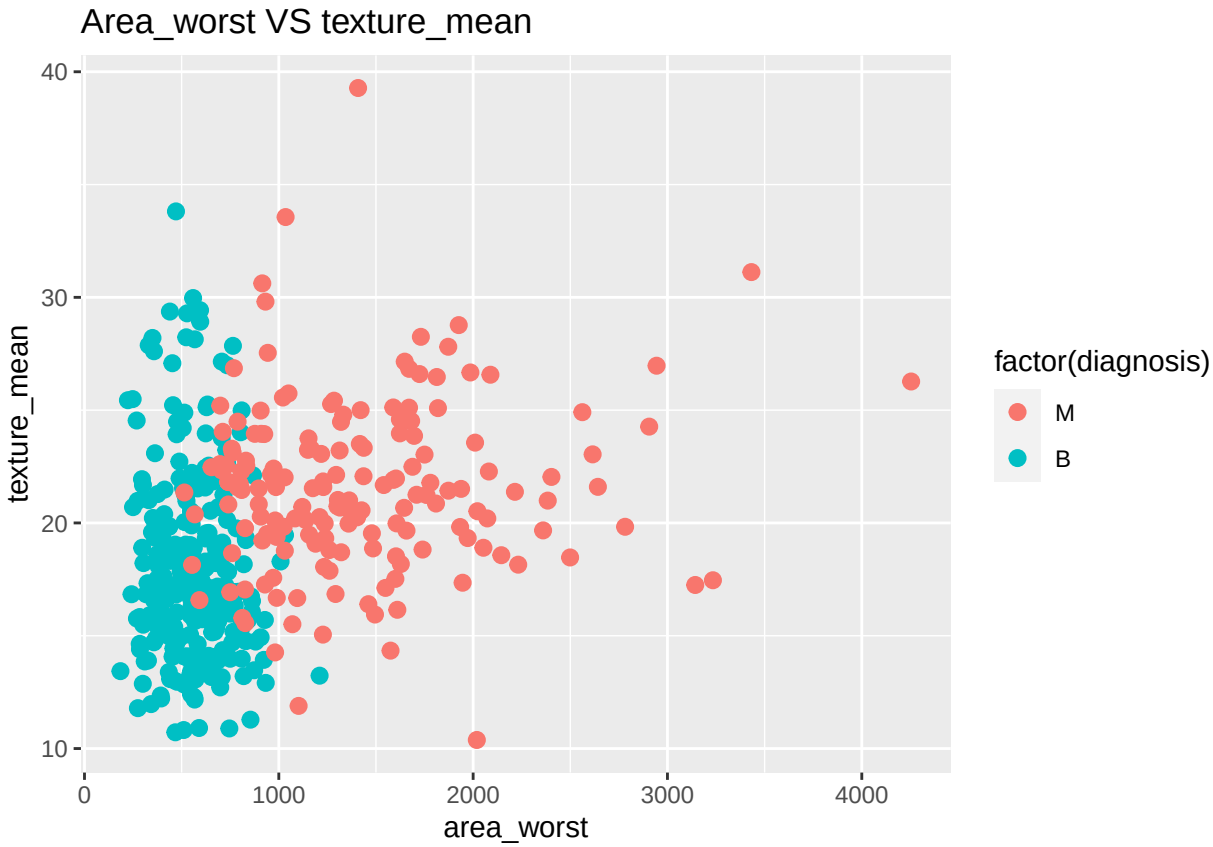

```
ggplot(train, aes(x =area_mean , y = perimeter_mean, colour = factor(diagnosis)))+
  geom_point(size=2.5) + ggtitle("Area_mean VS Perimeter_mean")
```




```
ggplot(train, aes(x =area_worst , y = radius_worst, colour = factor(diagnosis)))+
  geom_point(size=2.5)+ ggtitle("Area_worst VS Radius_worst")
```



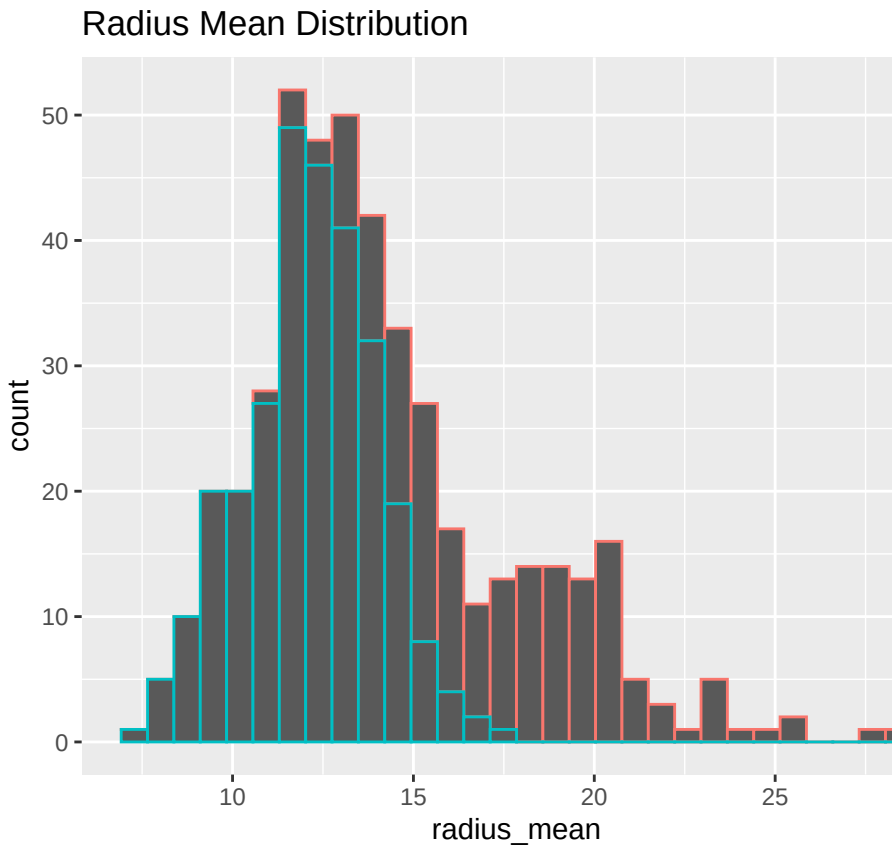

```
ggplot(train, aes(x =area_worst , y = texture_mean, colour = factor(diagnosis)))+
  geom_point(size=2.5)+ ggtitle("Area_worst VS texture_mean")
```



The visualization of the scatter plot shows that they do have strong postive correlation. It also shows the grouping of the diagnosis. This means that diagnosis labels can be distinguished as theyre not always founded to be in the same population space.

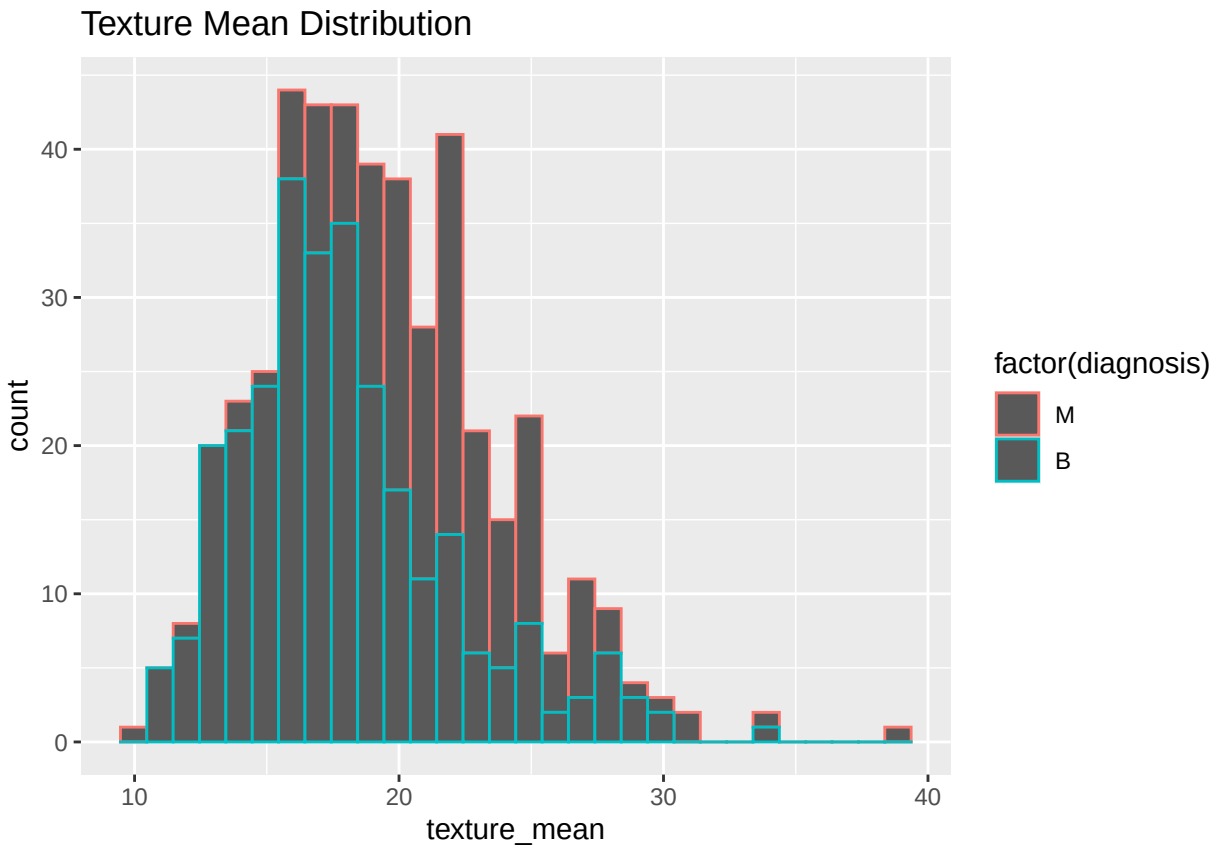
Benign is seen to always be located at the beginning of our scatter plot while Malignant is at at the far end . This could be useful to use for the inferential part of our model.

```
train %>%
  ggplot(aes(x = radius_mean,colour=factor(diagnosis))) +
  geom_histogram()+ ggtitle('Radius Mean Distribution')
```

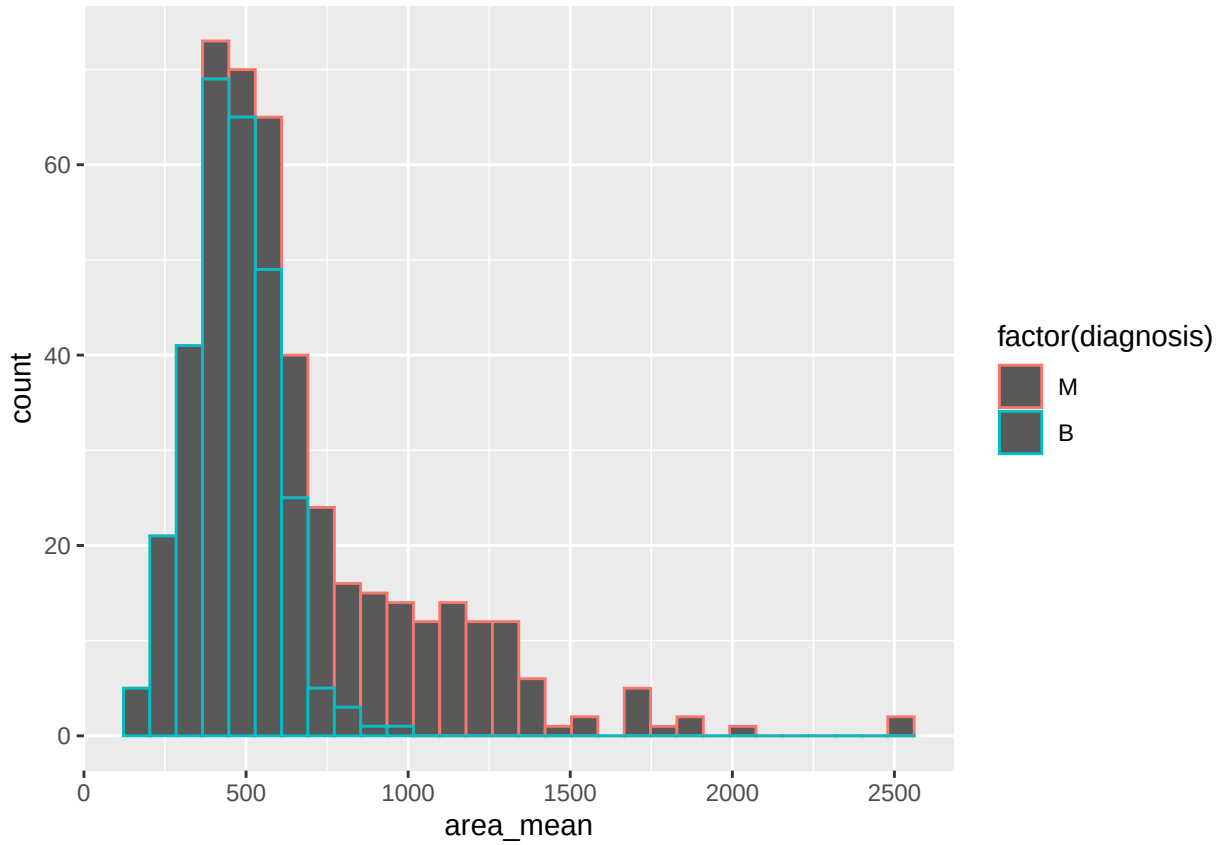


Distribution of the observation collected

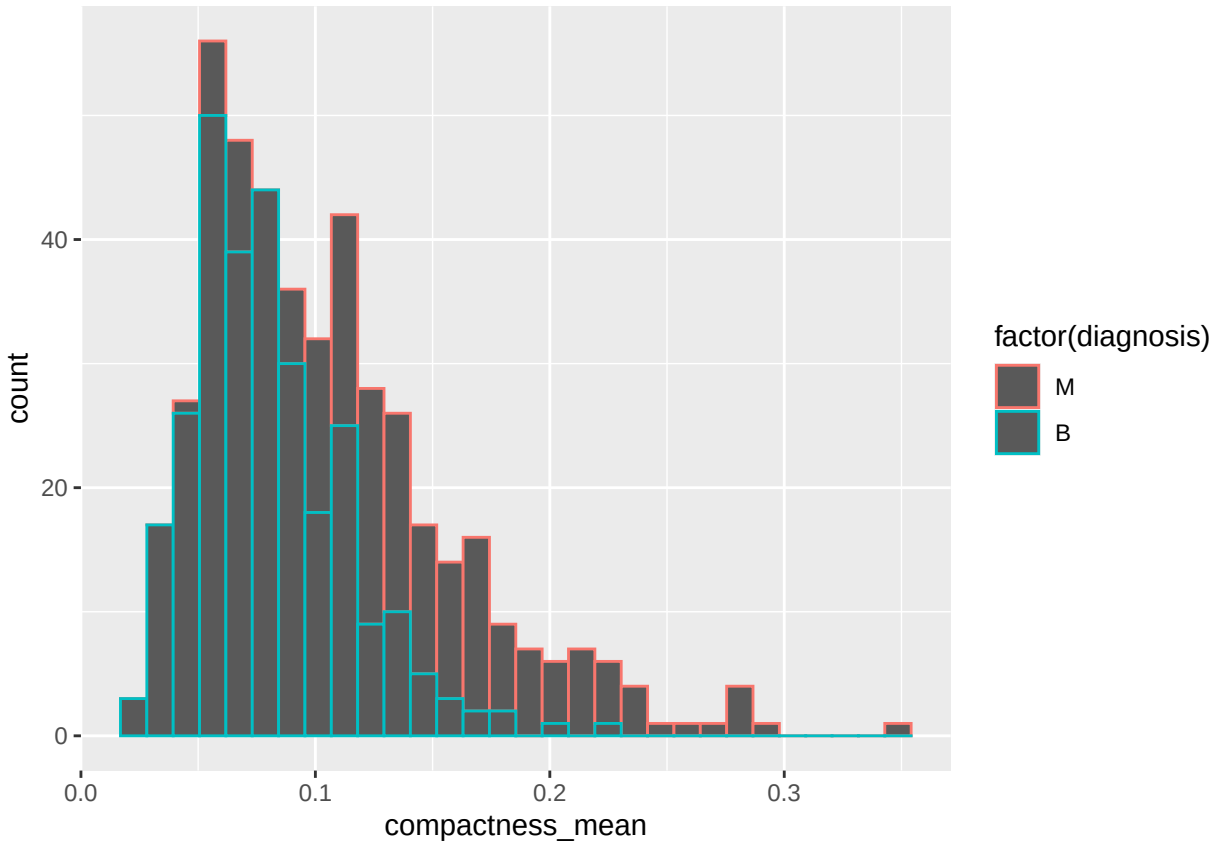
```
train %>%
  ggplot(aes(x = texture_mean,colour=factor(diagnosis))) +
  geom_histogram()+ ggtitle('Texture Mean Distribution')
```




```
train %>%
  ggplot(aes(x = area_mean,colour=factor(diagnosis))) +
  geom_histogram()
```




```
train %>%
  ggplot(aes(x = compactness_mean,colour=factor(diagnosis))) +
  geom_histogram()
```



The visualization above shows that its distribution are right skewed and not approximating a normal distribution. The Benign population lies mostly on the lower end of the distribution. The Malignant population is spread out more evenly through out the distribution but still is on a right skewed form.

Model Building

Recipe Building and Tuning

The recipe I created induces the predictor variable and response variable of my model. There's also a step we included to standarize and scale all our predictor variables by normalzing them.

```
diagnosis_recipe <- recipe(diagnosis~texture_mean+radius_mean+perimeter_mean+area_mean+compactness_mean
  step_normalize(all_predictors())
```

The predictor variables are texture_mean ,radius_mean ,perimeter_mean, area_mean,compactness_mean , concavity_mean.

Preparing and Fitting models

I will now use the 10 k fold validation set to be used to fit the four classification models. The 4 models I chose are widely used for classification problems. Models :

Logestic Regression

Decission Tree

RandomForest

Boosted Trees

For each model I will assign them there engine and mode to be used in context to our goal. Based on the models Hyperparameter we will tune them in order to find the best parameter .Then add the model to the workflow that combines both our custom model and recipe to be used.

Logistic regression model will have no tuning parameters. I received a syntax error that I couldnt resolve.

All of these models will fit onto a cross validation and will give us the mean accuracy and roc score achieved for the 10 folds we assigned earlier in data splitting

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

log_wkflow <- workflow() %>%
  add_recipe(diagnosis_recipe) %>%
  add_model(log_reg)
```

Logestic Regression Assigned the engine to be GLM and mode to classification

```
tree_spec <- decision_tree() %>%
  set_engine("rpart")
class_tree_spec <- tree_spec %>%
  set_mode("classification")

auto_wrkflw2=workflow()%>%
  add_recipe((diagnosis_recipe)) %>%
  add_model(class_tree_spec %>% set_args(cost_complexity = tune()))

param_grid <- grid_regular(cost_complexity(range = c(-3, -1)), levels = 10)
```

Decision Tree Assigned the engine to be rpart and mode to be classification. We tune only the cost complexity parameters at only 10 levels

```
rf_spec <- rand_forest(mtry = tune(),trees=tune(),min_n=tune())%>%
  set_engine("ranger", importance = 'impurity') %>%
  set_mode("classification")


auto_wrkflw3=workflow()%>%
  add_recipe((diagnosis_recipe)) %>%
  add_model(rf_spec)

grid_pen=grid_regular(mtry(range = c(1,6)),trees(range = c(1,10)),min_n(range = c(1,10)),
                      levels =6)
```

Random Forest Assigned the engine to be ranger and mode to be classification. The parameters that will be tuned are mtry , trees , and min_n . It must be between 1-6 for mtry because we have 6 predictors in our model.

```
boost_spec <- boost_tree(trees = tune() )%>%
  set_engine("xgboost") %>%
  set_mode("classification")



auto_wrkflw4=workflow()%>%
  add_recipe((diagnosis_recipe)) %>%
  add_model(boost_spec)

grid_trees=grid_regular(trees(range = c(10,2000)),levels =10)
```

Boosted trees Assigned the engine to be xgboost and mode to be classification.The parameter being tuned is trees from (10-2,000).

Fitting our models

I will fit all the models I created above. We will fit them onto our K folds to do cross validation before we fit it onto our testing and training set .

The reason for the cross validation is to find the best hyper parameters and to not spoil our testing set.

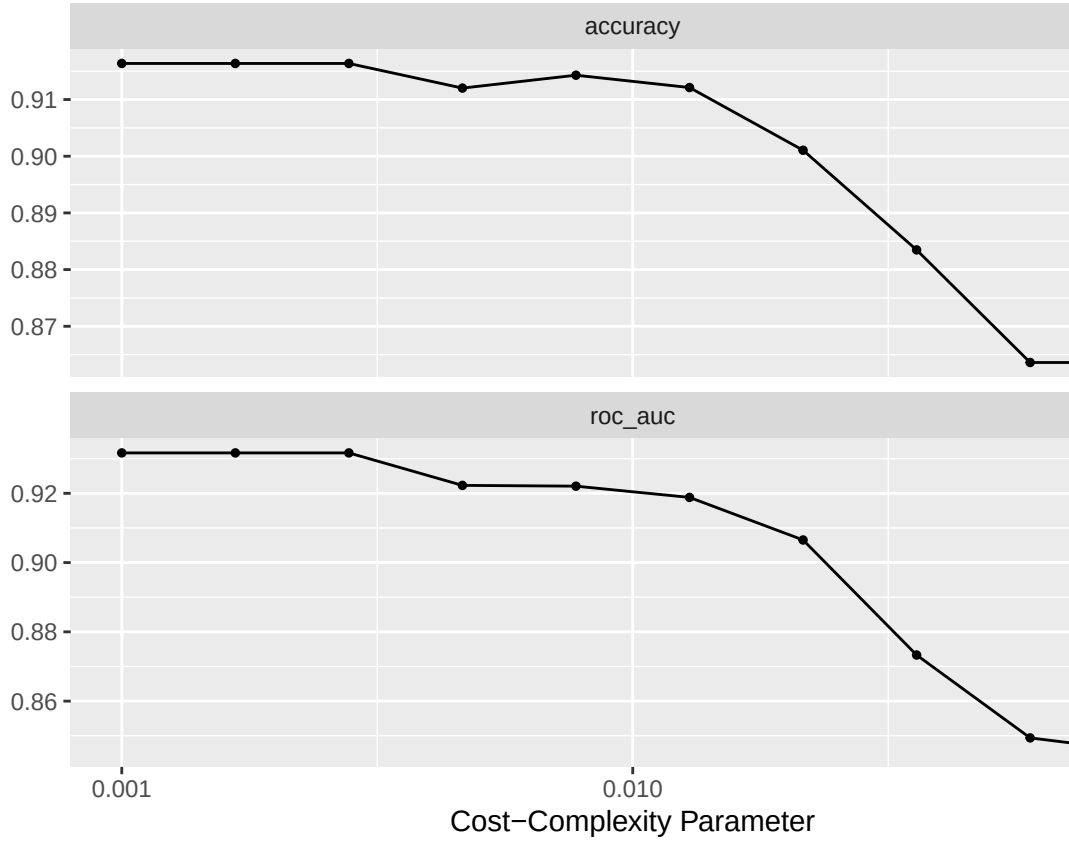
```
## logestic regression model

log_fit=fit_resamples(log_wkflow,K_folds)


## DT model
tune_res2 <- tune_grid(
  auto_wrkflw2,
  resamples = K_folds,
  grid = param_grid,
  metrics = metric_set(roc_auc,accuracy)
)

## RF model
tune_res3=tune_grid(
  object = auto_wrkflw3,
  resamples = K_folds,
  grid = grid_pen,
  metrics = metric_set(roc_auc,accuracy)
)


## Boosting model
tune_res4=tune_grid(
  object = auto_wrkflw4,
```

```
  resamples = K_folds,
  grid = grid_trees,
  metrics = metric_set(roc_auc,accuracy)
)
```

Analyzation on Performance models

Now that we fitted out model to the K folds it is now time to analyze there performances along with the tuning involved. The visualization lets us see how well are model is fitted based on the tuning parameters.

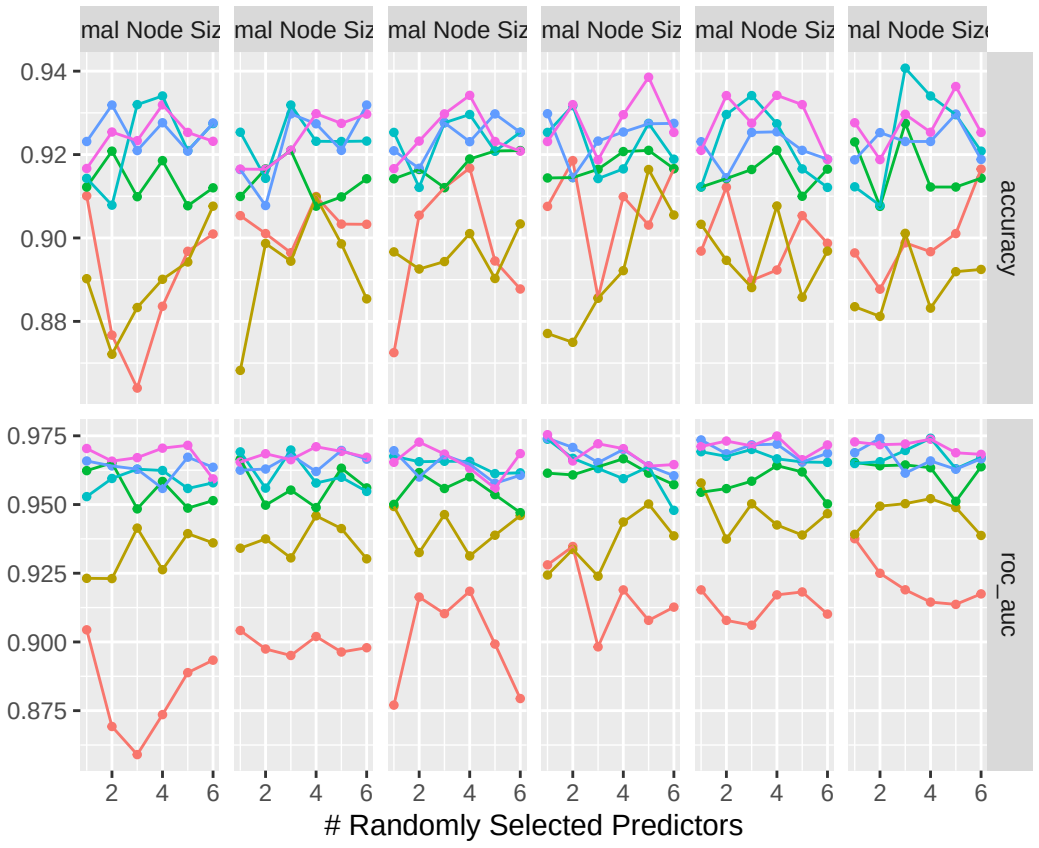
```
autoplot(tune_res2)
```



Decision Tree Performance

A Decision Tree with low complexity value will fit the best . A low complexity value results in high accuracy score and a high ROC as well

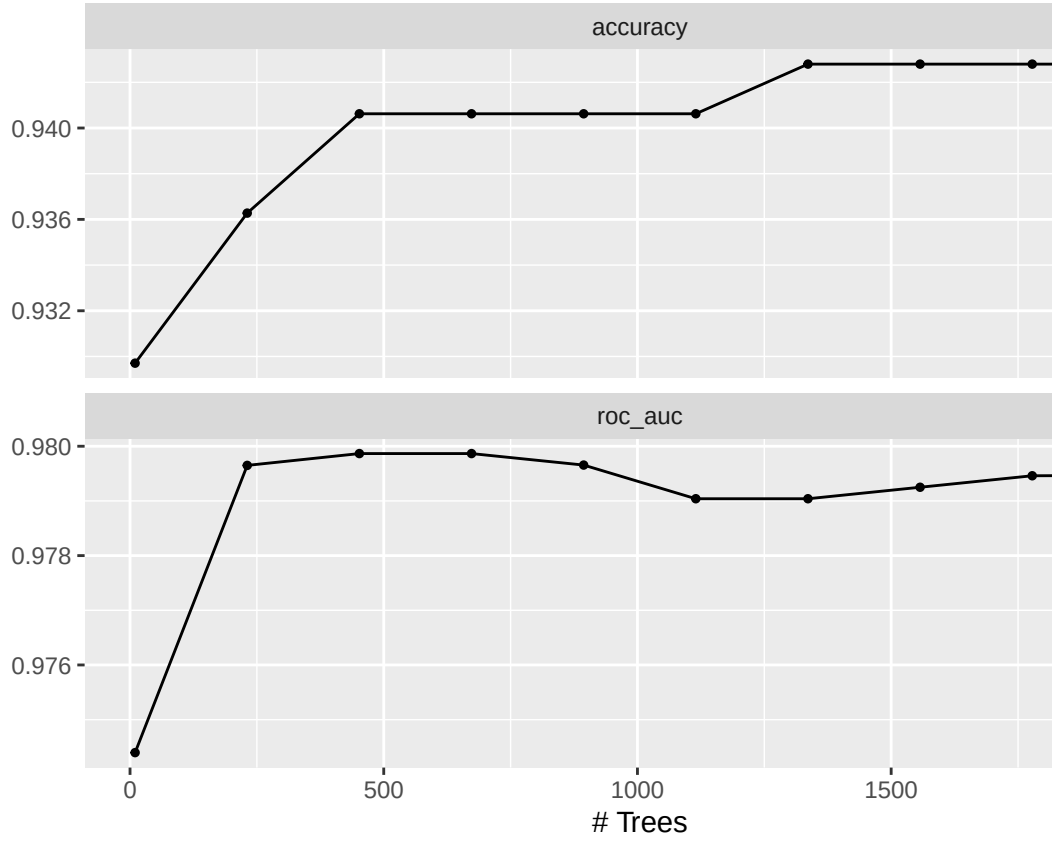
```
autoplot(tune_res3)
```



Random Forest Performance

A Random forest with high number of trees will fit the best.A high tree count results in high accuracy score and a high ROC as well .

```
autoplot(tune_res4)
```



Boosting Trees Performance

A Random forest with high number of trees will fit the best.A high tree count results in high accuracy score and a high ROC as well .

Select the best tuned parameters based on ROC scoring metric .

I will now select the models parameter with highest ROC score to be considrered as the best model. I will use the best model to fit it on the whole training set and see how it performs.

```
metrics_2=collect_metrics(tune_res2)
best2=select_best(tune_res2)
best2
```

```
## # A tibble: 1 x 2
##   cost_complexity .config
##             <dbl> <chr>
## 1           0.001 Preprocessor1_Model01
```

Decision tree best parameters :

- Cost complexity =.001

```
metrics_3=collect_metrics(tune_res3)
best3=select_best(tune_res3)
best3
```

```
## # A tibble: 1 x 4
##    mtry trees min_n .config
##   <int> <int> <int> <chr>
## 1     1    10     6 Preprocessor1_Model139
```

Random Forest best parameters :

- Mtry =1
- Tress =10
- Min_n=6

```
metrics_4=collect_metrics(tune_res4)
best4=select_best(tune_res4)
best4
```

```
## # A tibble: 1 x 2
##   trees .config
##   <int> <chr>
## 1   452 Preprocessor1_Model03
```

Boosted Trees :

- Trees =452

Fit the best models to the training set

Based on my models best hyper parameters . I will fit them onto my training set and check how well it performed.

```
log_fit2=fit(log_wkflow,train)

predict(log_fit2, new_data = train, type = "class") %>%
  bind_cols(train) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
```

Log fit and accuracy

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.932
```

```
DT_final=finalize_workflow(auto_wrkflw2,best2)

DT_fit=fit(DT_final,train)

predict(DT_fit, new_data = train, type = "class") %>%
  bind_cols(train) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
```

Decision Tree fit and accuracy

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.958
```

```
RF_final=finalize_workflow(auto_wrkflw3,best3)

RF_fit=fit(RF_final,train)

predict(RF_fit, new_data = train, type = "class") %>%
  bind_cols(train) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
```

Random Forest fit and accuracy

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.978
```

```
Boost_final=finalize_workflow(auto_wrkflw4,best4)

Boost_fit=fit(Boost_final,train)

predict(Boost_fit, new_data = train, type = "class") %>%
  bind_cols(train) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
```

Boosting fit and accuracy

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

All models performed really well all showing accuracy higher than 90%. The model that received the highest score is the Boosting Tree. I remember in class we talked about how Tree based models always produce the best results and always ends up as the optimal one to use. This is proof that the theory we discuss in class is true.

Fitting the best models into the testing set

Now that I have fitted our model onto our cross validation data set and chose the model with the best parameters. I fitted the best models onto the the training set and now we arrive at training our data set

```
log_final_fit=fit(log_wkflow,test)
predict(log_final_fit, new_data = test, type = "class") %>%
  bind_cols(test) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.913
```

```
DT_final_fit=fit(DT_final,test)

predict(DT_final_fit, new_data = test, type = "class") %>%
  bind_cols(test) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.922
```

```
RF_final_fit=fit(RF_final,test)

predict(RF_final_fit, new_data = test, type = "class") %>%
  bind_cols(test) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.974
```

```
Boost_final_fit=fit(Boost_final,test)

predict(Boost_final_fit, new_data = test, type = "class") %>%
  bind_cols(test) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
```

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

All models achieved high accuracys . The best model is still the Boosted tree model. The performance of the boosted model is 100% accuracy. Boosted model is the best model to use for this classification project.

Best model overall The model that fit our model the best is the boosting tree with the parameter of tree set at 452. It was able to achieve an accuracy score of 1. This image classification problem is best suited to be used in a boosting tree model.The quality of the predictions is very high .

Conclusion

The goal I set for this project can open the door for doctors in understanding breast cancer. I was able to achieve a really high accuracy score for all my models of choice .

The model that fitted the data set the best is the Boosted Tree model . The model achieve 100% for both the training and testing data set.The models hyper parameter is 452 trees.

For this project that deals with classification I used a logistic regression model , decision tree model , random forest model , and a boosting tree model. I chose these models because in the discussion we have in lecture Tree based model always works the best for classification. This is evidence that it is true . All models were able to achieve an accuracy score over 90% when we fit it onto our training and testing data set. The K fold cross validation with the tuning helped us find the best optimal model that fitted our data the best.

I was surprised with my models performance because I didnt think any of my models would achieve such high accuracy score . When I started this project my goal was to hit at least 80% accuracy score and all my models beat that threshold . It surprised me how well the model were able to fit my data even when I reduce the dimension .

The next step would be the inferential part. Now that I created a model with such high accuracy score we are able find relationship between features.By finding trend and patterns that affect predictor variables will allows us to understand cancer better . Being able to understand the features relationship in response to the predictor variables will illustrate key points to look at .The key points can help point out early or on the rise cancer patient and save lives.

Overall , I enjoyed working on this project . It let me explored the health care side of data science . I have now done a project using industry data , research data , and no healthcare data . The project helped me add more stuff to my portfolio and have better judgement in creating these models.