# DH_Challenge_2

Dagmawe Haileslassie

05/09/2022

```
#knitr::opts_chunk$set(echo=FALSE, warning=FALSE, message=FALSE, fig.show="hide", results=FALSE)
knitr::opts_chunk$set(warning=FALSE, message=FALSE)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidymodels)
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip
```

```
## -- Attaching packages --------------------------------------- tidymodels 0.1.4 --
```

```
## v broom        0.7.12     v rsample      0.1.1
## v dials        0.1.0      v tune         0.1.6
## v infer        1.0.0      v workflows    0.2.4
## v modeldata    0.1.1      v workflowsets 0.1.0
## v parsnip      0.2.1      v yardstick    0.0.9
## v recipes      0.1.17
```

```
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## x tune::tune()      masks parsnip::tune()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
tidymodels_prefer()
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-3
```

```
library(vip)
library(mlbench)

library(rpart)

library(rpart.plot)

library(keras)
library(dplyr)
library(magrittr)
library(neuralnet)
```

## What is my Data and What do I plan to do?

Early diagnosis of cancer is critical for its successful treatment. Ultimatley, there is a high demand for accurate and cheap diagnostic methods. In this project I wanted to explore the applicability of 1. Decision tree machine learning techniques - A random forest model - Normal Decision Tree 2. Neural networks and 3. Basic Logistic regression for breast cancer diagnosis using digitized images of tissue samples. I obtained the data from UC Irvine Machine Learning Repository ("Breast Cancer Wisconsin data set" created by William H. Wolberg, W. Nick Street, and Olvi L. Mangasarian).

## Why I picked the Data.

The most accurate traditional method for a diagnosis when it comes to breast cancer is a rather invasive technique, called breast biopsy, where a small piece of breast tissue is surgically removed, and then the tissue sample has to be examined by a specialist. However, a much less invasive technique can be used, where the samples can be obtained by a minimally invasive fine needle aspirate method.

As seen in our data the sample obtained by this method can be easily digitized and used for computationally based diagnosis. This can ultimatley increase processing speed and on a big scale can make the process significantly cheaper.

## Deep dive into Cancer.tbl

In the code chunk below we import the dataset data.csv and select out ID and X which are colomns I found unuseful when building any of the aforementioned models. When taking a close look we can see that all the colomns (variables) are numerical except for our classifiable data that shows wheather or not the diagnosis is benign or malignant.

```
Cancer <- read.csv("~/Mscs 341 S22/Submit Section A/Project_2/Data/data.csv", header = TRUE)
Cancer.tbl <- Cancer%>%
  select(-id, -X)
str(Cancer.tbl)
```

```
## 'data.frame':    569 obs. of  31 variables:
```

```
##  $ diagnosis              : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ 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 ...
```

```
dim(Cancer.tbl)
```

```
## [1] 569  31
```

And as usual we will set-up or training/testing dataset:

```
cancer.split <- initial_split(Cancer.tbl, prop=0.8)
cancer.train.tbl <- training(cancer.split)

cancer.test.tbl <- testing(cancer.split)
```

## Lasso Classification

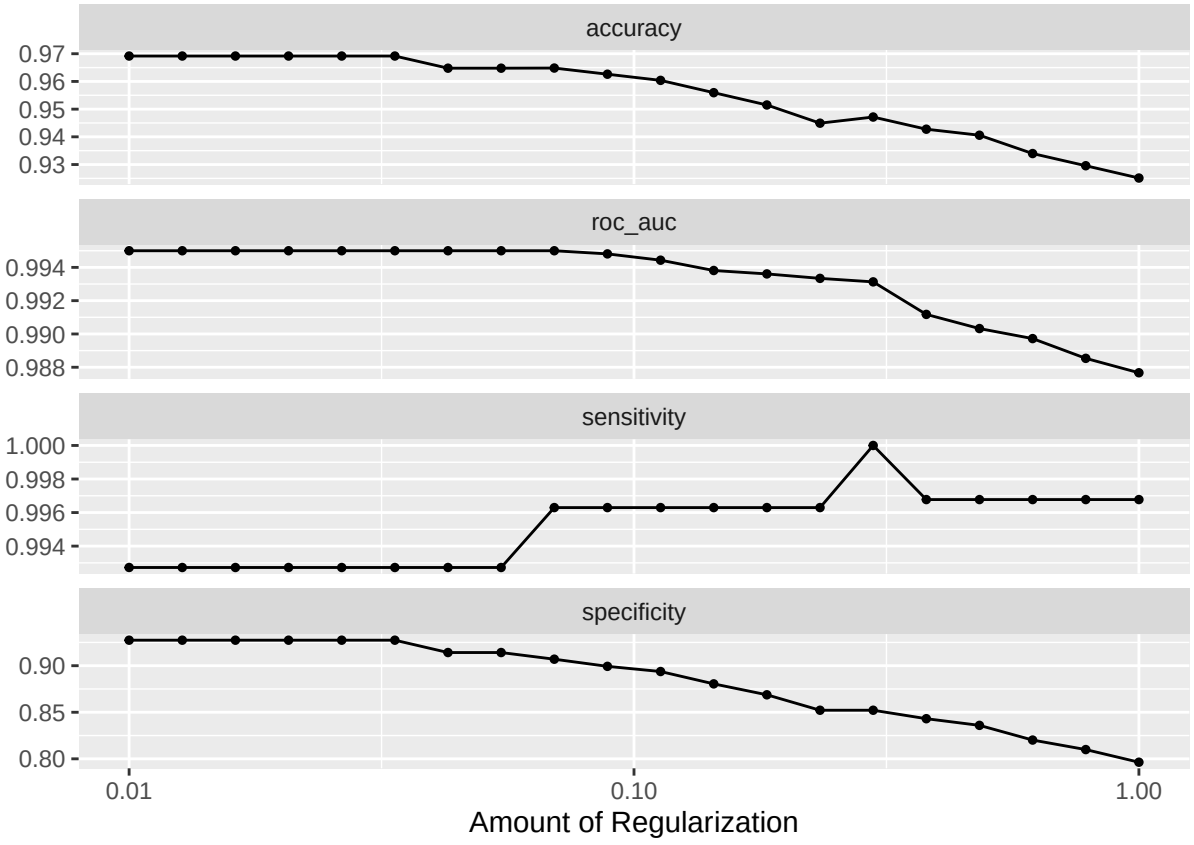
The first thing we're going to do is build a Lasso classification model that will allow us to predict the diagnosis based on the different variables. Furthermore I would like to identify a small number of the important features (variables) to use when building the tree model and Neural Net later in this project. The reason why I chose a lasso regression is because of LASSO's ability to identify a small subset of variables.

```
cancer.model <-
  logistic_reg(mixture = 0, penalty=tune()) %>%
  set_mode("classification") %>%
  set_engine("glmnet")
```

```
cancer.recipe <-
  recipe(formula = diagnosis~ ., data = cancer.train.tbl) %>%
  step_normalize(all_predictors())

cancer.wf <- workflow() %>%
  add_recipe(cancer.recipe) %>%
  add_model(cancer.model)
```

Next we are going to create a grid between -2 and 0 on the log-scale with 20 values. I will then use tune_grid() and plot the effect of the penalty in the classification accuracy of the LASSO model.

```
set.seed(1234)
cancer.folds<- vfold_cv(cancer.train.tbl, v = 10)

penalty.grid <-
  grid_regular(penalty(range = c(-2, 0)), levels = 20)

tune.res.lasso <- tune_grid(
  cancer.wf,
  resamples = cancer.folds,
  grid = penalty.grid,
  metrics = metric_set(accuracy, roc_auc, sensitivity, specificity))
autoplot(tune.res.lasso)
```




When looking at the autoplot we can see that the accuracy of the model decreases after penality = 0.10.

```
show_best(tune.res.lasso, metric = "accuracy")
```

```
## # A tibble: 5 x 7
##   penalty .metric  .estimator  mean     n std_err .config
##     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
## 1  0.01   accuracy binary     0.969    10 0.00590 Preprocessor1_Model01
## 2  0.0127 accuracy binary     0.969    10 0.00590 Preprocessor1_Model02
## 3  0.0162 accuracy binary     0.969    10 0.00590 Preprocessor1_Model03
## 4  0.0207 accuracy binary     0.969    10 0.00590 Preprocessor1_Model04
## 5  0.0264 accuracy binary     0.969    10 0.00590 Preprocessor1_Model05
```

```
(best.penalty <- select_by_one_std_err(tune.res.lasso,
                                       metric = "accuracy",
                                       desc(penalty)))
```

```
## # A tibble: 1 x 9
##   penalty .metric  .estimator  mean     n std_err .config           .best .bound
##     <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <fct>             <dbl>  <dbl>
## 1  0.0695 accuracy binary     0.965    10 0.00817 Preprocessor1_Mo~ 0.969  0.963
```

```
cancer.final.wf <- finalize_workflow(cancer.wf, best.penalty)
cancer.final.fit <- fit(cancer.final.wf, data = cancer.train.tbl)

augment(cancer.final.fit, new_data = cancer.test.tbl) %>%
  conf_mat(truth = diagnosis, estimate = .pred_class)
```

```
##           Truth
## Prediction  B  M
##          B 72  1
##          M  0 41
```

```
augment(cancer.final.fit, new_data = cancer.test.tbl) %>%
  accuracy(truth = diagnosis, estimate = .pred_class)
```

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

## Lasso Classification Result Analysis

When looking at the confusion matrix for patients with benign cancer our model predicted a 100% of the test dataset, in our malignant cancer cell there were 3 misdiagnosed patients in our model.

The accuracy of this model is about 97.4%

Let's now look at a table of estimates.

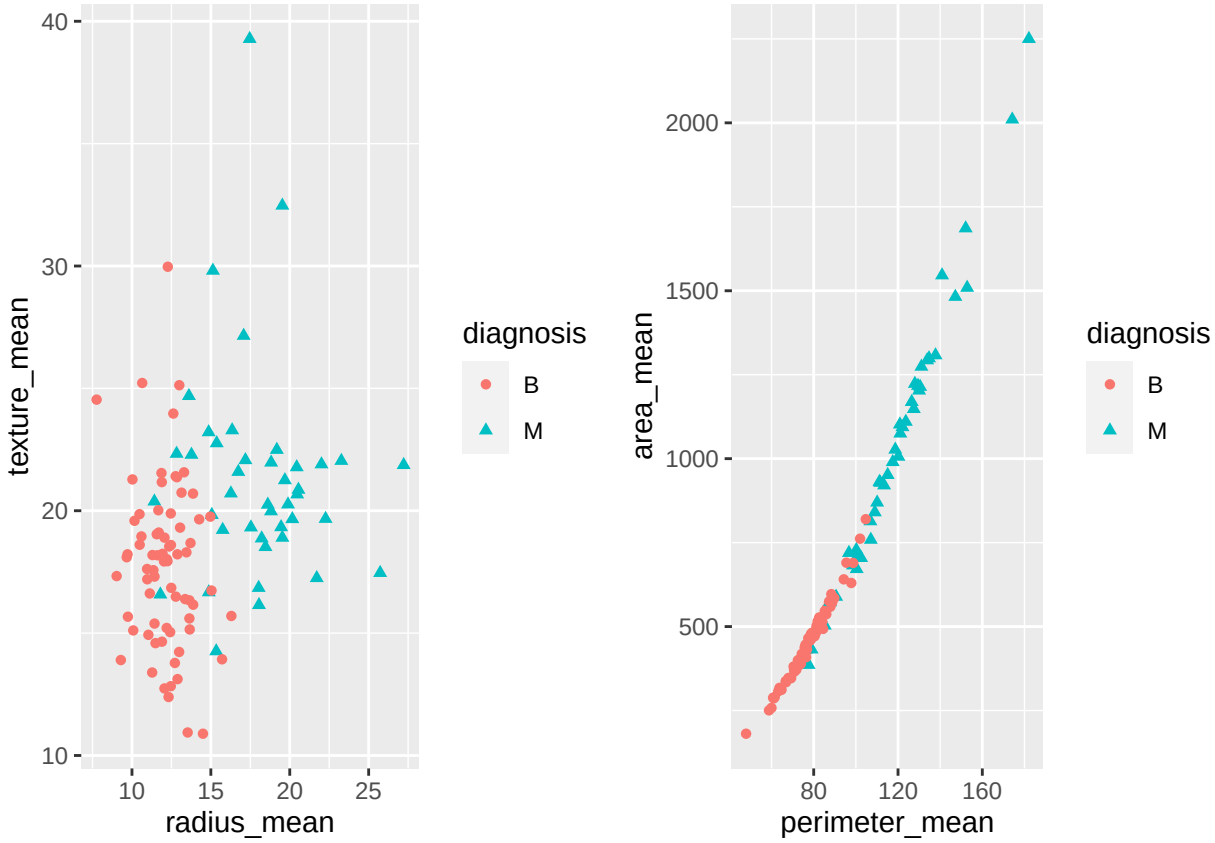
```
tidy(cancer.final.fit) %>% filter(estimate!=0)
```

```
## # A tibble: 31 x 3
##    term                estimate penalty
##    <chr>                  <dbl>   <dbl>
##  1 (Intercept)          -0.585   0.0695
##  2 radius_mean           0.284   0.0695
##  3 texture_mean          0.260   0.0695
```

```
##  4 perimeter_mean        0.278   0.0695
##  5 area_mean             0.269   0.0695
##  6 smoothness_mean       0.120   0.0695
##  7 compactness_mean      0.0653  0.0695
##  8 concavity_mean        0.241   0.0695
##  9 concave.points_mean   0.294   0.0695
## 10 symmetry_mean         0.0491  0.0695
## # ... with 21 more rows
```

From the table of estimates and terms we can see that the variables that have the significant effects on the model area are radius_mean, texture_mean, Perimeter_mean, area_mean and Smoothness_mean.

```
# library(gridExtra)
gg1 <- ggplot (cancer.test.tbl, aes(x=radius_mean, y=texture_mean, color=diagnosis, shape=diagnosis))+
    geom_point()
gg2 <- ggplot (cancer.test.tbl, aes(x=perimeter_mean, y=area_mean, color=diagnosis, shape=diagnosis))+
    geom_point()

grid.arrange(gg1,gg2,ncol=2)
```



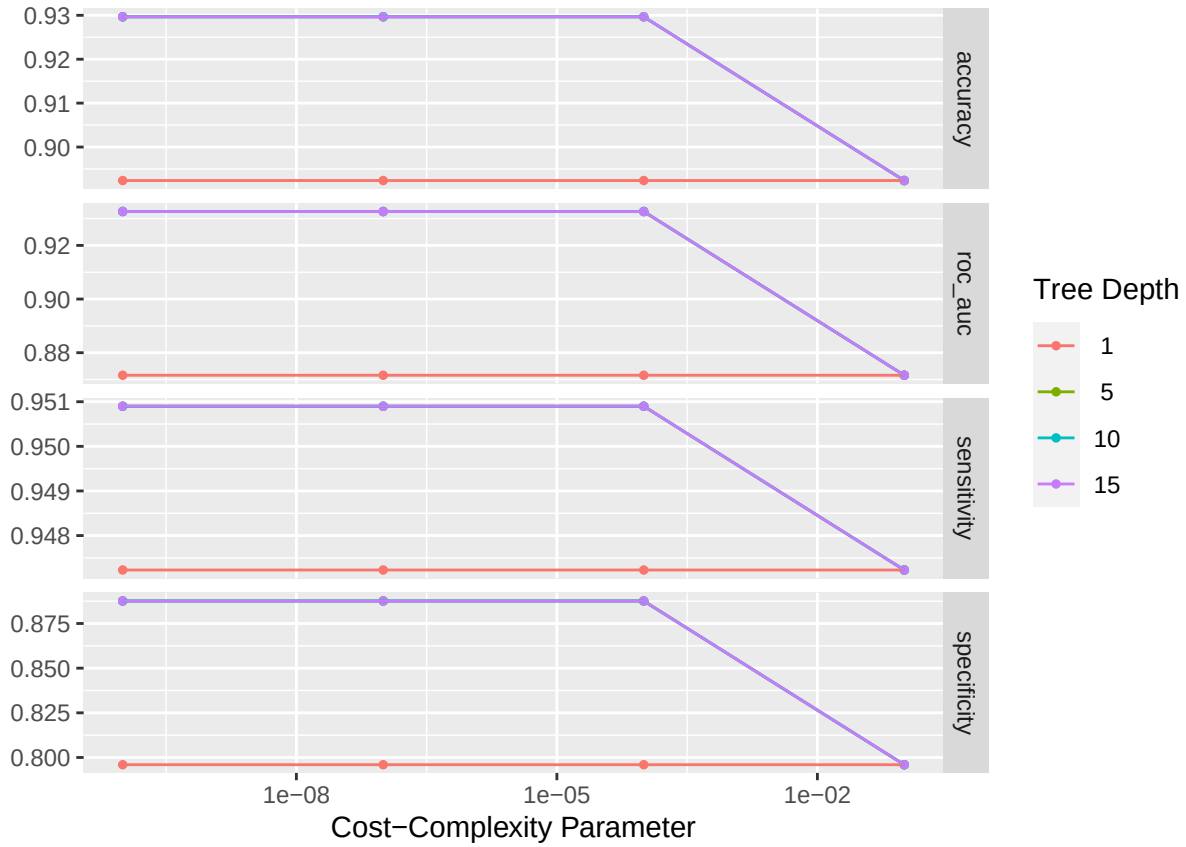

## Decisions trees

Decisions trees introduce a completely new idea for making predictions. The fundamental idea, as the name implies, is to use a **tree** as the means of making a decisions. The tree is built on a sequence of decisions based on the predictor variables.

```
  cancer.tree.model <-
    decision_tree(tree_depth = tune(), cost_complexity = tune()) %>%
    set_mode("classification") %>%
    set_engine("rpart")

  cancer.tree.recipe <- recipe(diagnosis ~ .,
                  data=cancer.train.tbl)

  cancer.tree.wflow <- workflow() %>%
      add_recipe(cancer.tree.recipe) %>%
      add_model(cancer.tree.model)

  # Create the cross-validation dataset
  cancer.tree.folds <- vfold_cv(cancer.train.tbl, v = 10)

  #Set up the grid
  cancer.tree.grid <-
    grid_regular(cost_complexity(), tree_depth(), levels = 4)


  tune.res.tree <-
    tune_grid(
      cancer.tree.wflow,
      resamples = cancer.folds,
      grid = cancer.tree.grid,
      metrics = metric_set(accuracy, roc_auc, sensitivity, specificity))
  tune.res.tree
```

```
## # Tuning results
## # 10-fold cross-validation
## # A tibble: 10 x 4
##    splits           id     .metrics          .notes
##    <list>           <chr>  <list>            <list>
##  1 <split [409/46]> Fold01 <tibble [64 x 6]> <tibble [0 x 1]>
##  2 <split [409/46]> Fold02 <tibble [64 x 6]> <tibble [0 x 1]>
##  3 <split [409/46]> Fold03 <tibble [64 x 6]> <tibble [0 x 1]>
##  4 <split [409/46]> Fold04 <tibble [64 x 6]> <tibble [0 x 1]>
##  5 <split [409/46]> Fold05 <tibble [64 x 6]> <tibble [0 x 1]>
##  6 <split [410/45]> Fold06 <tibble [64 x 6]> <tibble [0 x 1]>
##  7 <split [410/45]> Fold07 <tibble [64 x 6]> <tibble [0 x 1]>
##  8 <split [410/45]> Fold08 <tibble [64 x 6]> <tibble [0 x 1]>
##  9 <split [410/45]> Fold09 <tibble [64 x 6]> <tibble [0 x 1]>
## 10 <split [410/45]> Fold10 <tibble [64 x 6]> <tibble [0 x 1]>
```

```
  autoplot(tune.res.tree)
```




```
  show_best(tune.res.tree, metric = "accuracy")
```

```
## # A tibble: 5 x 8
##   cost_complexity tree_depth .metric  .estimator  mean     n std_err .config
##             <dbl>      <int> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
## 1    0.0000000001          5 accuracy binary     0.930    10 0.00975 Preprocess~
## 2    0.0000001             5 accuracy binary     0.930    10 0.00975 Preprocess~
## 3    0.0001                5 accuracy binary     0.930    10 0.00975 Preprocess~
## 4    0.0000000001         10 accuracy binary     0.930    10 0.00975 Preprocess~
## 5    0.0000001            10 accuracy binary     0.930    10 0.00975 Preprocess~
```

```
  (best.penalty <- select_by_one_std_err(tune.res.tree,
                                         metric = "accuracy",
                                         -cost_complexity))
```

```
## # A tibble: 1 x 10
##   cost_complexity tree_depth .metric  .estimator  mean     n std_err .config
##             <dbl>      <int> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
## 1          0.0001          5 accuracy binary     0.930    10 0.00975 Preprocess~
## # ... with 2 more variables: .best <dbl>, .bound <dbl>
```

```
  cancer.final.wf <- finalize_workflow(cancer.tree.wflow,
                                       best.penalty)

  cancer.final.fit <- fit(cancer.final.wf,  cancer.train.tbl)

  cancer.final.rs <- last_fit(cancer.final.wf,
```

```
                           cancer.split)

  collect_metrics(cancer.final.rs)
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <fct>
## 1 accuracy binary         0.939 Preprocessor1_Model1
## 2 roc_auc  binary         0.928 Preprocessor1_Model1
```

```
  augment(cancer.final.fit, new_data = cancer.test.tbl) %>%
    conf_mat(truth = diagnosis, estimate = .pred_class)
```

```
##           Truth
## Prediction  B  M
##          B 68  3
##          M  4 39
```

```
  augment(cancer.final.fit, new_data = cancer.test.tbl)%>%
    accuracy(truth = diagnosis, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.939
```

Let's see the variable importance of our decision tree

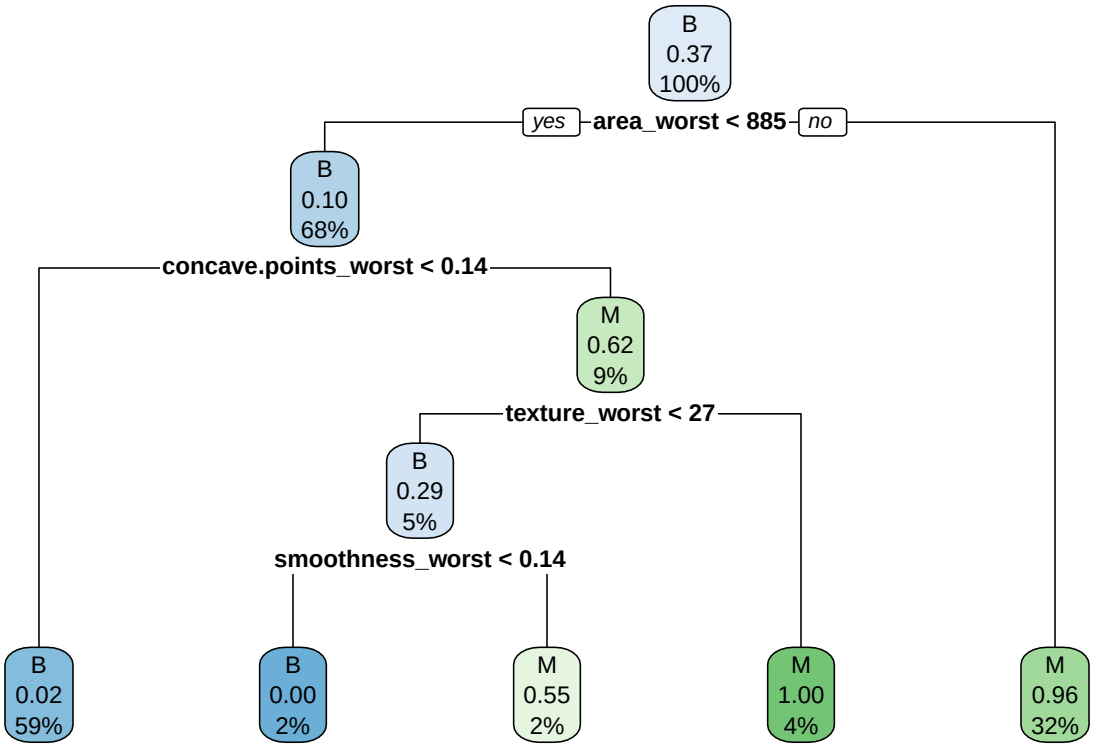
```
imp.tbl.dt <- cancer.final.fit %>%
  extract_fit_engine() %>%
  vip::vi()
imp.tbl.dt
```

```
## # A tibble: 19 x 2
##    Variable             Importance
##    <chr>                     <dbl>
##  1 area_worst               147.
##  2 radius_worst             144.
##  3 perimeter_worst          135.
##  4 area_mean                129.
##  5 perimeter_mean           128.
##  6 radius_mean              128.
##  7 concave.points_worst      28.4
##  8 concavity_worst           11.1
##  9 concave.points_mean       10.5
## 10 texture_worst              9.89
## 11 compactness_worst          8.61
## 12 texture_mean               7.69
## 13 concavity_mean             7.38
## 14 smoothness_worst           6.96
## 15 compactness_mean           5.53
## 16 symmetry_worst             4.95
## 17 texture_se                 4.40
## 18 smoothness_mean            2.18
## 19 perimeter_se               1.87
```

Let's visualize our model on our training dataset using `parttree` and let's look at how the final model looks

as a tree.

```
cancer.final.fit %>%
  extract_fit_engine() %>%
  rpart.plot(roundint=FALSE)
```




## Random Forest Model

```
ranger_recipe <-
  recipe(formula = diagnosis ~ ., data = cancer.train.tbl)

ranger_spec <-
  rand_forest(trees = 100, mtry=28) %>%
  set_mode("classification") %>%
  set_engine("ranger",importance = "impurity")

ranger_workflow <-
  workflow() %>%
  add_recipe(ranger_recipe) %>%
  add_model(ranger_spec)

cancer.forest.model <- fit(ranger_workflow, cancer.train.tbl)
```

```
augment(cancer.forest.model, cancer.test.tbl) %>%
  accuracy(truth=diagnosis, estimate= .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.982
```

```
augment(cancer.forest.model, cancer.test.tbl) %>%
  conf_mat(truth=diagnosis, estimate= .pred_class)
```

```
##           Truth
## Prediction  B  M
##          B 71  1
##          M  1 41
```

```
imp.tbl.dt.forest <- cancer.forest.model %>%
  extract_fit_engine() %>%
  vip::vi()
imp.tbl.dt.forest
```

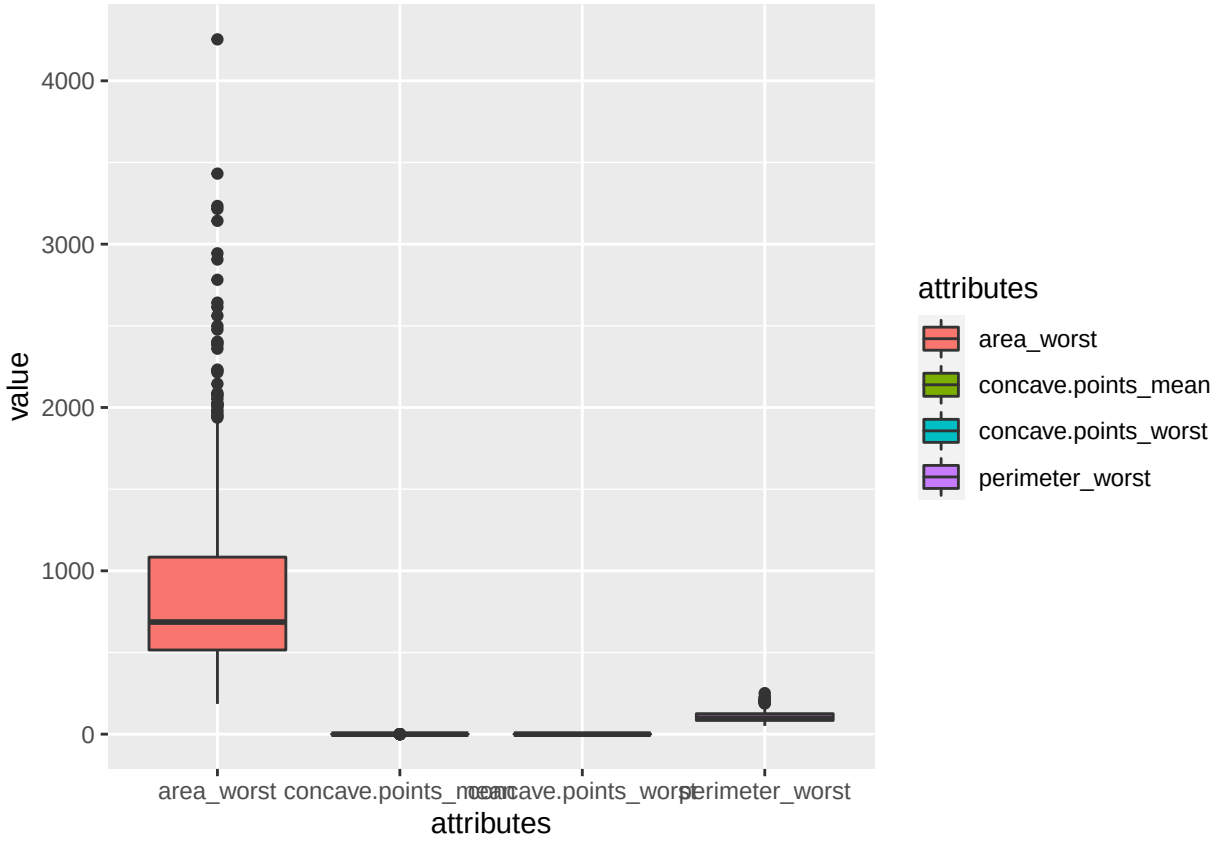
```
## # A tibble: 30 x 2
##    Variable             Importance
##    <chr>                     <dbl>
##  1 area_worst                51.9
##  2 perimeter_worst           42.0
##  3 concave.points_worst      36.9
##  4 concave.points_mean       30.5
##  5 radius_worst              17.0
##  6 texture_worst              4.62
##  7 concavity_worst            3.82
##  8 texture_mean               3.06
##  9 area_se                    2.85
## 10 smoothness_worst           2.56
## # ... with 20 more rows
```

# Neural Net

So before we get started with the Neural Network, I wanted to understand what it is. A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

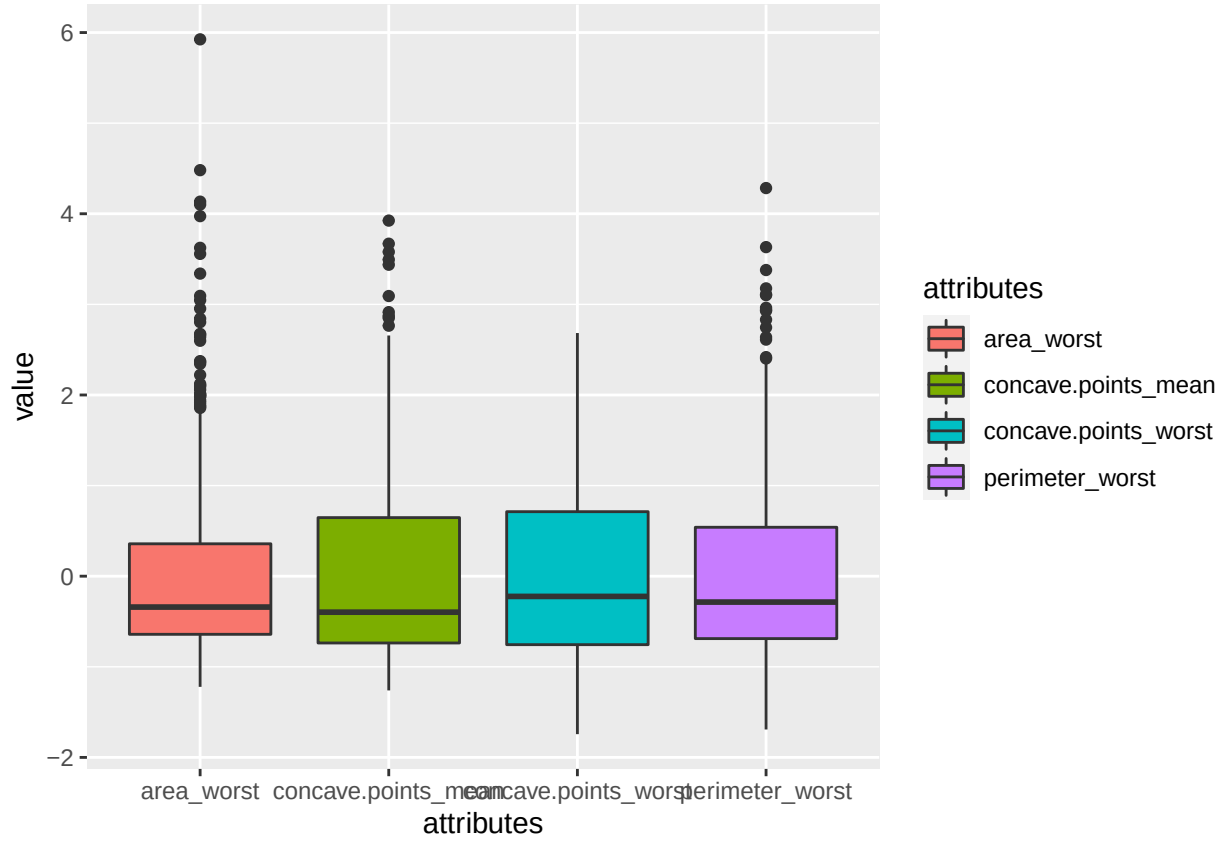
Let's first look at the distribution of benign and malignant in the cancer.tbl dataset to see if we need to do any preprocessing. I am going to draw a boxplot to see if the dataset needs to be scaled and if there are any outliers. To that end, let me create a function to draw boxplots.

```
Cancer.tbl.nn <- Cancer.tbl %>%
  select(concave.points_worst, perimeter_worst, area_worst, concave.points_mean, diagnosis)
draw_boxplot <- function(){
  Cancer.tbl.nn %>%
    pivot_longer(1:4, names_to="attributes")%>%
    ggplot(aes(attributes, value, fill=attributes)) +
      geom_boxplot()
}
draw_boxplot()
```




We can observe that the columns have different scales and the 'Sepal.Width' column has outliers. First, let us get rid of the outliers. I am going to use the squish method to remove the outliers. Here, note that I will not be removing the outlying data. Instead, I will only be setting the outlying rows of data to the maximum or minimum value.

```
Cancer.tbl.nn <- Cancer.tbl.nn%>%
  mutate(across(1:4, scale))
draw_boxplot()
```



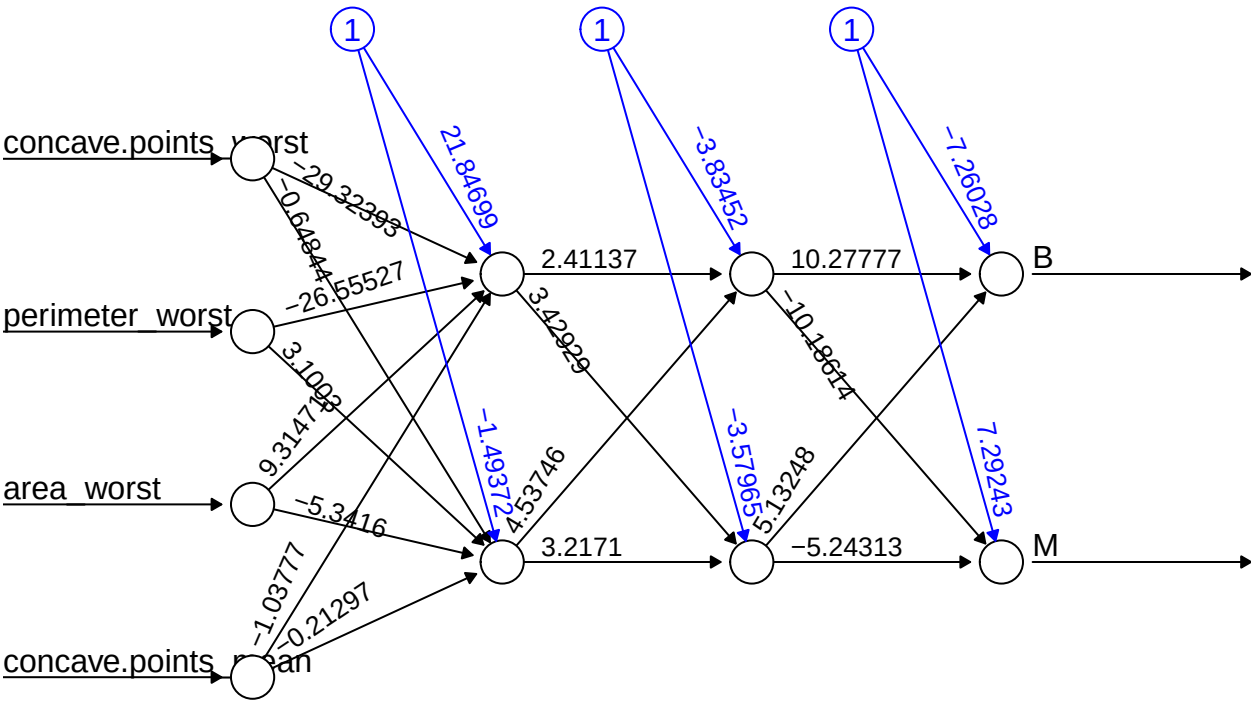

Let's divide up our new dataset into testing and training datasets.

```
cancer.split.nn <- initial_split(Cancer.tbl.nn, prop=0.8)
cancer.train.tbl.nn <- training(cancer.split.nn)
cancer.test.tbl.nn <- testing(cancer.split.nn)
```

To create a neural network, I am going to use the neuralnet package. I will be using the default settings and will be using two hidden layers with two neurons on each. By default, neuralnet uses the logistic function as the activation function.

```
nn=neuralnet(diagnosis ~.,
             data=cancer.train.tbl.nn, hidden=c(2,2), linear.output = FALSE)
```

```
plot(nn, rep = 'best')
```




Error: 13.916399  Steps: 2666

```
predict <- function(data){
  prediction <- data.frame(neuralnet::compute(nn,
                                              data.frame(data[,-5]))$net.result)
  labels <- c("B", "M")
  prediction_label <- data.frame(max.col(prediction)) %>%
    mutate(prediction=labels[max.col.prediction.]) %>%
    select(2) %>%
    unlist()
  table(data$diagnosis, prediction_label)
}
```

```
predict(cancer.test.tbl.nn)
```

```
##    prediction_label
##       B  M
##   B 75  2
##   M  2 35
```

## Summary of the accuracy of the models

Let's summarize what we did so far and finally analyse the accuracy of these models. First we built a lasso classification model, the accuracy of this model was a 98% the most important variables when running this model are radius_mean, texture_mean, Perimeter_mean, area_mean and Smoothness_mean. Next we moved on to a decision tree and the accuracy was 93.9% the most important variables when running this model were concave.points_worst, perimeter_worst, area_worst, concave.points_mean. Lastly we tried to

do a neural net with the above 4 important variables we got an accuracy of (96%). This may have been caused by the number of variables being significantly smaller but we can see an improve in the accuracy than the decision tree.