Final Project

Tamar Saad 207256991 & Or Arbel 207016098 & Rachel Weinberger 208812628

31/07/2022

In this report we explored a breast cancer dataset. This dataset contains data that was extracted from pictures of nuclei in breast mass, and 10 different features were taken: size, texture, etc. Since every picture contains many nuclei, each feature has 3 values- mean, worst and standard error, making total of 30 features. In this report we will go through data processing, feature selection and machine learning tools. Due to limitation of the number of pages, we only include here our final decisions of the options we used- normalization method, ML algorithms choices etc.

Upload and examine the data

```r
data<-read.csv("data.csv")
data<-subset(data, select=-X)
str(data)
```

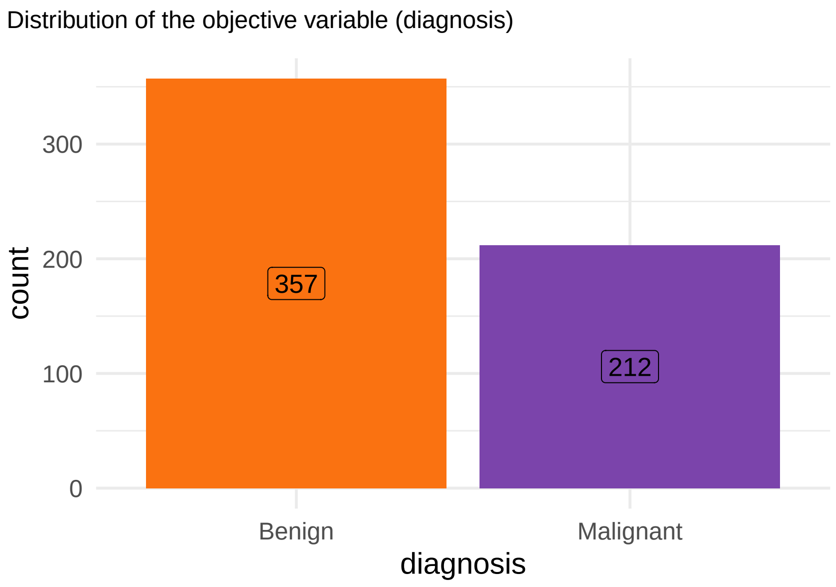
```
## 'data.frame':    569 obs. of  32 variables:
##  $ id                     : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844
##  $ diagnosis              : chr  "M" "M" "M" "M" ...
##  $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
##  $ texture_mean           : 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 ...
```

In this dataset we have 30 different features of 569 samples of breast cancer. We actually have 32 columns, but one of them is the ID of the sample and one is the diagnosis, which is the classification and therefore are not considered features.

```
## [1] "number of NAs in dataset: 0"
```

We don't have NAs at all, so there are no missing values.

Lets look at the classification distribution:



The distribution of the objective variable is not 1:1 but biased. We decided to continue normally, and apply changes only if we'll face problems in the future.

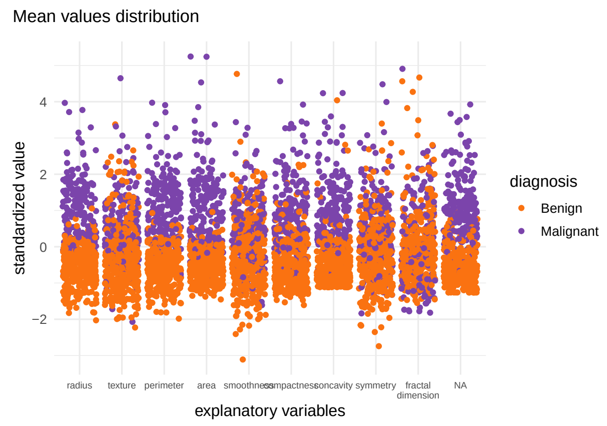
Checking the distribution of the features:

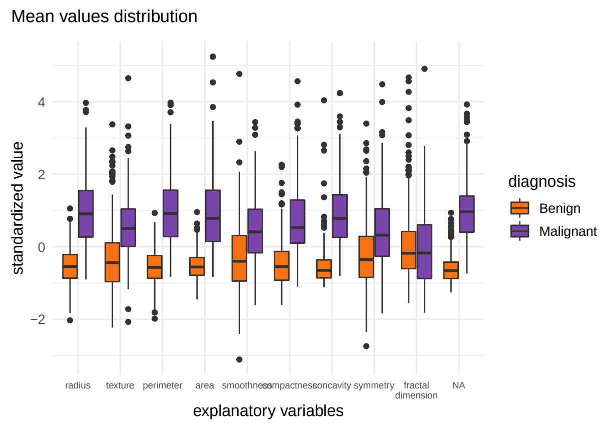
We divided the features into 3 different groups:

mean values, standard error values and worst values.

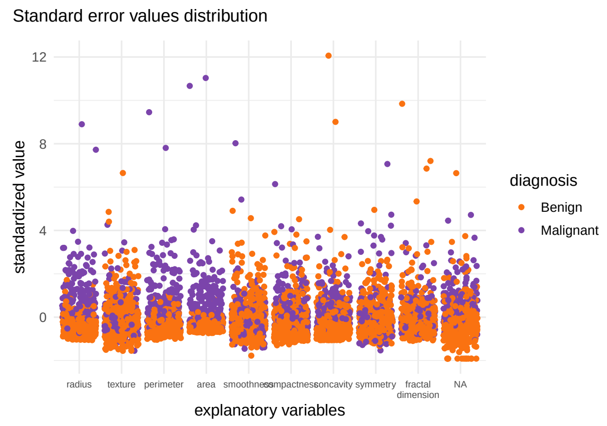
We wanted to check if the distribution's differences between malignant and benign diagnosis.

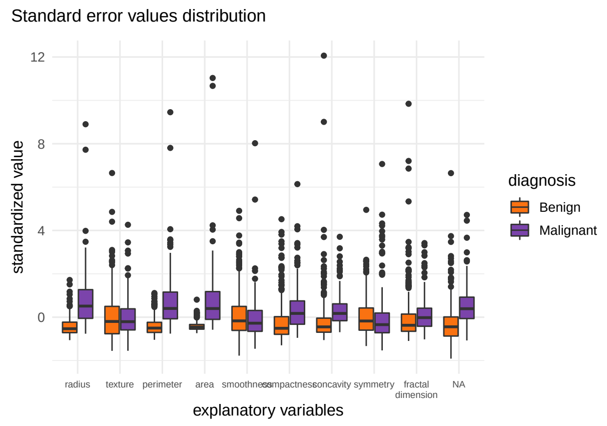
Visualize the results for mean values, with jitter plot and box plot



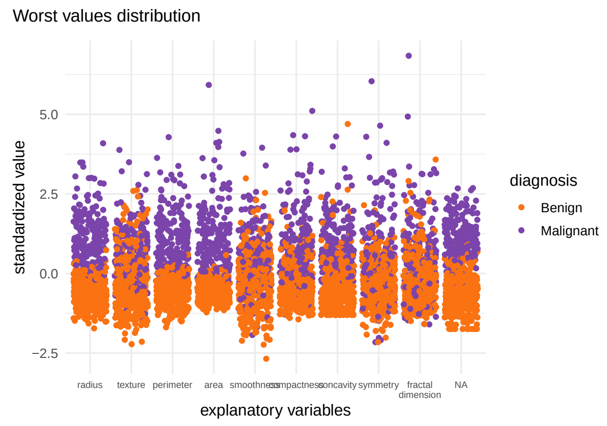


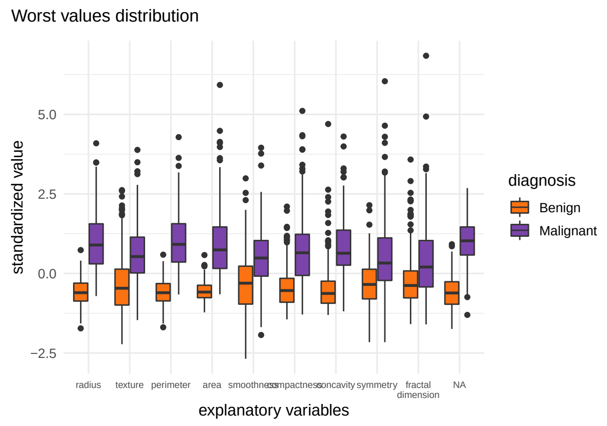
Visualize the results for standard error values





Visualize the results for worst values



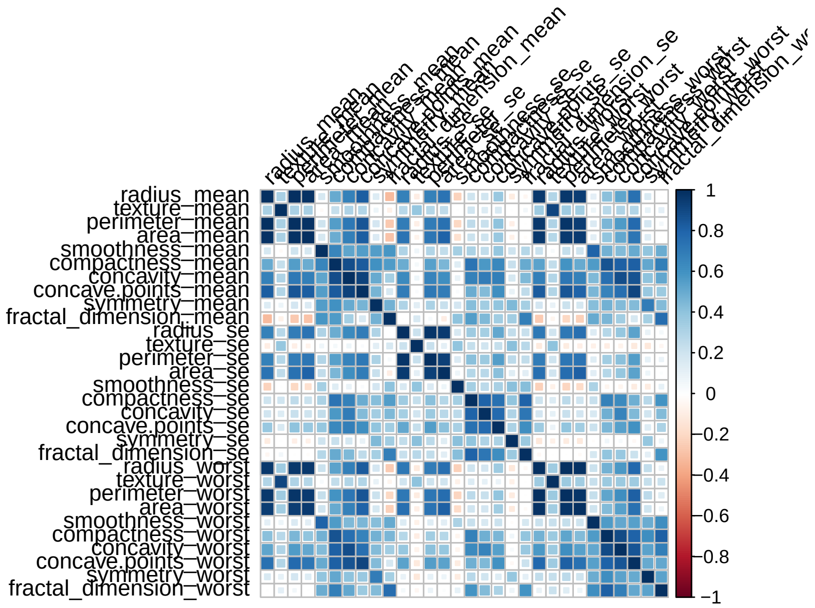


We can see for all the groups that not all features has the same distribution of malignant and benign samples. It means that there are differences between the features of the groups, so maybe ML tools could help us to classify the samples.

Next, we wanted to see if there are correlations between the features.

Correlation matrix:

```
cor(clean_data[,-(ncol(clean_data))]) %>%
  corrplot(method = "square", tl.col = "black", tl.srt = 45,
         sig.level = 0.05)
```



We can see that there are correlations mostly in the areas of features from the same type:

Worst values with worst values, and mean values with mean values. The big dark squares make us suspect that perhaps we have multicollinearity, which can deflect the ML algorithms.

In order to check if we indeed have multicolliearirty, we used VIF.

```
## [1] "the VIF value of radius mean is: 3806.11529639797"
```

```
## [1] "the VIF value of texture mean is: 11.8840480563618"
```

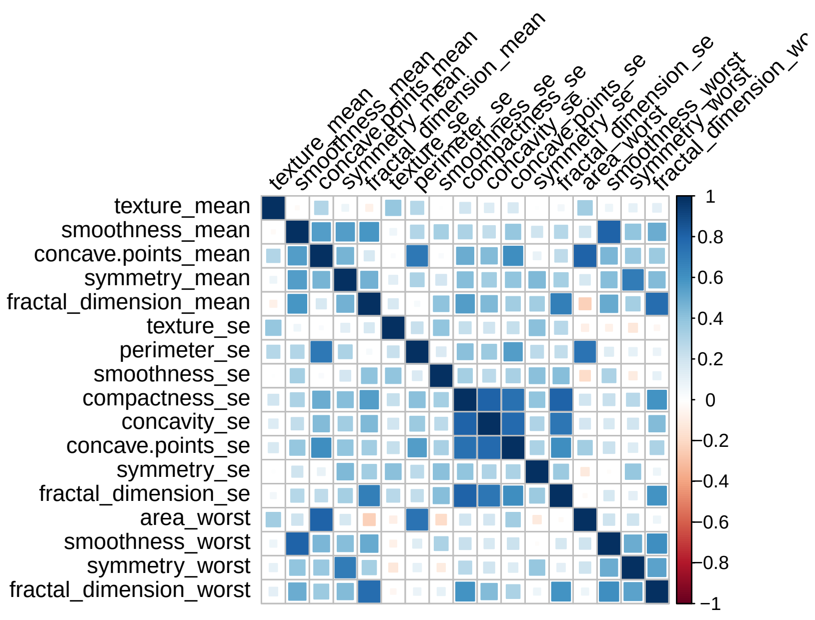
As we can see, the VIF scores are very high:

If the largest VIF value is 5 or more, multicollinearity exists. If it is 10 or more, multicollinearity is evaluated as serious. Therefore, we wanted to remove the correlated values.

We set the VIF threshold to 10, since many of the features were removed with this threshold already, and we didn't want to lose too much data. We decided to go with this threshold, and if we faced problems with classification in the future- change it respectively. After selecting the remaining features we created the correlation matrix again to check the results.

```
# Clearing columns with high multicollinearity
data_custom <- vif_func(X, thresh=10, trace=T)
```

```
## [1] "Number of remaining features: 17"
```



As we can see, there is a significantly smaller number of correlated features, as wanted.

Normalization:

We normalized by min-max and by Z-score, and wanted to check which one gave better results.

Normalize the data by MIN/MAX:

```
# create the min/max normalization function:
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x)))
}
#normalize the data without the classification column
minmax_data <- as.data.frame(lapply(X_2, normalize))
str(minmax_data)
```

```
## 'data.frame':    569 obs. of  17 variables:
##  $ texture_mean           : num  0.0227 0.2726 0.3903 0.3608 0.1566 ...
##  $ smoothness_mean        : num  0.594 0.29 0.514 0.811 0.43 ...
##  $ concave.points_mean    : num  0.731 0.349 0.636 0.523 0.518 ...
##  $ symmetry_mean          : num  0.686 0.38 0.51 0.776 0.378 ...
##  $ fractal_dimension_mean : num  0.606 0.141 0.211 1 0.187 ...
##  $ texture_se             : num  0.1205 0.0826 0.0943 0.1759 0.0931 ...
##  $ perimeter_se           : num  0.369 0.124 0.18 0.127 0.221 ...
##  $ smoothness_se          : num  0.159 0.119 0.151 0.251 0.332 ...
##  $ compactness_se         : num  0.3514 0.0813 0.284 0.5432 0.1679 ...
##  $ concavity_se           : num  0.1357 0.047 0.0968 0.143 0.1436 ...
##  $ concave.points_se      : num  0.301 0.254 0.39 0.354 0.357 ...
##  $ symmetry_se            : num  0.3116 0.0845 0.2057 0.7281 0.1362 ...
##  $ fractal_dimension_se   : num  0.183 0.0911 0.127 0.2872 0.1458 ...
##  $ area_worst             : num  0.451 0.435 0.375 0.094 0.342 ...
##  $ smoothness_worst       : num  0.601 0.348 0.484 0.915 0.437 ...
##  $ symmetry_worst         : num  0.598 0.234 0.404 1 0.158 ...
##  $ fractal_dimension_worst: num  0.419 0.223 0.213 0.774 0.143 ...
```

Normalize the data by z-score:

```
# z-score normalization
zscore_data <- as.data.frame(scale(X_2))
str(zscore_data)
```

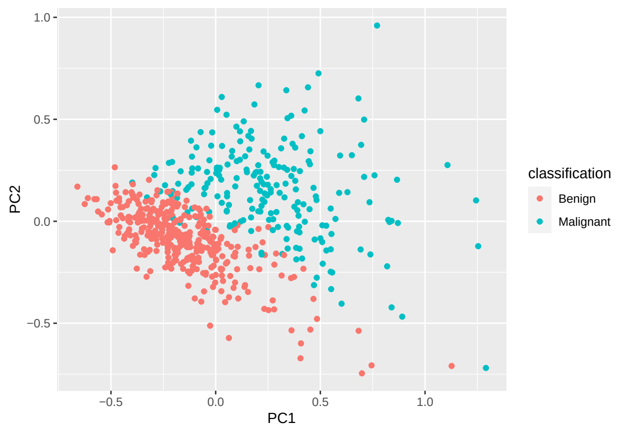
```
## 'data.frame':    569 obs. of  17 variables:
##  $ texture_mean           : num  -2.072 -0.353 0.456 0.254 -1.151 ...
##  $ smoothness_mean        : num  1.567 -0.826 0.941 3.281 0.28 ...
##  $ concave.points_mean    : num  2.53 0.548 2.035 1.45 1.427 ...
##  $ symmetry_mean          : num  2.21557 0.00139 0.93886 2.86486 -0.00955 ...
##  $ fractal_dimension_mean : num  2.254 -0.868 -0.398 4.907 -0.562 ...
##  $ texture_se             : num  -0.565 -0.875 -0.779 -0.11 -0.79 ...
##  $ perimeter_se           : num  2.831 0.263 0.85 0.286 1.272 ...
##  $ smoothness_se          : num  -0.214 -0.605 -0.297 0.689 1.482 ...
##  $ compactness_se         : num  1.3157 -0.6923 0.8143 2.7419 -0.0485 ...
##  $ concavity_se           : num  0.723 -0.44 0.213 0.819 0.828 ...
##  $ concave.points_se      : num  0.66 0.26 1.42 1.11 1.14 ...
##  $ symmetry_se            : num  1.148 -0.805 0.237 4.729 -0.361 ...
##  $ fractal_dimension_se   : num  0.9063 -0.0994 0.2933 2.0457 0.4989 ...
##  $ area_worst             : num  2 1.89 1.46 -0.55 1.22 ...
##  $ smoothness_worst       : num  1.307 -0.375 0.527 3.391 0.22 ...
##  $ symmetry_worst         : num  2.748 -0.244 1.151 6.041 -0.868 ...
##  $ fractal_dimension_worst: num  1.935 0.281 0.201 4.931 -0.397 ...
```

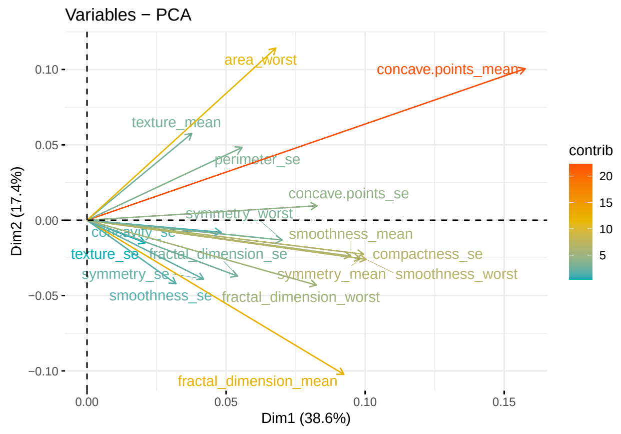
We tried to see which of the normalization gives us the best clustering results.

We performed dimension reduction with PCA, and applied loading arrows to see the weight of every feature an the principle components.

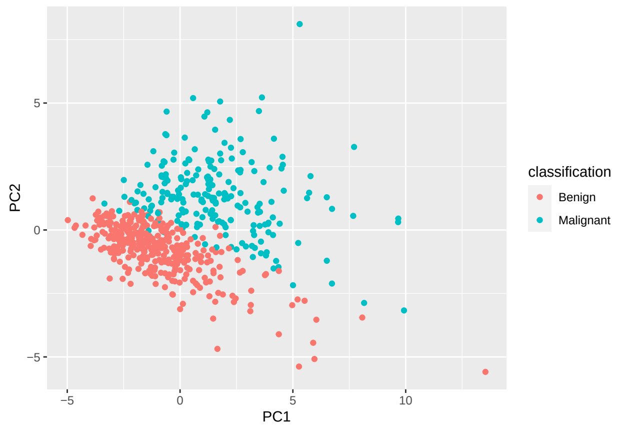
With Min-Max normalization:

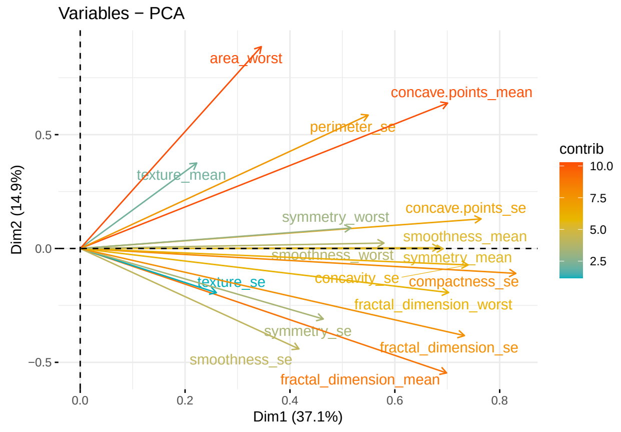
```
pca <- prcomp(minmax_data)


pca_to_show <- data.frame(
  PC1 = pca$x[, 1],
  PC2 = pca$x[, 2],
  classification = as.factor(clean_data$diagnosis)
)

ggplot(pca_to_show, aes(x = PC1, y = PC2, col = classification)) +
  geom_point()

fviz_pca_var(pca,
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE      # Avoid text overlapping
             )
```





With Z-Score normalization:





There is a good classification in both normalization methods, so we can assume that ML tools will be effective in analyzing the data.

When applying both z-score and minmax, we got similar results, but z-score gave us slightly better results. Therefore, we decided to continue with Z-score normalization.

Machine learning algorithms:

We chose to use the following algorithms:

- SVM
- Decision Tree
- Random Forest
- KNN

We chose to use 2 linear algorithms (SVM & KNN) because we saw good clustering in PCA, which is a linear algorithm for dimension reduction. We therefore concluded that linear models could help us in classification.

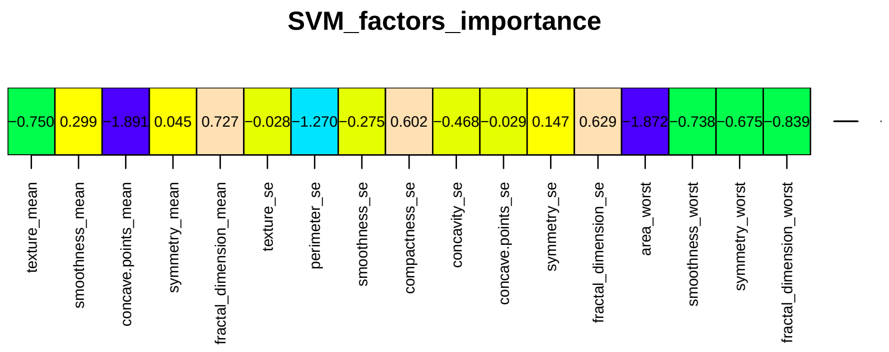
In addition, we wanted to apply non-linear algorithms as well, which is why we also included DT and RF.

SVM algorithm:

We used the default hyper parameters, since they gave us the best results. We split the data into train set (450 samples, 80%) and test set (119 samples, 20%)

```
## [1] "accuracy: 96.6386554621849"
```

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
## 
## 
## Total Observations in Table:  119 
## 
## 
##               | predicted type 
##   actual type |         B |         M | Row Total | 
## --------------|-----------|-----------|-----------|
##             B |        73 |         2 |        75 | 
##               |     0.613 |     0.017 |           | 
## --------------|-----------|-----------|-----------|
##             M |         2 |        42 |        44 | 
##               |     0.017 |     0.353 |           | 
## --------------|-----------|-----------|-----------|
## Column Total |        75 |        44 |       119 | 
## --------------|-----------|-----------|-----------|
## 
## 
```



As we can see, SVM gave us pretty good result on the test set.

The most important factors were area_worst and concave.points_mean.

Decision Tree (rpart) algorithm:

To prevent over fitting, we used the hyper parameters minsplit = 50, maxdepth = 3, and cp = 0.001.

We split the data into train set (450 samples, 80%) and test set (119 samples, 20%).

The outcomes are pretty good- TP=0.954.

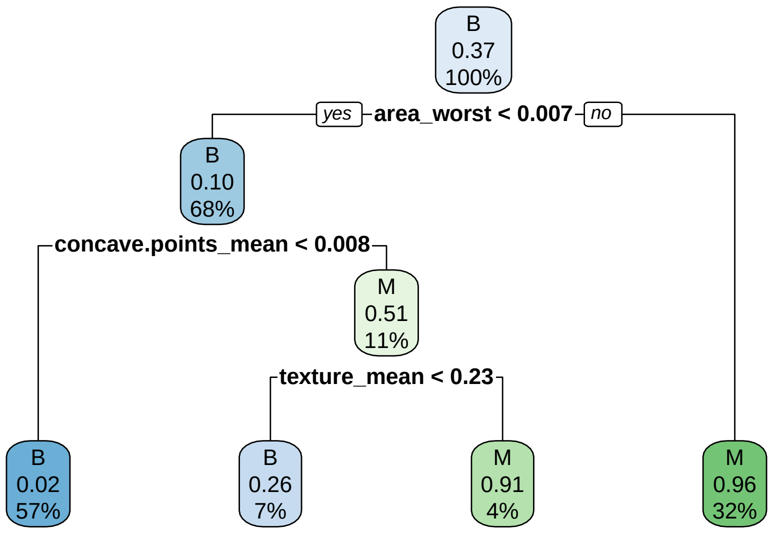
Each node shows:

- the predicted class,
- the predicted probability to be Malignant,
- the percentage of observations in the node.

```
set.seed(0)
diagnosis = data$diagnosis
cart <- rpart(diagnosis ~ .,
              data = zscore_data, method = "class",
              control=rpart.control(minsplit=50,
                                    maxdepth=3,
                                    cp=0.001))
p <- predict(cart, type="class")
```

```
## [1] "accuracy: 95.4305799648506"
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  569
##
##
##              | predicted type
##  actual type |         B |         M | Row Total |
## -------------|-----------|-----------|-----------|
##            B |       347 |        10 |       357 |
##              |     0.610 |     0.018 |           |
## -------------|-----------|-----------|-----------|
##            M |        16 |       196 |       212 |
##              |     0.028 |     0.344 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       363 |       206 |       569 |
## -------------|-----------|-----------|-----------|
##
##
```



RANDOM FOREST:

We split the data into train set (427 samples, 75%) and test set (142 samples, 25%)

We run the model with 3 different options: 500 trees, 900 trees, and 1500 trees. We chose those numbers because they are big enough to enable the models learning, but not too big to increase the running time and get to over fitting.

```
set.seed(0)
# the model #
## z score ##
rf_z_500 <- ranger(diagnosis ~.,
             data = train_z,
             mtry = 5, num.trees = 500, write.forest=TRUE, importance = "permutation")

rf_z_1500 <- ranger(diagnosis ~.,
             data = train_z,
             mtry = 5, num.trees = 1500, write.forest=TRUE, importance = "permutation")

rf_z_900 <- ranger(diagnosis ~.,
             data = train_z,
             mtry = 5, num.trees = 900, write.forest=TRUE, importance = "permutation")
```

Prediction of the model:

```
## [1] "Confusion matrix of Random Forest with 500 trees:"

## Confusion Matrix and Statistics
##
##              Reference
## Prediction   Benign Malignant
##    Benign        87         3
##   Malignant      3        50
##
##                Accuracy : 0.958
##                  95% CI : (0.9109, 0.9844)
##     No Information Rate : 0.6294
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9101
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9434
##             Specificity : 0.9667
##          Pos Pred Value : 0.9434
##          Neg Pred Value : 0.9667
##              Prevalence : 0.3706
##          Detection Rate : 0.3497
##    Detection Prevalence : 0.3706
##       Balanced Accuracy : 0.9550
##
##        'Positive' Class : Malignant
##
```

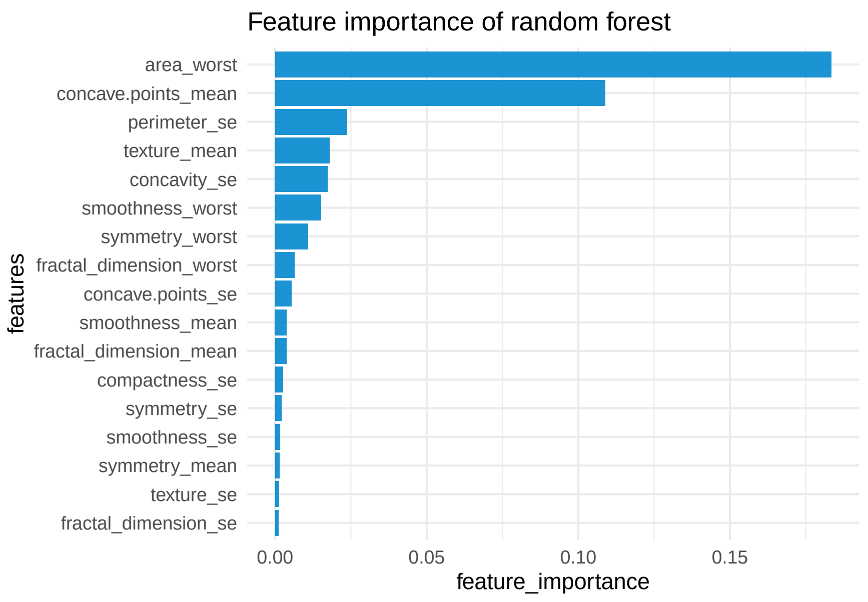
```
## [1] "Confusion matrix of Random Forest with 1500 trees:"
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign        87         2
##   Malignant      3        51
##
##                Accuracy : 0.965
##                  95% CI : (0.9203, 0.9886)
##     No Information Rate : 0.6294
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9253
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9623
##             Specificity : 0.9667
##          Pos Pred Value : 0.9444
##          Neg Pred Value : 0.9775
##              Prevalence : 0.3706
##          Detection Rate : 0.3566
##    Detection Prevalence : 0.3776
##       Balanced Accuracy : 0.9645
##
##        'Positive' Class : Malignant
##
```

```
## [1] "Confusion matrix of Random Forest with 900 trees:"
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign        87         2
##   Malignant      3        51
##
##                Accuracy : 0.965
##                  95% CI : (0.9203, 0.9886)
##     No Information Rate : 0.6294
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9253
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9623
##             Specificity : 0.9667
##          Pos Pred Value : 0.9444
##          Neg Pred Value : 0.9775
##              Prevalence : 0.3706
##          Detection Rate : 0.3566
##    Detection Prevalence : 0.3776
##       Balanced Accuracy : 0.9645
##
##        'Positive' Class : Malignant
##
```

We can see that we received similar results from RF with 900 and 1500 trees, and also that we didn't see a difference between min-max and z-score

Feature importance of random forest:

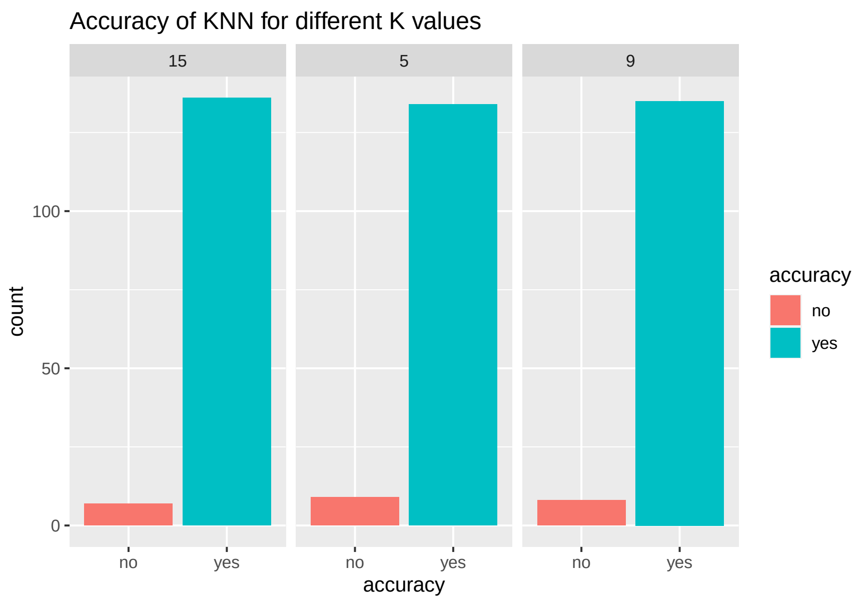


We can see here that the most important features for RF are similar to those in SVM, even though SVM is a linear model and RF is not. This repetitiveness increases the reliability of the models.

KNN ALGORITHM:

We split the data into train set (427 samples, 75%) and test set (142 samples, 25%)

While testing the ultimate K for the algorithm, we tried different values and presented here three of them. Choosing the values, we wanted the numbers to be odd and proportionate to the size of the data.



As we can see the best K is 15.

```
## [1] "Results of KNN for K = 15:"

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  143
##
##
##                              | results_zscore_test_15$.pred_class
## results_zscore_test_15$truth |    Benign | Malignant | Row Total |
## -----------------------------|-----------|-----------|-----------|
##                       Benign |        89 |         1 |        90 |
##                              |     0.989 |     0.011 |     0.629 |
##                              |     0.937 |     0.021 |           |
##                              |     0.622 |     0.007 |           |
## -----------------------------|-----------|-----------|-----------|
##                   Malignant |         6 |        47 |        53 |
##                             |     0.113 |     0.887 |     0.371 |
##                             |     0.063 |     0.979 |           |
##                             |     0.042 |     0.329 |           |
## ----------------------------|-----------|-----------|-----------|
##                Column Total |        95 |        48 |       143 |
##                             |     0.664 |     0.336 |           |
## ----------------------------|-----------|-----------|-----------|
##
##
```

Visualize predictions on test split:

*The axes are the two most importance features as we saw in other algorithms.

Conclusion:

All models gave us good results, and apparently our data is convenient to classify. All of the models presented similar results, with Random Forest and SVM that seemed a bit better than the rest. We would probably give up on KNN, since it's a lazy algorithm and therefore will be the slowest to use. Moreover, it gives all the features equal weights, which is an incorrect approach in many cases. We saw in SVM, RF, DT and PCA that the most important features were 'area worst' and 'concave points mean'. These findings can teach us a lot about our data, and should be considered while applying more algorithms or EDA. It is also interesting to examine further the effect of these features on the diagnosis.