

# STA7734 Project Analysis

Yuan Du

11/26/2019

## Read data

Check first 6 rows of Data, and missing values

\* 32 columns, first two columns are ID and target variable: Diagnosis (B,M).

\* No Missing value

```
## # A tibble: 6 x 32
##   ID      Diagnosis radius texture perimeter  area smoothness compactness
##   <chr> <chr>      <dbl>  <dbl>    <dbl> <dbl>    <dbl>      <dbl>
## 1 8423~ M          18.0   10.4    123.  1001    0.118    0.278
## 2 8425~ M          20.6   17.8    133.  1326    0.0847   0.0786
## 3 8430~ M          19.7   21.2    130   1203    0.110    0.160
## 4 8434~ M          11.4   20.4     77.6  386.    0.142    0.284
## 5 8435~ M          20.3   14.3    135.  1297    0.100    0.133
## 6 8437~ M          12.4   15.7     82.6  477.    0.128    0.17
## # ... with 24 more variables: concavity <dbl>, concave_points <dbl>,
## #   symmetry <dbl>, fractal_dimension <dbl>, radiusSE <dbl>,
## #   textureSE <dbl>, perimeterSE <dbl>, areaSE <dbl>, smoothnessSE <dbl>,
## #   compactnessSE <dbl>, concavitySE <dbl>, concave_pointsSE <dbl>,
## #   symmetrySE <dbl>, fractal_dimensionSE <dbl>, radiusW <dbl>,
## #   textureW <dbl>, perimeterW <dbl>, areaW <dbl>, smoothnessW <dbl>,
## #   compactnessW <dbl>, concavityW <dbl>, concave_pointsW <dbl>,
## #   symmetryW <dbl>, fractal_dimensionW <dbl>
```

##	ID	Diagnosis	radius
##	0	0	0
##	texture	perimeter	area
##	0	0	0
##	smoothness	compactness	concavity
##	0	0	0
##	concave_points	symmetry	fractal_dimension
##	0	0	0
##	radiusSE	textureSE	perimeterSE
##	0	0	0
##	areaSE	smoothnessSE	compactnessSE
##	0	0	0
##	concavitySE	concave_pointsSE	symmetrySE
##	0	0	0
##	fractal_dimensionSE	radiusW	textureW
##	0	0	0
##	perimeterW	areaW	smoothnessW
##	0	0	0
##	compactnessW	concavityW	concave_pointsW
##	0	0	0
##	symmetryW	fractal_dimensionW	
##	0	0	

Descriptive Statistics on numeric variables and target variable: Diagnosis

##	vars	n	mean	sd	median	trimmed	mad	min
## Diagnosis*	1	569	NaN	NA	NA	NaN	NA	Inf
## radius	2	569	14.13	3.52	13.37	13.82	2.82	6.98
## texture	3	569	19.29	4.30	18.84	19.04	4.17	9.71
## perimeter	4	569	91.97	24.30	86.24	89.74	18.84	43.79
## area	5	569	654.89	351.91	551.10	606.13	227.28	143.50
## smoothness	6	569	0.10	0.01	0.10	0.10	0.01	0.05
## compactness	7	569	0.10	0.05	0.09	0.10	0.05	0.02
## concavity	8	569	0.09	0.08	0.06	0.08	0.06	0.00
## concave_points	9	569	0.05	0.04	0.03	0.04	0.03	0.00
## symmetry	10	569	0.18	0.03	0.18	0.18	0.03	0.11
## fractal_dimension	11	569	0.06	0.01	0.06	0.06	0.01	0.05
## radiusSE	12	569	0.41	0.28	0.32	0.36	0.16	0.11
## textureSE	13	569	1.22	0.55	1.11	1.16	0.47	0.36
## perimeterSE	14	569	2.87	2.02	2.29	2.51	1.14	0.76
## areaSE	15	569	40.34	45.49	24.53	31.69	13.63	6.80
## smoothnessSE	16	569	0.01	0.00	0.01	0.01	0.00	0.00
## compactnessSE	17	569	0.03	0.02	0.02	0.02	0.01	0.00
## concavitySE	18	569	0.03	0.03	0.03	0.03	0.02	0.00
## concave_pointsSE	19	569	0.01	0.01	0.01	0.01	0.01	0.00
## symmetrySE	20	569	0.02	0.01	0.02	0.02	0.01	0.01
## fractal_dimensionSE	21	569	0.00	0.00	0.00	0.00	0.00	0.00
## radiusW	22	569	16.27	4.83	14.97	15.73	3.65	7.93
## textureW	23	569	25.68	6.15	25.41	25.39	6.42	12.02
## perimeterW	24	569	107.26	33.60	97.66	103.42	25.01	50.41
## areaW	25	569	880.58	569.36	686.50	788.02	319.65	185.20
## smoothnessW	26	569	0.13	0.02	0.13	0.13	0.02	0.07
## compactnessW	27	569	0.25	0.16	0.21	0.23	0.13	0.03
## concavityW	28	569	0.27	0.21	0.23	0.25	0.20	0.00
## concave_pointsW	29	569	0.11	0.07	0.10	0.11	0.07	0.00
## symmetryW	30	569	0.29	0.06	0.28	0.28	0.05	0.16
## fractal_dimensionW	31	569	0.08	0.02	0.08	0.08	0.01	0.06
##	max	range	skew	kurtosis	se			
## Diagnosis*	-Inf	-Inf	NA	NA	NA			
## radius	28.11	21.13	0.94	0.81	0.15			
## texture	39.28	29.57	0.65	0.73	0.18			
## perimeter	188.50	144.71	0.99	0.94	1.02			
## area	2501.00	2357.50	1.64	3.59	14.75			
## smoothness	0.16	0.11	0.45	0.82	0.00			
## compactness	0.35	0.33	1.18	1.61	0.00			
## concavity	0.43	0.43	1.39	1.95	0.00			
## concave_points	0.20	0.20	1.17	1.03	0.00			
## symmetry	0.30	0.20	0.72	1.25	0.00			
## fractal_dimension	0.10	0.05	1.30	2.95	0.00			
## radiusSE	2.87	2.76	3.07	17.45	0.01			
## textureSE	4.88	4.52	1.64	5.26	0.02			
## perimeterSE	21.98	21.22	3.43	21.12	0.08			
## areaSE	542.20	535.40	5.42	48.59	1.91			
## smoothnessSE	0.03	0.03	2.30	10.32	0.00			
## compactnessSE	0.14	0.13	1.89	5.02	0.00			
## concavitySE	0.40	0.40	5.08	48.24	0.00			
## concave_pointsSE	0.05	0.05	1.44	5.04	0.00			
## symmetrySE	0.08	0.07	2.18	7.78	0.00			

## fractal_dimensionSE	0.03	0.03	3.90	25.94	0.00
## radiusW	36.04	28.11	1.10	0.91	0.20
## textureW	49.54	37.52	0.50	0.20	0.26
## perimeterW	251.20	200.79	1.12	1.04	1.41
## areaW	4254.00	4068.80	1.85	4.32	23.87
## smoothnessW	0.22	0.15	0.41	0.49	0.00
## compactnessW	1.06	1.03	1.47	2.98	0.01
## concavityW	1.25	1.25	1.14	1.57	0.01
## concave_pointsW	0.29	0.29	0.49	-0.55	0.00
## symmetryW	0.66	0.51	1.43	4.37	0.00
## fractal_dimensionW	0.21	0.15	1.65	5.16	0.00

```
## B M
## 357 212
```

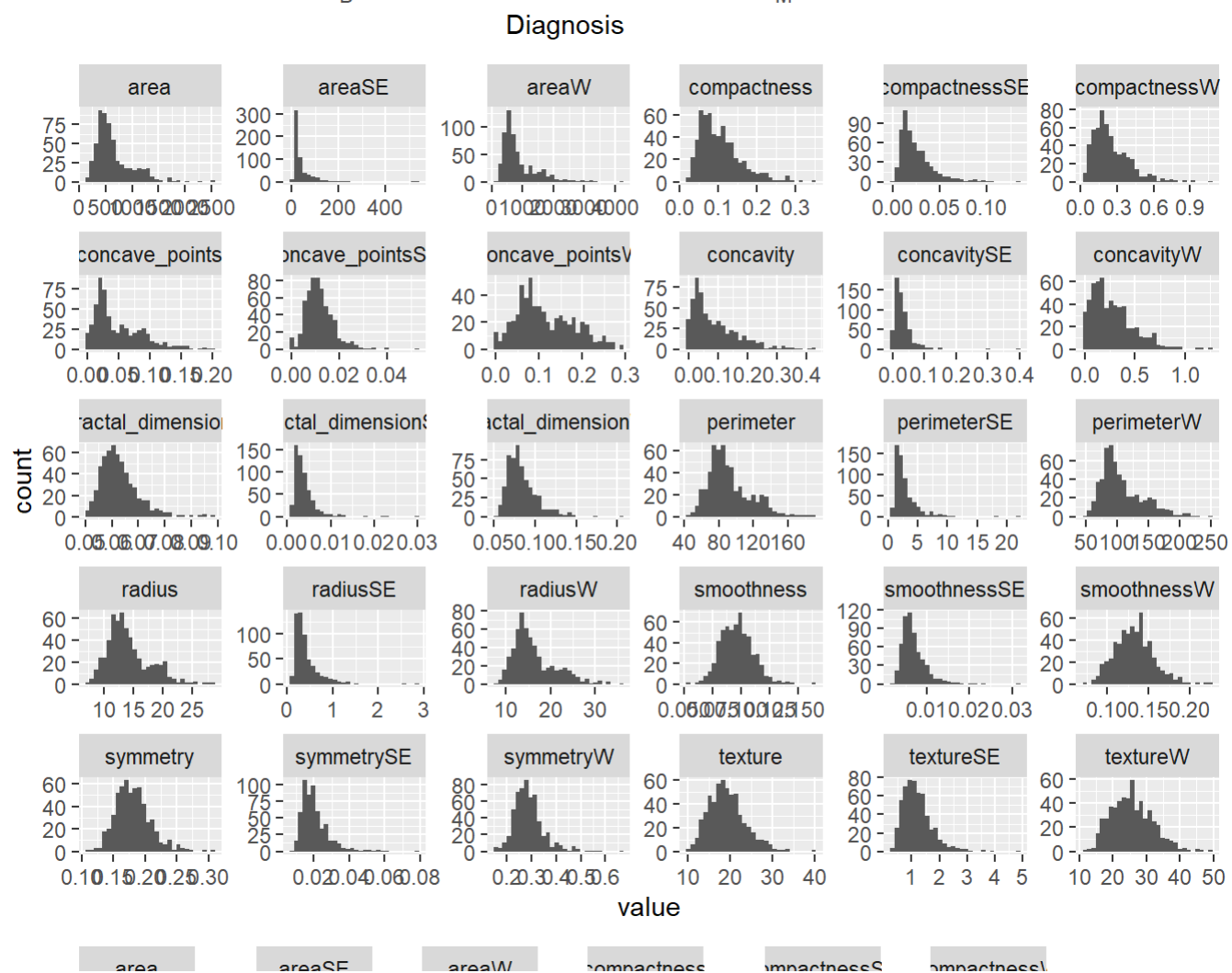
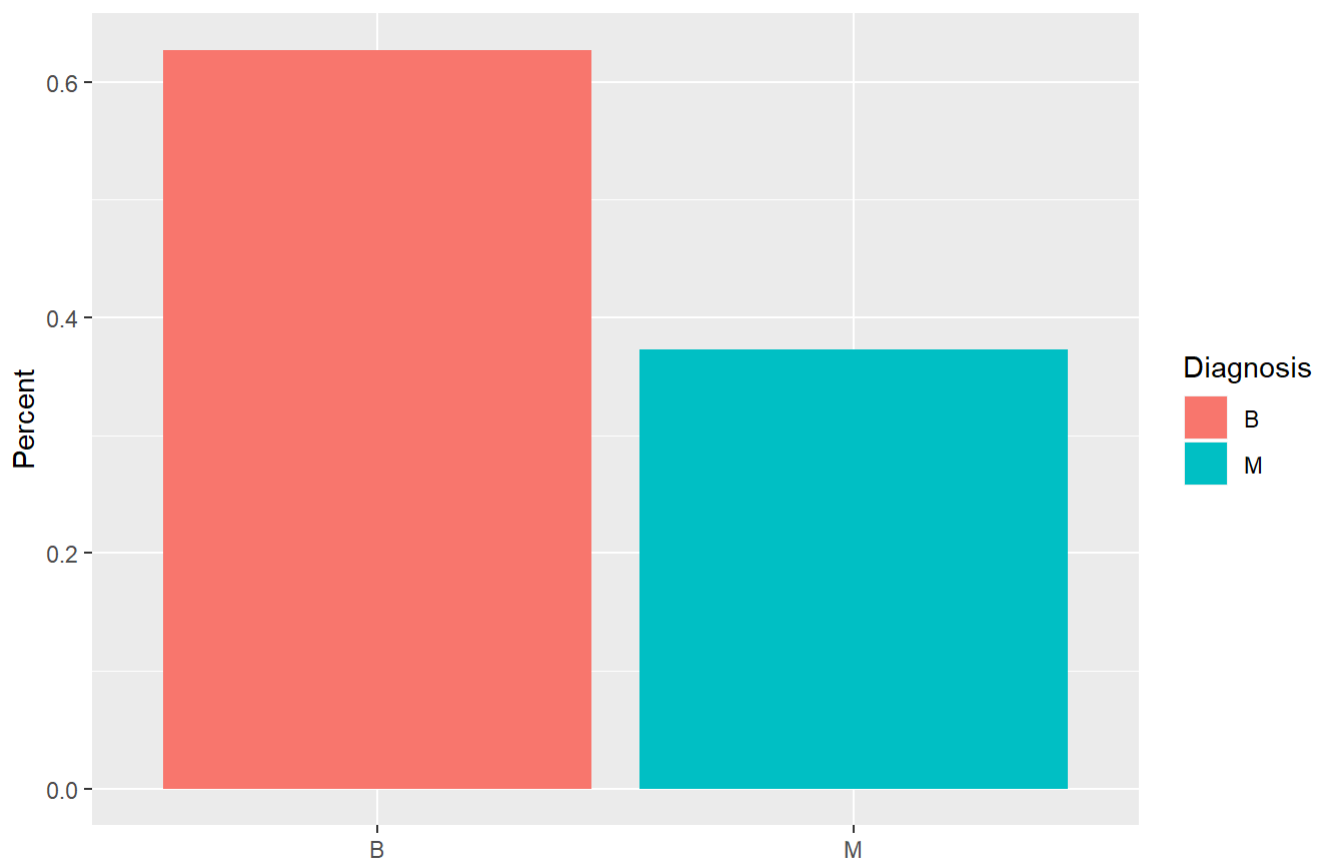
## Data visualization

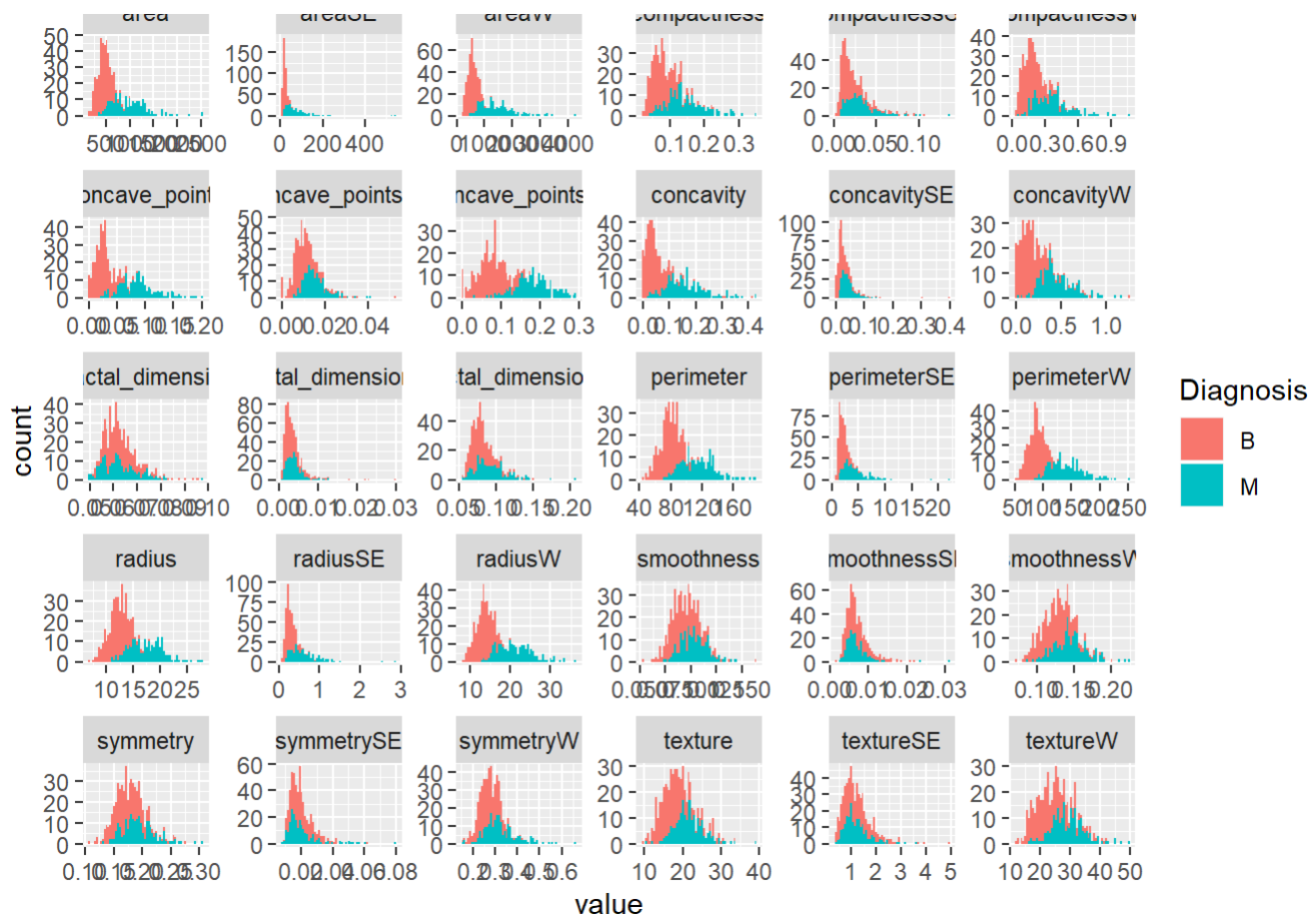
- \* Plot Diagnosis;
- \* Histogram for all Variables;
- \* Histogram for all Variables by Diagnosis.

**There are differences between B and M Diagnosis such as area, compactness, perimeter, radius, etc**

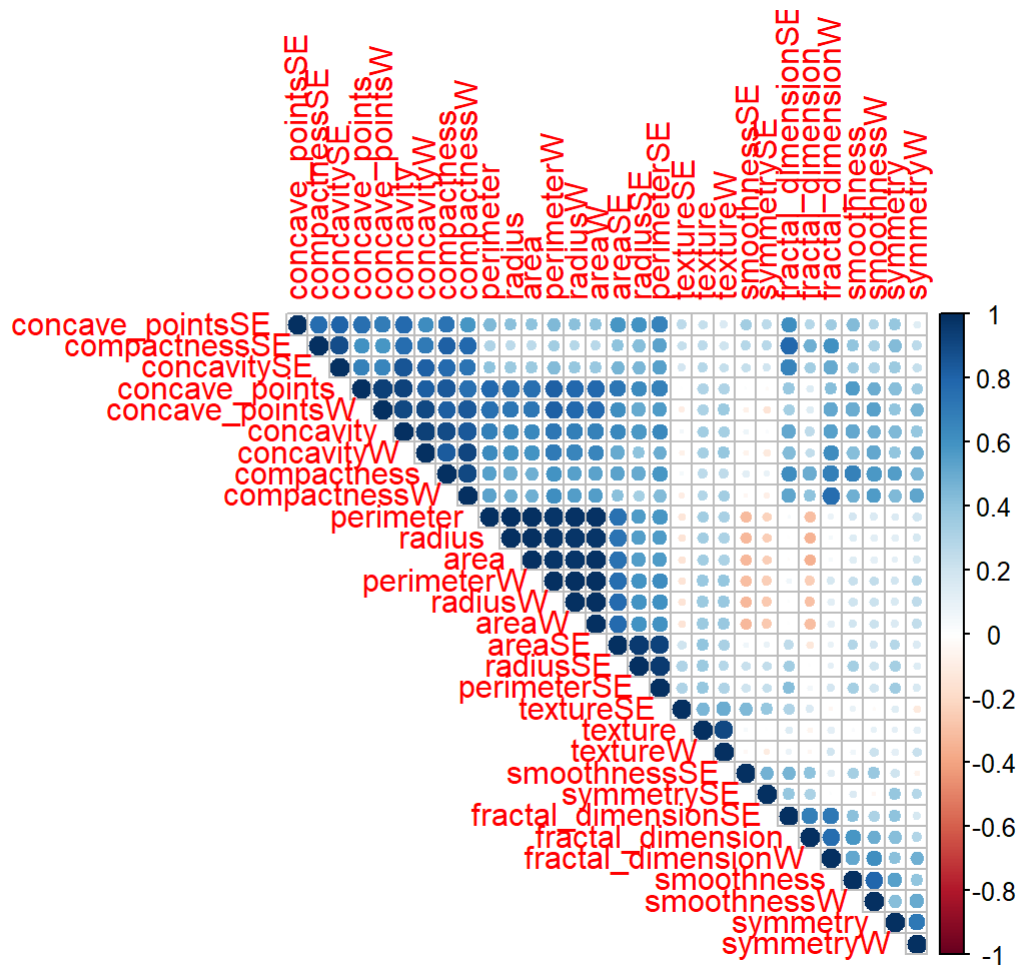


Diagnosis percentages





Correlaion analysis by spearman since most variables are non normal distributed



Drop variables with multicollinearity, where correlation  $> 0.9$ , and remove the variable with the largest mean absolute correlation



```
## Compare row 7 and column 28 with corr 0.905
## Means: 0.579 vs 0.417 so flagging column 7
## Compare row 28 and column 8 with corr 0.937
## Means: 0.557 vs 0.405 so flagging column 28
## Compare row 6 and column 26 with corr 0.901
## Means: 0.541 vs 0.396 so flagging column 6
## Compare row 27 and column 26 with corr 0.915
## Means: 0.508 vs 0.385 so flagging column 27
## Compare row 23 and column 21 with corr 0.994
## Means: 0.497 vs 0.375 so flagging column 23
## Compare row 21 and column 24 with corr 0.999
## Means: 0.462 vs 0.366 so flagging column 21
## Compare row 24 and column 3 with corr 0.981
## Means: 0.436 vs 0.359 so flagging column 24
## Compare row 3 and column 1 with corr 0.998
## Means: 0.401 vs 0.353 so flagging column 3
## Compare row 1 and column 4 with corr 1
## Means: 0.356 vs 0.35 so flagging column 1
## Compare row 14 and column 13 with corr 0.927
## Means: 0.385 vs 0.346 so flagging column 14
## Compare row 13 and column 11 with corr 0.958
## Means: 0.397 vs 0.345 so flagging column 13
## Compare row 22 and column 2 with corr 0.909
## Means: 0.241 vs 0.346 so flagging column 2
## All correlations <= 0.9
```

## Highly correlated variables

```
## [1] "concavity"      "concave_pointsW" "compactness"
## [4] "concavityW"      "perimeterW"      "radiusW"
## [7] "areaW"           "perimeter"       "radius"
## [10] "areaSE"          "perimeterSE"     "texture"
```

## Number of features left

```
## [1] 18
```

Data Transformation/ Dimension Reduction by using PCA, after dropping label.

PCA components summary for raw data

```
## Importance of components:
##          PC1    PC2    PC3    PC4    PC5    PC6
## Standard deviation  3.6444 2.3857 1.67867 1.40735 1.28403 1.09880
## Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025
## Cumulative Proportion 0.4427 0.6324 0.72636 0.79239 0.84734 0.88759
##          PC7    PC8    PC9    PC10   PC11   PC12
## Standard deviation  0.82172 0.69037 0.6457 0.59219 0.5421 0.51104
## Proportion of Variance 0.02251 0.01589 0.0139 0.01169 0.0098 0.00871
## Cumulative Proportion 0.91010 0.92598 0.9399 0.95157 0.9614 0.97007
##          PC13   PC14   PC15   PC16   PC17   PC18
## Standard deviation  0.49128 0.39624 0.30681 0.28260 0.24372 0.22939
## Proportion of Variance 0.00805 0.00523 0.00314 0.00266 0.00198 0.00175
## Cumulative Proportion 0.97812 0.98335 0.98649 0.98915 0.99113 0.99288
##          PC19   PC20   PC21   PC22   PC23   PC24
## Standard deviation  0.22244 0.17652 0.1731 0.16565 0.15602 0.1344
## Proportion of Variance 0.00165 0.00104 0.0010 0.00091 0.00081 0.0006
## Cumulative Proportion 0.99453 0.99557 0.9966 0.99749 0.99830 0.9989
##          PC25   PC26   PC27   PC28   PC29   PC30
## Standard deviation  0.12442 0.09043 0.08307 0.03987 0.02736 0.01153
## Proportion of Variance 0.00052 0.00027 0.00023 0.00005 0.00002 0.00000
## Cumulative Proportion 0.99942 0.99969 0.99992 0.99997 1.00000 1.00000
```

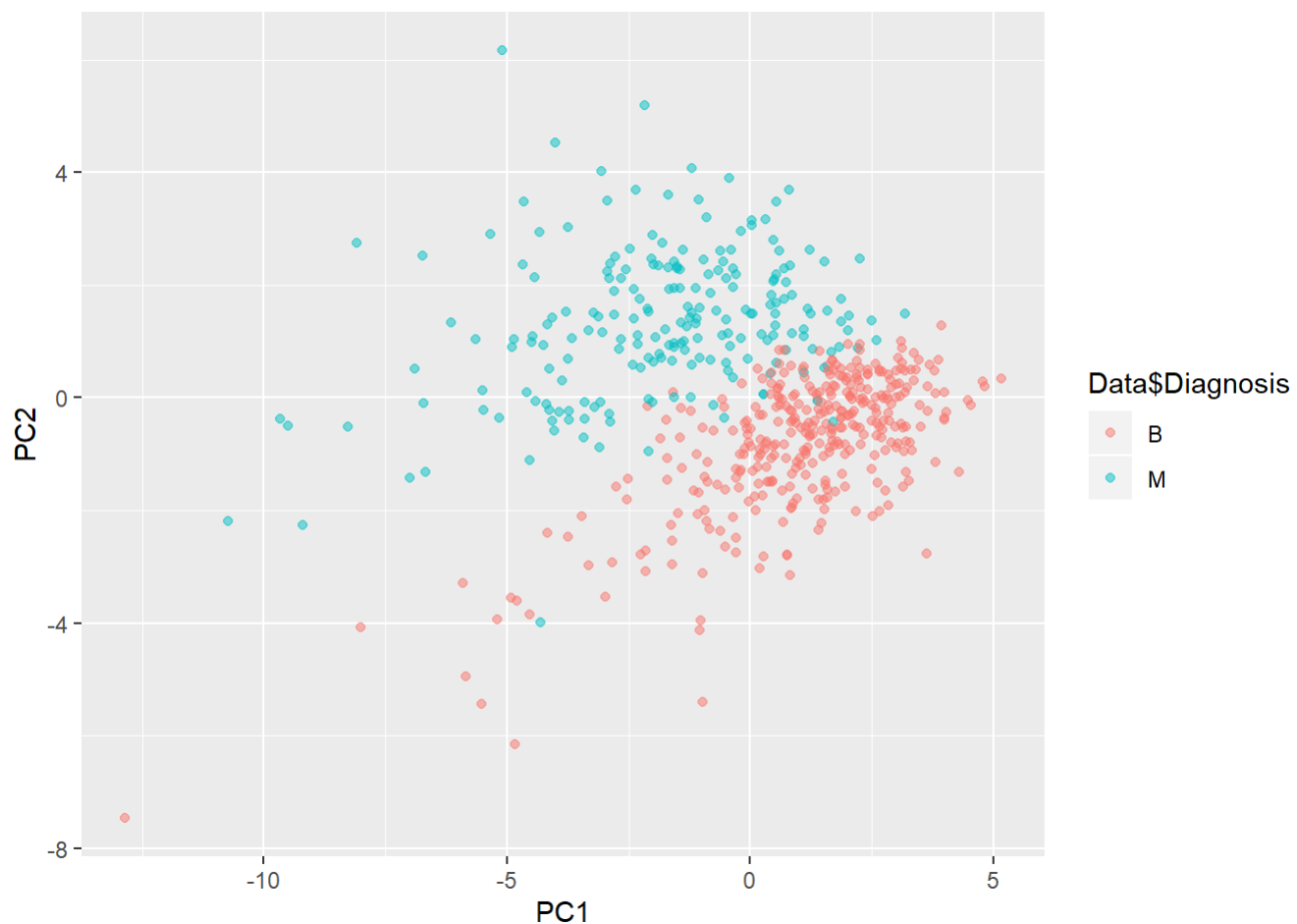
## PCA without highly correlated variables

```
## Importance of components:
##          PC1    PC2    PC3    PC4    PC5    PC6
## Standard deviation  2.6151 1.6197 1.5256 1.21914 1.12496 1.09311
## Proportion of Variance 0.3799 0.1457 0.1293 0.08257 0.07031 0.06638
## Cumulative Proportion 0.3799 0.5257 0.6550 0.73756 0.80787 0.87425
##          PC7    PC8    PC9    PC10   PC11   PC12
## Standard deviation  0.68529 0.66200 0.56110 0.49260 0.42947 0.40531
## Proportion of Variance 0.02609 0.02435 0.01749 0.01348 0.01025 0.00913
## Cumulative Proportion 0.90034 0.92469 0.94218 0.95566 0.96591 0.97503
##          PC13   PC14   PC15   PC16   PC17   PC18
## Standard deviation  0.3698 0.3576 0.25151 0.2326 0.2079 0.15547
## Proportion of Variance 0.0076 0.0071 0.00351 0.0030 0.0024 0.00134
## Cumulative Proportion 0.9826 0.9897 0.99325 0.9962 0.9987 1.00000
```

## Plot first 2 PCA components

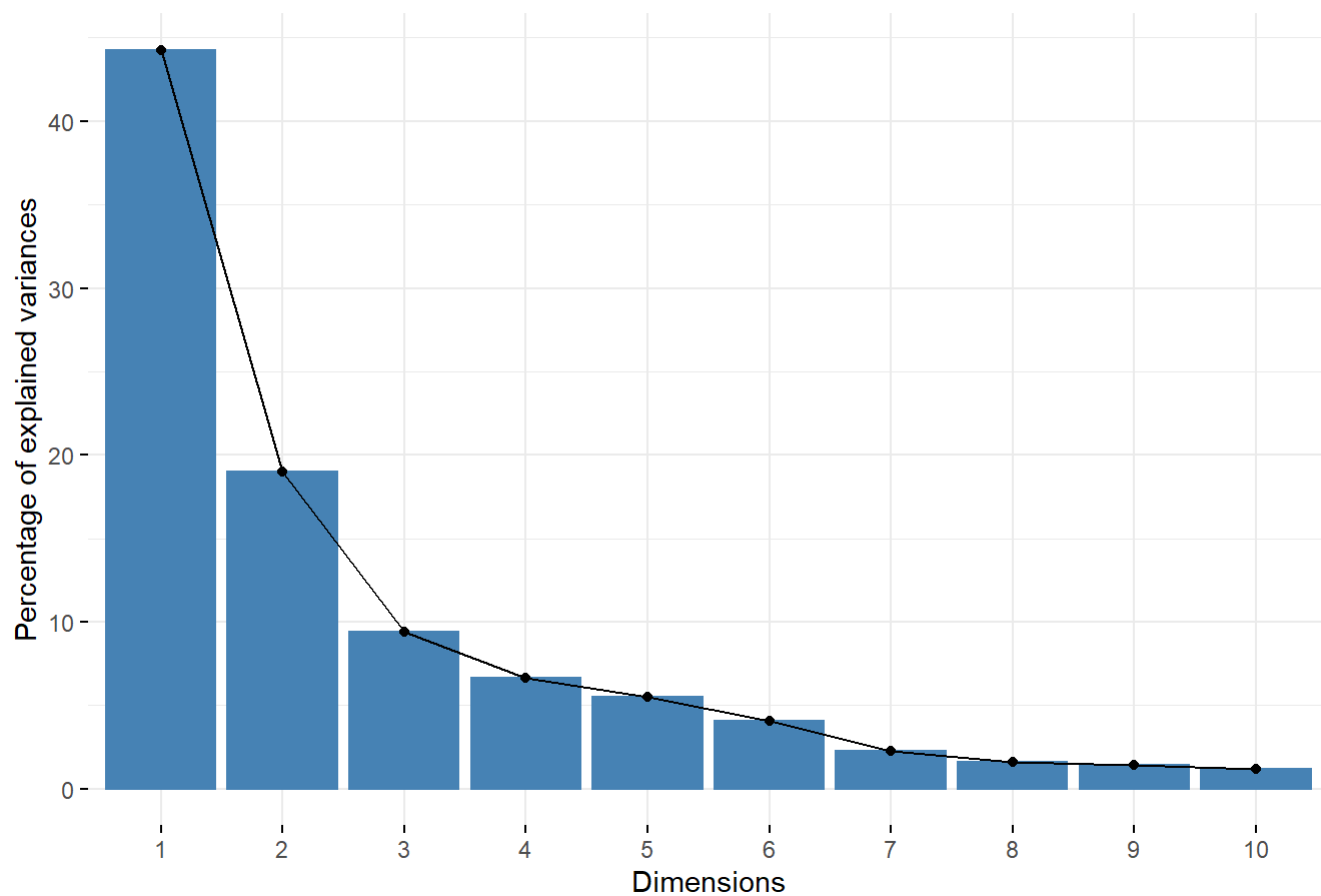


Plot PCA without highly correlated variables on the first 2 components

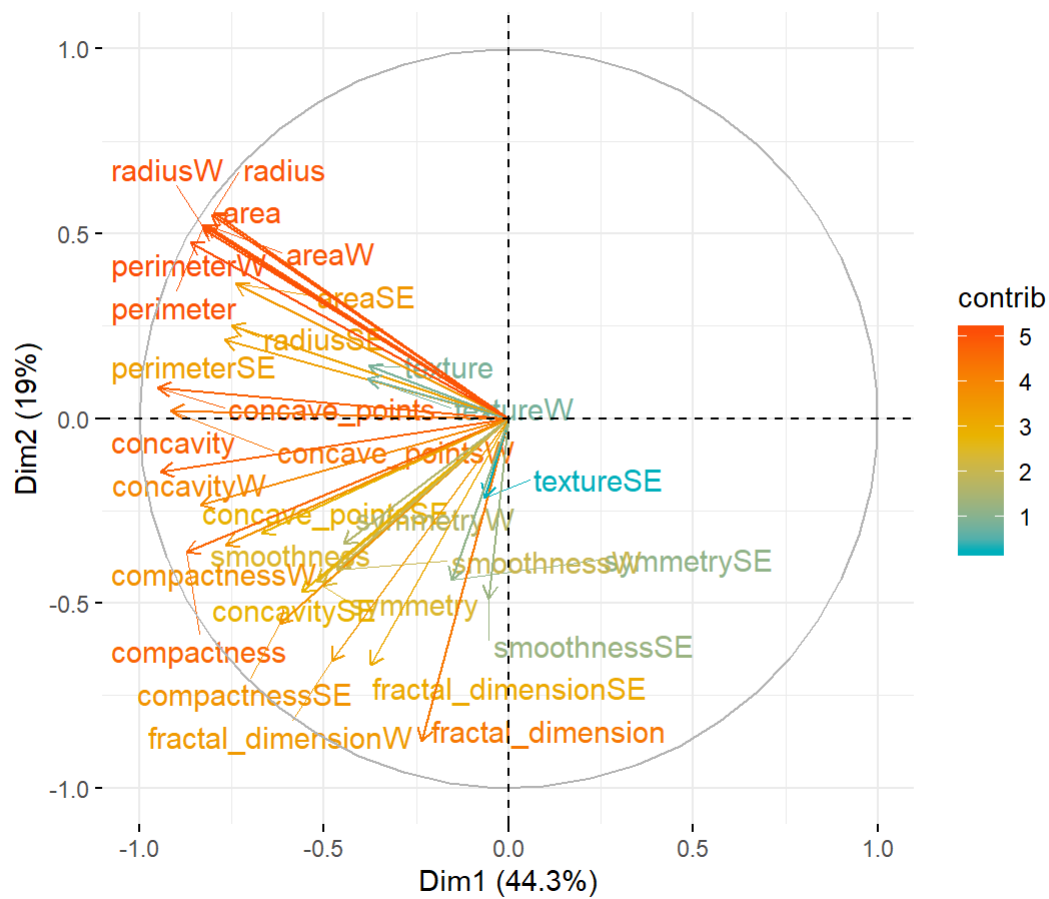


Visualize which variables are the most influential.  
Individuals with a similar profile are grouped together

Scree plot

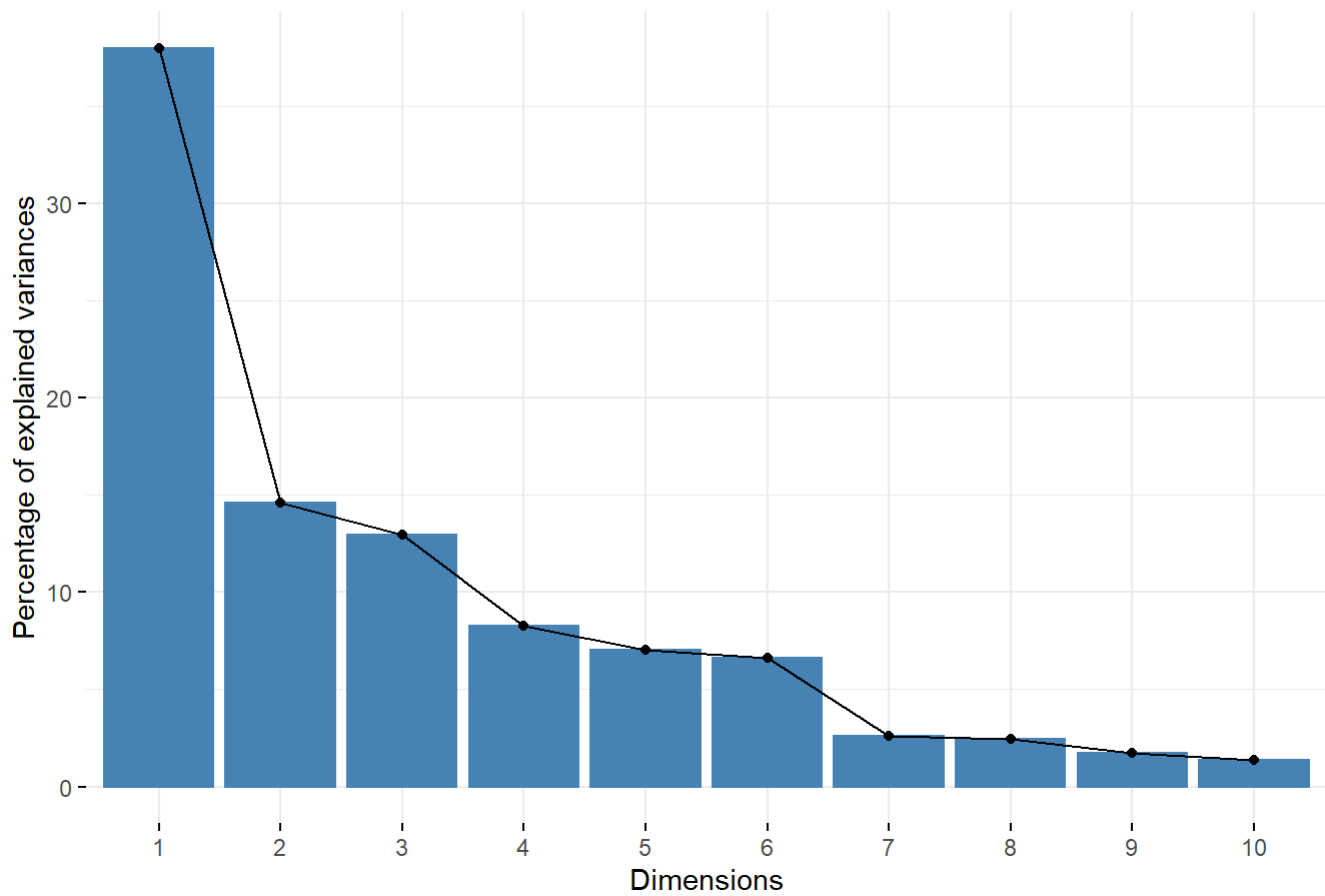


Variables - PCA

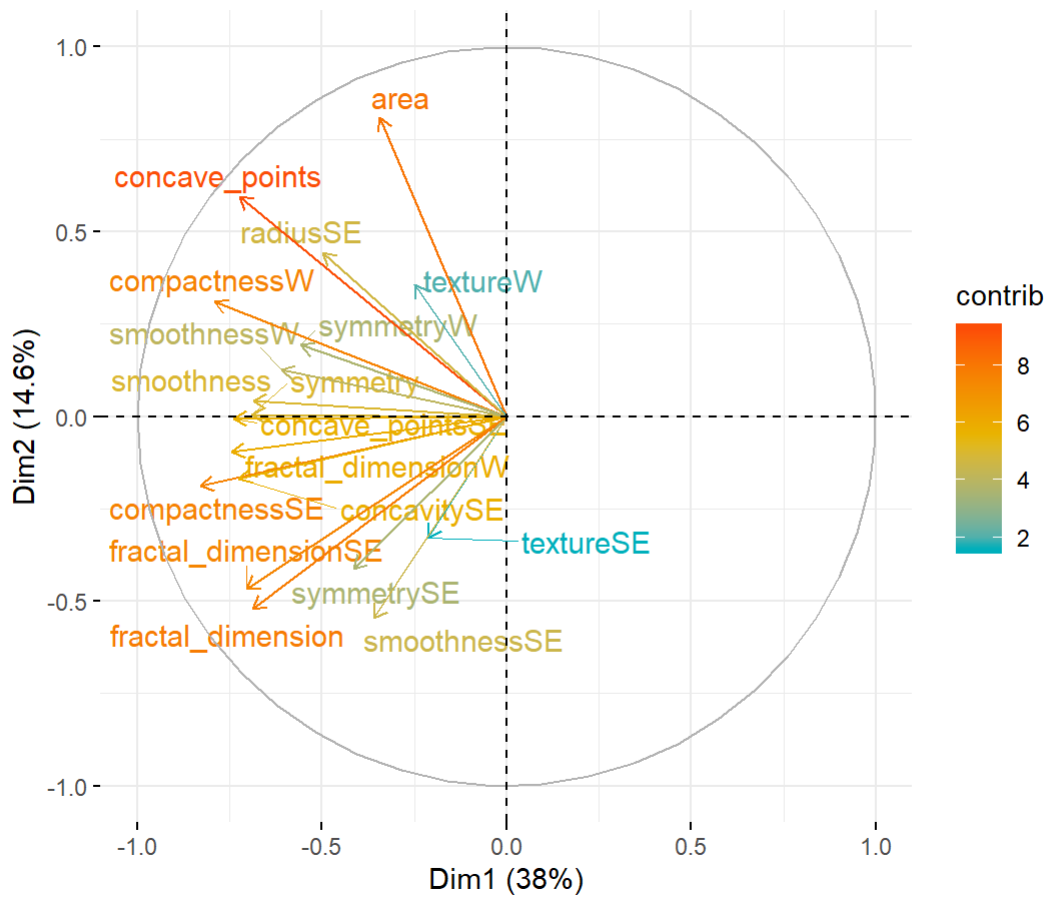


After correlated variables were dropped

Scree plot



Variables - PCA



**Correlated variables such as area, perimeters, radius are grouped together and they are important contributors**

## **Model Training**

Split data into training and test

split data into training and test after dropping correlated variables

Svm with original data by Using library(e1071), data is not scaled;

\* Default parameters: Cost =1, gamma = 1/(data dimension)

first gamma:

```
## [1] 0.03333333
```



```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##           B 71 42
##           M  0  0
##
##           Accuracy : 0.6283
##           95% CI : (0.5324, 0.7174)
##           No Information Rate : 0.6283
##           P-Value [Acc > NIR] : 0.542
##
##           Kappa : 0
##
##           Mcnemar's Test P-Value : 2.509e-10
##
##           Sensitivity : 0.0000
##           Specificity : 1.0000
##           Pos Pred Value :      NaN
##           Neg Pred Value : 0.6283
##           Prevalence : 0.3717
##           Detection Rate : 0.0000
##           Detection Prevalence : 0.0000
##           Balanced Accuracy : 0.5000
##
##           'Positive' Class : M
##

```

Svm with after dropped correlated variables with default setting by using library(e1071)

second gamma:

```
## [1] 0.05555556
```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B   M
##           B 68 15
##           M  3 27
##
##           Accuracy : 0.8407
##           95% CI : (0.76, 0.9028)
##           No Information Rate : 0.6283
##           P-Value [Acc > NIR] : 6.001e-07
##
##           Kappa : 0.6378
##
##           McNemar's Test P-Value : 0.009522
##
##           Sensitivity : 0.6429
##           Specificity : 0.9577
##           Pos Pred Value : 0.9000
##           Neg Pred Value : 0.8193
##           Prevalence : 0.3717
##           Detection Rate : 0.2389
##           Detection Prevalence : 0.2655
##           Balanced Accuracy : 0.8003
##
##           'Positive' Class : M
##

```

Use caret package to fit models with default parameter tuning with 5 fold cross validation, repeated 5 times; Caret “svmRadial” model uses kernlab package; No data scaling.

$$* C = 2^{((1:len)-3)} \text{ (C=0.25)}$$

\* Sigma is defined by Kernlab package’s sigest function, sigest estimates are based upon the 0.1 and 0.9 quantile of  $\|x - x'\|^2$ . Basically any value in between those two bounds will produce good results. A vector of length 3 defining the range (0.1 quantile, median and 0.9 quantile) of the sigma hyperparameter.

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc (classification)
## parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
## Hyperparameter : sigma = 0.0445118095276661
##
## Number of Support Vectors : 121
##
## Objective Function Value : -50.3072
## Training error : 0.013158
## Probability model included.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction B  M
##           B 71  3
##           M  0 39
##
##           Accuracy : 0.9735
##           95% CI : (0.9244, 0.9945)
##           No Information Rate : 0.6283
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9423
##
## Mcnemar's Test P-Value : 0.2482
##
##           Sensitivity : 0.9286
##           Specificity : 1.0000
##           Pos Pred Value : 1.0000
##           Neg Pred Value : 0.9595
##           Prevalence : 0.3717
##           Detection Rate : 0.3451
##           Detection Prevalence : 0.3451
##           Balanced Accuracy : 0.9643
##
##           'Positive' Class : M
##
```

Use caret package to fit models after dropped correlated variables with default parameter tuning with 5 fold cross validation, repeated 5 times; No data scaling.

**It's the same result with previous model without dropped correlated variables but it's less sensitive on positive predicting and more sensitive on negative predicting.**

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc (classification)
## parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
## Hyperparameter : sigma = 0.0642190212977282
##
## Number of Support Vectors : 130
##
## Objective Function Value : -62.7148
## Training error : 0.013158
## Probability model included.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##           B 70  3
##           M  1 39
##
##           Accuracy : 0.9646
##           95% CI : (0.9118, 0.9903)
## No Information Rate : 0.6283
## P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9235
##
## Mcnemar's Test P-Value : 0.6171
##
##           Sensitivity : 0.9286
##           Specificity : 0.9859
##           Pos Pred Value : 0.9750
##           Neg Pred Value : 0.9589
##           Prevalence : 0.3717
##           Detection Rate : 0.3451
## Detection Prevalence : 0.3540
##           Balanced Accuracy : 0.9572
##
##           'Positive' Class : M
##
```

**Scale variables and cross validation**

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc (classification)
## parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
## Hyperparameter : sigma = 0.0445118095276661
##
## Number of Support Vectors : 121
##
## Objective Function Value : -50.3072
## Training error : 0.013158
## Probability model included.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##           B 71  3
##           M  0 39
##
##           Accuracy : 0.9735
##           95% CI : (0.9244, 0.9945)
##           No Information Rate : 0.6283
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9423
##
## Mcnemar's Test P-Value : 0.2482
##
##           Sensitivity : 0.9286
##           Specificity : 1.0000
##           Pos Pred Value : 1.0000
##           Neg Pred Value : 0.9595
##           Prevalence : 0.3717
##           Detection Rate : 0.3451
##           Detection Prevalence : 0.3451
##           Balanced Accuracy : 0.9643
##
##           'Positive' Class : M
##
```

Scale variables with pca threshold = 0.8. 0.8 is the sweet spot.

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc (classification)
## parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
## Hyperparameter : sigma = 0.285658961570517
##
## Number of Support Vectors : 128
##
## Objective Function Value : -64.8713
## Training error : 0.02193
## Probability model included.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##           B 70  1
##           M  1 41
##
##           Accuracy : 0.9823
##           95% CI : (0.9375, 0.9978)
##           No Information Rate : 0.6283
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9621
##
## Mcnemar's Test P-Value : 1
##
##           Sensitivity : 0.9762
##           Specificity : 0.9859
##           Pos Pred Value : 0.9762
##           Neg Pred Value : 0.9859
##           Prevalence : 0.3717
##           Detection Rate : 0.3628
##           Detection Prevalence : 0.3717
##           Balanced Accuracy : 0.9811
##
##           'Positive' Class : M
##
```

Scale variables with dropped variables and with pca = 0.95;

**The results doesn't improve much anymore, with increasing threshold of pca, the performance increases on the 4th digit and it's more stable**

```

## Support Vector Machines with Radial Basis Function Kernel
##
## 456 samples
## 18 predictor
## 2 classes: 'B', 'M'
##
## Pre-processing: centered (18), scaled (18), principal component
## signal extraction (18)
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 365, 365, 364, 365, 365, 365, ...
## Resampling results across tuning parameters:
##
## C      ROC      Sens      Spec
## 0.25  0.9911843  0.9656866  0.9482353
## 0.50  0.9919437  0.9705989  0.9494118
## 1.00  0.9928896  0.9706110  0.9482353
##
## Tuning parameter 'sigma' was held constant at a value of 0.09535176
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.09535176 and C = 1.

```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B    M
##           B 285    7
##           M   1 163
##
##           Accuracy : 0.9825
##           95% CI : (0.9657, 0.9924)
##           No Information Rate : 0.6272
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9622
##
## Mcnemar's Test P-Value : 0.0771
##
##           Sensitivity : 0.9588
##           Specificity : 0.9965
##           Pos Pred Value : 0.9939
##           Neg Pred Value : 0.9760
##           Prevalence : 0.3728
##           Detection Rate : 0.3575
##           Detection Prevalence : 0.3596
##           Balanced Accuracy : 0.9777
##
##           'Positive' Class : M
##

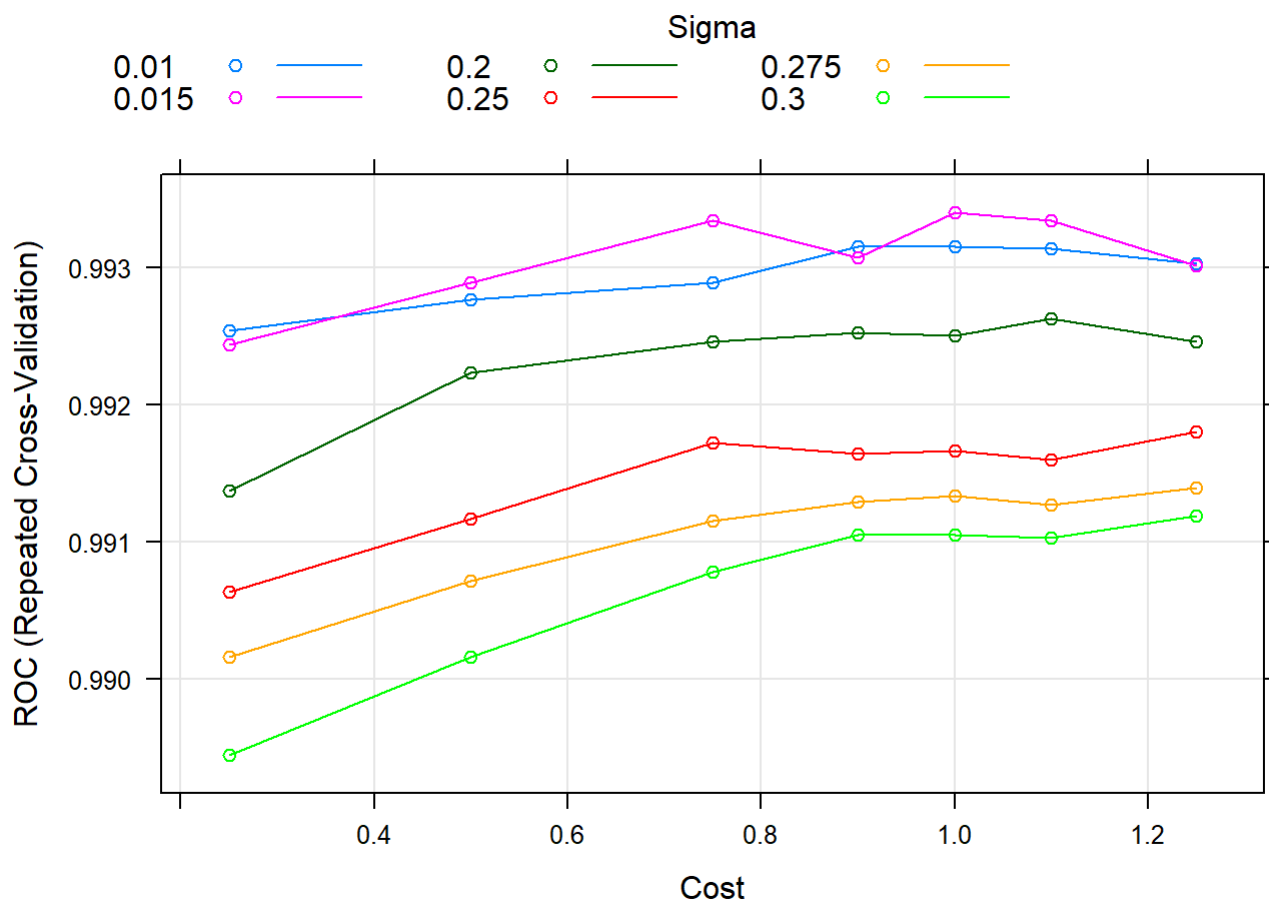
```

# Tune SVM parameters by grid search, total 42 pairs

```
svmGrid <- expand.grid(sigma = c(.01, .015, 0.2, 0.25, 0.275, 0.3),  
  C = c(0.25, 0.5, 0.75, 0.9, 1, 1.1, 1.25))  
  
nrow(svmGrid)
```

```
## [1] 42
```

Plot the results on the feature selected data with pca, by grid search



Performance on test data with the best parameters

```
pred_svm_tune <- predict(svm_tune, train_drop_raw)  
cm_svm_tune <- confusionMatrix(pred_svm_tune, as.factor(train_drop_raw$Diagnosis), positive =  
  "M")  
cm_svm_tune
```



```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B    M
##           B 279    9
##           M   7 161
##
##           Accuracy : 0.9649
##           95% CI : (0.9436, 0.9798)
##           No Information Rate : 0.6272
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9248
##
##           Mcnemar's Test P-Value : 0.8026
##
##           Sensitivity : 0.9471
##           Specificity : 0.9755
##           Pos Pred Value : 0.9583
##           Neg Pred Value : 0.9688
##           Prevalence : 0.3728
##           Detection Rate : 0.3531
##           Detection Prevalence : 0.3684
##           Balanced Accuracy : 0.9613
##
##           'Positive' Class : M
##

```

*#Plotting the Resampling Profile: Examine the relationship between the estimates of performance and the tuning parameters*

**The best parameters on training data doesn't guarantee the best performance on the test data. It could be easily overfitting. The final values used for the model were  $\sigma = 0.015$  and  $C = 1$ .**

**Instead of just doing grid search, could we add another regularization parameter to reduce over fitting, or**

**instead of choosing the best pair of parameters ( $C$ ,  $\sigma$ ), allow more errors and use another algorithm, for example, select top 3 pairs on test dataset to find the best pair of parameters?**

**Neural network**

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B   M
##           B 71  4
##           M  0 38
##
##           Accuracy : 0.9646
##           95% CI : (0.9118, 0.9903)
##           No Information Rate : 0.6283
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9227
##
##           McNemar's Test P-Value : 0.1336
##
##           Sensitivity : 0.9048
##           Specificity : 1.0000
##           Pos Pred Value : 1.0000
##           Neg Pred Value : 0.9467
##           Prevalence : 0.3717
##           Detection Rate : 0.3363
##           Detection Prevalence : 0.3363
##           Balanced Accuracy : 0.9524
##
##           'Positive' Class : M
##

```

## Neural network with pca

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B   M
##           B 71  3
##           M  0 39
##
##           Accuracy : 0.9735
##           95% CI : (0.9244, 0.9945)
##           No Information Rate : 0.6283
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9423
##
##           McNemar's Test P-Value : 0.2482
##
##           Sensitivity : 0.9286
##           Specificity : 1.0000
##           Pos Pred Value : 1.0000
##           Neg Pred Value : 0.9595
##           Prevalence : 0.3717
##           Detection Rate : 0.3451
##           Detection Prevalence : 0.3451
##           Balanced Accuracy : 0.9643
##
##           'Positive' Class : M
##

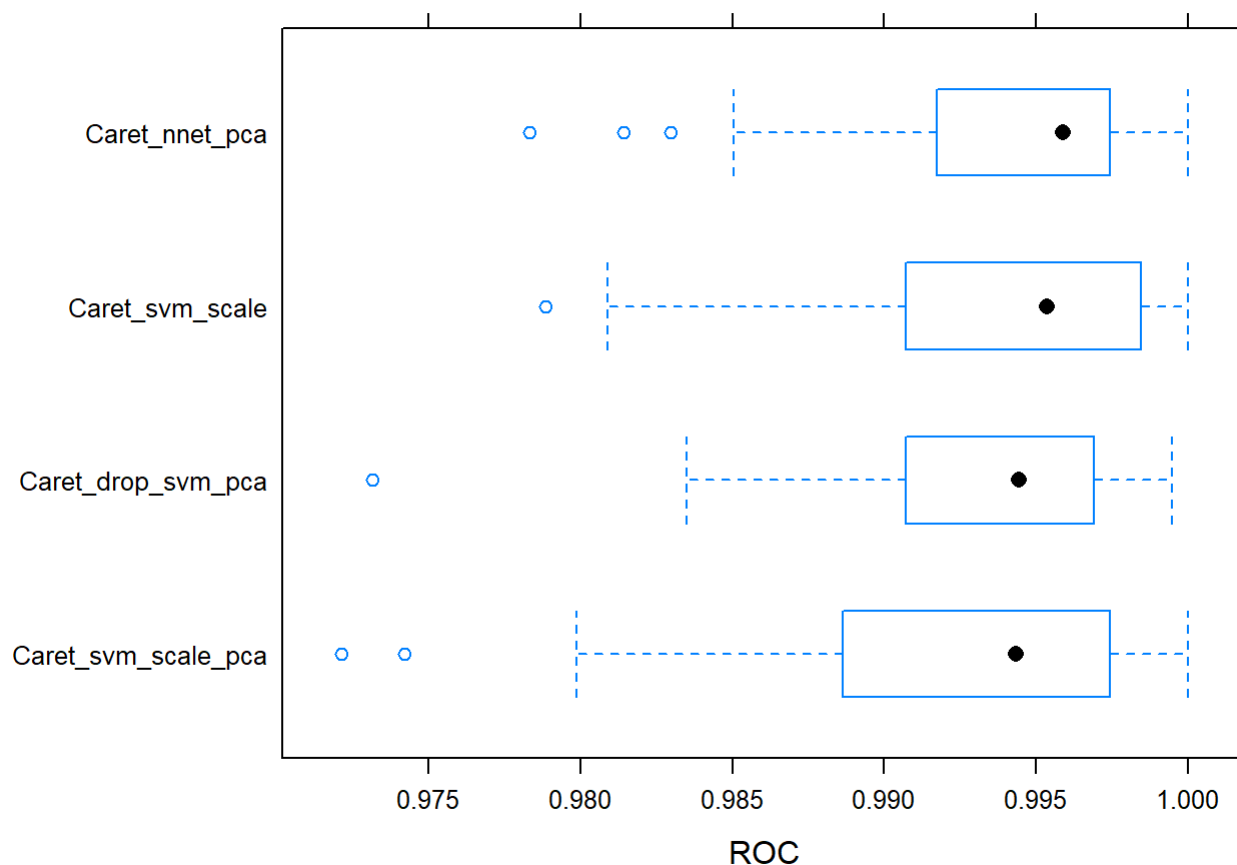
```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B   M
##           B 71  3
##           M  0 39
##
##           Accuracy : 0.9735
##           95% CI : (0.9244, 0.9945)
##           No Information Rate : 0.6283
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.9423
##
##           McNemar's Test P-Value : 0.2482
##
##           Sensitivity : 0.9286
##           Specificity : 1.0000
##           Pos Pred Value : 1.0000
##           Neg Pred Value : 0.9595
##           Prevalence : 0.3717
##           Detection Rate : 0.3451
##           Detection Prevalence : 0.3451
##           Balanced Accuracy : 0.9643
##
##           'Positive' Class : M
##

```

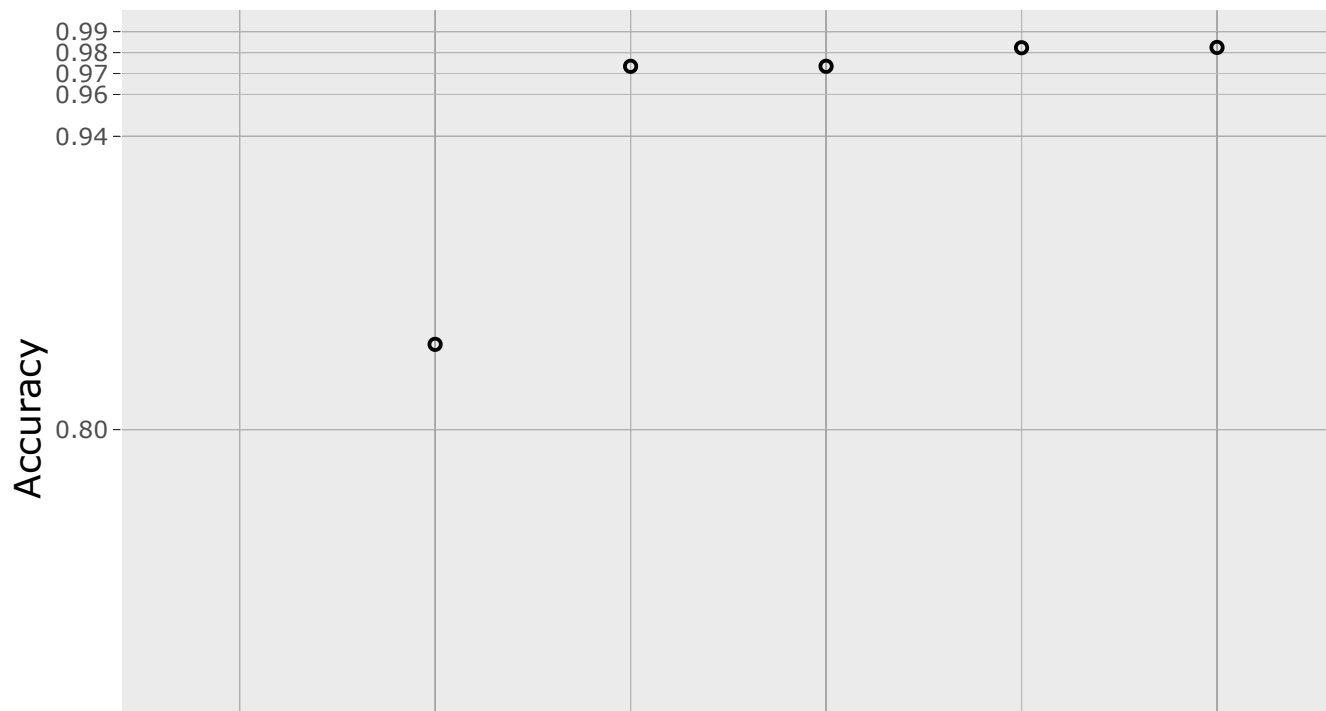
## Models evaluation by caret package with cross validation on training model

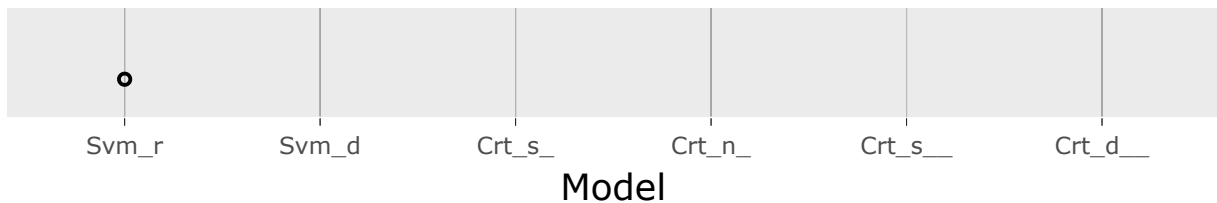


Overall model evaluation on test data

**Final model achieved accuracy of 0.9825.**

Interactive Model Performance





## Takeaways:

1. Scaling variables is very important. and tuning parameters by using resampling technique for example, cross validation is more important.
2. Reduce multicollinearity could stabilize the model performance.
3. Dimension deduction (using PCA here) without feature selection is not robust. It would take time to find a sweet spot on the optimal PCA value.
4. In another word, with the appropriate feature selection, it reduces the variation of dimension deduction with a stable result.
5. The best tuning parameters on training could be overfitting. Neural network tends to overfit the model.
6. Grid search method could be improved by combining performance on test data instead of solely relying on training data.
7. A good SVM could perform better than neural network and could explain important features.

## Appendix

```

knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(dplyr)
library(ggplot2)
library(psych)
library(corrplot)
library(e1071)
library(caret)
library(plotly)
library(plyr)

Data <- read_csv("C:/Work/Project/STA7734/SVM-Asymptotic-Normality/Data/data.csv",
  col_names = T,
  col_types=cols(ID =col_character(),
    Diagnosis=col_character(),
    "3" = col_double(),
    "4" = col_double(),
    "5" = col_double(),
    "6" = col_double(),
    "7" = col_double(),
    "8" = col_double(),
    "9" = col_double(),
    "10" = col_double(),
    "11" = col_double(),
    "12" = col_double(),
    "13" = col_double(),
    "14" = col_double(),
    "15" = col_double(),
    "16" = col_double(),
    "17" = col_double(),
    "18" = col_double(),
    "19" = col_double(),
    "20" = col_double(),
    "21" = col_double(),
    "22" = col_double(),
    "23" = col_double(),
    "24" = col_double(),
    "25" = col_double(),
    "26" = col_double(),
    "27" = col_double(),
    "28" = col_double(),
    "29" = col_double(),
    "30" = col_double(),
    "31" = col_double(),
    "32" = col_double()
  )) %>%
dplyr::rename( "radius" = "3",
  "texture" = "4",
  "perimeter" = "5",
  "area" = "6",
  "smoothness" = "7",
  "compactness" = "8",
  "concavity" = "9",

```

```

"concave_points" = "10",
"symmetry" = "11",
"fractal_dimension" = "12",
"radiusSE" = "13",
"textureSE" = "14",
"perimeterSE" = "15",
"areaSE" = "16",
"smoothnessSE" = "17",
"compactnessSE" = "18",
"concavitySE" = "19",
"concave_pointsSE" = "20",
"symmetrySE" = "21",
"fractal_dimensionSE" = "22",
"radiusW" = "23",
"textureW" = "24",
"perimeterW" = "25",
"areaW" = "26",
"smoothnessW" = "27",
"compactnessW" = "28",
"concavityW" = "29",
"concave_pointsW" = "30",
"symmetryW" = "31",
"fractal_dimensionW" = "32")

```

*#Check specs/data type of the data:*

```

spec(Data)
head(Data)
#tail(Data)
sapply(Data, function(x) sum(is.na(x)))

```

*Data\_drop <- Data[,-1] #drop ID*

*psych::describe(Data\_drop) # Describe data*

*#psych::describeBy(Data\_drop,Data\_drop\$Diagnosis) #Describe data by Diagnosis*

```
Data_drop$Diagnosis <- as.factor(Data_drop$Diagnosis)
```

```
summary(Data_drop$Diagnosis)
```

```

ggplot(Data,aes(x = Diagnosis) ) +
  geom_bar(aes(y = ..count../sum(..count..), fill = Diagnosis)) +
  ylab("Percent") +
  ggtitle("Diagnosis percentages")

```

*#Plot histogram for all Variables*

```

Data_drop %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
    geom_histogram() +
    facet_wrap(~key, scales = "free")

```

*#Plot histogram for all Variables by Diagnosis*

```

Data_drop %>%
  gather(key = "Var", value = "value", -Diagnosis) %>%
  ggplot(aes(x=value, fill=Diagnosis)) +

```



```

    geom_histogram(bins = 60) +
    facet_wrap(~Var, scales = "free")
#create correlation matrix
Cor <- Data_drop %>%
  select(-Diagnosis) %>%
  cor(method="spearman")

corrplot(Cor, type="upper", order="hclust")
#Removing all features with a correlation higher than 0.7, keeping the feature with the lower mean.

highlyCor <- colnames(Data_drop[, -1])[findCorrelation(Cor, cutoff = 0.9, verbose = TRUE)]
highlyCor #highly correlated variables

Data_drop_cor <- Data_drop[, which(!colnames(Data_drop) %in% highlyCor)]#Drop highly correlated variables

ncol(Data_drop_cor)-1
#Drop Diagnosis
Data_pca <- prcomp(Data_drop[, -1], center=TRUE, scale=TRUE)
summary(Data_pca)#need 10 components for variance over 95%
Data_pca_re <- prcomp(Data_drop_cor[, -1], center=TRUE, scale=TRUE)
summary(Data_pca_re)#7 components have variance over 97%
pca_df <- as.data.frame(Data_pca$x)
ggplot(pca_df, aes(x=PC1, y=PC2, col=Data$Diagnosis)) + geom_point(alpha=0.5)
pca_df_re <- as.data.frame(Data_pca_re$x)
ggplot(pca_df_re, aes(x=PC1, y=PC2, col=Data$Diagnosis)) + geom_point(alpha=0.5)
library(factoextra)
fviz_eig(Data_pca)
fviz_pca_var(Data_pca,
  col.var = "contrib", # Color by contributions to the PC
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
  repel = TRUE      # Avoid text overlapping
)
fviz_eig(Data_pca_re)
fviz_pca_var(Data_pca_re,
  col.var = "contrib", # Color by contributions to the PC
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
  repel = TRUE      # Avoid text overlapping
)

set.seed(1234) # so that the indices will be the same when re-run
trainIndices = createDataPartition(Data_drop$Diagnosis, p=.8, list=F)

train_raw = Data_drop %>%
  slice(trainIndices)

nrow(train_raw)

test_raw = Data_drop %>%
  slice(-trainIndices)
head(test_raw)

```

```

#verify if the randomization process is correct
prop.table(table(train_raw$Diagnosis))
prop.table(table(test_raw$Diagnosis))
#for multicolinearty dropped Data_drop_cor
set.seed(1234) # so that the indices will be the same when re-run
trainIndices_drop = createDataPartition(Data_drop_cor$Diagnosis, p=.8, list=F)

train_drop_raw = Data_drop_cor %>%
  slice(trainIndices_drop)

nrow(train_drop_raw)
head(train_drop_raw)

test_drop_raw = Data_drop_cor %>%
  slice(-trainIndices_drop)
head(test_drop_raw)

#verify if the randomization process is correct
prop.table(table(train_drop_raw$Diagnosis))
prop.table(table(test_drop_raw$Diagnosis))

svm_raw <- svm(Diagnosis ~ ., data=train_raw, scale = FALSE, kernel =
"radial", type = "C-classification")
#svm_pca <- svm(Diagnosis ~ ., data=Data_pca)
svm_raw$gamma

pred_svm <- predict(svm_raw, test_raw)

cm_svm <- confusionMatrix(pred_svm, as.factor(test_raw$Diagnosis), positive = "M")
cm_svm
#summary(svm_raw)
svm_drop_raw <- svm(Diagnosis ~ ., data=train_drop_raw, scale = FALSE, kernel =
"radial", type = "C-classification")
#svm_pca <- svm(Diagnosis ~ ., data=Data_pca)
svm_drop_raw$gamma

pred_drop_svm <- predict(svm_drop_raw, test_drop_raw)

cm_drop_svm <- confusionMatrix(pred_drop_svm, as.factor(test_drop_raw$Diagnosis), positive = "M"
)
cm_drop_svm
#fitControl <- trainControl(method = "none", classProbs = TRUE)
fitControl <- trainControl(method="repeatedcv",
                           number = 5,
                           repeats =5,
                           preProcOptions = list(thresh = 0.8), # threshold for pca preprocess
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

set.seed(825)
svm_raw_caret <- train(Diagnosis~.,
                      data = train_raw,
                      method="svmRadial",
                      metric="ROC",
                      trace=FALSE,

```

```

        trControl=fitControl)
svm_raw_caret$finalModel
pred_svm_caret <- predict(svm_raw_caret, test_raw)
cm_svm_caret <- confusionMatrix(pred_svm_caret, as.factor(test_raw$Diagnosis), positive = "M")
cm_svm_caret

#fitControl <- trainControl(method = "none", classProbs = TRUE)
fitControl <- trainControl(method="repeatedcv",
                           number = 5,
                           repeats =5,
                           preProcOptions = list(thresh = 0.8), # threshold for pca preprocess
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

set.seed(825)
svm_drop_caret <- train(Diagnosis~.,
                        data = train_drop_raw,
                        method="svmRadial",
                        metric="ROC",
                        trace=FALSE,
                        trControl=fitControl)
svm_drop_caret$finalModel
pred_drop_svm_caret <- predict(svm_drop_caret, test_drop_raw)
cm_drop_svm_caret <- confusionMatrix(pred_drop_svm_caret, as.factor(test_drop_raw$Diagnosis), positive = "M")
cm_drop_svm_caret

fitControl <- trainControl(method="repeatedcv",
                           number = 5,
                           repeats =5,
                           preProcOptions = list(thresh = 0.8), # threshold for pca preprocess
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

set.seed(825)
svm_scale <- train(Diagnosis~.,
                  data = train_raw,
                  method="svmRadial",
                  metric="ROC",
                  preProcess=c('center', 'scale'),
                  trace=FALSE,
                  trControl=fitControl)
svm_scale$finalModel
pred_svm_scale <- predict(svm_scale, test_raw)
cm_svm_scale <- confusionMatrix(pred_svm_scale, as.factor(test_raw$Diagnosis), positive = "M")
cm_svm_scale

fitControl <- trainControl(method="repeatedcv",
                           number = 5,
                           repeats =5,
                           preProcOptions = list(thresh = 0.8), # threshold for pca preprocess
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

set.seed(825)
svm_scale_pca <- train(Diagnosis~.,

```

```

        data = train_raw,
        method="svmRadial",
        metric="ROC",
        preProcess=c('center', 'scale','pca'),
        trace=FALSE,
        trControl=fitControl)
svm_scale_pca$finalModel
pred_svm_scale_pca <- predict(svm_scale_pca, test_raw)
cm_svm_scale_pca <- confusionMatrix(pred_svm_scale_pca, as.factor(test_raw$Diagnosis), positive
= "M")
cm_svm_scale_pca

fitControl <- trainControl(method="repeatedcv",
                           number = 5,
                           repeats =5,
                           preProcOptions = list(thresh = 0.95), # threshold for pca preprocess
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

set.seed(825)
svm_drop_pca <- train(Diagnosis~.,
                     data = train_drop_raw,
                     method="svmRadial",
                     metric="ROC",
                     preProcess=c('center', 'scale','pca'),
                     trace=FALSE,
                     trControl=fitControl
                     #tuneLength = 8
                     )

svm_drop_pca
pred_svmdrop_pca <- predict(svm_drop_pca, train_drop_raw)
cm_svmdrop_pca <- confusionMatrix(pred_svmdrop_pca, as.factor(train_drop_raw$Diagnosis), positiv
e = "M")
cm_svmdrop_pca

svmGrid <- expand.grid(sigma = c(.01, .015, 0.2,0.25, 0.275,0.3),
                      C = c(0.25, 0.5, 0.75, 0.9, 1, 1.1, 1.25))

nrow(svmGrid)
fitControl <- trainControl(method="repeatedcv",
                           number = 5,
                           repeats =5,
                           preProcOptions = list(thresh = 0.8), # threshold for pca preprocess
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

set.seed(825)
svm_tune <- train(Diagnosis ~ .,
                 data = train_drop_raw,
                 method="svmRadial",
                 metric="ROC",
                 preProcess=c('center', 'scale','pca'),
                 trace=FALSE,
                 trControl=fitControl,
                 tuneGrid = svmGrid)

#svm_tune

```

*#ten fold The final values used for the model were  $\sigma = 0.015$  and  $C = 0.9$ .*  
*#five fold The final values used for the model were  $\sigma = 0.015$  and  $C = 0.75$ .*  
*#The final values used for the model were  $\sigma = 0.015$  and  $C = 1$ .*

```
plot(svm_tune)
pred_svm_tune <- predict(svm_tune, train_drop_raw)
cm_svm_tune <- confusionMatrix(pred_svm_tune, as.factor(train_drop_raw$Diagnosis), positive =
"M")
cm_svm_tune
```

*#Plotting the Resampling Profile: Examine the relationship between the estimates of performance and the tuning parameters*

```
fitControl <- trainControl(method="repeatedcv",
                           number = 5,
                           repeats =5,
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)
```

```
set.seed(825)
model_nnet <- train(Diagnosis~.,
                   data = train_drop_raw,
                   method="nnet",
                   metric="ROC",
                   preProcess=c('center', 'scale'),
                   trace=FALSE,
                   tuneLength=10,
                   trControl=fitControl)
```

*#model\_nnet*

```
pred_nnet <- predict(model_nnet, test_drop_raw)
cm_nnet <- confusionMatrix(pred_nnet, as.factor(test_drop_raw$Diagnosis), positive = "M")
```

*# #grid search for nnet*

```
# nnetGrid <- expand.grid(size = seq(from = 1, to = 10, by = 1),
#                          decay = seq(from = 0.1, to = 0.5, by = 0.1))
```

*cm\_nnet*

```
set.seed(825)
```

```
fitControl <- trainControl(method="repeatedcv",
                           number = 5,
                           repeats =5,
                           preProcOptions = list(thresh = 0.8), # threshold for pca preprocess
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)
```

```
nnet_pca <- train(Diagnosis~.,
                 data = train_raw,
                 method = "nnet",
                 metric = "ROC",
                 preProcess=c('center', 'scale','pca'),
                 trace=FALSE,
                 tuneLength=10,
                 trControl = fitControl)
```

*#nnet\_pca*

```
pred_nnet_pca <- predict(nnet_pca, test_raw)
cm_nnet_pca <- confusionMatrix(pred_nnet_pca, as.factor(test_raw$Diagnosis), positive = "M")
cm_nnet_pca
```

```

cm_nnet_pca
model_list <- list(Caret_svm_scale= svm_scale, Caret_svm_scale_pca=svm_scale_pca,
                  Caret_drop_svm_pca=svm_drop_pca, Caret_nnet_pca=nnet_pca)
resamples <- resamples(model_list)
bwplot(resamples, metric = "ROC")
overall <- data.frame(model = rep(c("Svm_raw", "Svm_drop","Caret_svm_scale", "Caret_svm_scale_pca",
"Caret_drop_svm_pca", "Caret_nnet_pca")),
                    rbind(cm_svm$overall,
                          cm_drop_svm$overall,
                          cm_svm_scale$overall,
                          cm_svm_scale_pca$overall,
                          cm_svmdrop_pca$overall,
                          cm_nnet_pca$overall))

#overall
overall_gather <- overall[,1:3] %>%
  gather(measure, value, Accuracy:Kappa)
#overall_gather
byClass <- data.frame(model = rep(c("Svm_raw", "Svm_drop","Caret_svm_scale", "Caret_svm_scale_pca",
"Caret_drop_svm_pca", "Caret_nnet_pca")),
                    rbind(cm_svm$byClass,
                          cm_drop_svm$byClass,
                          cm_svm_scale$byClass,
                          cm_svm_scale_pca$byClass,
                          cm_svmdrop_pca$byClass,
                          cm_nnet_pca$byClass))

#byClass
byClass_gather <- byClass[,1:3] %>%
  gather(measure, value, Sensitivity:Specificity)
#byClass_gather

overall_byClass_gather <- rbind(overall_gather, byClass_gather)
overall_byClass_gather <- within(overall_byClass_gather, model <- factor(model, levels = c("Svm_raw",
"Svm_drop","Caret_svm_scale", "Caret_nnet_pca", "Caret_svm_scale_pca", "Caret_drop_svm_pca")))

#overall_byClass_gather
OV<-overall_byClass_gather %>%
  filter(measure == "Accuracy") %>%
  arrange(value)

#OV

Per<-ggplot(OV, aes(x = model, y = value)) +
  geom_point(shape=1) +
  scale_x_discrete(labels = abbreviate) +
  scale_y_continuous(breaks=c(0.6,0.8, 0.94, 0.96,0.97,0.98, 0.99))

Performance <- ggplotly(Per) %>%
  layout(title = 'Interactive Model Performance',
        xaxis = list(title = 'Model'),
        yaxis = list (title = 'Accuracy'))
Performance

```