

# Your Compact Report

Your Name

November 05, 2025

## Contents

<b>1</b>	<b>Load data</b>	<b>1</b>
<b>2</b>	<b>EDA, pre-processing and data cleaning</b>	<b>3</b>
2.1	Missing values . . . . .	3
2.2	Outliers, features distributions - scales . . . . .	3
2.3	Normalization . . . . .	4
2.4	Elimination of irrelevant variables . . . . .	4
2.5	Elimination of redundant variables . . . . .	5
2.6	Normalization of the variables . . . . .	7

## 1 Load data

```
## diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1          M      17.99      10.38     122.80    1001.0      0.11840
## 2          M      20.57      17.77     132.90    1326.0      0.08474
## 3          M      19.69      21.25     130.00    1203.0      0.10960
## 4          M      11.42      20.38      77.58     386.1      0.14250
## 5          M      20.29      14.34     135.10    1297.0      0.10030
## 6          M      12.45      15.70      82.57     477.1      0.12780
## compactness_mean concavity_mean concave_points_mean symmetry_mean
## 1        0.27760      0.3001      0.14710      0.2419
## 2        0.07864      0.0869      0.07017      0.1812
## 3        0.15990      0.1974      0.12790      0.2069
## 4        0.28390      0.2414      0.10520      0.2597
## 5        0.13280      0.1980      0.10430      0.1809
## 6        0.17000      0.1578      0.08089      0.2087
## fractal_dimension_mean radius_se texture_se perimeter_se area_se
## 1        0.07871     1.0950      0.9053      8.589    153.40
## 2        0.05667     0.5435      0.7339      3.398    74.08
## 3        0.05999     0.7456      0.7869      4.585    94.03
## 4        0.09744     0.4956      1.1560      3.445    27.23
## 5        0.05883     0.7572      0.7813      5.438    94.44
## 6        0.07613     0.3345      0.8902      2.217    27.19
## smoothness_se compactness_se concavity_se concave_points_se symmetry_se
## 1        0.006399    0.04904     0.05373     0.01587    0.03003
## 2        0.005225    0.01308     0.01860     0.01340    0.01389
## 3        0.006150    0.04006     0.03832     0.02058    0.02250
## 4        0.009110    0.07458     0.05661     0.01867    0.05963
## 5        0.011490    0.02461     0.05688     0.01885    0.01756
## 6        0.007510    0.03345     0.03672     0.01137    0.02165
## fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## 1        0.006193    25.38       17.33      184.60    2019.0
## 2        0.003532    24.99       23.41      158.80    1956.0
```

```

## 3      0.004571    23.57    25.53    152.50   1709.0
## 4      0.009208    14.91    26.50     98.87   567.7
## 5      0.005115    22.54    16.67    152.20   1575.0
## 6      0.005082    15.47    23.75    103.40   741.6
##   smoothness_worst compactness_worst concavity_worst concave_points_worst
## 1      0.1622     0.6656    0.7119     0.2654
## 2      0.1238     0.1866    0.2416     0.1860
## 3      0.1444     0.4245    0.4504     0.2430
## 4      0.2098     0.8663    0.6869     0.2575
## 5      0.1374     0.2050    0.4000     0.1625
## 6      0.1791     0.5249    0.5355     0.1741
##   symmetry_worst fractal_dimension_worst
## 1      0.4601     0.11890
## 2      0.2750     0.08902
## 3      0.3613     0.08758
## 4      0.6638     0.17300
## 5      0.2364     0.07678
## 6      0.3985     0.12440

##
## Structure of the clean data:

## 'data.frame': 569 obs. of 31 variables:
## $ diagnosis : Factor w/ 2 levels "B","M": 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 ...

```

## 2 EDA, pre-processing and data cleaning

### 2.1 Missing values

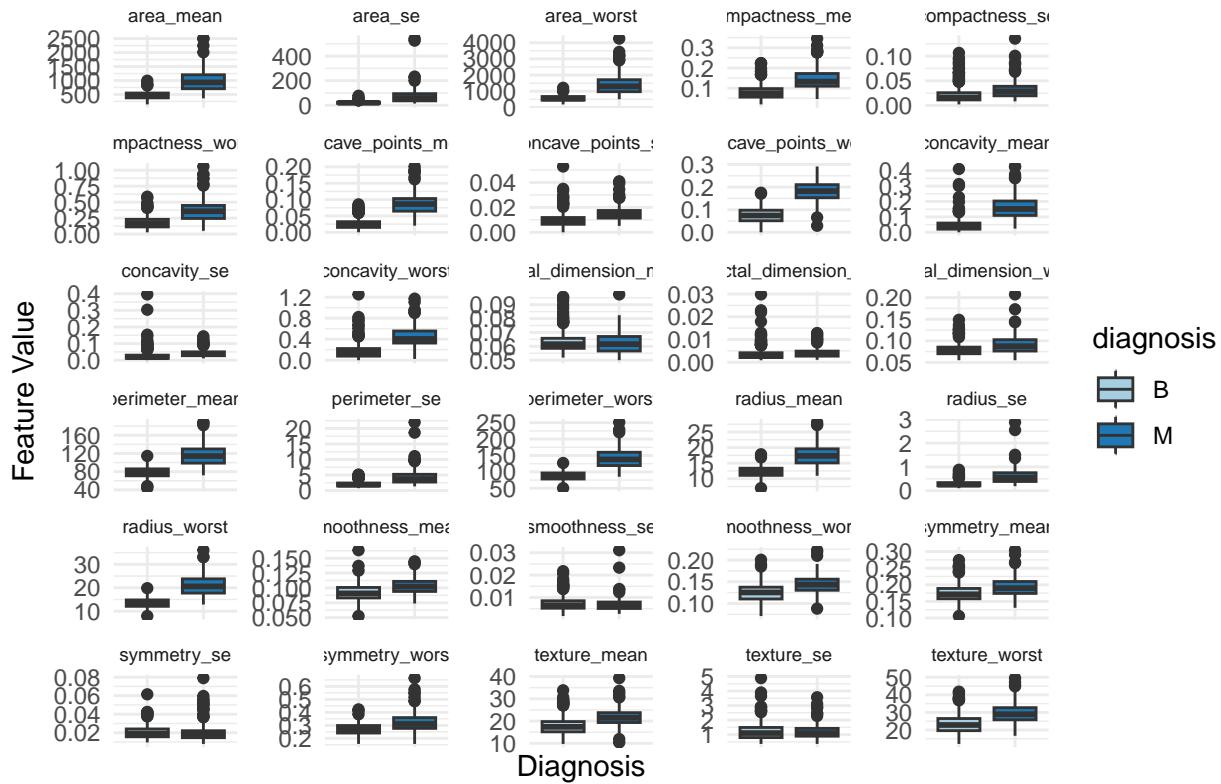
```
## Total missing values in the entire dataset: 0
```

### 2.2 Outliers, features distributions - scales

```
# To plot all 30 features at once, we first "pivot" the data
# into a "long" format. This is the standard tidyverse way.
data_long <- data_clean %>%
  pivot_longer(
    cols = -diagnosis,           # Pivot every column *except* 'diagnosis'
    names_to = "feature",        # New column for the feature name
    values_to = "value"          # New column for its value
  )

ggplot(data_long, aes(x = diagnosis, y = value, fill = diagnosis)) +
  geom_boxplot() +
  facet_wrap(~feature, scales = "free_y", ncol = 5) +
  scale_fill_brewer(palette = "Paired") +
  labs(
    title = "Feature Distributions by Diagnosis (Malignant vs. Benign)",
    x = "Diagnosis",
    y = "Feature Value"
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_blank(), # Hide x-axis labels for a cleaner look
    axis.ticks.x = element_blank(),
    strip.text = element_text(size = 7) # Smaller facet labels
  )
```

## Feature Distributions by Diagnosis (Malignant vs. Benign)



- Outliers (Anomalous Value):** We immediately observed the presence of numerous outliers, represented by individual observations outside the IQR range. These are particularly present in features like `area_mean`, `area_worst`, and `concavity_worst`. However, because those trust the original source, we know that these are not data entry errors but likely represent true, extreme biological values. Therefore, we made the decision not to remove these outliers, as they are part of the real-world problem. We will proceed with the full dataset, relying on our models to be robust enough to handle this variance.

### 2.3 Normalization

Our plot also highlights the vast difference in scales across features. Gradient-based models like the Perceptron and Logistic Regression (GLM) are sensitive to feature scales. Without normalization, features with larger magnitudes would dominate the learning process. This plot provides a clear justification for standardizing the features before modeling.

### 2.4 Elimination of irrelevant variables

For nearly every feature, the boxplot for the Malignant (M) class is visually distinct from the boxplot for the Benign (B) class. The medians, quartiles, and overall ranges show clear separation. This is an excellent sign, as it indicates that our features contain strong predictive information, and we can expect our models to perform well.

```
# With AI
nzv_check <- nearZeroVar(data_clean, saveMetrics = TRUE)

# Filter to see only the features that *might* be a problem
# A "TRUE" in the 'nzv' column means it's a candidate for removal.
problematic_features <- nzv_check[nzv_check$nzv == TRUE, ]

# Print the results
if (nrow(problematic_features) > 0) {
  cat("Found", nrow(problematic_features), "near-zero variance features:\n")
  print(problematic_features)
```

```

} else {
  cat("No near-zero variance features found. All features have sufficient variance.\n")
}

```

```
## No near-zero variance features found. All features have sufficient variance.
```

First, we checked for “irrelevant” features by running a “Near-Zero Variance” (NZV) test. This test finds any feature that is constant or nearly constant, as a feature that doesn’t change cannot be a predictor. We used the `caret::nearZeroVar()` function to check all 30 features.

we note that we used AI to help confirm this was the correct tool for this task.

The NZV test showed that all 30 features have enough variance, so we did not eliminate any features at this step.

## 2.5 Elimination of redundant variables

Our primary tool for this step was correlation analysis. We first calculated the correlation matrix for all numeric features and visualized it using a heatmap.

```

# Correlation matrix.
# numeric features selections
numeric_features <- data_clean %>%
  select_if(is.numeric)

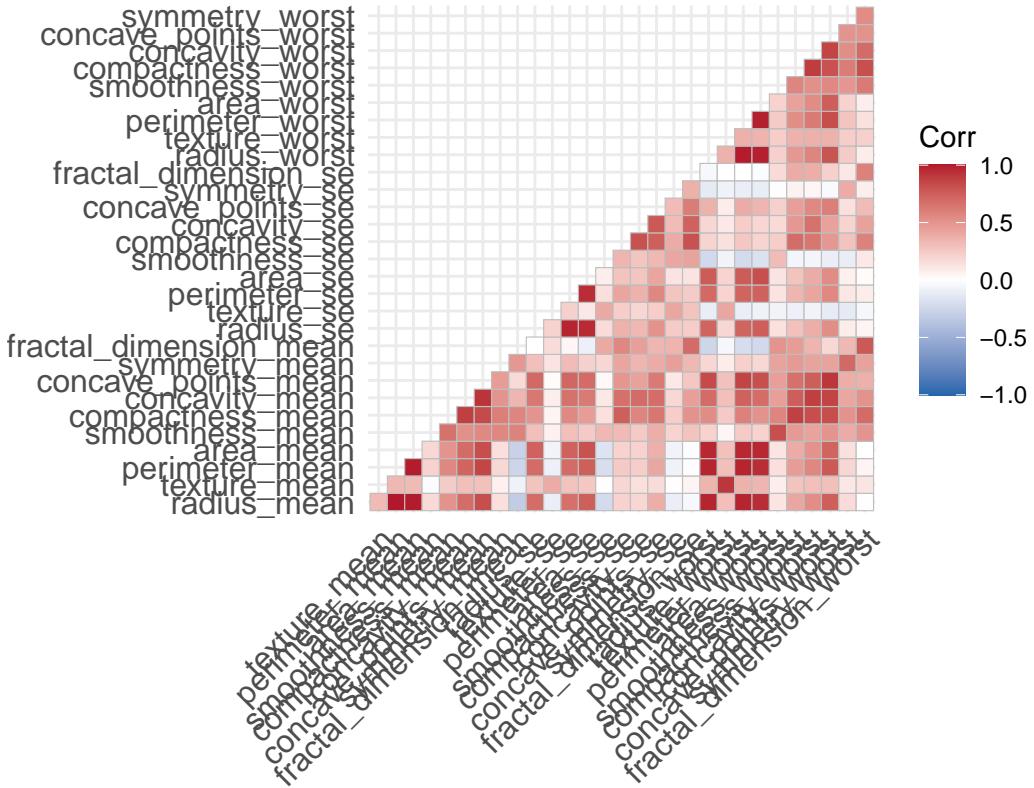
cor_matrix <- cor(numeric_features)

# Plot the correlation heatmap
ggcorrplot(cor_matrix,
  type = "lower",
  lab = FALSE,
  colors = c("#2166AC", "white", "#B2182B"),
  title = "Correlation Heatmap of 30 Numeric Features"
)

## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()``.
## i See also `vignette("ggplot2-in-packages")` for more information.
## i The deprecated feature was likely used in the ggcorrplot package.
## Please report the issue at <https://github.com/kassambara/ggcorrplot/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

```

### Correlation Heatmap of 30 Numeric Features



```
# Correlation and cutoff

highly_correlated_features <- findCorrelation(cor_matrix, cutoff = 0.90, names = TRUE)

cat("Found", length(highly_correlated_features), "redundant features to remove (cutoff > 0.90):\n")

## Found 10 redundant features to remove (cutoff > 0.90):
print(highly_correlated_features)

## [1] "concavity_mean"      "concave_points_mean" "perimeter_worst"
## [4] "radius_worst"        "perimeter_mean"       "area_worst"
## [7] "radius_mean"         "perimeter_se"        "area_se"
## [10] "texture_mean"

# Create the reduced Dataset

data_clean_reduced <- data_clean %>%
  select(-all_of(highly_correlated_features))

cat("\nOriginal data had", ncol(data_clean), "columns (30 features + 1 target).\n")

##
## Original data had 31 columns (30 features + 1 target).
cat("Reduced data has", ncol(data_clean_reduced), "columns.\n")

## Reduced data has 21 columns.
```

```

head(data_clean_reduced)

## diagnosis area_mean smoothness_mean compactness_mean symmetry_mean
## 1      M 1001.0        0.11840        0.27760        0.2419
## 2      M 1326.0        0.08474        0.07864        0.1812
## 3      M 1203.0        0.10960        0.15990        0.2069
## 4      M  386.1        0.14250        0.28390        0.2597
## 5      M 1297.0        0.10030        0.13280        0.1809
## 6      M  477.1        0.12780        0.17000        0.2087
## fractal_dimension_mean radius_se texture_se smoothness_se compactness_se
## 1          0.07871    1.0950      0.9053      0.006399      0.04904
## 2          0.05667    0.5435      0.7339      0.005225      0.01308
## 3          0.05999    0.7456      0.7869      0.006150      0.04006
## 4          0.09744    0.4956      1.1560      0.009110      0.07458
## 5          0.05883    0.7572      0.7813      0.011490      0.02461
## 6          0.07613    0.3345      0.8902      0.007510      0.03345
## concavity_se concave_points_se symmetry_se fractal_dimension_se texture_worst
## 1      0.05373      0.01587      0.03003      0.006193     17.33
## 2      0.01860      0.01340      0.01389      0.003532     23.41
## 3      0.03832      0.02058      0.02250      0.004571     25.53
## 4      0.05661      0.01867      0.05963      0.009208     26.50
## 5      0.05688      0.01885      0.01756      0.005115     16.67
## 6      0.03672      0.01137      0.02165      0.005082     23.75
## smoothness_worst compactness_worst concavity_worst concave_points_worst
## 1      0.1622       0.6656       0.7119       0.2654
## 2      0.1238       0.1866       0.2416       0.1860
## 3      0.1444       0.4245       0.4504       0.2430
## 4      0.2098       0.8663       0.6869       0.2575
## 5      0.1374       0.2050       0.4000       0.1625
## 6      0.1791       0.5249       0.5355       0.1741
## symmetry_worst fractal_dimension_worst
## 1      0.4601       0.11890
## 2      0.2750       0.08902
## 3      0.3613       0.08758
## 4      0.6638       0.17300
## 5      0.2364       0.07678
## 6      0.3985       0.12440

```

While the heatmap is excellent for visualization, we use a correlation matrix and a cutoff of 0.90 (standard threshold) for reducing feature redundancy. This function automatically identifies the minimum number of features to remove to ensure that no two features in the remaining dataset have a correlation greater than 0.90.

This analysis identified 10 features as redundant.

## 2.6 Normalization of the variables

```

# --- 6. Final Pre-Processing: Data Splitting and Normalization ---
# This is Point 8 from the project guidelines.

# Set a seed for 100% reproducible results (as per teacher feedback)
set.seed(123)

# --- 6.1. Process the FULL 30-feature dataset ---

# 1. Split 'data_clean' (Full dataset)
train_index_full <- createDataPartition(data_clean$diagnosis, p = 0.80, list = FALSE)
train_data_full   <- data_clean[train_index_full, ]

```

```

test_data_full      <- data_clean[-train_index_full, ]

# 2. "Fit" the normalization parameters (mean/sd) on the training data
# We exclude column 1 ('diagnosis') from the calculation
preproc_params_full <- preProcess(train_data_full[, -1], method = c("center", "scale"))

# 3. "Transform" both datasets using the *training* parameters
train_data_full_norm <- predict(preproc_params_full, train_data_full)
test_data_full_norm <- predict(preproc_params_full, test_data_full)

# --- 6.2. Process the REDUCED 20-feature dataset ---

# 1. Split 'data_clean_reduced'
train_index_reduced <- createDataPartition(data_clean_reduced$diagnosis, p = 0.80, list = FALSE)
train_data_reduced    <- data_clean_reduced[train_index_reduced, ]
test_data_reduced     <- data_clean_reduced[-train_index_reduced, ]

# 2. "Fit" normalization parameters on the *reduced* training data
# We exclude column 1 ('diagnosis') from the calculation
preproc_params_reduced <- preProcess(train_data_reduced[, -1], method = c("center", "scale"))

# 3. "Transform" both reduced datasets
train_data_reduced_norm <- predict(preproc_params_reduced, train_data_reduced)
test_data_reduced_norm <- predict(preproc_params_reduced, test_data_reduced)

# --- 6.3. Verification ---
cat("--- Full Dataset (Normalized) ---\n")

## --- Full Dataset (Normalized) ---
print(head(train_data_full_norm[, 2:6])) # Show first few features

##      radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## 1  1.1241421   -2.0986642    1.2990423  1.0150778   1.5774897
## 2  1.8576695   -0.3551160    1.7159119  1.9472293  -0.8034854
## 3  1.6074741    0.4659323    1.5962166  1.5944458   0.9550126
## 4  -0.7437938    0.2606702   -0.5673774  -0.7485529   3.2822283
## 5  1.7780619   -1.1643677    1.8067151  1.8640527   0.2971674
## 6  -0.4509515   -0.8434983   -0.3614191  -0.4875505   2.2424085

cat("\n--- Reduced Dataset (Normalized) ---\n")

##
## --- Reduced Dataset (Normalized) ---
print(head(train_data_reduced_norm[, 2:6])) # Show first few features

##      area_mean smoothness_mean compactness_mean symmetry_mean
## 1  0.9677074    1.5017818    3.2046013   2.20651622
## 2  1.8814951   -0.8169078   -0.4777606  -0.03664695
## 3  1.5356616    0.8955884    1.0262037   0.91309429
## 4  -0.7611790    3.1619249    3.3212020   2.86431365
## 5  1.7999571    0.2549523    0.5246355  -0.04773343
## 6  -0.5053184    2.1493065    1.2131350   0.97961313
##      fractal_dimension_mean
## 1                  2.2132800
## 2                 -0.8651498

```

```
## 3 -0.4014299
## 4 4.8293866
## 5 -0.5634525
## 6 1.8529193
```