Cheat Sheets and 5 ChatGPT Quiz Questions

Due Mar 3 by 11:59pm **Points** 5 **Submitting** a file upload **Available** until Mar 5 at 11:59pm

This assignment was locked Mar 5 at 11:59pm.

Upload 2 pages of visual notes/cheat sheets and 5 ChatGPT Quiz Questions and answers related to the exam. Two random student-generated ChatGPT quiz questions will be used assuming the question is relevant and the answers are correct.

Cheat Sheet

Linear Regression

Simple linear regression

$$y = \beta_0 + \beta_1 x + \varepsilon$$

For multiple regression $x = x_1, x_2 \cdots x_n$

RSS → ~~Residual~~ Residual Squared Sum

$$\sum_{i=1}^{N} \varepsilon_i^2 = \sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_i)^2$$

$$\left.\begin{aligned} \hat{\beta}_1 &= \sum_{i=1}^{N} (y_i - \bar{y})(x_i - \bar{x}) \Big/ \sum_{i=1}^{N} (x_i - \bar{x})^2 \\ \hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x} \end{aligned}\right\} \rightarrow \text{Least squares estimates}$$

* Correlation measures the amount of linear dependence/relationship between two variables

Pearson correlation → r $\quad -1 < r < +1$

Standard Error (SE)

95% confidence interval → $\hat{\beta}_1 \pm 2SE(\hat{\beta}_1)$

$t = \hat{\beta}_1 - 0 / SE(\hat{\beta}_1)$ (This will have a t-distribution with n-2 degrees of freedom)

Probability of observing any value equal to $|t|$ or larger is called p-value

To reject null hypothesis, $p > 0.05$ for simple linear regression.

For multiple regression,

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \cdots$$

For hypothesis testing, F statistic is used

$$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)}$$

$R^2 = 1 - SS_{residuals}/SS_{total}$ $\qquad 0 \geq R^2 \geq 1$

Q-Q plot used to check for normality

Regularization

It is the process of adding information in order to solve an ill-posed problem or to prevent overfitting.

$$\min_f \sum_{i=1}^{n} V(f(x_i), y_i) + \lambda R(f)$$

V is the underlying loss function that describes the cost of predicting f(x) when the label is y, λ is a parameter which controls the importance of the regularization term. R(f) is typically chosen to impose a penalty on complexity of f.

Mallows Cp

$$C_p = \frac{1}{n}(RSS + 2d\sigma^2)$$

d → total no of parameters, σ^2 → estimate of the variance

AIC

$$-2 \log L + 2 \cdot d$$

BIC

$$BIC = \frac{1}{n} (RSS +$$

Adjusted $R^2 = 1 - RSS/(n-d-1) \Big/ TSS(n-1)$

Ridge Regression

$$\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

$(\lambda \geq 0)$

Lasso

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

AutoML

Due Feb 28 by 11:59pm **Points** 100 **Submitting** a file upload **Available** until Apr 15 at 11:59pm

This assignment was locked Apr 15 at 11:59pm.

AutoML

In this assignment, you will an AutoML library like H2O.ai to create predictive models and interpret them. Find a significant relation for each algorithm of your choosing in your data. Create multivariate models.

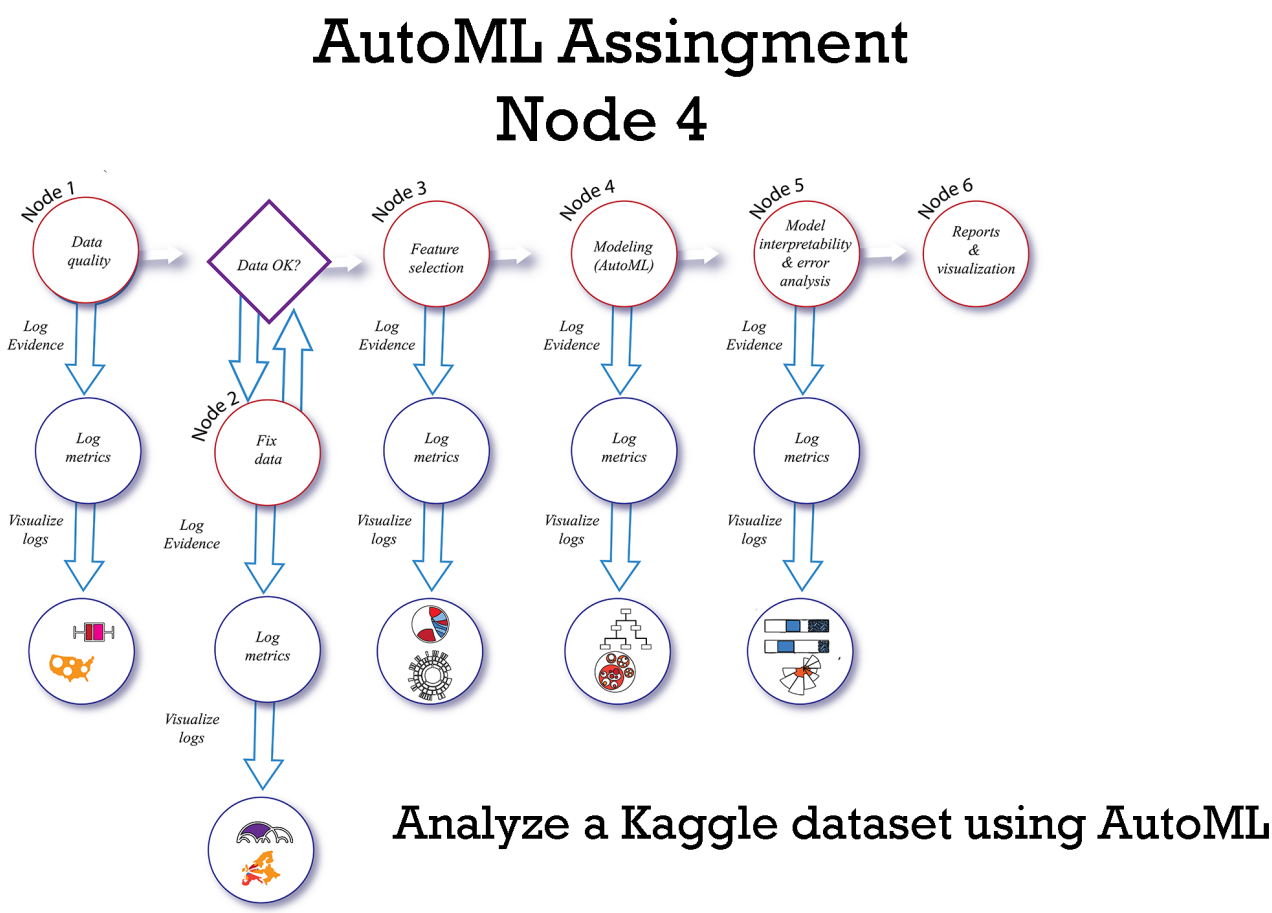
For the moment you will assume the data is good. In future assignments, you will check your data, fix data issues and do some feature engineering.

Starter code for AutoML is in this folder **https://github.com/aiskunks/YouTube/tree/main/A_Crash_Course_in_Statistical_Learning/AutoML (https://github.com/aiskunks/YouTube/tree/main/A_Crash_Course_in_Statistical_Learning/AutoML)**

In particular, this notebook should get you started

https://github.com/aiskunks/YouTube/blob/main/A_Crash_Course_in_Statistical_Learning/AutoML/CC_Kaggle_AutoML_Regression_Melbourn (https://github.com/aiskunks/YouTube/blob/main/A_Crash_Course_in_Statistical_Learning/AutoML/CC_Kaggle_AutoML_Regression_Melbourne_Housing.ipy

This assignment will focus on Node 4 of the simple analysis sequence.



Answer the following questions for all of the models:

* Is the relationship significant?

* Are any model assumptions violated?

* Is there any multicollinearity in the model?

* In the multivariate models are predictor variables independent of all the other predictor variables?

* In in multivariate models rank the most significant predictor variables and exclude insignificant ones from the model.

* Does the model make sense?

* Does regularization help?

* Which independent variables are significant?

* Which hyperparameters are important?

Scoring Rubric

Did I explain my idea clearly? (10 Points)

How effective are you at explaining what you are doing? You MUST write an abstract and a conclusion.

Did I give my data to the TA to add to ***https://github.com/nikbearbrown/AI_Research_Group/tree/main/Kaggle_Datasets*** ***(https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fnikbearbrown%2FAI_Research_Group%2Ftree%2Fmain%2FKaggle_Datasets&data=04%7C01%7Cgupta.shes%40northe*** *? (10 Points)*

A description of the data set and its fields must be given to a TA in markdown to be placed at **https://github.com/nikbearbrown/AI_Research_Group/tree/main/Kaggle_Datasets (https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fnikbearbrown%2FAI_Research_Group%2Ftree%2Fmain%2FKaggle_Datasets&data=04%7C01%7Cgupta.shes%40northe** under the appropriate directory (binary classification, multiclass classification, regression). If the dataset is greater than 50Mb a randomly sampled version just under 50Mb must be created and that is noted in the markdown description.

You MUST get approval for your dataset from the TAs.

Did I explain my evaluation clearly? (15 Points)

Just saying "accuracy" is not a clear explanation of an evaluation scheme. Clearly explain the evaluation scheme. Do the metrics make sense? You MUST explain how you are preventing overfitting.

Tables, graphs, and charts must support your evaluation.

It MUST run in Google Collab. You will also save the Google Collab notebook as a .ipynb notebook and upload that to Canvas . (5 Points)

What code is yours and what have you adapted? (10 Points)

You must explain what code you wrote and what you have done that is different. Failure to cite ANY code will result in a zero for this section.

Did I explain my code clearly? (15 Points) Your code review score will be scaled to a range of 0 to 10 and be used for this score.

Did I explain my licensing clearly? (5 Points) Failure to cite a clear license will result in a zero for this section.

Answers to listed questions (30 Points)

Answer the following questions for all of the models:

* Is the relationship significant?

* Are any model assumptions violated?

* Is there any multicollinearity in the model?

* In the multivariate models are predictor variables independent of all the other predictor variables?

* In in multivariate models rank the most significant predictor variables and exclude insignificant ones from the model.

* Does the model make sense?

* Does regularization help?

* Which independent variables are significant?

* Which hyperparameters are important?

* Coding professionalism?

AutoML (1) (3)

December 16, 2023

1 AutoML

AutoML is the process of automating the time-intensive practises of machine learning and predictive modelling.An AutoML tool aids the user by automating selection, composition and parameterization of machine learning models. This makes development a user-friendly and simple process,enabling an increase in scale,efficiency and productivity whilst upholding model quality.

H2O is a fully open source distributed in-memory machine learning platform which provides functionalities such as hyperparameter tuning and algorithm availability to produce a detailed output of the best performing models,for the given data.

[1]:
```
#installing the necessary libraries
!pip install h2o
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: h2o in /usr/local/lib/python3.8/dist-packages
(3.40.0.1)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-
packages (from h2o) (2.25.1)
Requirement already satisfied: tabulate in /usr/local/lib/python3.8/dist-
packages (from h2o) (0.8.10)
Requirement already satisfied: future in /usr/local/lib/python3.8/dist-packages
(from h2o) (0.16.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.8/dist-packages (from requests->h2o) (1.26.14)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.8/dist-packages (from requests->h2o) (2022.12.7)
Requirement already satisfied: chardet<5,>=3.0.2 in
/usr/local/lib/python3.8/dist-packages (from requests->h2o) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-
packages (from requests->h2o) (2.10)
```

[2]:
```
#package imports
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
```

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
import h2o
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.metrics import roc_auc_score,log_loss
from h2o.estimators import H2ORandomForestEstimator
from h2o.automl import H2OAutoML
from sklearn.ensemble import RandomForestClassifier
```

[3]:
```python
#initializing the h2o cluster
h2o.init()
```

```
Checking whether there is an H2O instance running at http://localhost:54321.
connected.

--------------------------  ----------------------------------
H2O_cluster_uptime:         1 hour 46 mins
H2O_cluster_timezone:       Etc/UTC
H2O_data_parsing_timezone:  UTC
H2O_cluster_version:        3.40.0.1
H2O_cluster_version_age:    20 days
H2O_cluster_name:           H2O_from_python_unknownUser_90kgb6
H2O_cluster_total_nodes:    1
H2O_cluster_free_memory:    3.060 Gb
H2O_cluster_total_cores:    2
H2O_cluster_allowed_cores:  2
H2O_cluster_status:         locked, healthy
H2O_connection_url:         http://localhost:54321
H2O_connection_proxy:       {"http": null, "https": null}
H2O_internal_security:      False
Python_version:             3.8.10 final
--------------------------  ----------------------------------
```

Abstract

Fine Needle Analysis(FNA) is a minimal invasive biopsy procedure which involves collecting breast tissue or fluid, used to detect the presence of cancerous cells. This notebook aims to provide various insights into the biopsy data by examining aspects such as:

- Statistical measures of distribution for predictor and response variables
- Correlation between independent features
- Correlation between each independent feature and the dependent feature

- Data Imputation(if required)

- Outlier Analysis

2 Dataset

FNA biopsy observations of 569 patients have been compiled into a dataset,with each row containing a patient's measurement variables and the pertaining diagnosis of breast cancer. Each indicator is represented by mean, standard error and mean of three largest readings. The indicators used are as follows:
* Mean of distances from the center to perimeter * Standard deviation of the gray-scale values pertaining to the biopsy image * Perimeter of cell nucleus * Area of cell nucleus * Variation of smoothness in radius lengths * Compactness of cell nucleus * Concavity severity * Number of Concave portions of the contour * Symmetry of the cell nucleus * Fractal Dimension of cell membrane

These indicators map to *radius*,*texture*,*perimeter*,*area*,*smoothness*,*compactness*,*concavity*,*concave points*,*symmetry* and *fractal_dimension* feature classes respectively. As mentioned previously, each feature class is represented by 3 values, resulting in **thirty independent features** of the dataset. It can be inferred that this is a case of binary classification.
The indicator of cancer-diagnosis(malignant or benign) is interpreted as **one dependent feature** of the dataset(represented by *diagnosis*).This is a categorical feature with two possible categories(M and B).
Note: An ID is assigned to each patient's diagnosis, inclusion of which can be ignored in our study.

[4]:
```
#Loading the dataset as a "H2O Frame"
df=h2o.import_file("https://raw.githubusercontent.com/SKashyap1999/DataScience/
↪main/Breast_Cancer-Wisconsin.csv")
```

```
Parse progress:
|                                        | (done) 100%
```

[5]:
```
df.describe(chunk_summary=True)
```

```
Rows:569
Cols:32

Chunk compression summary:
chunk_type    chunk_name        count    count_percentage    size    ␣
↪size_percentage
------------  ----------------  -------  ------------------  -------  ␣
↪-----------------
CBS           Binary            8        3.125               632  B   0.951435
C2S           2-Byte Fractions  179      69.9219             39.6 KB  60.9731
C4            4-Byte Integers   8        3.125               2.8 KB   4.24533
C4S           4-Byte Fractions  61       23.8281             21.9 KB  33.8301

Frame distribution summary:
                 size     number_of_rows    number_of_chunks_per_column   ␣
↪number_of_chunks
---------------  -------  ----------------  ----------------------------- ␣
↪------------------
127.0.0.1:54321  64.9 KB  569               8                              256
```

```
mean             64.9 KB  569              8                              256
min              64.9 KB  569              8                              256
max              64.9 KB  569              8                              256
stddev           0  B     0                0                              0
total            64.9 KB  569              8                              256


        id                  diagnosis    radius_mean         texture_mean      ␣
↪ perimeter_mean      area_mean          smoothness_mean       compactness_mean␣
↪     concavity_mean       concave points_mean    symmetry_mean       ␣
↪fractal_dimension_mean    radius_se           texture_se          perimeter_se␣
↪       area_se           smoothness_se         compactness_se       ␣
↪concavity_se          concave points_se     symmetry_se          ␣
↪fractal_dimension_se    radius_worst       texture_worst      perimeter_worst ␣
↪   area_worst         smoothness_worst      compactness_worst   ␣
↪concavity_worst      concave points_worst    symmetry_worst      ␣
↪fractal_dimension_worst
-------  ------------------  -----------  ------------------  ------------------␣
↪ ------------------  -----------------  -------------------- ␣
↪--------------------  -------------------  --------------------- ␣
↪-------------------  ------------------------  ------------------ ␣
↪------------------  ------------------  ----------------  --------------------␣
↪ --------------------  --------------------  --------------------- ␣
↪--------------------  ----------------------  ----------------- ␣
↪-----------------  ------------------  -----------------  --------------------␣
↪ -------------------  -------------------  ---------------------- ␣
↪-------------------  -------------------------
type     int                 enum         real                real              ␣
↪ real                real               real                  real            ␣
↪     real                 real                   real                 real    ␣
↪                real                real                real                ␣
↪real              real                  real                  real            ␣
↪     real                  real                  real                    real ␣
↪             real                real                real                real   ␣
↪              real                  real                 real                ␣
↪ real                 real
mins     8670.0                           6.981               9.71              ␣
↪ 43.79               143.5              0.05263               0.01938         ␣
↪     0.0                  0.0                    0.106                0.04996 ␣
↪                0.1115              0.3602              0.757               6.
↪802             0.001713              0.002252              0.0              ␣
↪   0.0                   0.007882              0.0008948               7.93   ␣
↪           12.02              50.41               185.2              0.07117  ␣
↪            0.02729              0.0                  0.0                    ␣
↪0.1565               0.05504
```

```
mean     30371831.432337433                14.127291739894554  19.289648506151142␣
↪ 91.96903339191564   654.8891036906855  0.0963602811950791    0.
↪1043409841827768    0.08879931581722321  0.04891914586994728    0.
↪18116186291739894  0.06279760984182778       0.4051720562390158  1.
↪2168534270650264  2.8660592267135323  40.337079086116   0.007040978910369069 ␣
↪0.025478138840070302  0.03189371634446398   0.011796137082601053  0.
↪020542298769771525  0.0037949038664323374   16.26918980667838  25.
↪67722319859402  107.26121265377857  880.5831282952549  0.13236859402460457   0.
↪2542650439367311   0.27218848330404216  0.11460622319859404     0.
↪2900755711775044   0.0839458172231986
maxs     911320502.0                       28.11                39.28              ␣
↪ 188.5               2501.0             0.1634                0.3454             ␣
↪     0.4268               0.2012                  0.304                0.09744 ␣
↪                  2.873               4.885               21.98                ␣
↪542.2             0.03113               0.1354                0.396             ␣
↪     0.05279               0.07895               0.02984                 36.04␣
↪              49.54               251.2               4254.0             0.2226 ␣
↪               1.058                1.252                0.291                 ␣
↪ 0.6638               0.2075
sigma    125020585.61222367                3.524048826212077   4.301035768166949 ␣
↪ 24.298981038754906  351.9141291816529  0.014064128137673621  0.
↪052812757932512194  0.07971980870789348  0.038802844859153605   0.
↪02741428133603572  0.007060362795084459      0.2773127329861039  0.
↪5516483926172023  2.0218545540421076  45.4910055161318  0.003002517943839066 ␣
↪0.017908179325677388  0.030186060322988408  0.006170285174046869  0.
↪008266371528798399  0.0026460709670891955   4.833241580469323  6.
↪146257623038319  33.602542269036356  569.3569926699489  0.02283242940483546 5 ␣
↪0.157336488913742    0.2086242806081323   0.06573234119594205     0.
↪06186746753751869  0.018061267348893986
zeros    0                                 0                   0                  ␣
↪ 0                   0                   0                     0                 ␣
↪     13                   13                     0                    0         ␣
↪                 0                   0                   0                   0␣
↪                0                    0                    13                    ␣
↪   13                    0                     0                       0        ␣
↪           0                   0                   0                   0        ␣
↪            0                    13                   13                        ␣
↪0                   0
missing  0                   0            0                   0                  ␣
↪ 0                   0                   0                     0                 ␣
↪     0                    0                      0                    0         ␣
↪                 0                   0                   0                   0␣
↪                0                    0                    0                     ␣
↪   0                     0                     0                       0        ␣
↪           0                   0                   0                   0        ␣
↪            0                    0                    0                        ␣
↪0                   0
```

```
0        842302.0            M            17.99                10.38
↪ 122.8               1001.0             0.1184                0.2776
↪     0.3001                0.1471                  0.2419                0.07871
↪                  1.095               0.9053              8.589
↪153.4             0.006399              0.04904               0.05373
↪     0.01587               0.03003                0.006193                25.38
↪             17.33              184.6               2019.0             0.1622
↪              0.6656               0.7119               0.2654
↪ 0.4601               0.1189
1        842517.0            M            20.57                17.77
↪ 132.9               1326.0             0.08474               0.07864
↪     0.0869                0.07017                 0.1812                0.05667
↪                  0.5435              0.7339              3.398
↪74.08             0.005225              0.01308               0.0186
↪     0.0134                0.01389                0.003532                24.99
↪             23.41              158.8               1956.0             0.1238
↪              0.1866               0.2416               0.186
↪ 0.275                0.08902
2        84300903.0          M            19.69                21.25
↪ 130.0               1203.0             0.1096                0.1599
↪     0.1974                0.1279                  0.2069                0.05999
↪                  0.7456              0.7869              4.585
↪94.03             0.00615               0.04006               0.03832
↪     0.02058               0.0225                 0.004571                23.57
↪             25.53              152.5               1709.0             0.1444
↪              0.4245               0.4504               0.243
↪ 0.3613               0.08758
3        84348301.0          M            11.42                20.38
↪ 77.58               386.1              0.1425                0.2839
↪     0.2414                0.1052                  0.2597                0.09744
↪                  0.4956              1.156               3.445
↪27.23             0.00911               0.07458               0.05661
↪     0.01867               0.05963                0.009208                14.91
↪             26.5               98.87               567.7              0.2098
↪              0.8663               0.6869               0.2575
↪ 0.6638               0.173
4        84358402.0          M            20.29                14.34
↪ 135.1               1297.0             0.1003                0.1328
↪     0.198                 0.1043                  0.1809                0.05883
↪                  0.7572              0.7813              5.438
↪94.44             0.01149               0.02461               0.05688
↪     0.01885               0.01756                0.005115                22.54
↪             16.67              152.2               1575.0             0.1374
↪              0.205                0.4                  0.1625
↪ 0.2364               0.07678
```

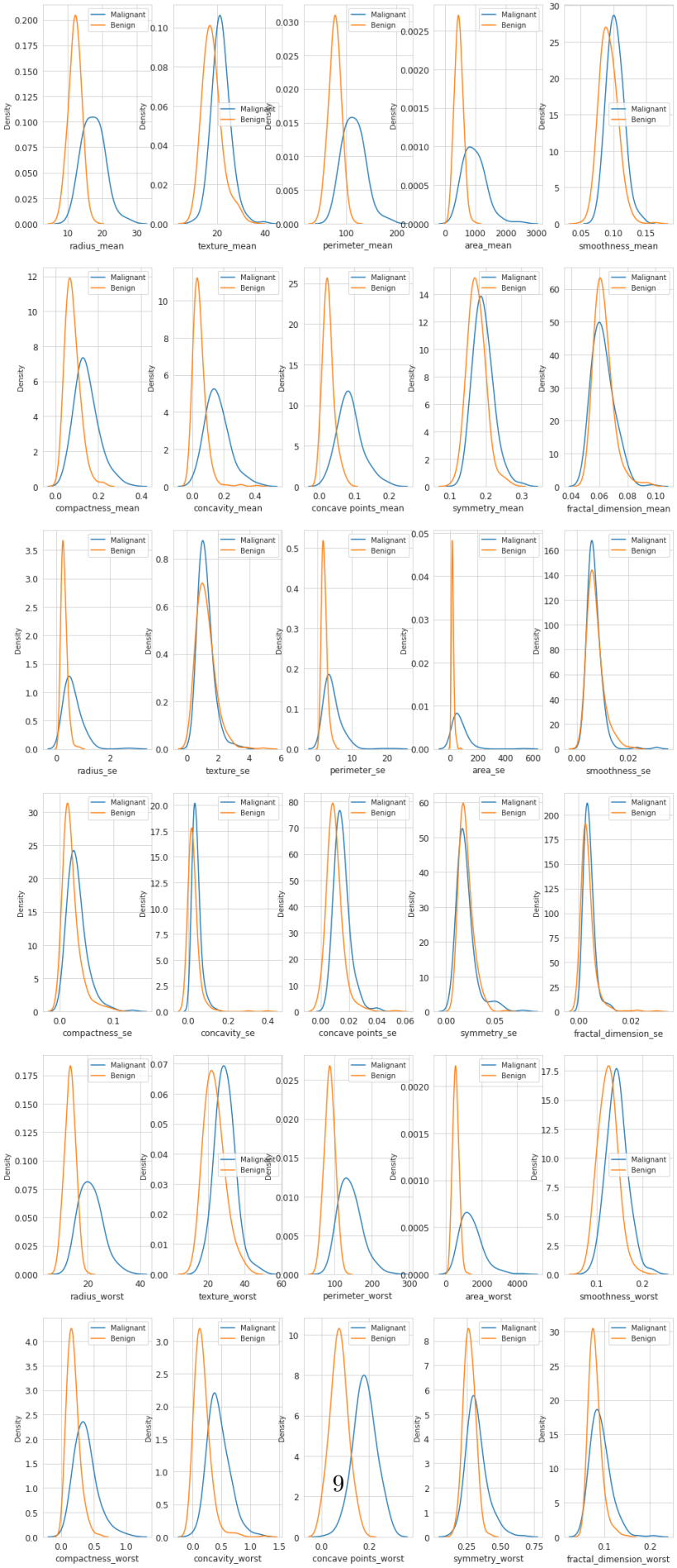
```
5        843786.0            M            12.45                15.7
↪ 82.57                477.1              0.1278                0.17
↪     0.1578                0.08089                0.2087                0.07613
↪                  0.3345              0.8902              2.217
↪27.19             0.00751               0.03345               0.03672
↪     0.01137               0.02165               0.005082                15.47
↪             23.75              103.4               741.6              0.1791
↪              0.5249               0.5355              0.1741
↪ 0.3985               0.1244
6        844359.0            M            18.25                19.98
↪ 119.6                1040.0             0.09463               0.109
↪     0.1127                0.074                0.1794                0.05742
↪                  0.4467              0.7732              3.18
↪53.91             0.004314              0.01382               0.02254
↪     0.01039               0.01369               0.002179                22.88
↪             27.66              153.2               1606.0             0.1442
↪              0.2576               0.3784              0.1932
↪ 0.3063               0.08368
7        84458202.0          M            13.71                20.83
↪ 90.2                 577.9              0.1189                0.1645
↪     0.09366               0.05985                0.2196                0.07451
↪                  0.5835              1.377               3.856
↪50.96             0.008805              0.03029               0.02488
↪     0.01448               0.01486               0.005412                17.06
↪             28.14              110.6               897.0              0.1654
↪              0.3682               0.2678              0.1556
↪ 0.3196               0.1151
8        844981.0            M            13.0                 21.82
↪ 87.5                 519.8              0.1273                0.1932
↪     0.1859                0.09353                0.235                 0.07389
↪                  0.3063              1.002               2.406
↪24.32             0.005731              0.03502               0.03553
↪     0.01226               0.02143               0.003749                15.49
↪             30.73              106.2               739.3              0.1703
↪              0.5401               0.539               0.206
↪ 0.4378               0.1072
9        84501001.0          M            12.46                24.04
↪ 83.97                475.9              0.1186                0.2396
↪     0.2273                0.08543                0.203                 0.08243
↪                  0.2976              1.599               2.039
↪23.94             0.007149              0.07217               0.07743
↪     0.01432               0.01789               0.01008                 15.09
↪             40.68              97.65               711.4              0.1853
↪              1.058                1.105               0.221
↪ 0.4366               0.2075
[569 rows x 32 columns]
```

The range of values across variable is seemingly very diverse, hence a scaling operation might be required.It is also observed that the minimum values of some columns is zero(columns pertaining to concavity and concave points). The common diagnosis for such occurences was observed to be benign(B), hinting at a proportional relationship.

The pdf of each predictor variable is computed, for each of the two diagnosis categories using the kdeplot to look for any discernable trends.

[6]:
```
df=df.drop(["id"],axis=1)
predictors=[f for f in df.columns if f not in ["diagnosis"]]
i = 0
t0 = df[df['diagnosis'] == 'M'].as_data_frame()
t1 = df[df['diagnosis'] == 'B'].as_data_frame()

sns.set_style('whitegrid')
plt.figure()
fig, ax = plt.subplots(6,5,figsize=(16,40))

for feature in predictors:
    i += 1
    plt.subplot(6,5,i)
    sns.kdeplot(t0[feature],bw_method=0.5,label='Malignant')
    sns.kdeplot(t1[feature],bw_method=0.5,label='Benign')
    plt.legend()
    plt.xlabel(feature, fontsize=12)
    locs, labels = plt.xticks()
    plt.tick_params(axis='both', which='major', labelsize=12)

plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```


Malignant
Benign
Density
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave points_worst
symmetry_worst
fractal_dimension_worst

By observing the above kde plots, it is inferred that most of the predictor variables have distributions that are overlapping. The realtionship between each of the predictor and target is not evident immediately, hence requiring further analysis.

[7]:
```
# Target variable class frequency
df_class_count=df.group_by("diagnosis").count()
df_class_count.get_frame()
```

[7]:
```
diagnosis      nrow
-----------  ------
B               357
M               212
[2 rows x 2 columns]
```

An enquiry into target class distribution shows that it is unbalanced. AutoML offers a hypeparameter known as "balance_classes" which undersamples or oversamples the dominant and non-dominant features respectively.

2.1 Multicollinearity

Multicollinearity refers to the situation where two or more independent variables (features) in a dataset have a strong correlation with each other.

[8]:
```
df_pd=df.as_data_frame(use_pandas=True)
#df_pd['diagnosis'] = df_pd['diagnosis'].map({'B':0, 'M':1})
df_pfeatures=df_pd.drop(["diagnosis"],axis=1)
vif_data = pd.DataFrame()
vif_data["feature"] = df_pfeatures.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(df_pfeatures.values, i)
                          for i in range(len(df_pfeatures.columns))]

print(vif_data)
```

```
                    feature           VIF
0               radius_mean  63306.172036
1              texture_mean    251.047108
2            perimeter_mean  58123.586079
3                 area_mean   1287.262339
4           smoothness_mean    393.398166
5          compactness_mean    200.980354
6            concavity_mean    157.855046
7       concave points_mean    154.241268
8             symmetry_mean    184.426558
9    fractal_dimension_mean    629.679874
```

```
10                radius_se    236.665738
11               texture_se     24.675367
12             perimeter_se    211.396334
13                  area_se     72.466468
14            smoothness_se     26.170243
15           compactness_se     44.919651
16             concavity_se     33.244099
17        concave points_se     53.698656
18              symmetry_se     37.176452
19     fractal_dimension_se     27.532631
20             radius_worst   9674.742602
21            texture_worst    343.004387
22          perimeter_worst   4487.781270
23               area_worst   1138.759252
24         smoothness_worst    375.597155
25        compactness_worst    132.884276
26          concavity_worst     86.310362
27     concave points_worst    148.673180
28           symmetry_worst    218.919805
29  fractal_dimension_worst    423.396723
```

The features are exhibiting very high multicollinearity. It is advisable to use models which are not affected by multicollinearity, such as decision trees,naive bayes and regularized linear models.

[9]:
```
# Training a RandomForestEstimator and observing the AUC and LogLoss metrics
train, valid = df.split_frame(ratios=[.8], seed=1234)
drf = H2ORandomForestEstimator(binomial_double_trees=True,balance_classes=True)
drf.train(x=list(df_pfeatures.columns),y="diagnosis",␣
 ↪training_frame=train,validation_frame=valid)
print("AUC and LogLoss values are "+str(drf.auc())+" , "+str(drf.logloss()))
```

```
drf Model Build progress:
|                                                      | (done) 100%
AUC and LogLoss values are 0.9975189562890278 , 0.11454204166410699
```

The H2ORandomForestEstimator class of H2O performs regularization by default.It uses several hyperparameters to control the regularization of the Random Forest model such as max_depth,min_rows and sample_rate.

Lets try to train a model on the RandomForestClassifier from sklearn.ensemble

[10]:
```
df_pd_target=df_pd['diagnosis'].map({'B':0,'M':1})

X=df_pfeatures
y=df_pd_target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

```
print("AUC and Logloss metrics of sklearn RandomForestClassifier is␣
↪"+str(roc_auc_score(y_test, y_pred))+" , "+str(log_loss(y_test, y_pred)))
```

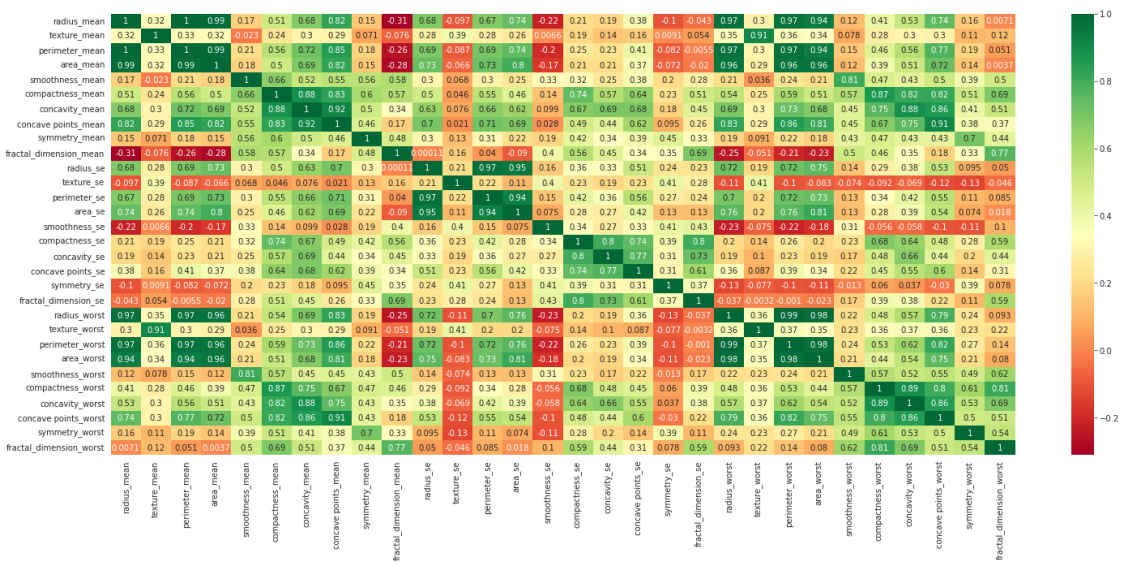
```
AUC and Logloss metrics of sklearn RandomForestClassifier is 0.9482051282051284
, 1.5148726418369638
```

We can observe that there is significant difference in AUC and LogLoss metrics when regularization is performed.(The h2o estimator has performed regularization by hyperparameter tuning)

2.1.1 HeatMap for correlation

[11]:
```
plt.figure(figsize=(25,10))
sns.heatmap(df_pd.corr(), annot=True, cmap='RdYlGn')
```

[11]: <AxesSubplot:>



2.2 Variable Significance

[12]:
```
model = sm.OLS(y, df_pfeatures).fit()
model.summary()
```

[12]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
==============================================================================
=======
Dep. Variable:               diagnosis   R-squared (uncentered):
0.853
```

```
Model:                            OLS   Adj. R-squared (uncentered):
0.844
Method:                 Least Squares   F-statistic:
103.9
Date:                Wed, 01 Mar 2023   Prob (F-statistic):
1.78e-202
Time:                        04:28:11   Log-Likelihood:
18.089
No. Observations:                 569   AIC:
23.82
Df Residuals:                     539   BIC:
154.1
Df Model:                          30
Covariance Type:            nonrobust
==============================================================================
===========
                              coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-----------
radius_mean                -0.3531      0.174     -2.024      0.043      -0.696
-0.010
texture_mean                0.0033      0.008      0.403      0.687      -0.013
0.019
perimeter_mean              0.0239      0.026      0.932      0.352      -0.026
0.074
area_mean                   0.0013      0.000      2.764      0.006       0.000
0.002
smoothness_mean             0.2626      2.056      0.128      0.898      -3.777
4.302
compactness_mean           -1.4791      1.224     -1.208      0.227      -3.884
0.926
concavity_mean              1.0316      1.063      0.970      0.332      -1.057
3.121
concave points_mean         3.0280      2.009      1.507      0.132      -0.918
6.974
symmetry_mean              -0.4348      0.748     -0.581      0.561      -1.905
1.035
fractal_dimension_mean    -18.6170      4.009     -4.644      0.000     -26.493
-10.741
radius_se                   0.3957      0.316      1.251      0.212      -0.226
1.017
texture_se                 -0.0008      0.038     -0.022      0.982      -0.075
0.073
perimeter_se               -0.0122      0.042     -0.292      0.770      -0.094
0.070
area_se                    -0.0017      0.001     -1.228      0.220      -0.005
```

```
0.001
smoothness_se              14.5322      6.749      2.153      0.032       1.276
27.789
compactness_se             -1.8382      2.173     -0.846      0.398      -6.108
2.431
concavity_se               -3.5134      1.326     -2.649      0.008      -6.118
-0.908
concave points_se          10.7609      5.559      1.936      0.053      -0.158
21.680
symmetry_se                 1.4318      2.780      0.515      0.607      -4.030
6.894
fractal_dimension_se        7.8849     11.454      0.688      0.492     -14.616
30.386
radius_worst                0.2336      0.059      3.993      0.000       0.119
0.349
texture_worst               0.0063      0.007      0.888      0.375      -0.008
0.020
perimeter_worst            -0.0054      0.006     -0.900      0.368      -0.017
0.006
area_worst                 -0.0011      0.000     -3.463      0.001      -0.002
-0.000
smoothness_worst           -0.0697      1.457     -0.048      0.962      -2.931
2.792
compactness_worst          -0.0775      0.389     -0.199      0.842      -0.842
0.687
concavity_worst             0.4385      0.274      1.603      0.110      -0.099
0.976
concave points_worst        0.5394      0.932      0.579      0.563      -1.291
2.370
symmetry_worst              0.6380      0.504      1.267      0.206      -0.351
1.627
fractal_dimension_worst     5.3423      2.420      2.208      0.028       0.589
10.095
==============================================================================
Omnibus:                       34.960   Durbin-Watson:                   1.837
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               39.782
Skew:                           0.617   Prob(JB):                     2.30e-09
Kurtosis:                       3.396   Cond. No.                     1.49e+06
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The condition number is large, 1.49e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

"""

The p-value of majority features is greater than 0.05, which hints at a revelation that they maybe insignificant.

3 Train and Test split

Train test split is a model validation procedure that allows you to simulate how a model would perform on new/unseen data. We reserve a fraction of the original dataset from training the model to test its performance on previously unseen data, after training the model.

[13]:
```
train, test = df.split_frame(ratios=[0.8], seed = 1)
```

4 Training AutoML Models

The H2OAutoML function is used to automate the supervised learning process. It returns an AutoML object, which contains a leaderboard of models which were trained on the data, based on a performance metric.

The AutoML interface is designed to have as few parameters as possible to acheive an uncomplicated implementation. It has parameters like: * balance_classes
Indicate if you want to oversample minority classes in order to balance the distribution of classes. * exclude_algos
List the algorithms to avoid while building the model. * max_models
Set a limit on the number of models that can be created during an AutoML run, excluding Stacked Ensemble models.

The train method initiates an AutoML job, a background process that generates several models using various algorithms automatically and monitors their performance in a scoreboard.

[14]:
```
#Initializing the H2OAutoML object with necessary hyperparameters and␣
↪performing the training operation
aml = H2OAutoML(max_models =25,exclude_algos=["DeepLearning"],sort_metric='auc',
                seed =1)
aml.train(training_frame=train,y='diagnosis')
```

```
AutoML progress:
|                                                      | (done) 100%
```

[14]:
```
Model Details
=============
H2OStackedEnsembleEstimator : Stacked Ensemble
Model Key: StackedEnsemble_BestOfFamily_1_AutoML_5_20230301_42812


Model Summary for Stacked Ensemble:
key                                      value
---------------------------------------  ----------------
Stacking strategy                        cross_validation
```

```
Number of base models (used / total)  4/5
# GBM base models (used / total)      1/1
# XGBoost base models (used / total)  1/1
# GLM base models (used / total)      1/1
# DRF base models (used / total)      1/2
Metalearner algorithm                 GLM
Metalearner fold assignment scheme    Random
Metalearner nfolds                    5
Metalearner fold_column
Custom metalearner hyperparameters    None

ModelMetricsBinomialGLM: stackedensemble
** Reported on train data. **

MSE: 0.005662005783955958
RMSE: 0.07524630079914864
LogLoss: 0.02843228604102565
AUC: 0.999661447312738
AUCPR: 0.9994926260108352
Gini: 0.999322894625476
Null degrees of freedom: 447
Residual degrees of freedom: 443
Null deviance: 594.7659284347429
Residual deviance: 25.47532829275898
AIC: 35.47532829275898

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.46297867597703835
       B    M    Error    Rate
-----  ---  ---  -------  -----------
B      278  0    0        (0.0/278.0)
M      1    169  0.0059   (1.0/170.0)
Total  279  169  0.0022   (1.0/448.0)

Maximum Metrics: Maximum metrics at their respective thresholds
metric                        threshold    value     idx
----------------------------  -----------  --------  -----
max f1                        0.462979     0.99705   133
max f2                        0.462979     0.995289  133
max f0point5                  0.462979     0.998818  133
max accuracy                  0.462979     0.997768  133
max precision                 1            1         0
max recall                    0.104154     1         150
max specificity               1            1         0
max absolute_mcc              0.462979     0.995266  133
max min_per_class_accuracy    0.462979     0.994118  133
max mean_per_class_accuracy   0.462979     0.997059  133
max tns                       1            278       0
```

```
max fns                     1            158       0
max fps                     2.64919e-06  278       399
max tps                     0.104154     170       150
max tnr                     1            1         0
max fnr                     1            0.929412  0
max fpr                     2.64919e-06  1         399
max tpr                     0.104154     1         150

Gains/Lift Table: Avg response rate: 37.95 %, avg score: 37.83 %
group    cumulative_data_fraction    lower_threshold    lift
cumulative_lift    response_rate    score        cumulative_response_rate
cumulative_score    capture_rate    cumulative_capture_rate    gain
cumulative_gain    kolmogorov_smirnov
-------  --------------------------  -----------------  ---------
-----------------  ---------------  -----------  --------------------------
------------------  --------------  -------------------------  --------
-----------------  --------------------
1        0.0111607                   1                  2.63529    2.63529
1                1            1                          1
0.0294118       0.0294118                  163.529   163.529
0.0294118
2        0.0200893                   1                  2.63529    2.63529
1                1            1                          1
0.0235294       0.0529412                  163.529   163.529
0.0529412
3        0.03125                     1                  2.63529    2.63529
1                1            1                          1
0.0294118       0.0823529                  163.529   163.529
0.0823529
4        0.0401786                   0.999999           2.63529    2.63529
1                0.999999     1                          1
0.0235294       0.105882                   163.529   163.529            0.105882
5        0.0513393                   0.999999           2.63529    2.63529
1                0.999999     1                          1
0.0294118       0.135294                   163.529   163.529            0.135294
6        0.100446                    0.999987           2.63529    2.63529
1                0.999994     1                          0.999997
0.129412        0.264706                   163.529   163.529            0.264706
7        0.151786                    0.999909           2.63529    2.63529
1                0.999962     1                          0.999985
0.135294        0.4                        163.529   163.529            0.4
8        0.200893                    0.999579           2.63529    2.63529
1                0.999786     1                          0.999936
0.129412        0.529412                   163.529   163.529            0.529412
9        0.301339                    0.976085           2.63529    2.63529
1                0.995562     1                          0.998478
0.264706        0.794118                   163.529   163.529            0.794118
```

```
10       0.399554                    0.14915            2.03636    2.48807
0.772727         0.722494     0.944134                    0.930638
0.2             0.994118                   103.636   148.807            0.958146
11       0.5                         0.0128536          0.0585621  2
0.0222222        0.0558341    0.758929                    0.754896
0.00588235      1                          -94.1438  100                0.805755
12       0.600446                    0.00276087         0          1.66543
0                0.00543321   0.63197                     0.629521            0
1                          -100      66.5428            0.643885
13       0.698661                    0.00100567         0          1.43131
0                0.00168829   0.543131                    0.541264            0
1                          -100      43.131             0.485612
14       0.799107                    0.000416434        0          1.2514
0                0.00068468   0.47486                     0.473314            0
1                          -100      25.1397            0.323741
15       0.899554                    0.000158494        0          1.11166
0                0.000275332  0.421836                    0.420493            0
1                          -100      11.1663            0.161871
16       1                           2.64919e-06        0          1
0                7.98402e-05  0.379464                    0.378264            0
1                          -100      0                  0

ModelMetricsBinomialGLM: stackedensemble
** Reported on cross-validation data. **

MSE: 0.02382567419402746
RMSE: 0.15435567431755615
LogLoss: 0.08775914847879622
AUC: 0.9940753279729159
AUCPR: 0.9924507004700177
Gini: 0.9881506559458317
Null degrees of freedom: 447
Residual degrees of freedom: 443
Null deviance: 595.8508793671208
Residual deviance: 78.63219703700142
AIC: 88.63219703700142

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.49034265320686354
       B    M    Error    Rate
-----  ---  ---  -------  ------------
B      272  6    0.0216   (6.0/278.0)
M      6    164  0.0353   (6.0/170.0)
Total  278  170  0.0268   (12.0/448.0)

Maximum Metrics: Maximum metrics at their respective thresholds
metric                       threshold    value     idx
---------------------------  -----------  --------  -----
```

```
max f1                        0.490343     0.964706  138
max f2                        0.400435     0.966042  142
max f0point5                  0.620917     0.977995  130
max accuracy                  0.620917     0.973214  130
max precision                 1            1         0
max recall                    0.00381516   1         244
max specificity               1            1         0
max absolute_mcc              0.620917     0.943289  130
max min_per_class_accuracy    0.400435     0.967626  142
max mean_per_class_accuracy   0.490343     0.971562  138
max tns                       1            278       0
max fns                       1            164       0
max fps                       8.48191e-08  278       399
max tps                       0.00381516   170       244
max tnr                       1            1         0
max fnr                       1            0.964706  0
max fpr                       8.48191e-08  1         399
max tpr                       0.00381516   1         244

Gains/Lift Table: Avg response rate: 37.95 %, avg score: 38.02 %
group    cumulative_data_fraction    lower_threshold    lift
cumulative_lift    response_rate    score        cumulative_response_rate
cumulative_score    capture_rate    cumulative_capture_rate    gain
cumulative_gain    kolmogorov_smirnov
-------  --------------------------  -----------------  --------
-----------------  ---------------  -----------  --------------------------
------------------  --------------  -------------------------  --------
-----------------  --------------------
1        0.0111607                   1                  2.63529   2.63529
1                1            1                          1
0.0294118       0.0294118                  163.529   163.529
0.0294118
2        0.0200893                   1                  2.63529   2.63529
1                1            1                          1
0.0235294       0.0529412                  163.529   163.529
0.0529412
3        0.03125                     0.999999           2.63529   2.63529
1                0.999999     1                          1
0.0294118       0.0823529                  163.529   163.529
0.0823529
4        0.0401786                   0.999998           2.63529   2.63529
1                0.999999     1                          0.999999
0.0235294       0.105882                   163.529   163.529            0.105882
5        0.0513393                   0.999997           2.63529   2.63529
1                0.999998     1                          0.999999
0.0294118       0.135294                   163.529   163.529            0.135294
6        0.100446                    0.999968           2.63529   2.63529
```

```
1                0.999987     1                           0.999993
0.129412        0.264706                    163.529   163.529            0.264706
7        0.151786                    0.999849           2.63529   2.63529
1                0.999922     1                           0.999969
0.135294        0.4                         163.529   163.529            0.4
8        0.200893                    0.999214           2.63529   2.63529
1                0.999606     1                           0.99988
0.129412        0.529412                    163.529   163.529            0.529412
9        0.301339                    0.963694           2.63529   2.63529
1                0.989884     1                           0.996548
0.264706        0.794118                    163.529   163.529            0.794118
10       0.399554                    0.308485           1.79679   2.42918
0.681818         0.691999     0.921788                    0.921687
0.176471        0.970588                    79.6791   142.918            0.920229
11       0.5                         0.0224055          0.234248  1.98824
0.0888889        0.104685     0.754464                    0.757557
0.0235294       0.994118                    -76.5752  98.8235            0.796276
12       0.600446                    0.00454649         0         1.65563
0                0.0103639    0.628253                    0.632562            0
0.994118                    -100      65.5631            0.634405
13       0.698661                    0.00156686         0.059893  1.43131
0.0227273        0.00275561   0.543131                    0.544027
0.00588235      1                           -94.0107  43.131             0.485612
14       0.799107                    0.000448398        0         1.2514
0                0.000945908  0.47486                     0.475762            0
1                           -100      25.1397            0.323741
15       0.899554                    0.000160003        0         1.11166
0                0.000298164  0.421836                    0.422671            0
1                           -100      11.1663            0.161871
16       1                           8e-08              0         1
0                8.22282e-05  0.379464                    0.380223            0
1                           -100      0                  0

Cross-Validation Metrics Summary:
                      mean         sd            cv_1_valid    cv_2_valid
cv_3_valid    cv_4_valid    cv_5_valid
--------------------  -----------  ------------  ------------  ------------
------------  ------------  ------------
accuracy              0.97979575   0.0053605754  0.989011      0.97959185
0.9767442     0.9756098     0.978022
auc                   0.9958996    0.0033730385  0.9989899     0.9934211
0.99765396    0.9981763     0.99125683
err                   0.020204252  0.0053605754  0.010989011   0.020408163
0.023255814   0.024390243   0.021978023
err_count             1.8          0.4472136     1.0           2.0           2.0
2.0           2.0
f0point5              0.96837467   0.016992733   0.9943182     0.9736842
```

```
0.9509202     0.95628417    0.96666664
f1                    0.97344774   0.0075009065  0.9859155     0.9736842
0.96875       0.9722222     0.96666664
f2                    0.97879326   0.009277866   0.9776536     0.9736842
0.9872612     0.98870057    0.96666664
lift_top_group        2.6514218    0.26300278    2.5277777     2.5789473
2.7741935     2.3428571     3.0333333
logloss               0.08182214   0.0195905     0.054581955   0.09690843
0.071641035   0.08268318    0.1032961
max_per_class_error   0.033268746  0.0066007157  0.027777778   0.02631579
0.036363635   0.04255319    0.033333335
---                   ---          ---           ---           ---           ---
---           ---
mean_per_class_error  0.019940384  0.0041273385  0.013888889   0.021491228
0.018181818   0.021276595   0.024863388
mse                   0.022814317  0.004051993   0.016539432   0.026925512
0.021253377   0.024563711   0.024789553
null_deviance         119.17017    7.631934      122.31455     130.92255
112.63653     113.07537     116.901886
pr_auc                0.9942155    0.0045734076  0.9985184     0.99145603
0.9958634     0.99760616    0.9876335
precision             0.96513814   0.024092661   1.0           0.9736842
0.93939394    0.9459459     0.96666664
r2                    0.90252686   0.018081514   0.9308267     0.8865822
0.90780646    0.8995949     0.8878239
recall                0.9825146    0.016175168   0.9722222     0.9736842     1.0
1.0           0.96666664
residual_deviance     14.722032    4.0284696     9.933916      18.994051
12.322259     13.560041     18.79989
rmse                  0.15053123   0.013904311   0.12860572    0.16408995
0.14578538    0.15672815    0.15744698
specificity           0.9776046    0.017116103   1.0           0.98333335
0.96363634    0.9574468     0.9836066
[22 rows x 8 columns]


[tips]
Use `model.explain()` to inspect the model.
--
Use `h2o.display.toggle_user_tips()` to switch on/off this section.
```

We may employ AUC and LogLoss metrics to judge the fit of our model(due to target being an unbalanced categorical variable). The training outcome informs of impressive AUC and LogLoss values for the test and validation performances.

- The AUC is a measure of the quality of a binary classification model's predictions. It is computed by calculating the area under the curve of a plot of the true positive rate against the false positive rate at various classification thresholds.

- LogLoss measures the difference between the predicted probability of the positive class and the true label, where the result is transformed by taking the logarithm. A lower logloss value indicates better performance of the model.

5 Leaderboard Outcome

An H2O leaderboard is a table that shows the performance metrics of different models trained using the H2O platform. The metrics could be different depending on the task, but they typically include some measure of accuracy, such as AUC,accuracy and logloss for binary classification.

[15]:
```
#leaderboard visualization
leaders=aml.leaderboard
leaders.head(rows=leaders.nrows)
```

[15]:
```
model_id                                                    auc    logloss
aucpr    mean_per_class_error      rmse        mse
-------------------------------------------------------  --------  ---------
--------  ----------------------  --------  ---------
StackedEnsemble_BestOfFamily_1_AutoML_5_20230301_42812  0.994075  0.0877591
0.992451               0.0284384  0.154356  0.0238257
StackedEnsemble_AllModels_1_AutoML_5_20230301_42812     0.993716  0.086839
0.992445               0.029581   0.151599  0.0229822
GLM_1_AutoML_5_20230301_42812                           0.99361   0.0863161
0.992574               0.0230427  0.148406  0.0220244
GBM_grid_1_AutoML_5_20230301_42812_model_4              0.992298  0.101091
0.989902               0.036606   0.168193  0.0282887
XGBoost_grid_1_AutoML_5_20230301_42812_model_8          0.992065  0.112759
0.989581               0.0431443  0.179943  0.0323795
GBM_4_AutoML_5_20230301_42812                           0.991473  0.110945
0.988604               0.0392298  0.176475  0.0311435
GBM_1_AutoML_5_20230301_42812                           0.991261  0.107109
0.989279               0.0402031  0.17133   0.0293539
GBM_2_AutoML_5_20230301_42812                           0.990774  0.113632
0.987655               0.0387431  0.179841  0.0323427
GBM_grid_1_AutoML_5_20230301_42812_model_3              0.99069   0.109755
0.988226               0.042171   0.176354  0.0311009
GBM_grid_1_AutoML_5_20230301_42812_model_5              0.990647  0.115317
0.987725               0.0513119  0.182357  0.0332539
GBM_3_AutoML_5_20230301_42812                           0.990605  0.111098
0.988234               0.0426576  0.178879  0.0319976
XGBoost_grid_1_AutoML_5_20230301_42812_model_2          0.990573  0.113951
0.988                  0.0402031  0.176979  0.0313214
XGBoost_3_AutoML_5_20230301_42812                       0.990521  0.107953
0.988613               0.0392298  0.176132  0.0310227
GBM_grid_1_AutoML_5_20230301_42812_model_2              0.989272  0.116847
0.986826               0.0408591  0.179142  0.0320918
XGBoost_2_AutoML_5_20230301_42812                       0.988997  0.122026
```

```
0.987189                0.0406898  0.181108  0.0328003
XGBoost_grid_1_AutoML_5_20230301_42812_model_3          0.988785  0.120181
0.986198                0.0472281  0.183191  0.0335589
XRT_1_AutoML_5_20230301_42812                           0.988754  0.12986
0.983407                0.0482226  0.199604  0.0398417
GBM_grid_1_AutoML_5_20230301_42812_model_1              0.988172  0.114562
0.986875                0.0460855  0.175271  0.0307199
XGBoost_grid_1_AutoML_5_20230301_42812_model_1          0.988129  0.126638
0.985671                0.0449429  0.184064  0.0338795
GBM_5_AutoML_5_20230301_42812                           0.987579  0.130634
0.984756                0.0467414  0.189186  0.0357915
XGBoost_grid_1_AutoML_5_20230301_42812_model_7          0.987304  0.135968
0.984408                0.0431443  0.191855  0.0368084
DRF_1_AutoML_5_20230301_42812                           0.986796  0.200014
0.98413                 0.0521371  0.19778   0.0391171
XGBoost_grid_1_AutoML_5_20230301_42812_model_4          0.98413   0.155733
0.980015                0.061934   0.207096  0.0428889
XGBoost_grid_1_AutoML_5_20230301_42812_model_6          0.982586  0.188536
0.975796                0.0681549  0.225084  0.0506626
XGBoost_grid_1_AutoML_5_20230301_42812_model_5          0.980777  0.192021
0.972474                0.0756665  0.228297  0.0521195
XGBoost_1_AutoML_5_20230301_42812                       0.976566  0.192017
0.971567                0.0678375  0.22753   0.0517697
XGBoost_grid_1_AutoML_5_20230301_42812_model_9          0.973254  0.252126
0.966545                0.0737198  0.253772  0.0644001
[27 rows x 7 columns]
```

[16]:
```
aml.leader.model_performance(test_data=test)
```

[16]:
```
ModelMetricsBinomialGLM: stackedensemble
** Reported on test data. **

MSE: 0.012585102897409134
RMSE: 0.11218334500900361
LogLoss: 0.05239402630588414
AUC: 0.9972875226039783
AUCPR: 0.9957565798651044
Gini: 0.9945750452079567
Null degrees of freedom: 120
Residual degrees of freedom: 116
Null deviance: 156.78875756908377
Residual deviance: 12.679354366023961
AIC: 22.67935436602396

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.728251993322077
       B    M    Error    Rate
```

```
-----  ---  ---  -------  -----------
B      79   0    0        (0.0/79.0)
M      1    41   0.0238   (1.0/42.0)
Total  80   41   0.0083   (1.0/121.0)

Maximum Metrics: Maximum metrics at their respective thresholds
metric                       threshold    value     idx
---------------------------  -----------  --------  -----
max f1                       0.728252     0.987952  37
max f2                       0.728252     0.980861  37
max f0point5                 0.728252     0.995146  37
max accuracy                 0.728252     0.991736  37
max precision                1            1         0
max recall                   0.0755553    1         47
max specificity              1            1         0
max absolute_mcc             0.728252     0.981829  37
max min_per_class_accuracy   0.728252     0.97619   37
max mean_per_class_accuracy  0.728252     0.988095  37
max tns                      1            79        0
max fns                      1            39        0
max fps                      1.84394e-05  79        117
max tps                      0.0755553    42        47
max tnr                      1            1         0
max fnr                      1            0.928571  0
max fpr                      1.84394e-05  1         117
max tpr                      0.0755553    1         47

Gains/Lift Table: Avg response rate: 34.71 %, avg score: 35.23 %
group    cumulative_data_fraction    lower_threshold    lift
cumulative_lift    response_rate    score        cumulative_response_rate
cumulative_score    capture_rate    cumulative_capture_rate    gain
cumulative_gain    kolmogorov_smirnov
-------  --------------------------  -----------------  --------
-----------------  ---------------  -----------  --------------------------
------------------  --------------  -------------------------  --------
-----------------  --------------------
1        0.0165289                   1                   2.88095   2.88095
1                1            1                           1
0.047619        0.047619                   188.095   188.095            0.047619
2        0.0247934                   1                   2.88095   2.88095
1                1            1                           1
0.0238095       0.0714286                  188.095   188.095
0.0714286
3        0.0330579                   1                   2.88095   2.88095
1                1            1                           1
0.0238095       0.0952381                  188.095   188.095
0.0952381
```

```
4        0.0413223                    1                    2.88095   2.88095
1                1            1                            1
0.0238095       0.119048                    188.095   188.095            0.119048
5        0.0578512                    0.999999             2.88095   2.88095
1                0.999999     1                            1
0.047619        0.166667                    188.095   188.095            0.166667
6        0.107438                     0.999983             2.88095   2.88095
1                0.999993     1                            0.999996
0.142857        0.309524                    188.095   188.095            0.309524
7        0.157025                     0.999975             2.88095   2.88095
1                0.999978     1                            0.999991
0.142857        0.452381                    188.095   188.095            0.452381
8        0.206612                     0.999335             2.88095   2.88095
1                0.999746     1                            0.999932
0.142857        0.595238                    188.095   188.095            0.595238
9        0.305785                     0.951192             2.88095   2.88095
1                0.987315     1                            0.99584
0.285714        0.880952                    188.095   188.095            0.880952
10       0.404959                     0.0901846            0.960317  2.41059
0.333333         0.410936     0.836735                    0.852598
0.0952381       0.97619                     -3.96825  141.059            0.874925
11       0.504132                     0.0308129            0.240079  1.98361
0.0833333        0.0544159    0.688525                    0.695579
0.0238095       1                           -75.9921  98.3607            0.759494
12       0.603306                     0.0028408            0         1.65753
0                0.0134158    0.575342                    0.583442            0
1                         -100      65.7534            0.607595
13       0.702479                     0.00136928           0         1.42353
0                0.00199129   0.494118                    0.501355            0
1                         -100      42.3529            0.455696
14       0.801653                     0.000444032          0         1.24742
0                0.0007483    0.43299                     0.439424            0
1                         -100      24.7423            0.303797
15       0.900826                     0.000112215          0         1.11009
0                0.000226276  0.385321                    0.391072            0
1                         -100      11.0092            0.151899
16       1                            1.84394e-05          0         1
0                5.39391e-05  0.347107                    0.352294            0
1                         -100      0                  0
```

[17]:
```
explain_model = aml.explain(frame = test, figsize =␣
↪(8,6),exclude_explanations=['leaderboard','varimp_heatmap'])
```

6 Confusion Matrix

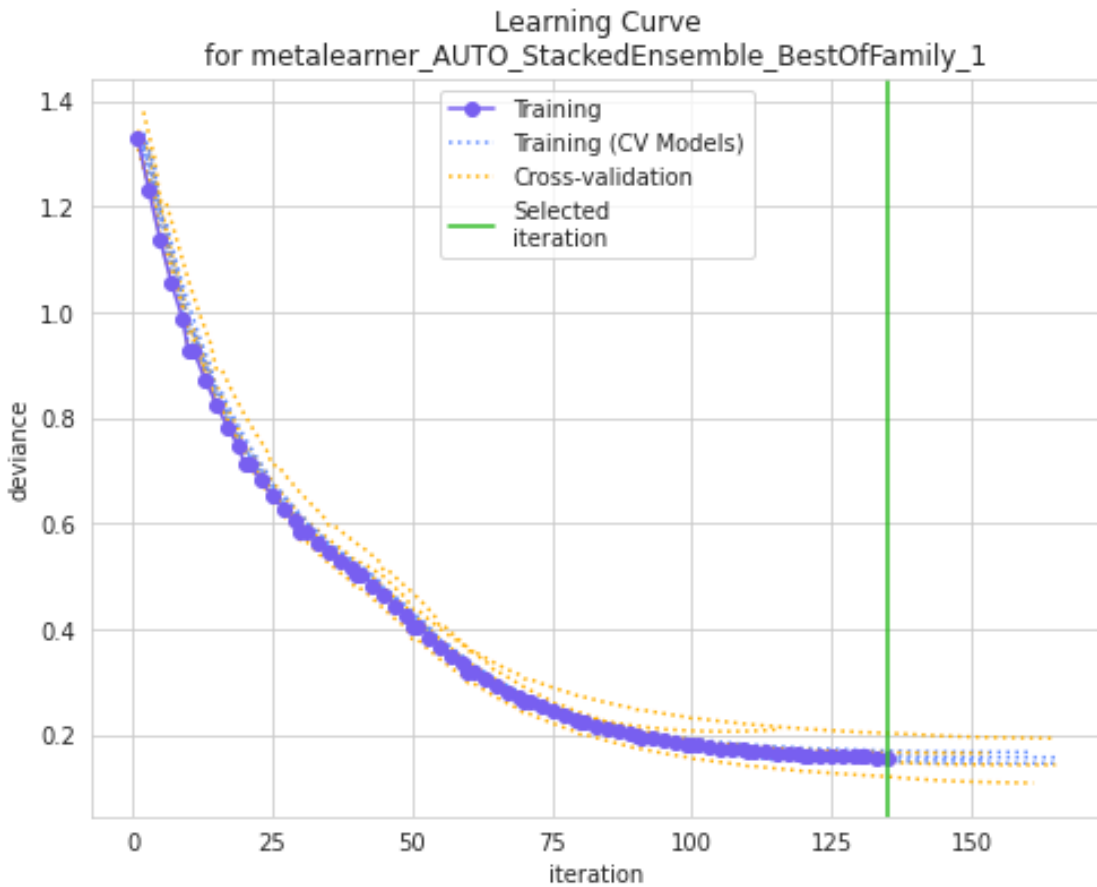
Confusion matrix shows a predicted class vs an actual class.

6.1 StackedEnsemble_BestOfFamily_1_AutoML_5_20230301_42812

```
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.728251993322077
       B    M    Error    Rate
-----  ---  ---  -------  -----------
B      79   0    0        (0.0/79.0)
M      1    41   0.0238   (1.0/42.0)
Total  80   41   0.0083   (1.0/121.0)
```

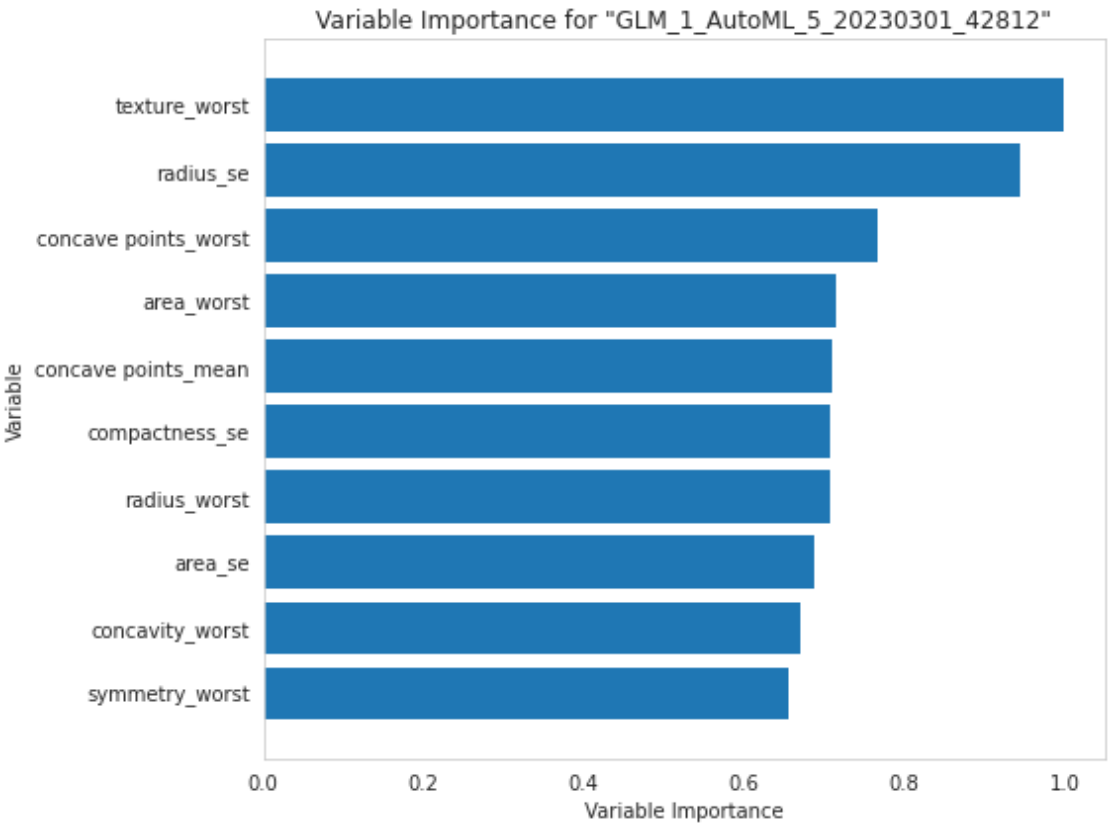
7 Learning Curve Plot

Learning curve plot shows the loss function/metric dependent on number of iterations or trees for tree-based algorithms. This plot can be useful for determining whether the model overfits.



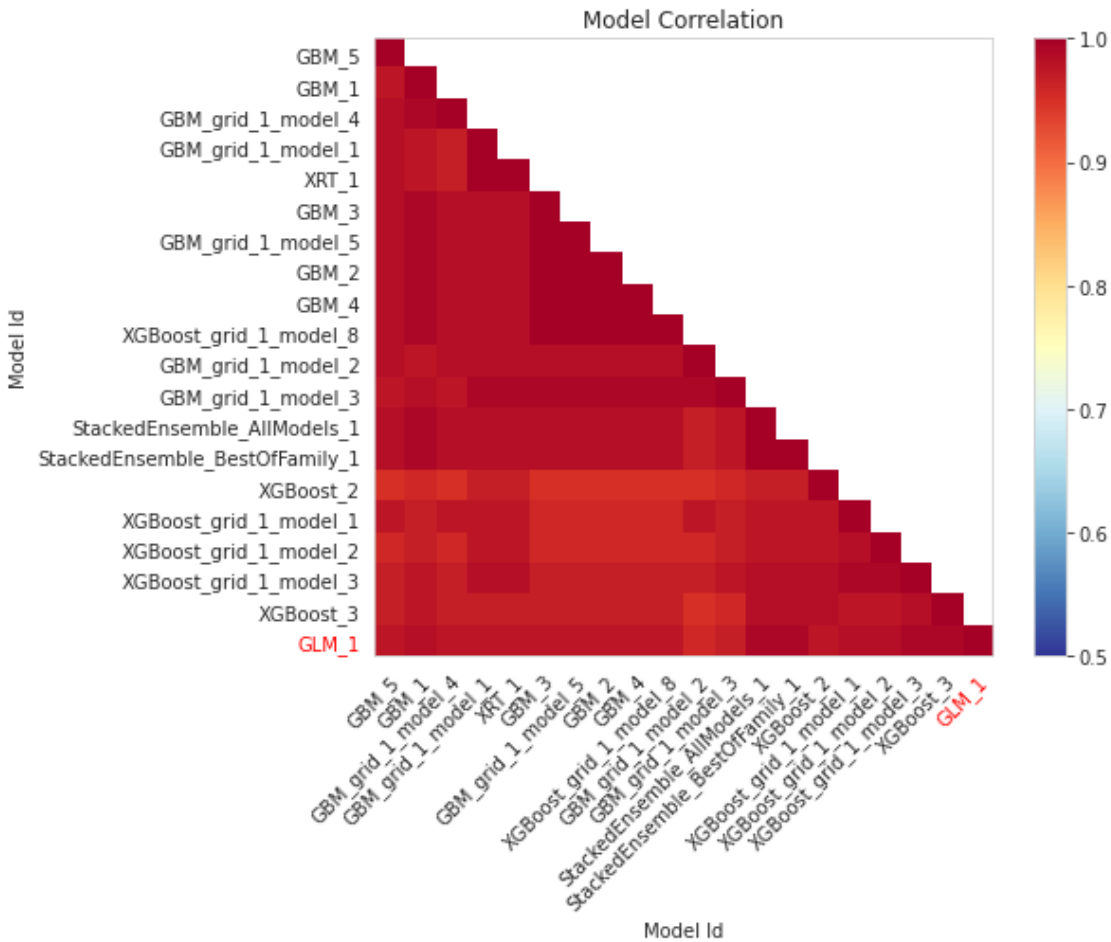
8 Variable Importance

The variable importance plot shows the relative importance of the most important variables in the model.



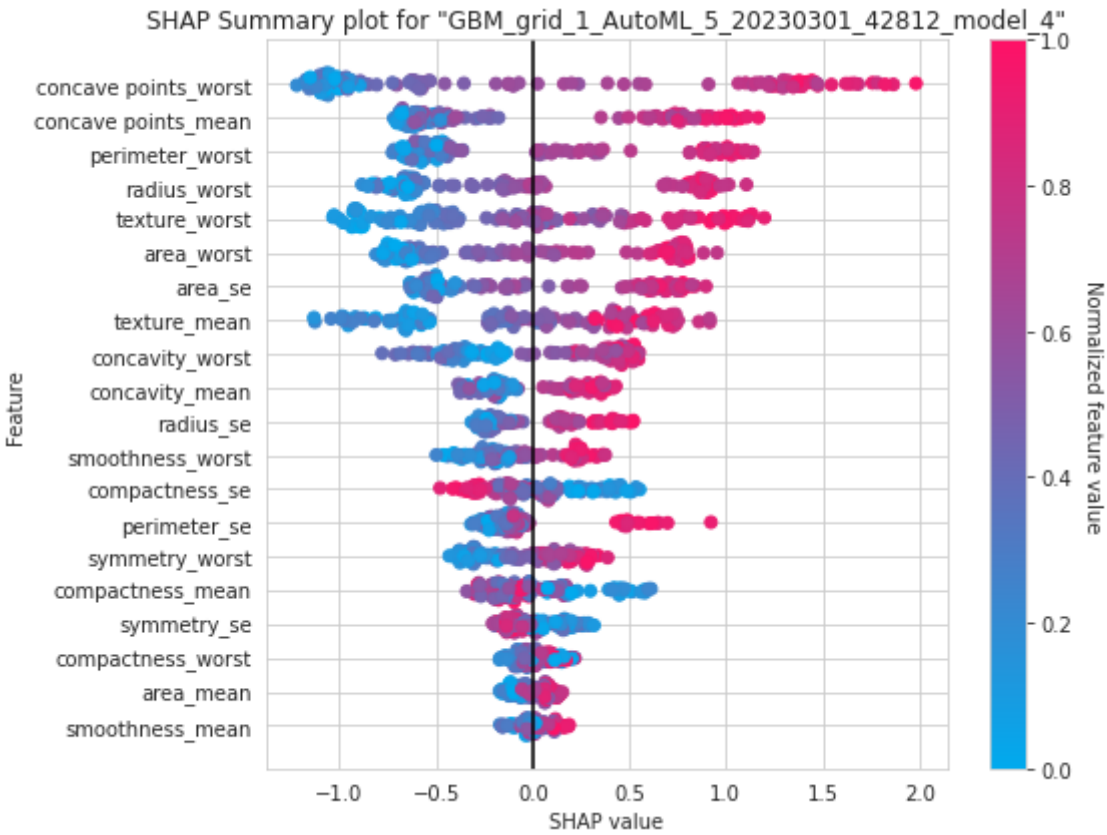
9 Model Correlation

This plot shows the correlation between the predictions of the models. For classification, frequency of identical predictions is used. By default, models are ordered by their similarity (as computed by hierarchical clustering). Interpretable models, such as GAM, GLM, and RuleFit are highlighted using red colored text.



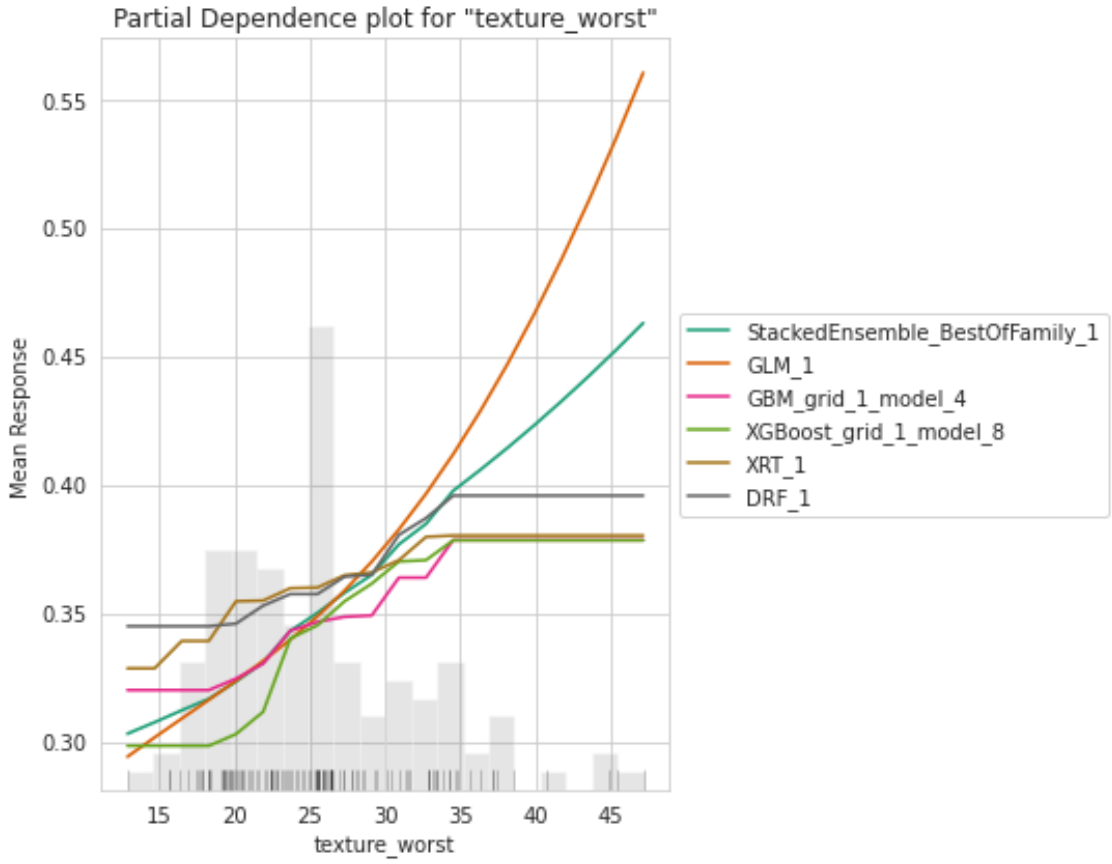
10 SHAP Summary

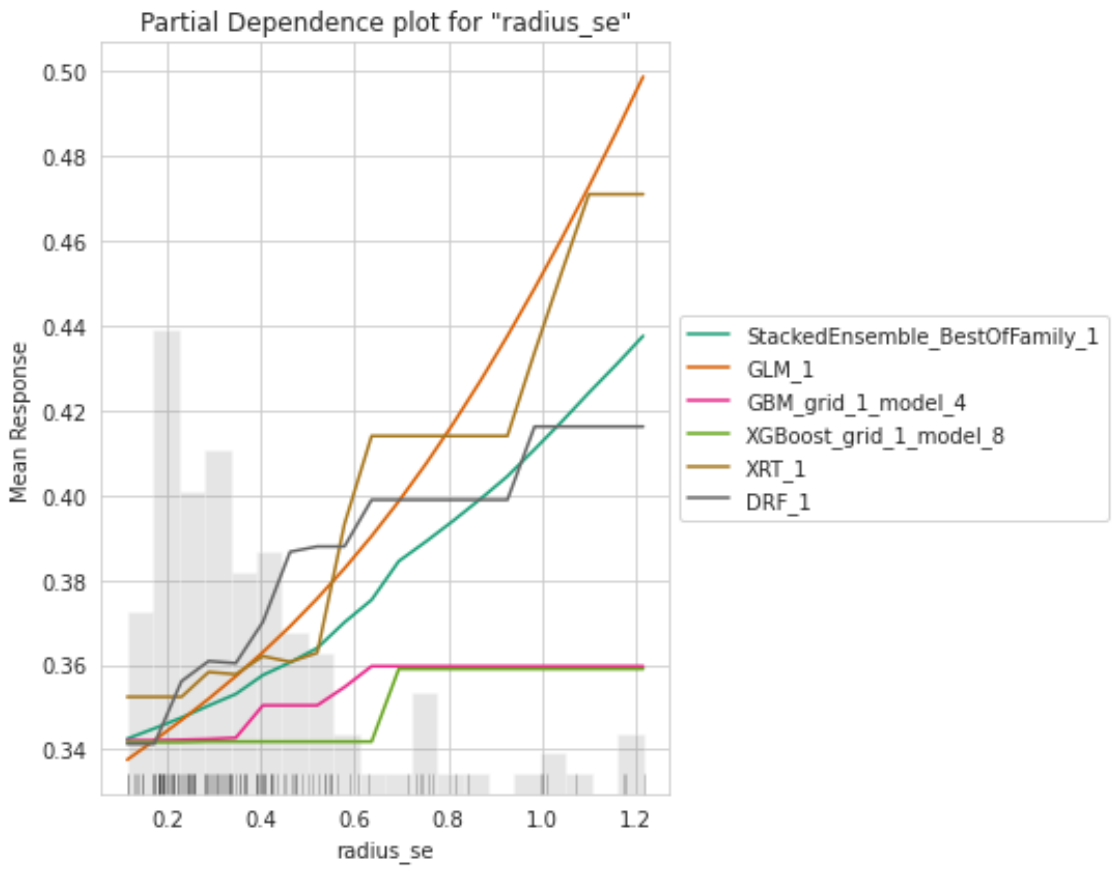
SHAP summary plot shows the contribution of the features for each instance (row of data). The sum of the feature contributions and the bias term is equal to the raw prediction of the model, i.e., prediction before applying inverse link function.

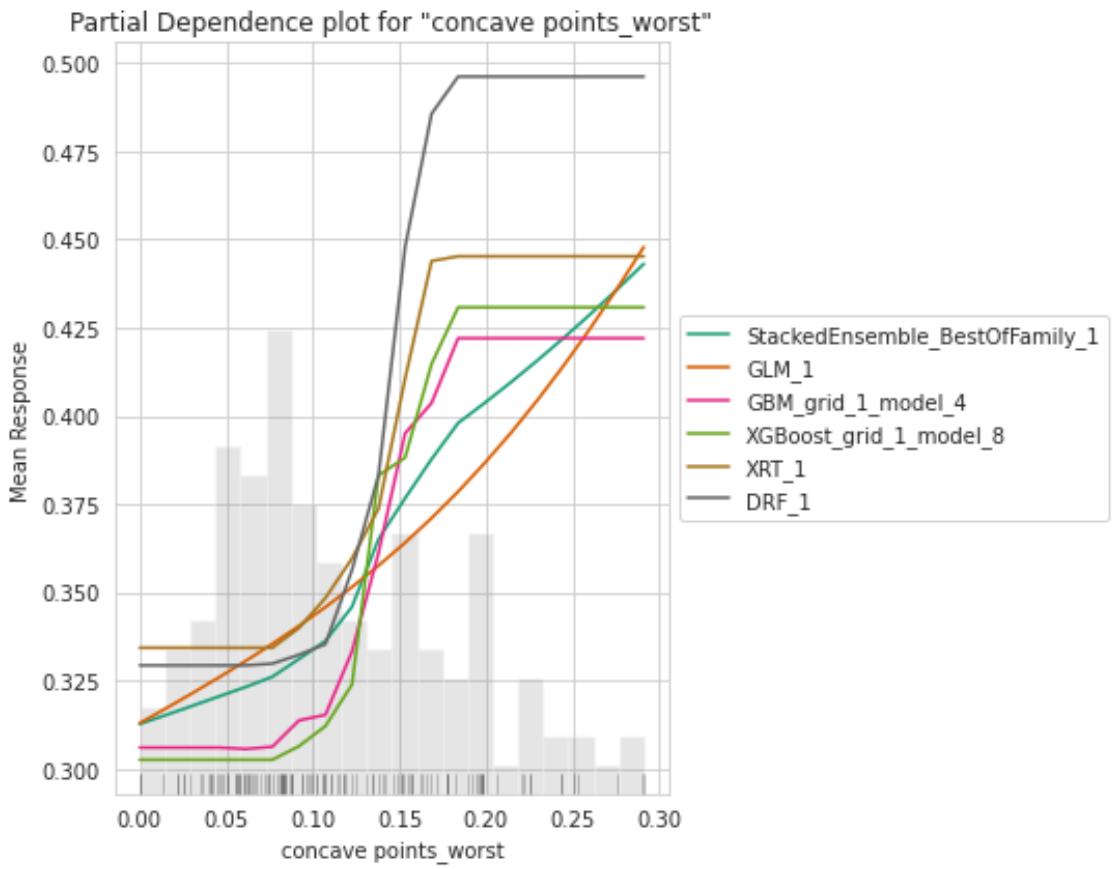


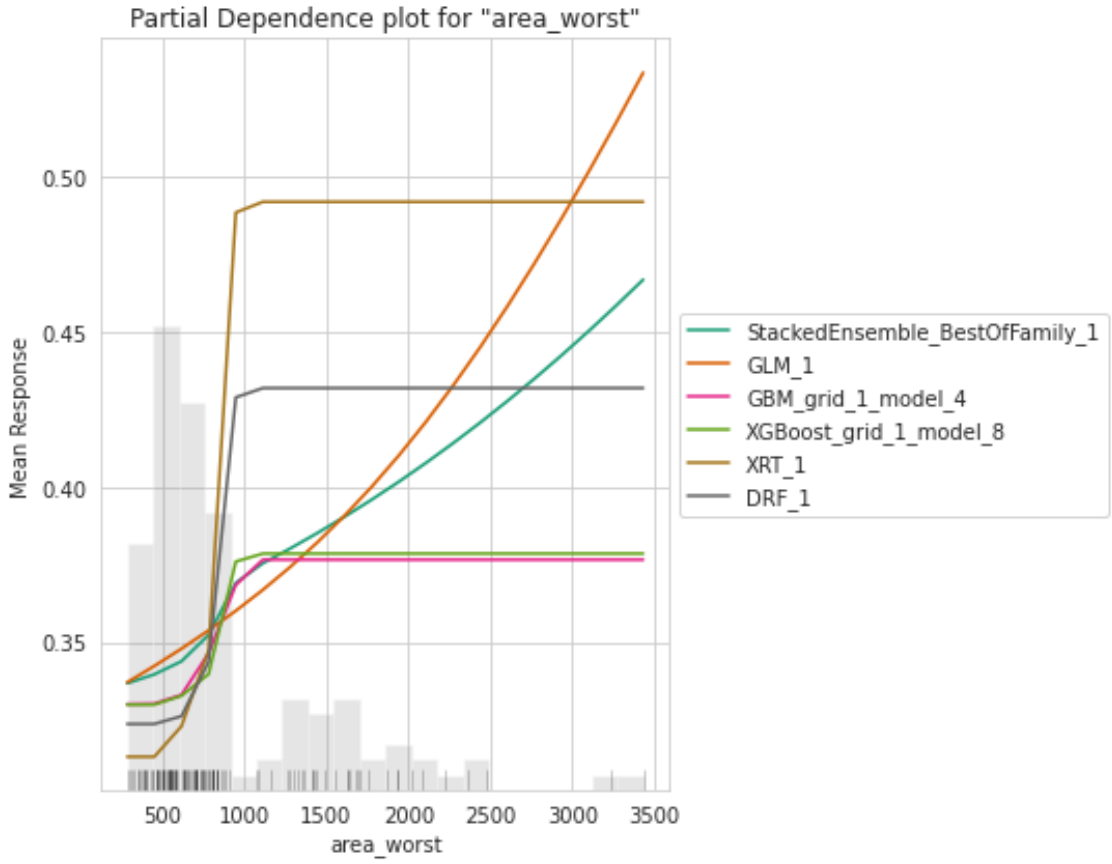
11 Partial Dependence Plots

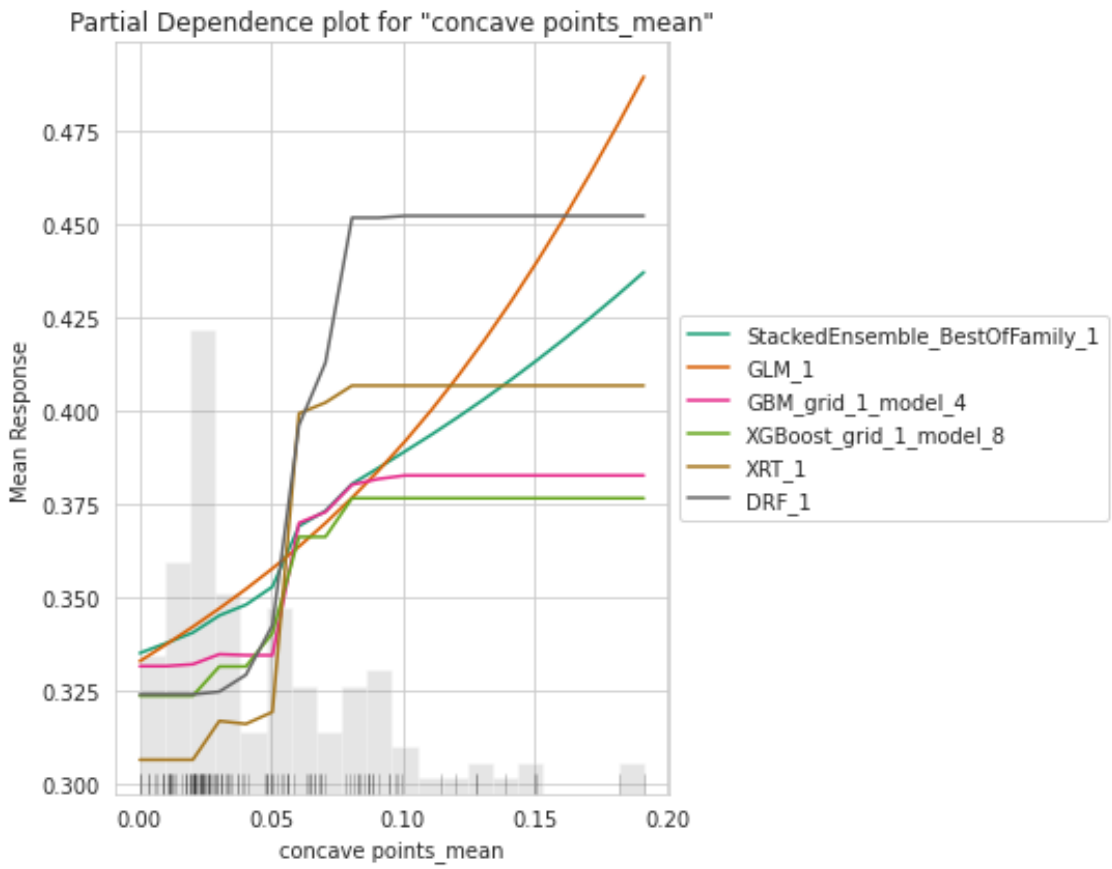
Partial dependence plot (PDP) gives a graphical depiction of the marginal effect of a variable on the response. The effect of a variable is measured in change in the mean response. PDP assumes independence between the feature for which is the PDP computed and the rest.


Partial Dependence plot for "texture_worst"
Mean Response
0.55
0.50
0.45
0.40
0.35
0.30
15
20
25
30
35
40
45
texture_worst
StackedEnsemble_BestOfFamily_1
GLM_1
GBM_grid_1_model_4
XGBoost_grid_1_model_8
XRT_1
DRF_1


Partial Dependence plot for "radius_se"
Mean Response
0.50
0.48
0.46
0.44
0.42
0.40
0.38
0.36
0.34
0.2
0.4
0.6
0.8
1.0
1.2
radius_se
StackedEnsemble_BestOfFamily_1
GLM_1
GBM_grid_1_model_4
XGBoost_grid_1_model_8
XRT_1
DRF_1


Partial Dependence plot for "concave points_worst"
Mean Response
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0.00
0.05
0.10
0.15
0.20
0.25
0.30
concave points_worst
StackedEnsemble_BestOfFamily_1
GLM_1
GBM_grid_1_model_4
XGBoost_grid_1_model_8
XRT_1
DRF_1


Partial Dependence plot for "area_worst"
Mean Response
0.50
0.45
0.40
0.35
500
1000
1500
2000
2500
3000
3500
area_worst
StackedEnsemble_BestOfFamily_1
GLM_1
GBM_grid_1_model_4
XGBoost_grid_1_model_8
XRT_1
DRF_1



11.0.1 Partial Dependency Plots

Partial Dependency Plots (PDPs) are a powerful visualization tool used in machine learning to help understand the relationship between a set of input features and the model's output. Specifically, PDPs help to visualize the relationship between a specific feature and the model's predicted outcome while holding all other features constant.

For most of the models, the trend in mean response for each variable, is tending to reach a saturation value. it means that the relationship between the predictor variable and the predicted outcome has become flat or linear, and there is no additional information to be gained by increasing or decreasing the predictor variable further.

11.0.2 SHAP Summary

The SHAP summary plot provides feature importances which are model agnostic. It includes a vertical line that represents the baseline value for the predicted outcome. Features that are above the line have a positive effect on the predicted outcome, and those that are below the line have a negative effect. By analyzing a SHAP summary plot, its possible to gain insights into how a model

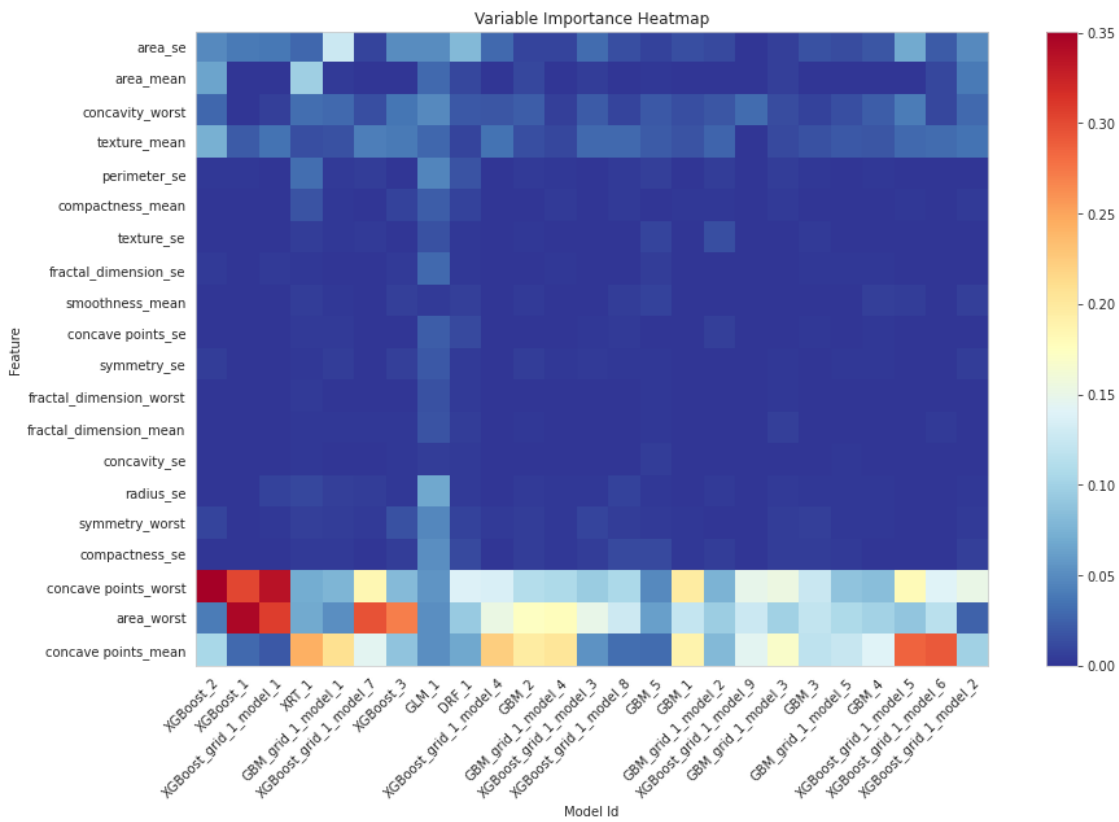
is making predictions and to identify potential areas for improvement or further investigation.

11.0.3 Variable Importance and Learning Curve Interpretations

The variable importance plot of a GLM model is given above. The importance is given most to the "texture_worst" variable. A learning curve shows the relationship between the model's performance on the training and validation sets against the number of iterations or epochs. The learning curve can be used to identify whether the model is overfitting or underfitting the data, and to determine the optimal number of iterations or epochs to use when training the model. The plot above has exhibited a steady decrease(non divergent trends) in training and validation metrics, which rules out overfitting.

[18]:
```
h2o.varimp_heatmap(leaders)
```

[18]:
```
<h2o.plot._plot_result._MObject at 0x7f160ab599a0>
```



The variable importance heatmap is calculated using the feature importance scores generated by the machine learning model. H2O uses a variety of feature importance metrics, including Gini importance, permutation importance, and SHAP values, to determine the relative importance of each feature. The feature importance scores are then normalized and plotted as a heatmap.

12 Code Adoption

The code for constructing kdeplots has been adopted from a kaggle notebook. Everything else has been coded by myself, by learning and implementing.

13 Responses to posed questions

- **Is the relationship significant?**
  Going by the OLS Regression results,significance in relationship of majority of features can be called into question,due to undesirable p-values.
- **Are any model assumptions violated?**
  The problem is prediction of a binomial category, which is a doable task by AutoML.
- **Is there any multicollinearity in the model?**
  From the VIF readings, we can infer that there is a great amount of multicollinearity between the features.
- **In the multivariate models are predictor variables independent of all the other predictor variables?**
  By examining the correlation heatmap, there seems to be a linear dependency among predictor variables(as high as 0.99).
- **Does regularization help?**
  By comparing the metrics of generic RandomForestClassifier and h20 RandomForestEstimator(Regularized), we can confirm that regularization does improve model performance.
- ___ Which independent variables are significant?___
  The significant variables derived from OLS regression are radius_mean,area_mean,fractal_dimension_mean,smoothness_se,concavity_se etc.
- **Which hyperparameters are important?**
  Hyperparameters in randomforest model such as max_depth,min_rows and sample_rate are important. For the AutoML models, all hyperparameters in each model is tweaked and tested.
- **Coding professionalism?**
  Coding professionalism is kept intact.
- **Does the model make sense?**
  Yes, the model makes sense and AutoML can be used to get the desired pipeline.
- **In in multivariate models rank the most significant predictor variables and exclude insignificant ones from the model.**
  The most significant feature is fractal_dimension_mean and most insignificant feature is smoothness_worst.

14 References

- h2o website
- tech target
- towardsdatascience
- geeksforgeeks

Copyright (c) 2023 Sai Kashyap Cheruku

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

ML Data Cleaning and Feature Selection

Due Feb 4 by 11:59pm **Points** 100 **Submitting** a file upload **Available** until Apr 15 at 11:59pm

This assignment was locked Apr 15 at 11:59pm.

ML Data Cleaning and Feature Selection

In this assignment, you will use a dataset for predictive learning and check the quality of the data and determine which features are important.

Answer the following questions:

* What are the data types? (Only numeric and categorical)

* Are there missing values?

* What are the likely distributions of the numeric variables?

* Which independent variables are useful to predict a target (dependent variable)? (Use at least three methods)

* Which independent variables have missing data? How much?

* Do the training and test sets have the same data?

* In the predictor variables independent of all the other predictor variables?

* Which predictor variables are the most important?

* Do the ranges of the predictor variables make sense?

* What are the distributions of the predictor variables?

* Remove outliers and keep outliers (does if have an effect of the final predictive model)?

* Remove 1%, 5%, and 10% of your data randomly and impute the values back using at least 3 imputation methods. How well did the methods recover the missing values? That is remove some data, check the % error on residuals for numeric data and check for bias and variance of the error.

For categorical data, calculate the accuracy and a confusion matrix.

There is an example notebook for assignment one here

https://github.com/aiskunks/YouTube/blob/main/A_Crash_Course_in_Statistical_Learning/ML_Data_Cleaning_and_Feature_Selection/ML_Dat
(https://github.com/aiskunks/YouTube/blob/main/A_Crash_Course_in_Statistical_Learning/ML_Data_Cleaning_and_Feature_Selection/ML_Data_Cleaning_and

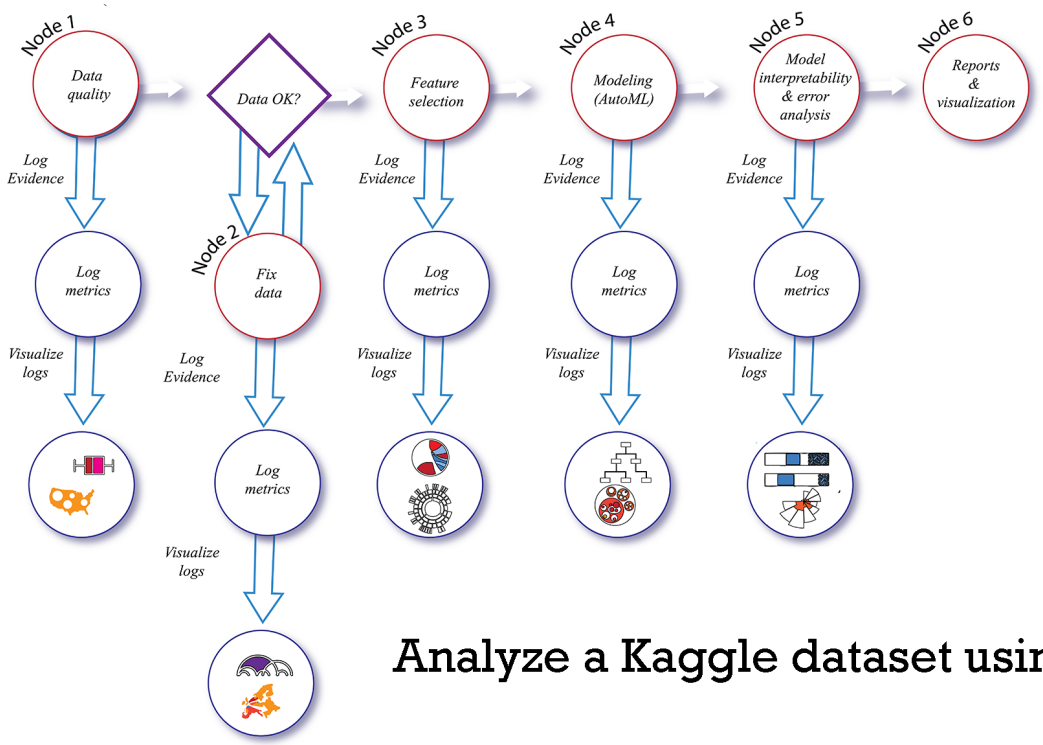
Imputation Methods for Missing Data

https://www.youtube.com/watch?v=fYhr8eF1ubo (https://www.youtube.com/watch?v=fYhr8eF1ubo)

(https://www.youtube.com/watch?v=fYhr8eF1ubo)

Nice EDA notebook **https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python**
(https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python)

AutoML Assingment
Node 4



Analyze a Kaggle dataset using AutoML

Scoring Rubric

Are my answers supported with data? (20 Points)

Tables, graphs, and charts must support your evaluation/answers.

It MUST run in Google Collab. You will also save the Google Collab notebook as a .ipynb notebook and upload that to Canvas . (5 Points)

Public dataset *(5 Points)*

Pick a public dataset that can be used for Regression or Classification. You MUST get approval for your dataset from the TAs.

What code is yours and what have you adapted? (5 Points)

You must explain what code you wrote and what you have done that is different. Failure to cite ANY code will result in a zero for this section.

Did I explain my code clearly? (15 Points) Your code review score will be scaled to a range of 0 to 10 and be used for this score.

Did I explain my licensing clearly? (5 Points) Failure to cite a clear license will result in a zero for this section.

Answers to listed questions (45 Points)

* Which independent variables are useful to predict a target (dependent variable)?

* Which independent variable have missing data? How much?

* Do the training and test sets have the same data?

* In the predictor variables independent of all the other predictor variables?

* Which predictor variables are the most important?

* Do the ranges of the predictor variables make sense?

* What are the distributions of the predictor variables?

Notes:

- **Normality** - When we talk about normality what we mean is that the data should look like a normal distribution. This is important because several statistic tests rely on this (e.g. t-statistics). In this exercise we'll just check univariate normality for 'SalePrice' (which is a limited approach). Remember that univariate normality doesn't ensure multivariate normality (which is what we would like to have), but it helps. Another detail to take into account is that in big samples (>200 observations) normality is not such an issue. However, if we solve normality, we avoid a lot of other problems (e.g. heteroscedacity) so that's the main reason why we are doing this analysis.
- **Homoscedasticity** - I just hope I wrote it right. Homoscedasticity refers to the 'assumption that dependent variable(s) exhibit equal levels of variance across the range of predictor variable(s)' **(Hair et al., 2013)** **(https://amzn.to/2uC3j9p)** . Homoscedasticity is desirable because we want the error term to be the same across all values of the independent variables.
- **Linearity**- The most common way to assess linearity is to examine scatter plots and search for linear patterns. If patterns are not linear, it would be worthwhile to explore data transformations. However, we'll not get into this because most of the scatter plots we've seen appear to have linear relationships.
- **Absence of correlated errors** - Correlated errors, like the definition suggests, happen when one error is correlated to another. For instance, if one positive error makes a negative error systematically, it means that there's a relationship between these variables. This occurs often in time series, where some patterns are time related. We'll also not get into this. However, if you detect something, try to add a variable that can explain the effect you're getting. That's the most common solution for correlated errors.

Data_Cleaning_and_Feature_Selection_Wisconsin_BC (2)

December 16, 2023

Abstract

Fine Needle Analysis(FNA) is a minimal invasive biopsy procedure which involves collecting breast tissue or fluid, used to detect the presence of cancerous cells. This notebook aims to provide various insights into the biopsy data by examining aspects such as:

- Statistical measures of distribution for predictor and response variables
- Correlation between independent features
- Correlation between each independent feature and the dependent feature

- Data Imputation(if required)

- Outlier Analysis

1 Dataset

FNA biopsy observations of 569 patients have been compiled into a dataset,with each row containing a patient's measurement variables and the pertaining diagnosis of breast cancer. Each indicator is represented by mean, standard error and mean of three largest readings. The indicators used are as follows:
* Mean of distances from the center to perimeter * Standard deviation of the gray-scale values pertaining to the biopsy image * Perimeter of cell nucleus * Area of cell nucleus * Variation of smoothness in radius lengths * Compactness of cell nucleus * Concavity severity * Number of Concave portions of the contour * Symmetry of the cell nucleus * Fractal Dimension of cell membrane

These indicators map to *radius*,*texture*,*perimeter*,*area*,*smoothness*,*compactness*,*concavity*,*concave points*,*symmetry* and *fractal_dimension* feature classes respectively. As mentioned previously, each feature class is represented by 3 values, resulting in **thirty independent features** of the dataset. The indicator of cancer-diagnosis(malignant or benign) is interpreted as **one dependent feature** of the dataset(represented by *diagnosis*).This is a categorical feature with two possible categories(M and B).
Note: An ID is assigned to each patient's diagnosis, inclusion of which can be ignored in our study.

[1]:
```
#Importing the necessary libraries for data analysis and visualization
import numpy as np
import scipy as sc
import pandas as pd
import seaborn as sns
```

```python
import matplotlib.pylab as plt
from matplotlib import pyplot
```

[2]:
```python
#Reading the csv formatted dataset into a pandas DataFrame
df=pd.read_csv("https://raw.githubusercontent.com/SKashyap1999/DataScience/main/
↪Breast_Cancer-Wisconsin.csv")
```

[3]:
```python
#Displaying the first 10 rows of the dataset
df.head(10)
```

[3]:
```
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1
4  84358402         M        20.29         14.34          135.10     1297.0
5    843786         M        12.45         15.70           82.57      477.1
6    844359         M        18.25         19.98          119.60     1040.0
7  84458202         M        13.71         20.83           90.20      577.9
8    844981         M        13.00         21.82           87.50      519.8
9  84501001         M        12.46         24.04           83.97      475.9

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760         0.30010              0.14710
1          0.08474           0.07864         0.08690              0.07017
2          0.10960           0.15990         0.19740              0.12790
3          0.14250           0.28390         0.24140              0.10520
4          0.10030           0.13280         0.19800              0.10430
5          0.12780           0.17000         0.15780              0.08089
6          0.09463           0.10900         0.11270              0.07400
7          0.11890           0.16450         0.09366              0.05985
8          0.12730           0.19320         0.18590              0.09353
9          0.11860           0.23960         0.22730              0.08543

   …  radius_worst  texture_worst  perimeter_worst  area_worst  \
0  …         25.38          17.33           184.60      2019.0
1  …         24.99          23.41           158.80      1956.0
2  …         23.57          25.53           152.50      1709.0
3  …         14.91          26.50            98.87       567.7
4  …         22.54          16.67           152.20      1575.0
5  …         15.47          23.75           103.40       741.6
6  …         22.88          27.66           153.20      1606.0
7  …         17.06          28.14           110.60       897.0
8  …         15.49          30.73           106.20       739.3
9  …         15.09          40.68            97.65       711.4

   smoothness_worst  compactness_worst  concavity_worst  concave points_worst  \
```

```
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625
5            0.1791             0.5249           0.5355                0.1741
6            0.1442             0.2576           0.3784                0.1932
7            0.1654             0.3682           0.2678                0.1556
8            0.1703             0.5401           0.5390                0.2060
9            0.1853             1.0580           1.1050                0.2210

   symmetry_worst  fractal_dimension_worst
0          0.4601                  0.11890
1          0.2750                  0.08902
2          0.3613                  0.08758
3          0.6638                  0.17300
4          0.2364                  0.07678
5          0.3985                  0.12440
6          0.3063                  0.08368
7          0.3196                  0.11510
8          0.4378                  0.10720
9          0.4366                  0.20750

[10 rows x 32 columns]
```

1.0.1 Dimensionality and Stats

[4]:
```
#Dimensions of the dataset
df.shape
```

[4]:
```
(569, 32)
```

[5]:
```
#Description of some statistical features pertaining to the dataset
df.describe()
```

[5]:
```
                 id  radius_mean  texture_mean  perimeter_mean    area_mean  \
count  5.690000e+02   569.000000    569.000000      569.000000   569.000000
mean   3.037183e+07    14.127292     19.289649       91.969033   654.889104
std    1.250206e+08     3.524049      4.301036       24.298981   351.914129
min    8.670000e+03     6.981000      9.710000       43.790000   143.500000
25%    8.692180e+05    11.700000     16.170000       75.170000   420.300000
50%    9.060240e+05    13.370000     18.840000       86.240000   551.100000
75%    8.813129e+06    15.780000     21.800000      104.100000   782.700000
max    9.113205e+08    28.110000     39.280000      188.500000  2501.000000

       smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
count       569.000000        569.000000      569.000000           569.000000
```

```
mean          0.096360          0.104341        0.088799             0.048919
std           0.014064          0.052813        0.079720             0.038803
min           0.052630          0.019380        0.000000             0.000000
25%           0.086370          0.064920        0.029560             0.020310
50%           0.095870          0.092630        0.061540             0.033500
75%           0.105300          0.130400        0.130700             0.074000
max           0.163400          0.345400        0.426800             0.201200

       symmetry_mean  …  radius_worst  texture_worst  perimeter_worst  \
count     569.000000  …    569.000000     569.000000       569.000000
mean        0.181162  …     16.269190      25.677223       107.261213
std         0.027414  …      4.833242       6.146258        33.602542
min         0.106000  …      7.930000      12.020000        50.410000
25%         0.161900  …     13.010000      21.080000        84.110000
50%         0.179200  …     14.970000      25.410000        97.660000
75%         0.195700  …     18.790000      29.720000       125.400000
max         0.304000  …     36.040000      49.540000       251.200000

        area_worst  smoothness_worst  compactness_worst  concavity_worst  \
count   569.000000        569.000000         569.000000       569.000000
mean    880.583128          0.132369           0.254265         0.272188
std     569.356993          0.022832           0.157336         0.208624
min     185.200000          0.071170           0.027290         0.000000
25%     515.300000          0.116600           0.147200         0.114500
50%     686.500000          0.131300           0.211900         0.226700
75%    1084.000000          0.146000           0.339100         0.382900
max    4254.000000          0.222600           1.058000         1.252000

       concave points_worst  symmetry_worst  fractal_dimension_worst
count            569.000000      569.000000               569.000000
mean               0.114606        0.290076                 0.083946
std                0.065732        0.061867                 0.018061
min                0.000000        0.156500                 0.055040
25%                0.064930        0.250400                 0.071460
50%                0.099930        0.282200                 0.080040
75%                0.161400        0.317900                 0.092080
max                0.291000        0.663800                 0.207500

[8 rows x 31 columns]
```

[6]:
```
# Dropping the 'id' column
df=df.drop('id',axis=1)
```

1.0.2 Data Imputation

After checking for the traditional NaN values using the .isnull() or .isna() methods, it is advisable to confirm the presence of other entries which may indicate an absence of data(presence of strings such as “None”,“No Value” etc).

This can be done by examining the datatypes of the columns, which are expected to be of float and object(string) types(response being object and predictors being float).

[7]:
```
#Checking for NaN values
df.isnull().sum()
```

[7]:
```
diagnosis                 0
radius_mean               0
texture_mean              0
perimeter_mean            0
area_mean                 0
smoothness_mean           0
compactness_mean          0
concavity_mean            0
concave points_mean       0
symmetry_mean             0
fractal_dimension_mean    0
radius_se                 0
texture_se                0
perimeter_se              0
area_se                   0
smoothness_se             0
compactness_se            0
concavity_se              0
concave points_se         0
symmetry_se               0
fractal_dimension_se      0
radius_worst              0
texture_worst             0
perimeter_worst           0
area_worst                0
smoothness_worst          0
compactness_worst         0
concavity_worst           0
concave points_worst      0
symmetry_worst            0
fractal_dimension_worst   0
dtype: int64
```

[8]:
```
#Examining the datatypes of each feature
df.dtypes
```

[8]:
```
diagnosis                  object
radius_mean               float64
texture_mean              float64
perimeter_mean            float64
area_mean                 float64
smoothness_mean           float64
```

```
compactness_mean           float64
concavity_mean             float64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
concave points_se          float64
symmetry_se                float64
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
dtype: object
```

- Due to the expected feature-datatype mapping being intact, we can confirm the absence of any "null" values and hence avoid imputation.

[9]:
```
df_target=df['diagnosis']
df=df.drop(['diagnosis'],axis=1)
```

[10]:
```
type(df_target)
```

[10]:
```
pandas.core.series.Series
```

2 Distributions

[11]:
```
#Boxplot for every independent feature
plt.rcParams.update({'figure.max_open_warning': 0})
plt.figure(figsize=(20,250))
cols=list(df.columns.values)
for a in range(1,30):
    plt.subplot(30,1,a)
    sns.boxplot(df[cols[a]])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
```

```
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
```

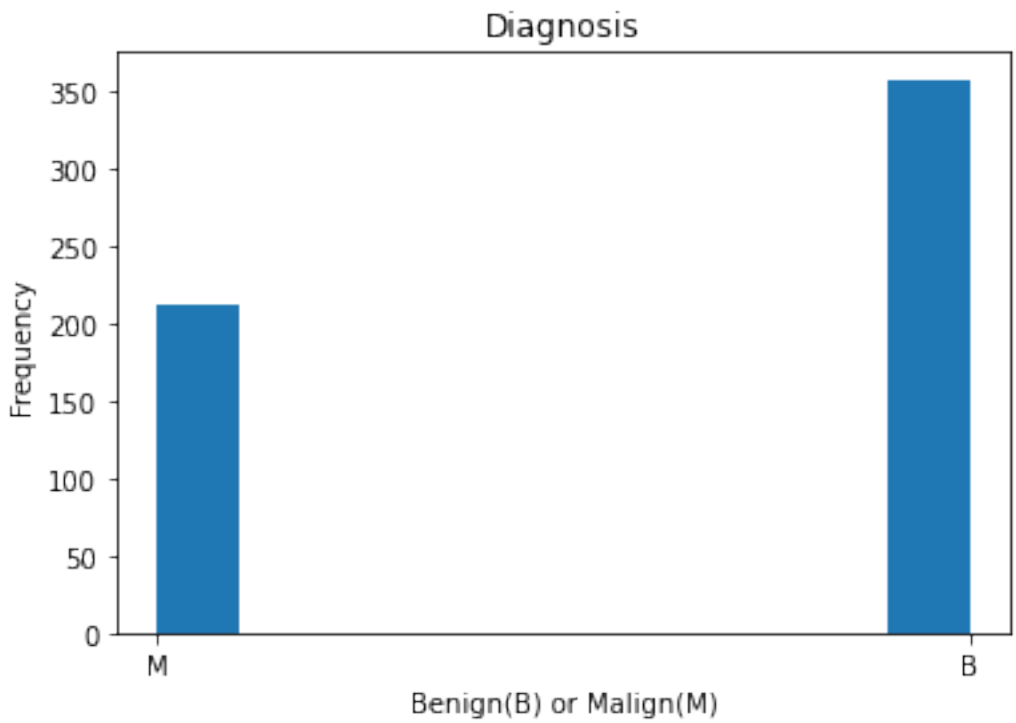
```
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
```

```
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

The range or spread of the data for independent features reveals that the difference in their ranges is very significant. A scaling operation should be performed on the dataset. Presence of outliers in most independent features is also observed

[12]:
```
#histogram plot for dependent feature
plt.hist(x=df_target)
plt.title("Diagnosis")
plt.xlabel("Benign(B) or Malign(M)")
plt.ylabel("Frequency")
```

[12]: Text(0, 0.5, 'Frequency')

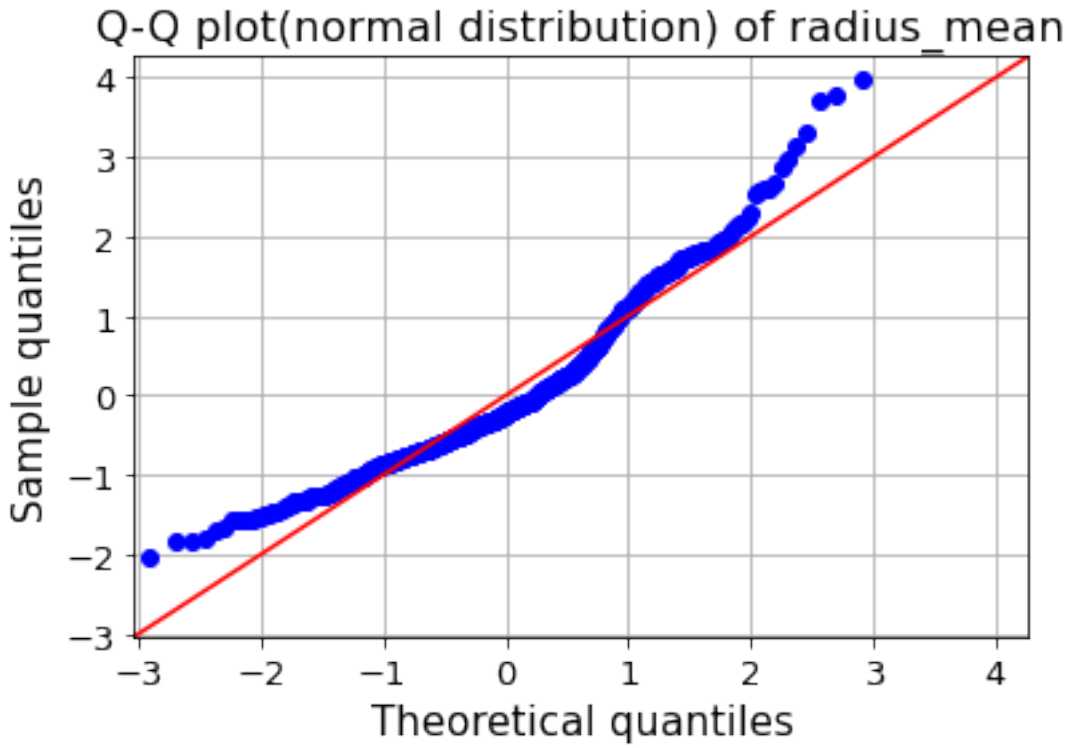


The bar graph shows a greater presence of "B" label in the dependent feature.

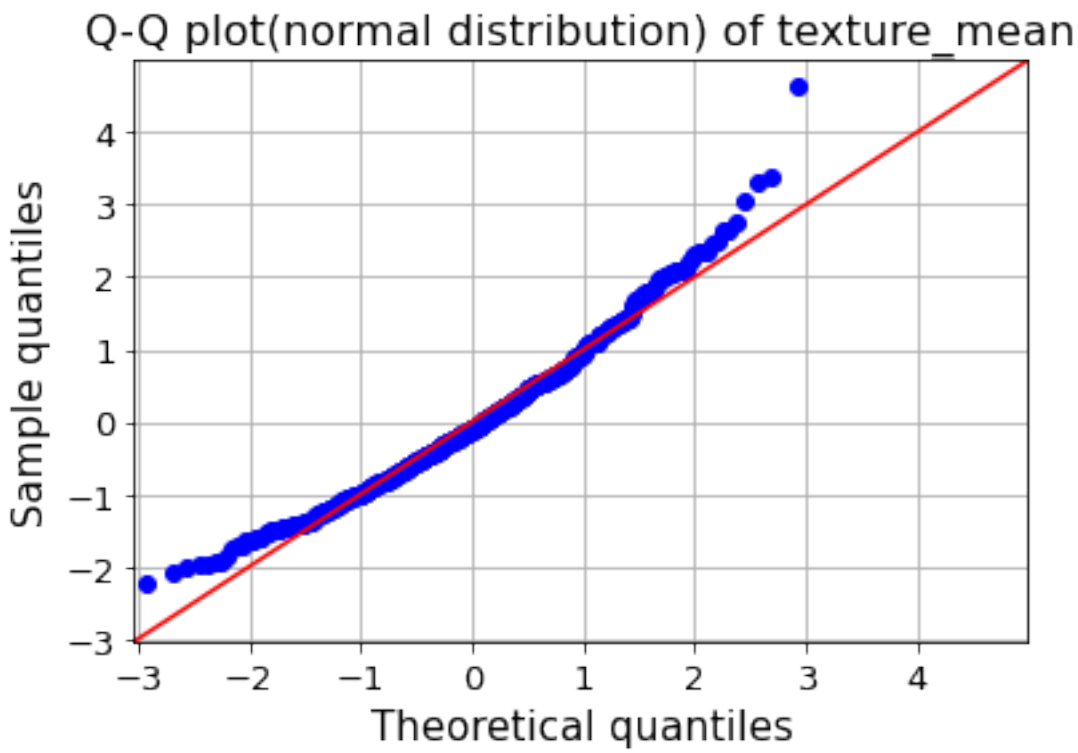
[13]:
```
#qqplot of independent features and various distributions

from statsmodels.graphics.gofplots import qqplot
for c in cols:
            plt.figure(figsize=(10,4))
            fig=qqplot(df[c],dist=sc.stats.norm,line="45",fit='True')
            plt.xticks(fontsize=13)
            plt.yticks(fontsize=13)
```

```
            plt.xlabel("Theoretical quantiles",fontsize=15)
            plt.ylabel("Sample quantiles",fontsize=15)
            plt.title("Q-Q plot(normal distribution) of {}".
↪format(c),fontsize=16)
            plt.grid(True)
            plt.show()
```

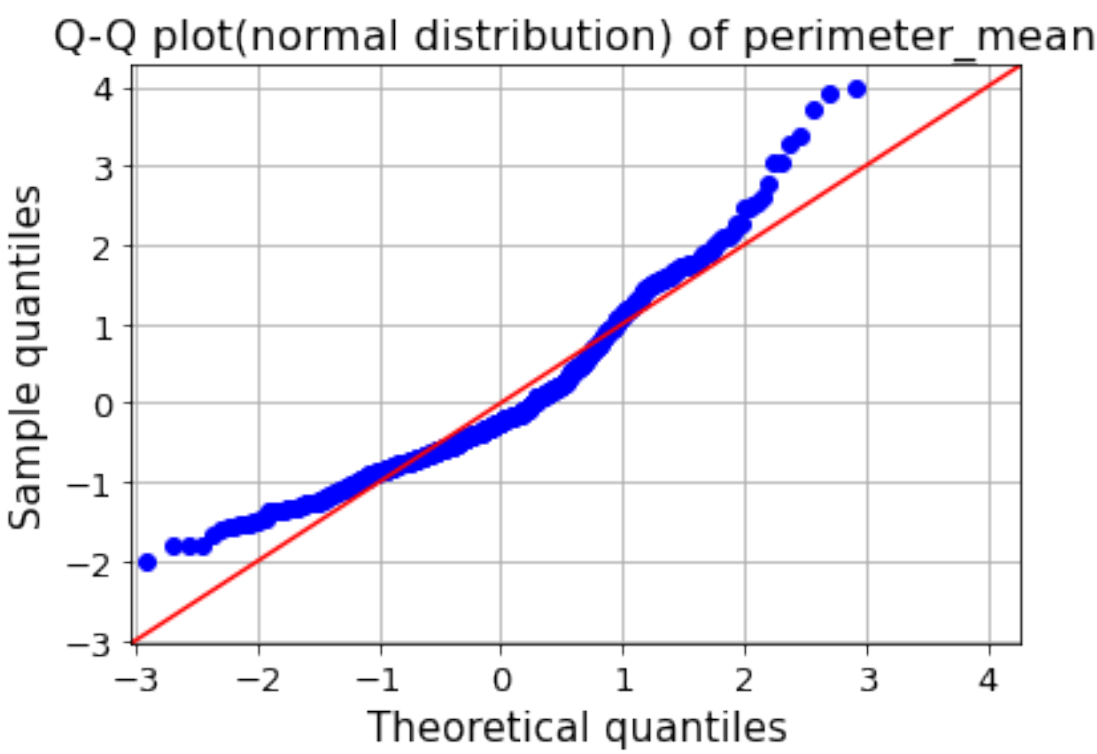
```
<Figure size 720x288 with 0 Axes>
```



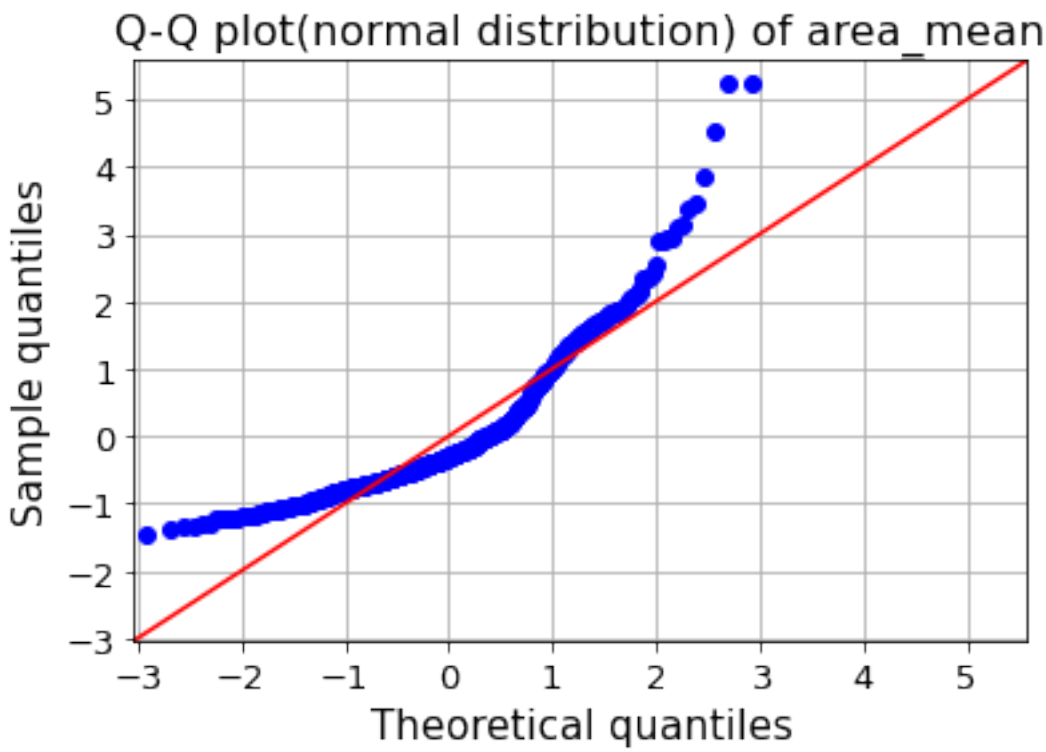

```
<Figure size 720x288 with 0 Axes>
```



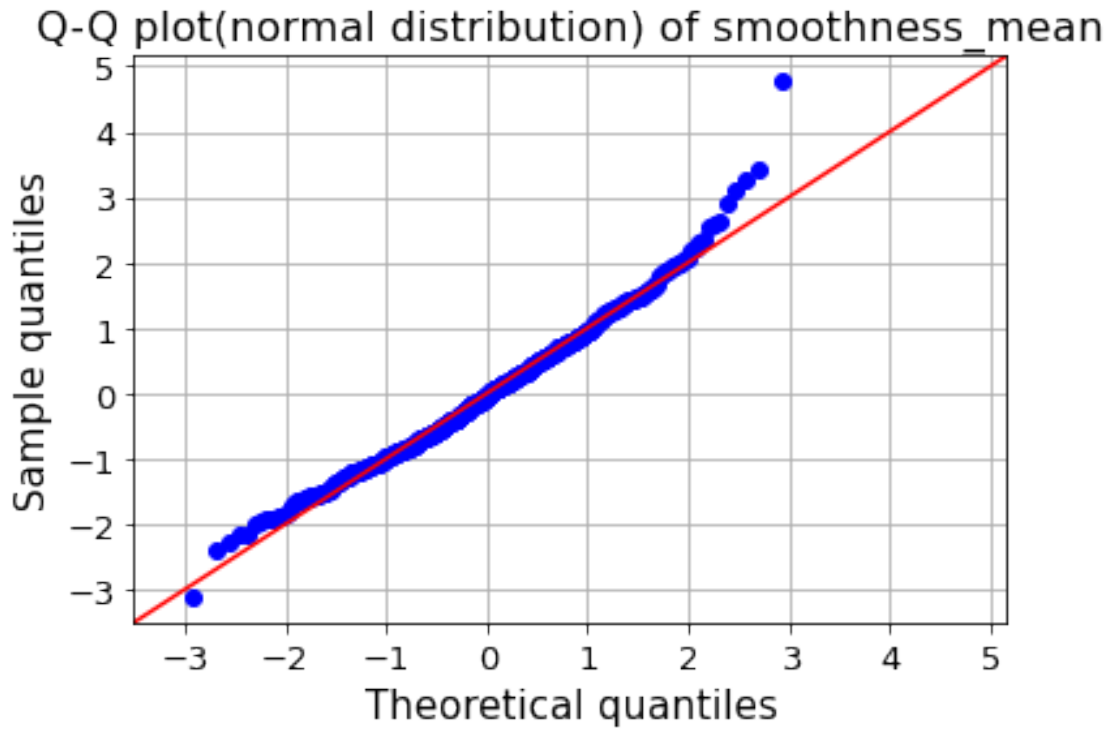

```
<Figure size 720x288 with 0 Axes>
```



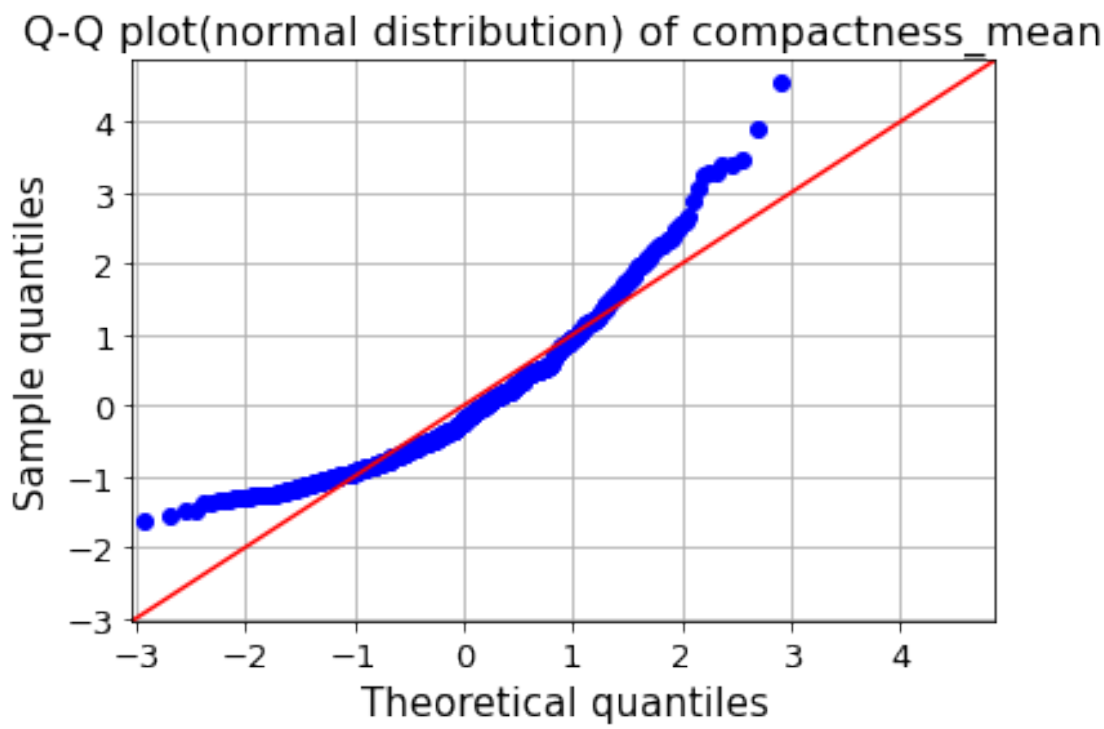
```
<Figure size 720x288 with 0 Axes>
```



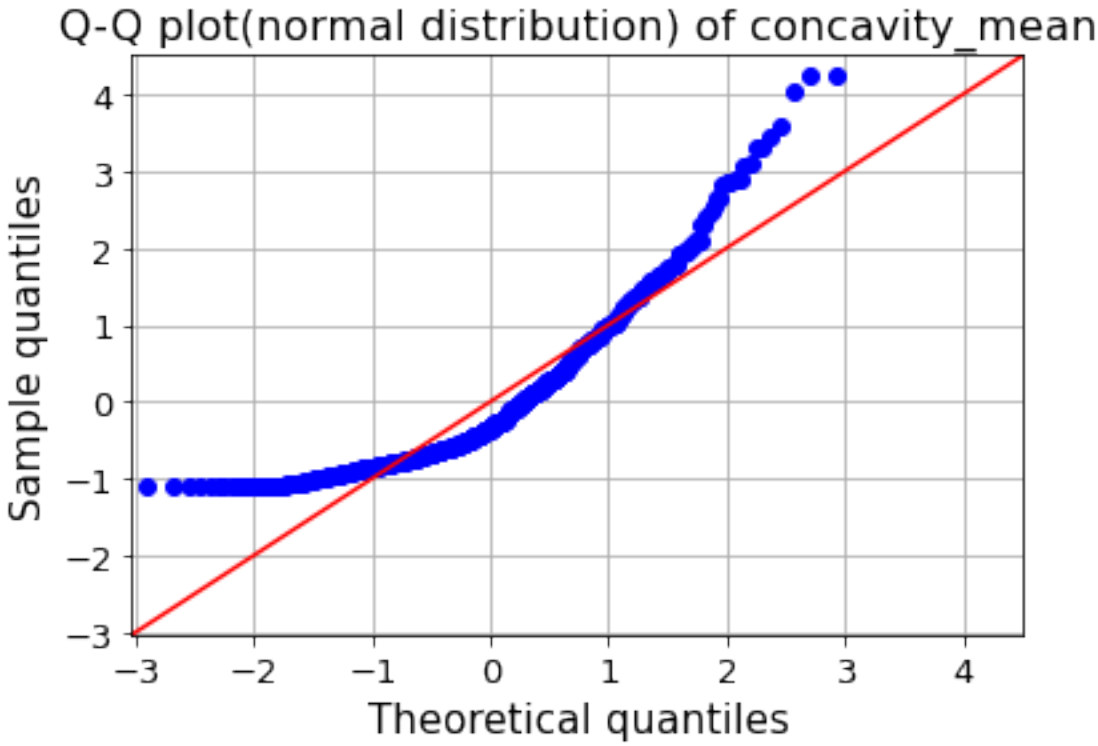

```
<Figure size 720x288 with 0 Axes>
```



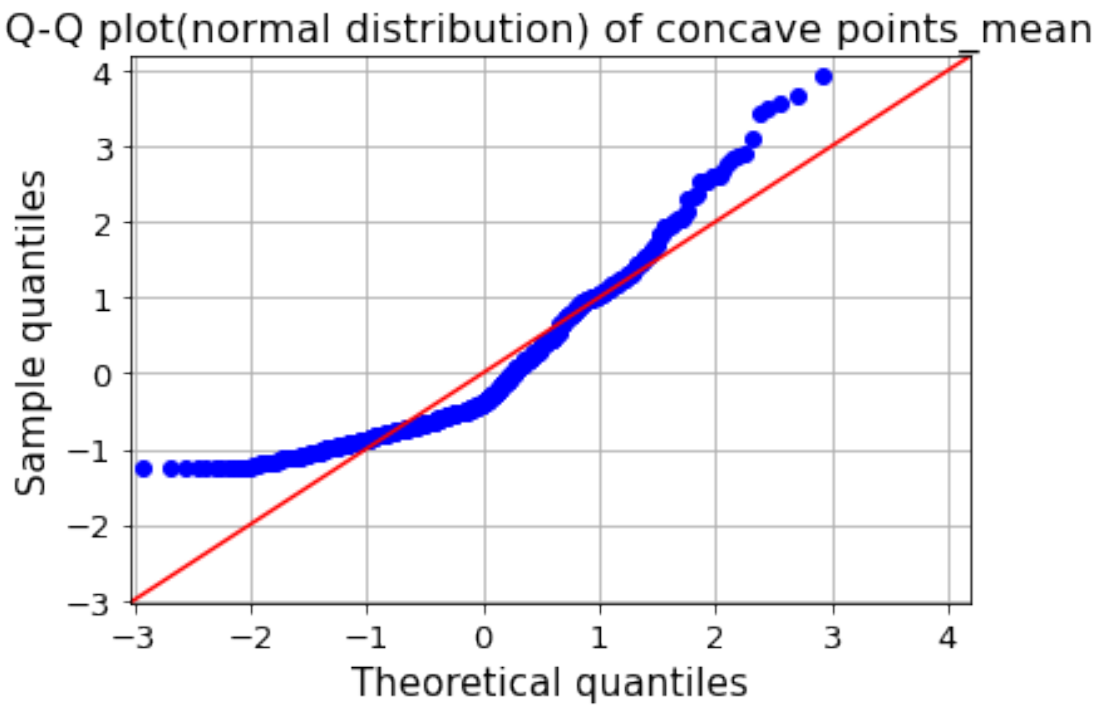

```
<Figure size 720x288 with 0 Axes>
```



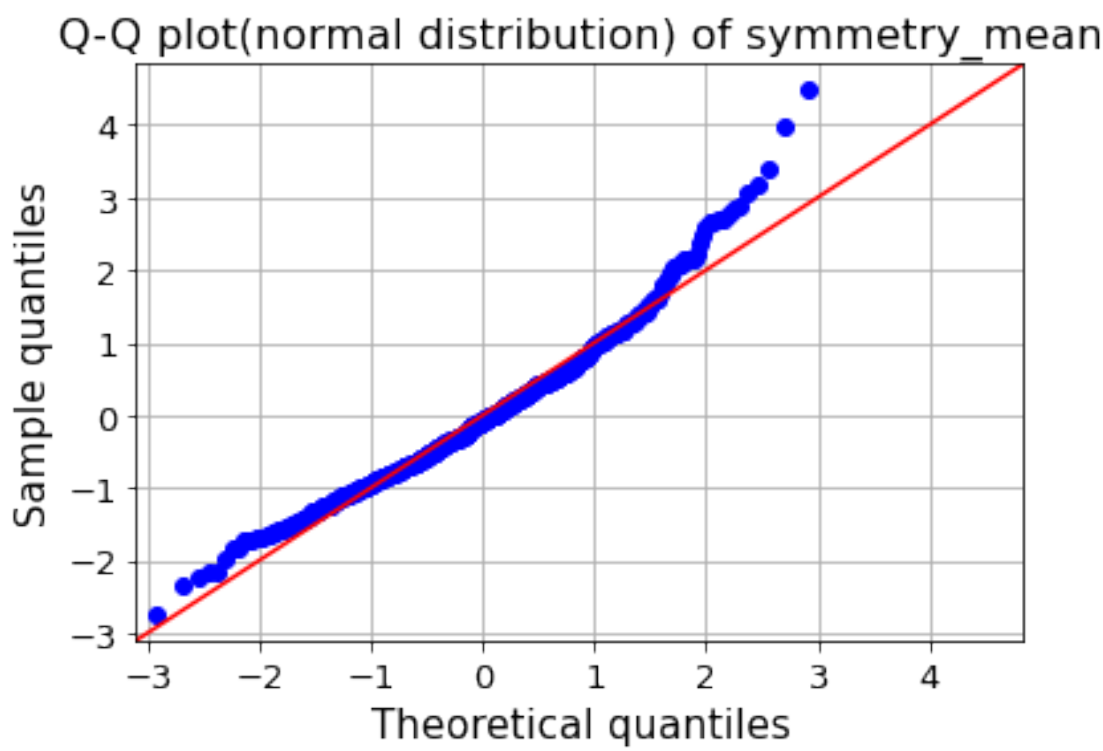
```
<Figure size 720x288 with 0 Axes>
```



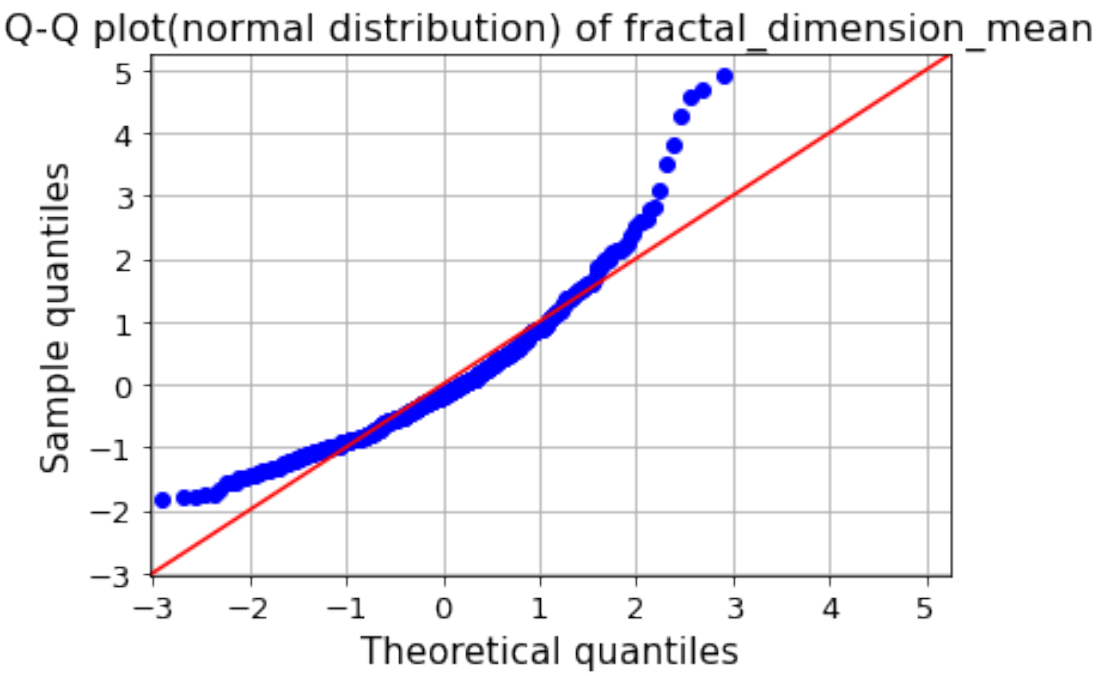

```
<Figure size 720x288 with 0 Axes>
```



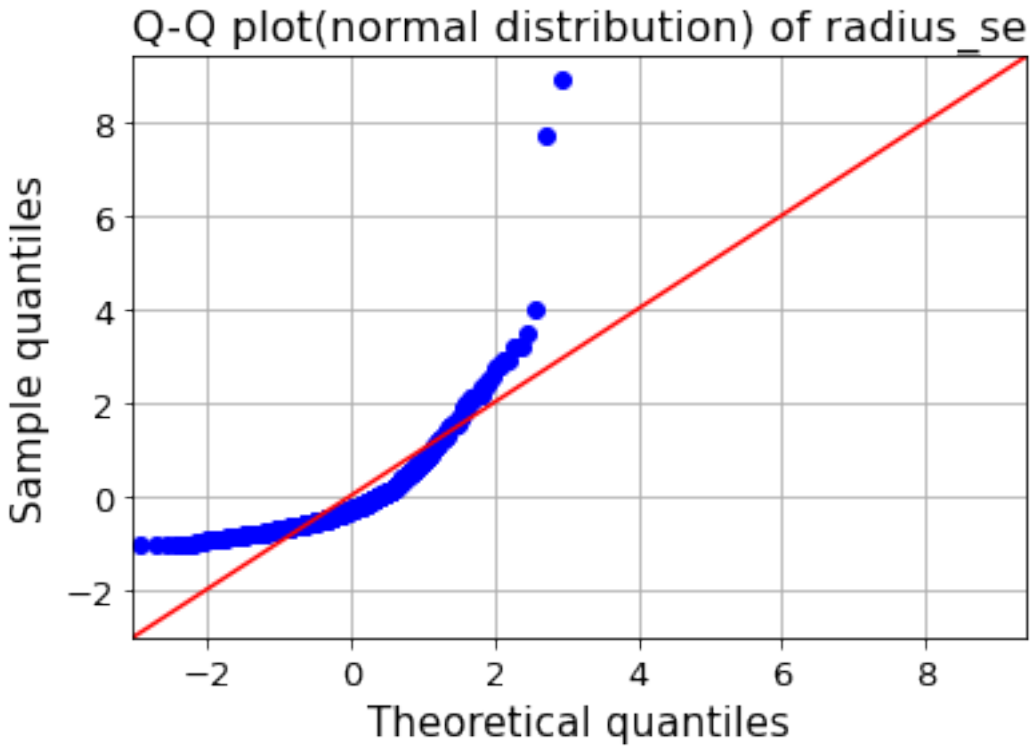

```
<Figure size 720x288 with 0 Axes>
```



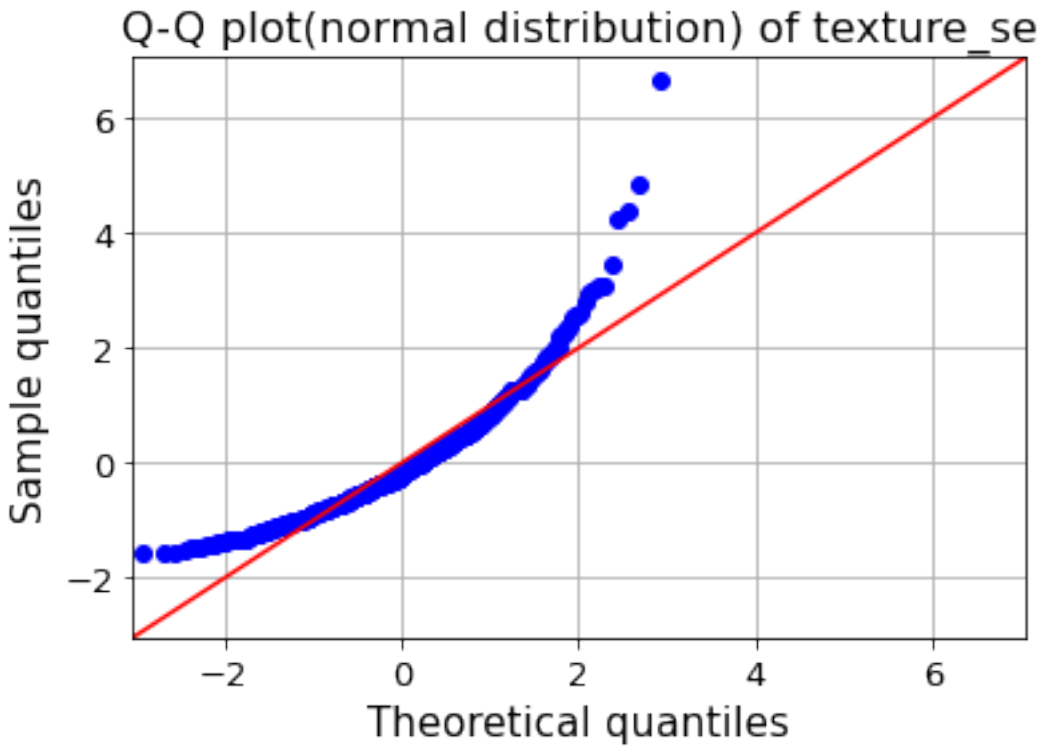
```
<Figure size 720x288 with 0 Axes>
```



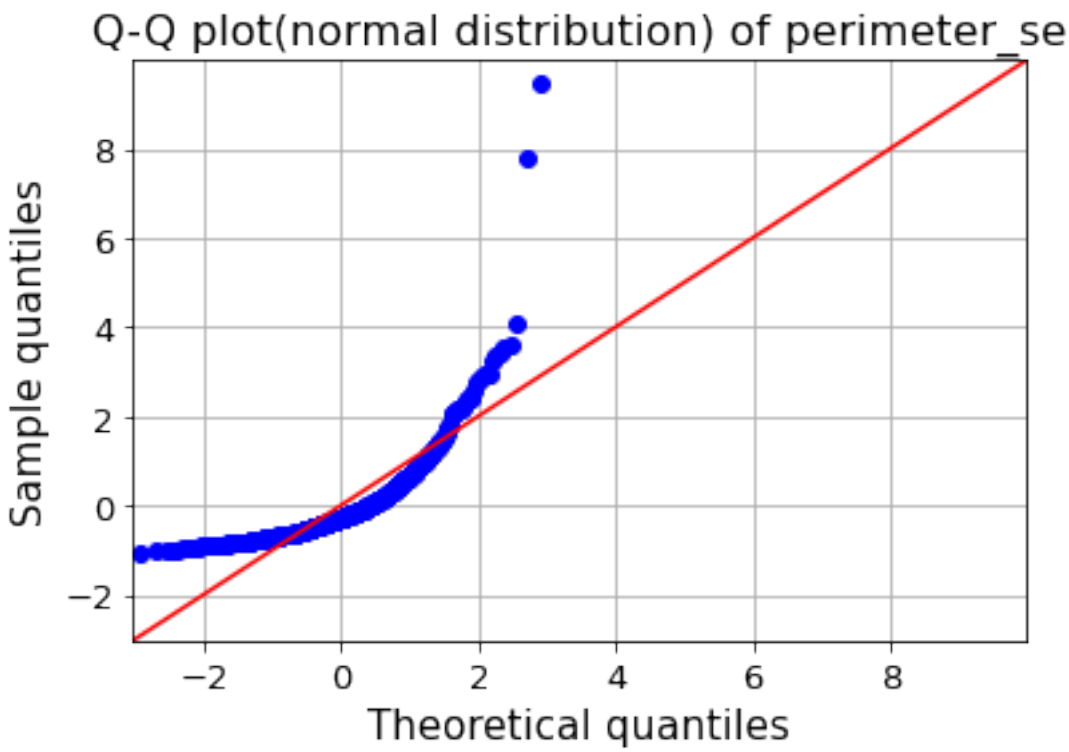

```
<Figure size 720x288 with 0 Axes>
```



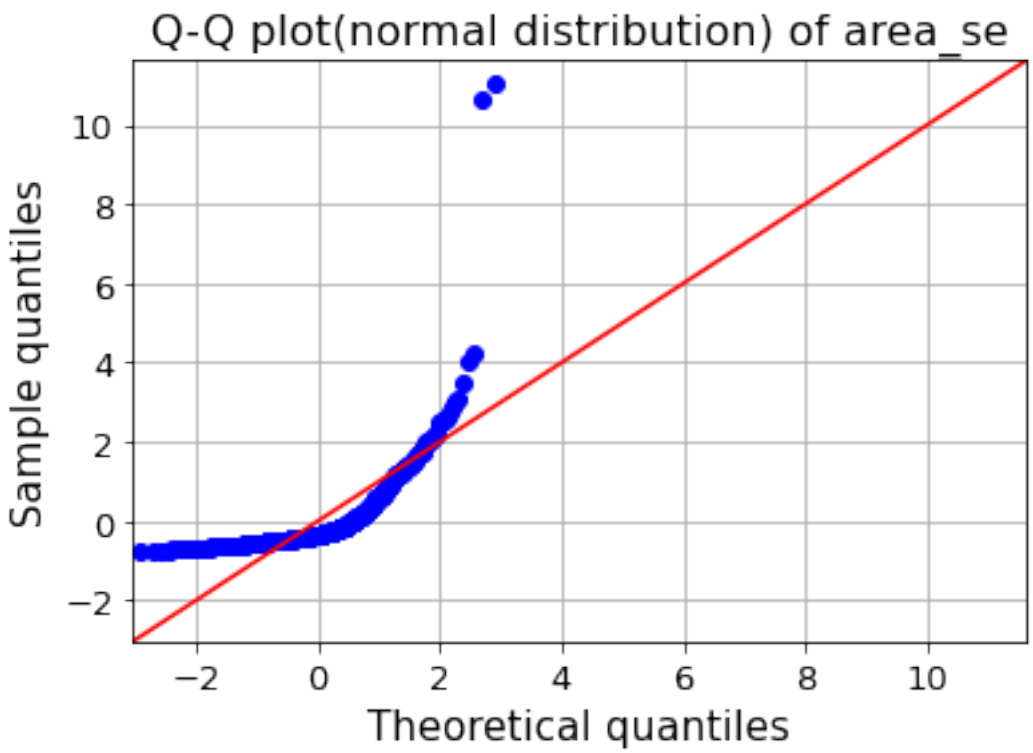

```
<Figure size 720x288 with 0 Axes>
```



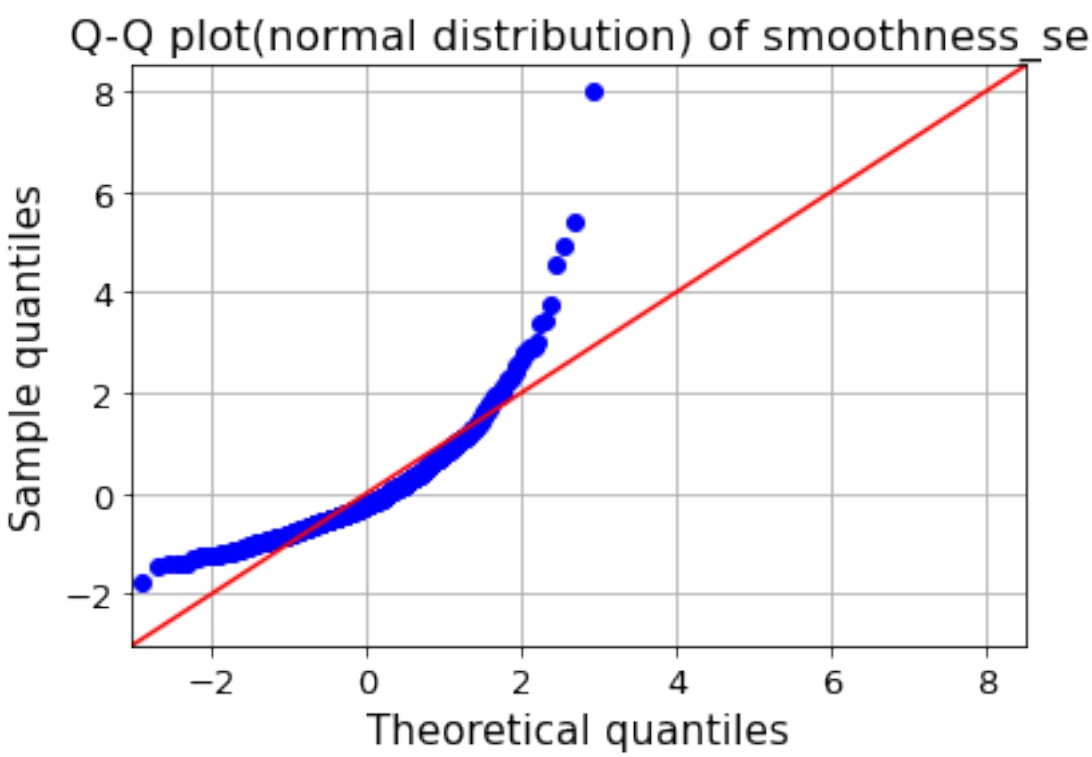
```
<Figure size 720x288 with 0 Axes>
```



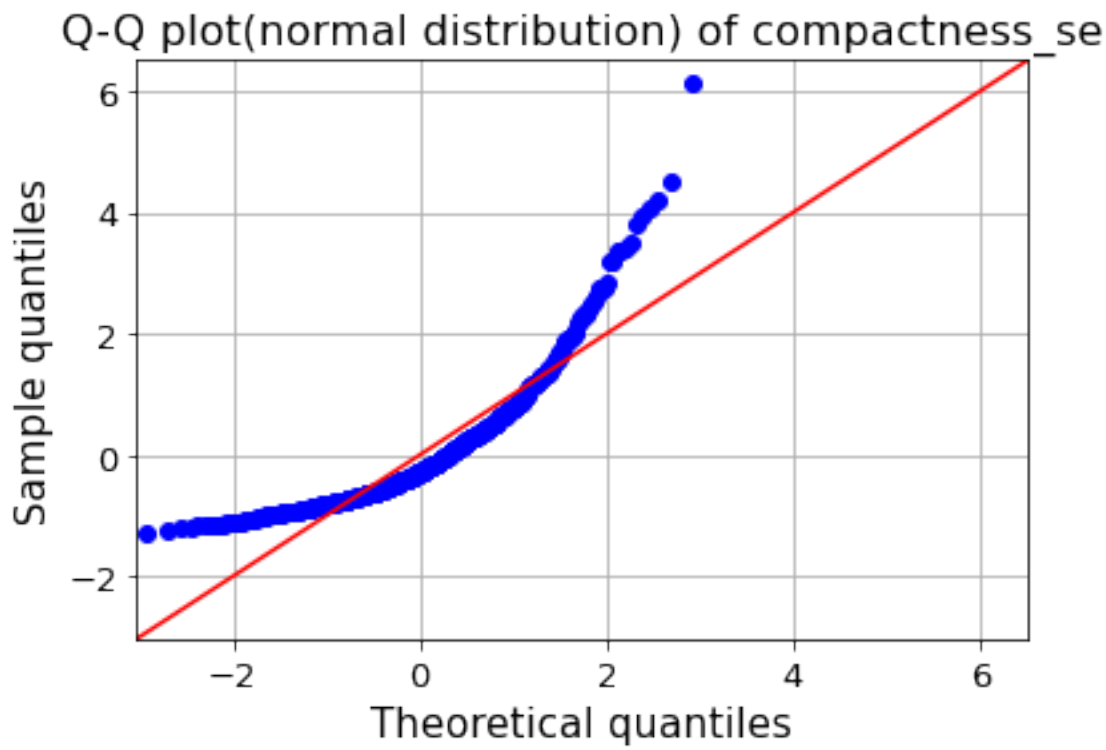

```
<Figure size 720x288 with 0 Axes>
```



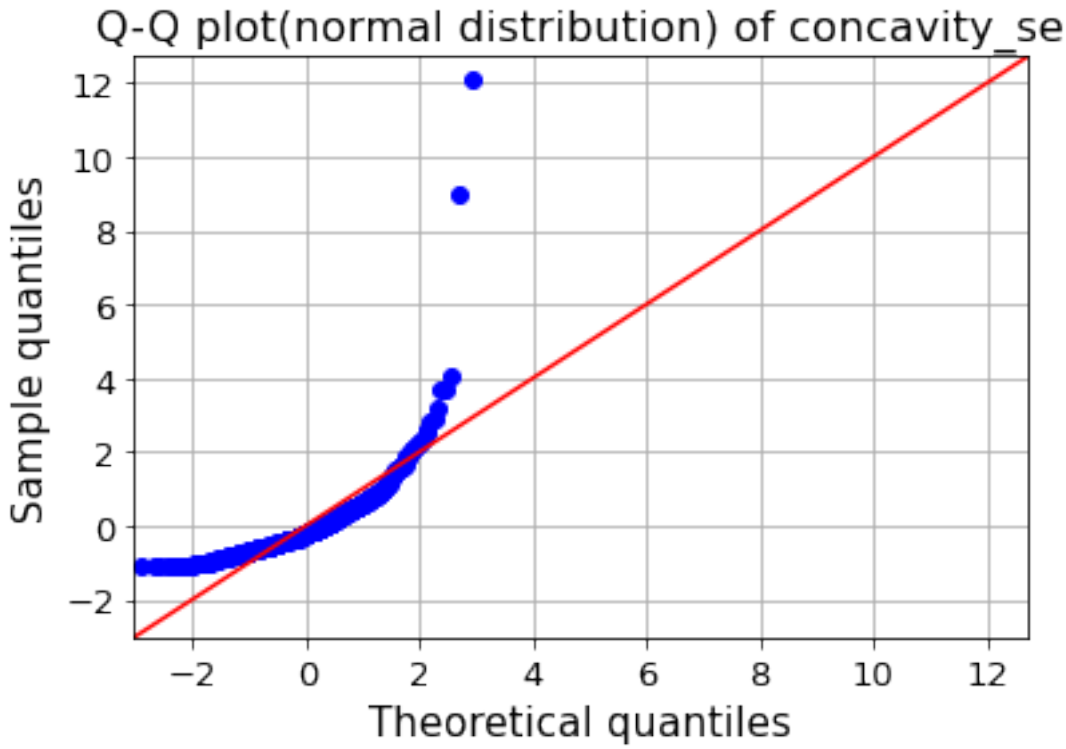

```
<Figure size 720x288 with 0 Axes>
```



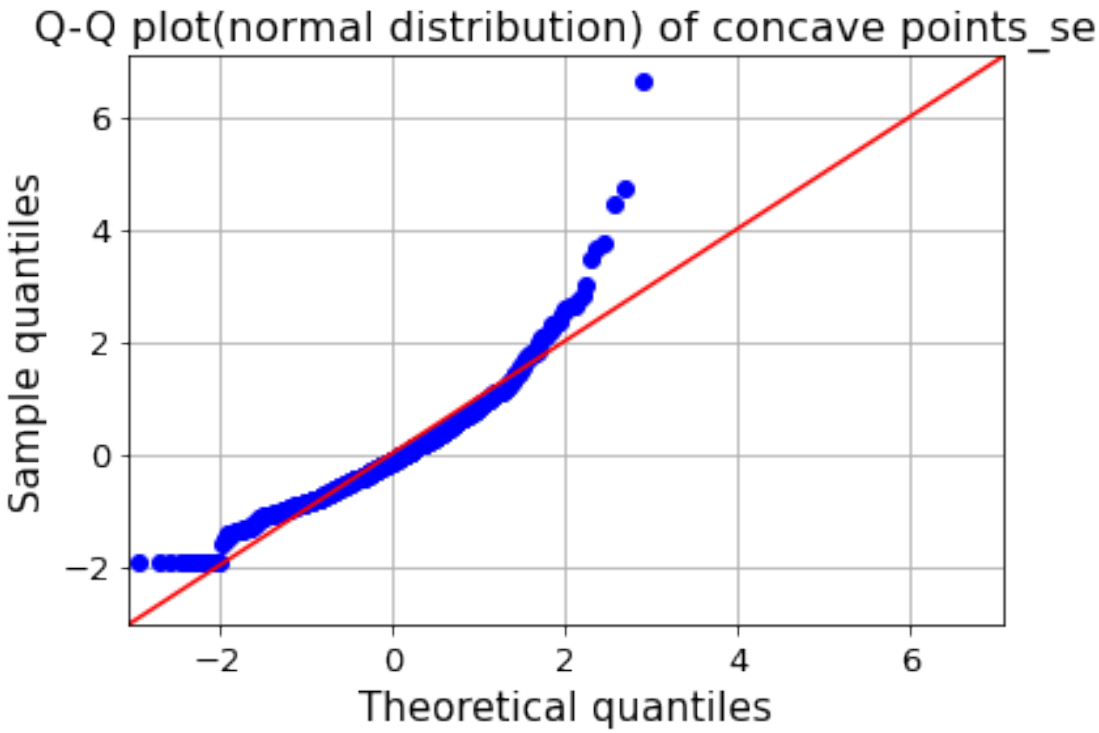
```
<Figure size 720x288 with 0 Axes>
```



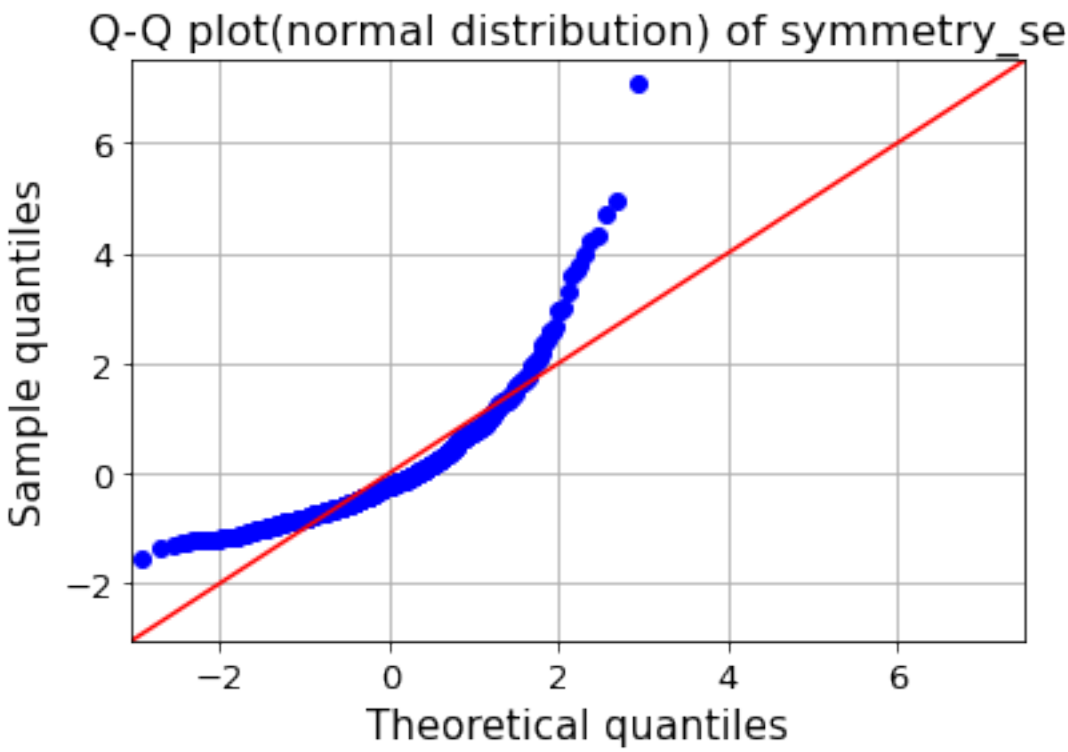

```
<Figure size 720x288 with 0 Axes>
```



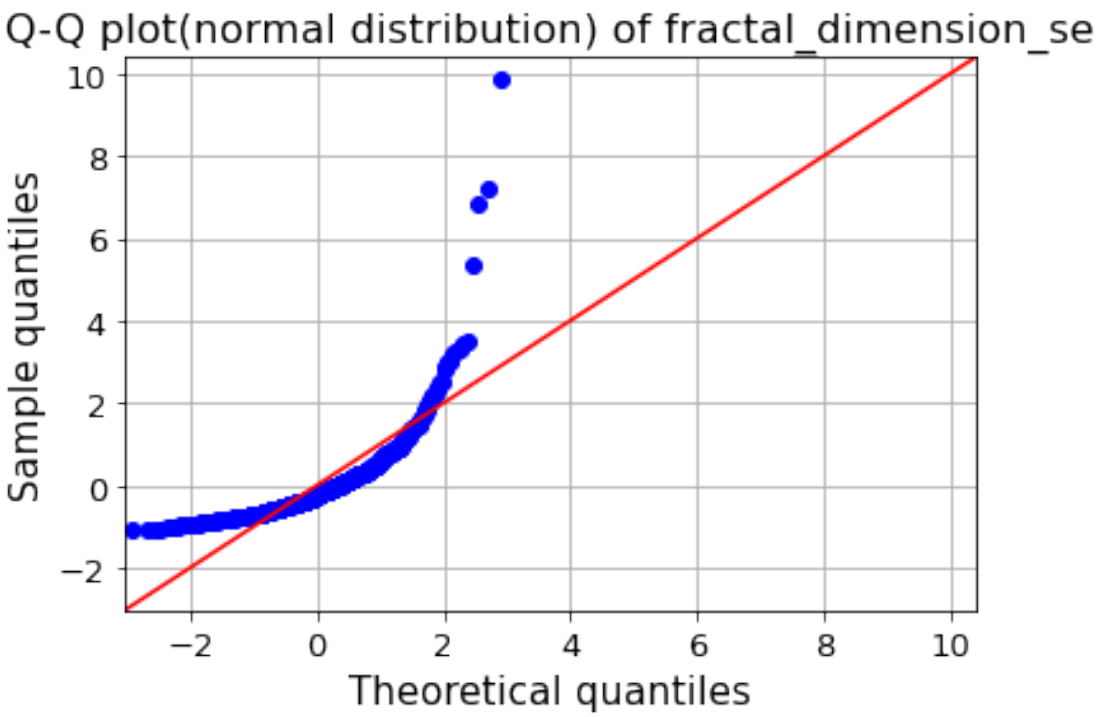

```
<Figure size 720x288 with 0 Axes>
```



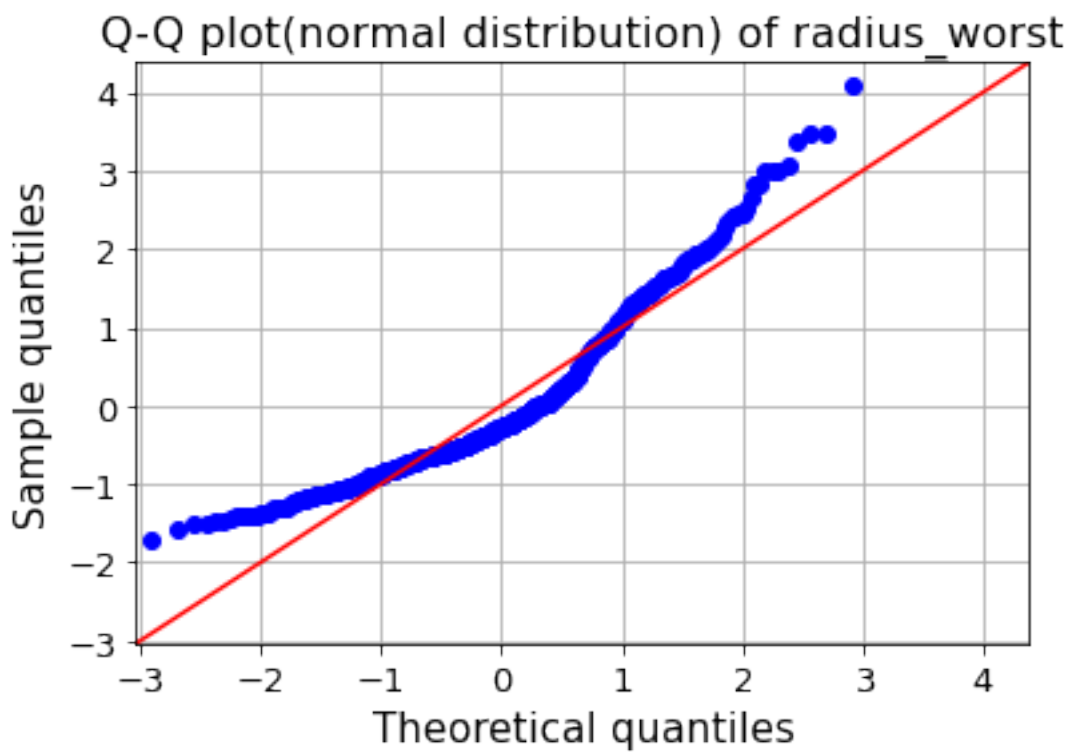
```
<Figure size 720x288 with 0 Axes>
```



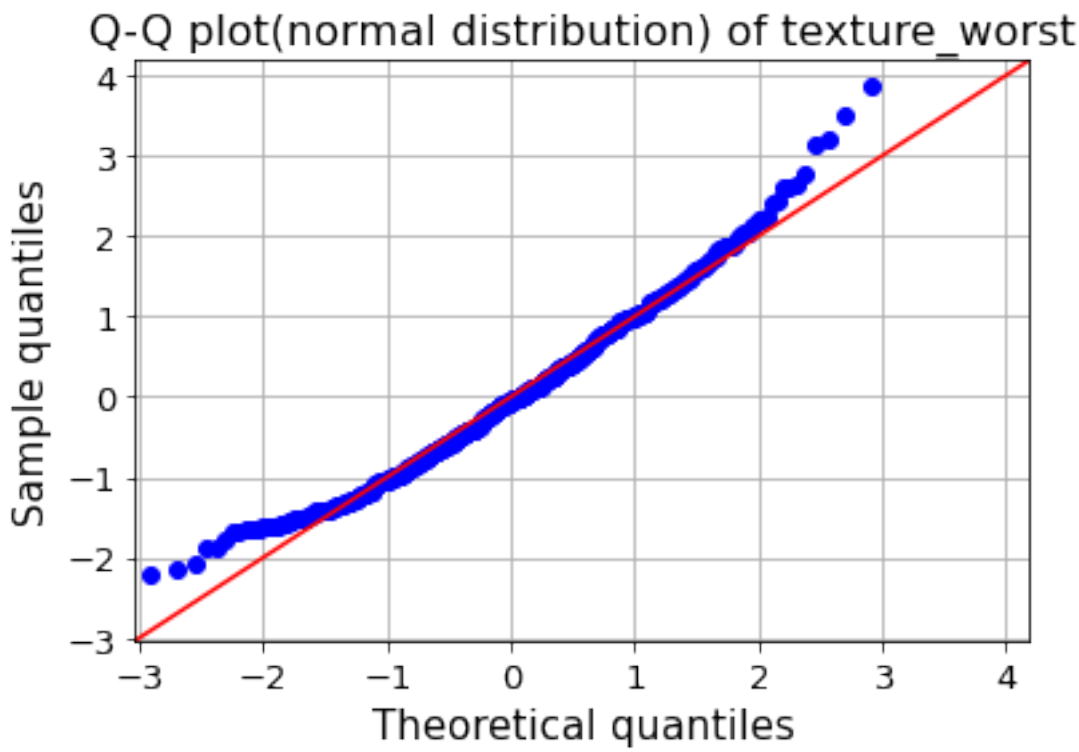

```
<Figure size 720x288 with 0 Axes>
```



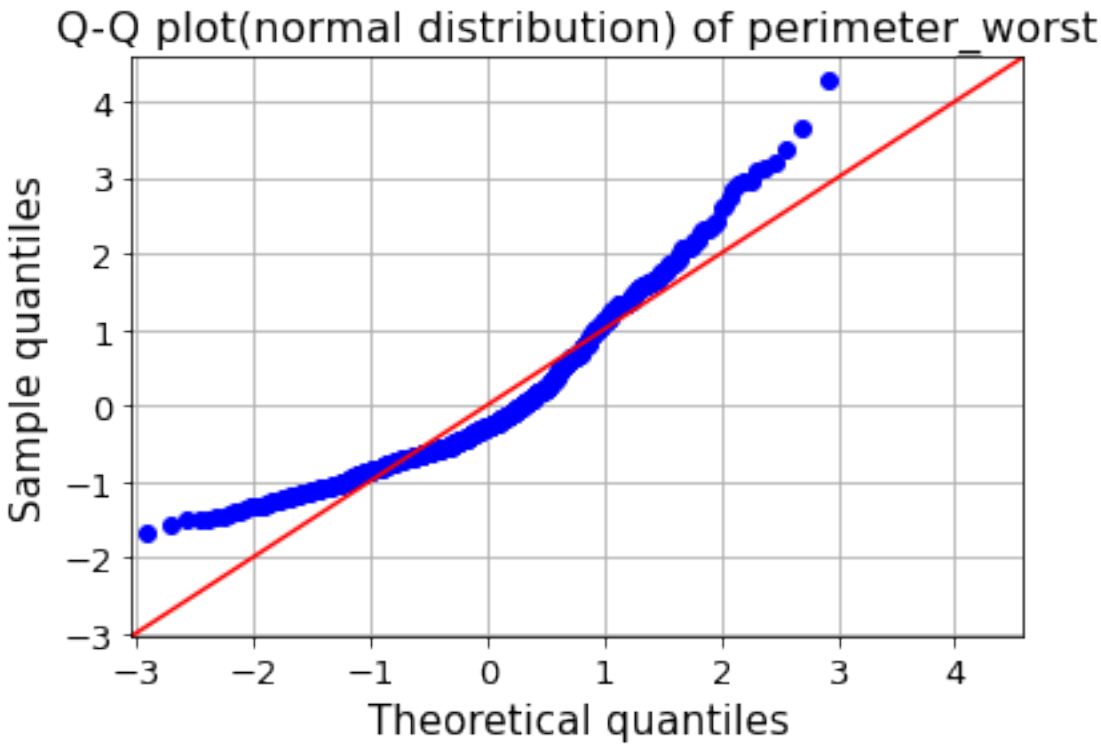

```
<Figure size 720x288 with 0 Axes>
```



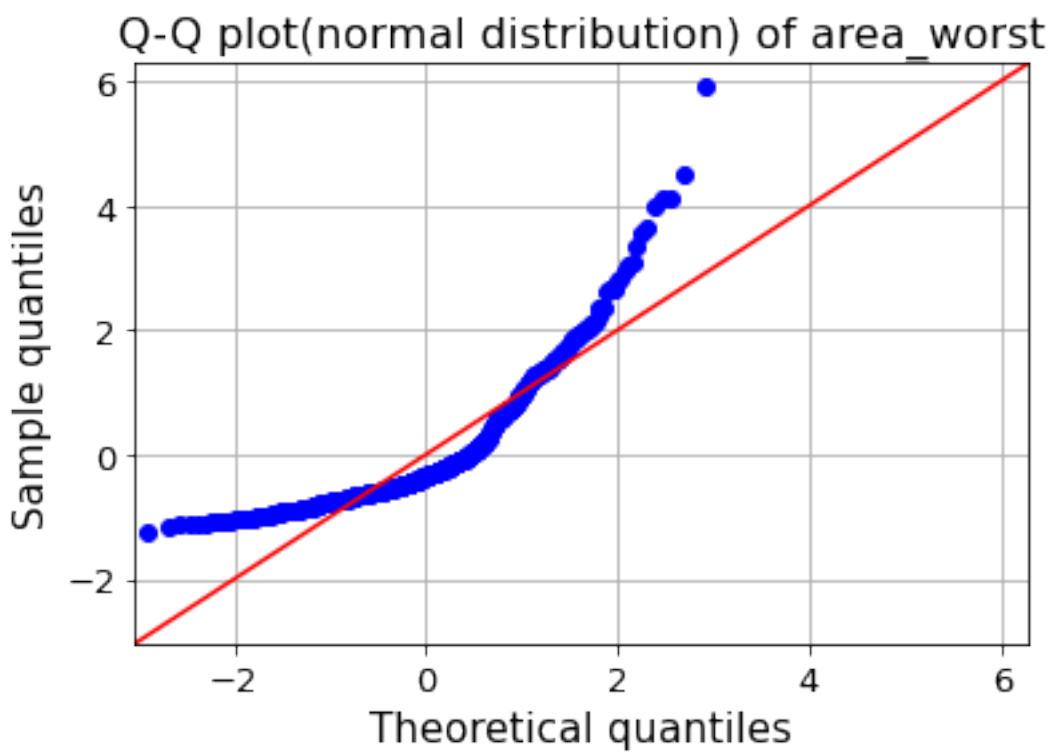
```
<Figure size 720x288 with 0 Axes>
```



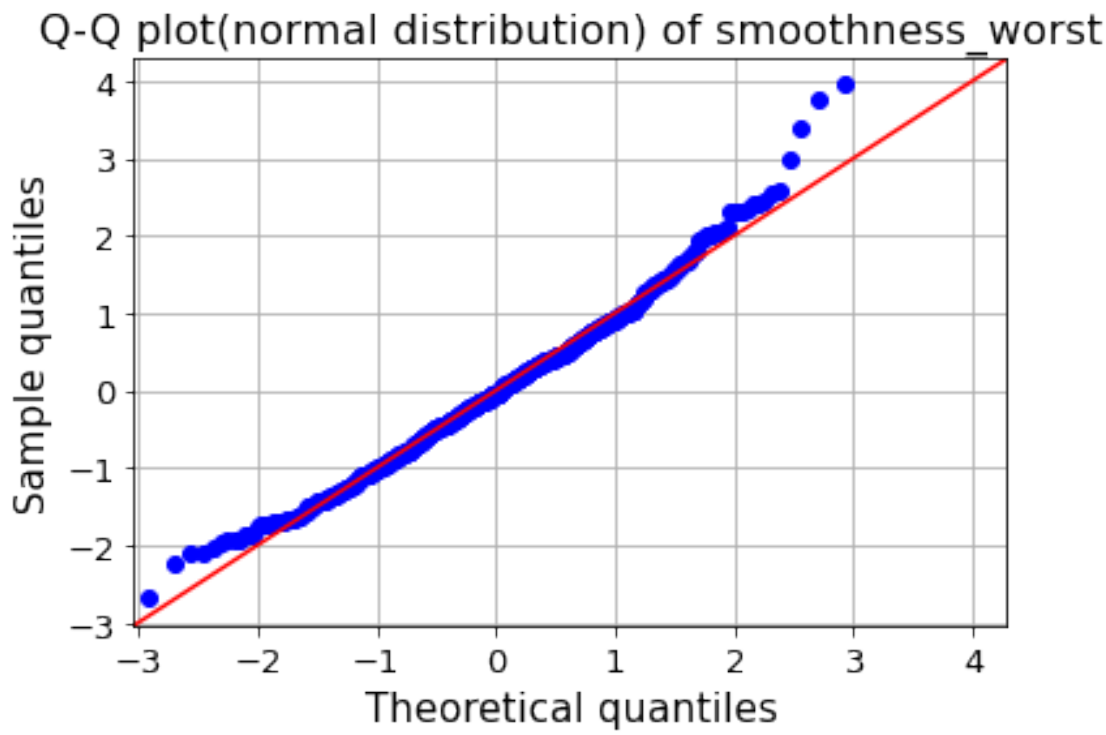

```
<Figure size 720x288 with 0 Axes>
```



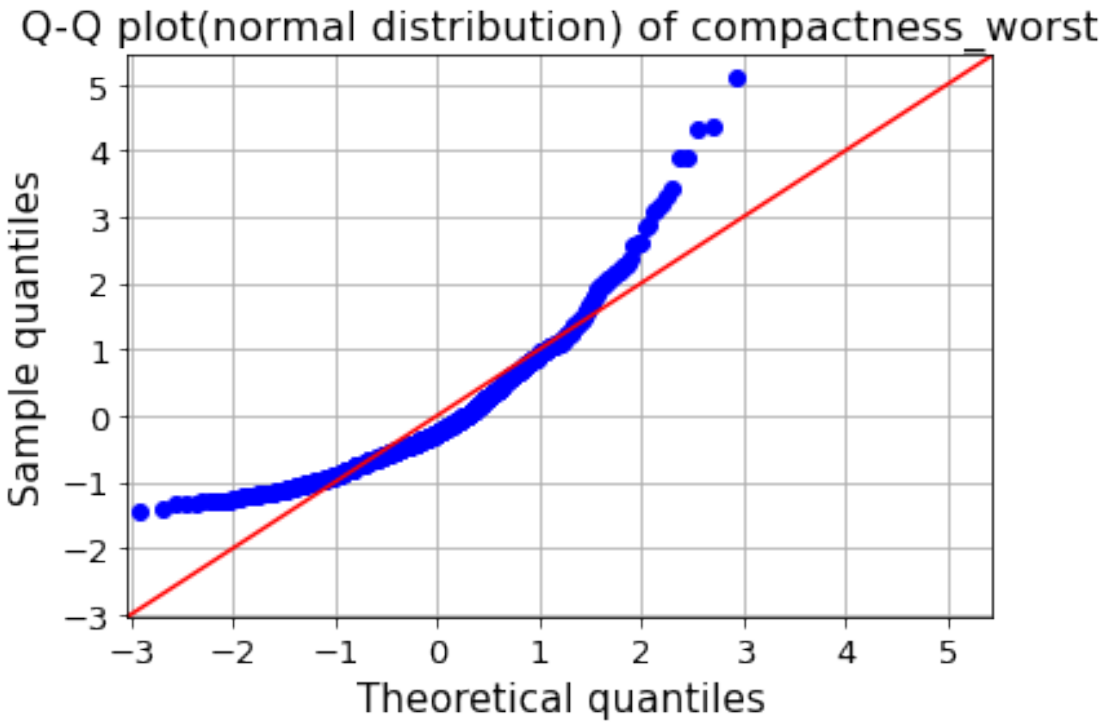

```
<Figure size 720x288 with 0 Axes>
```



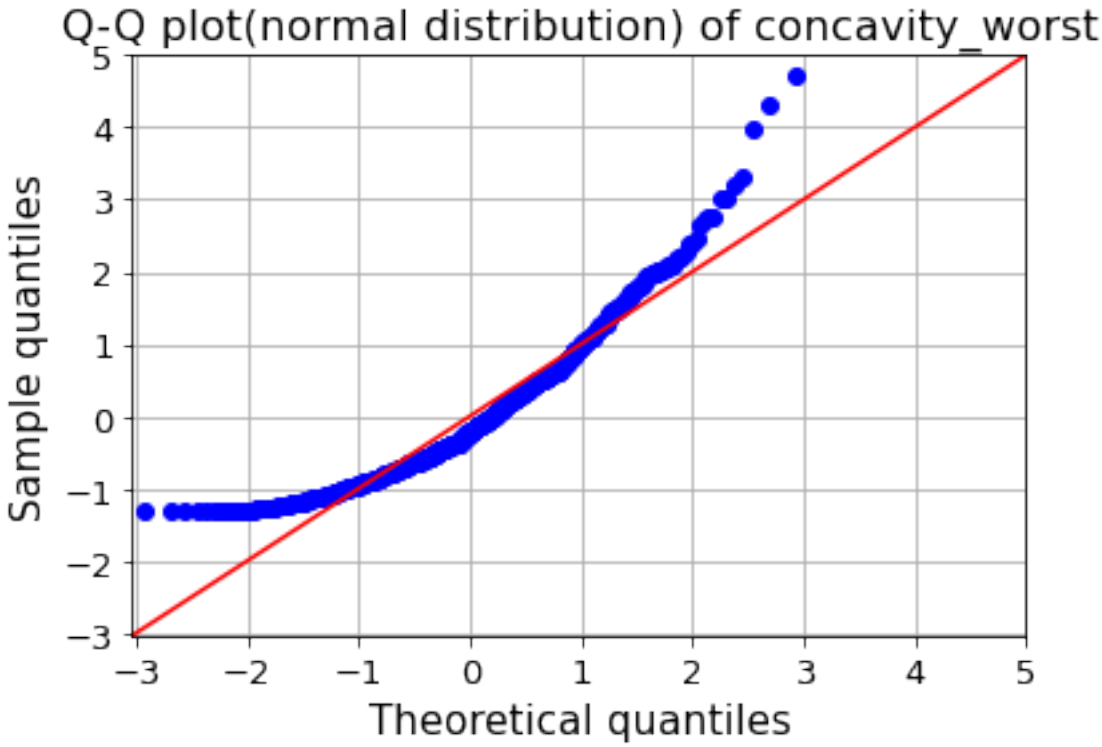
```
<Figure size 720x288 with 0 Axes>
```



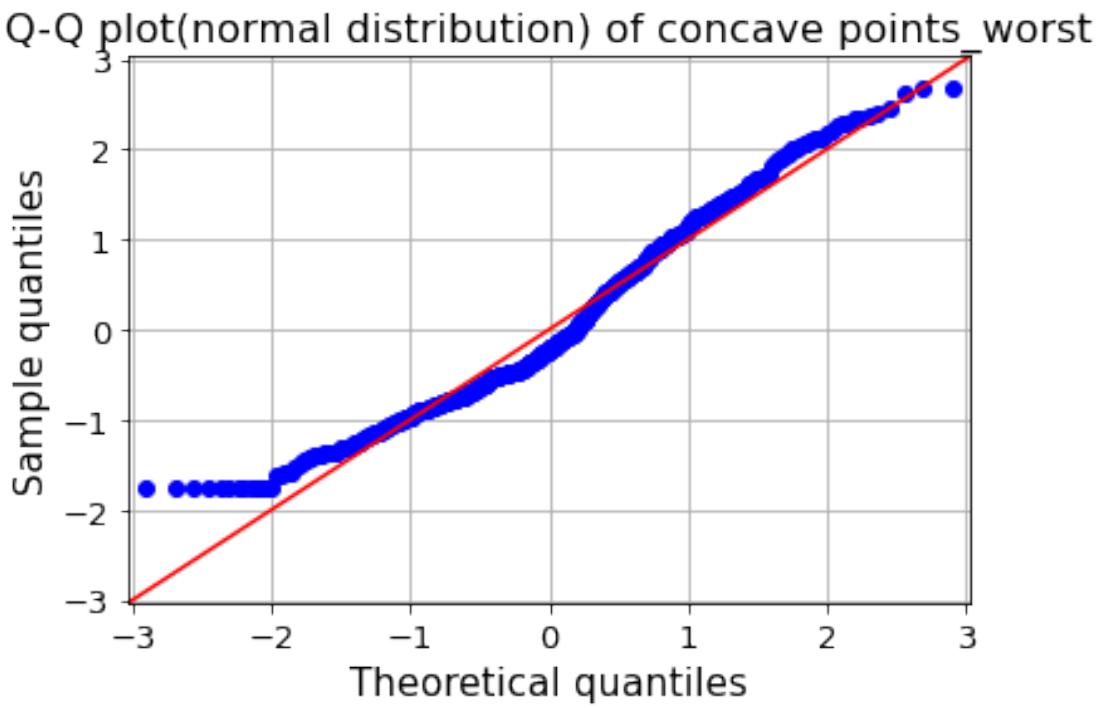

```
<Figure size 720x288 with 0 Axes>
```



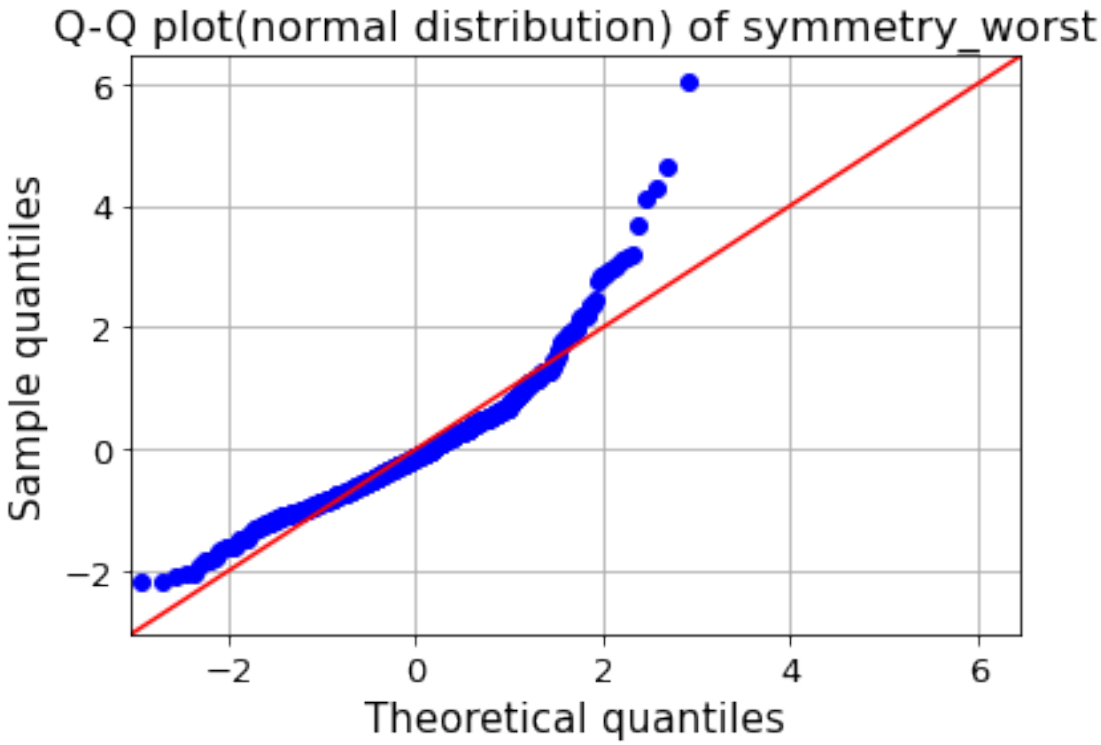

```
<Figure size 720x288 with 0 Axes>
```



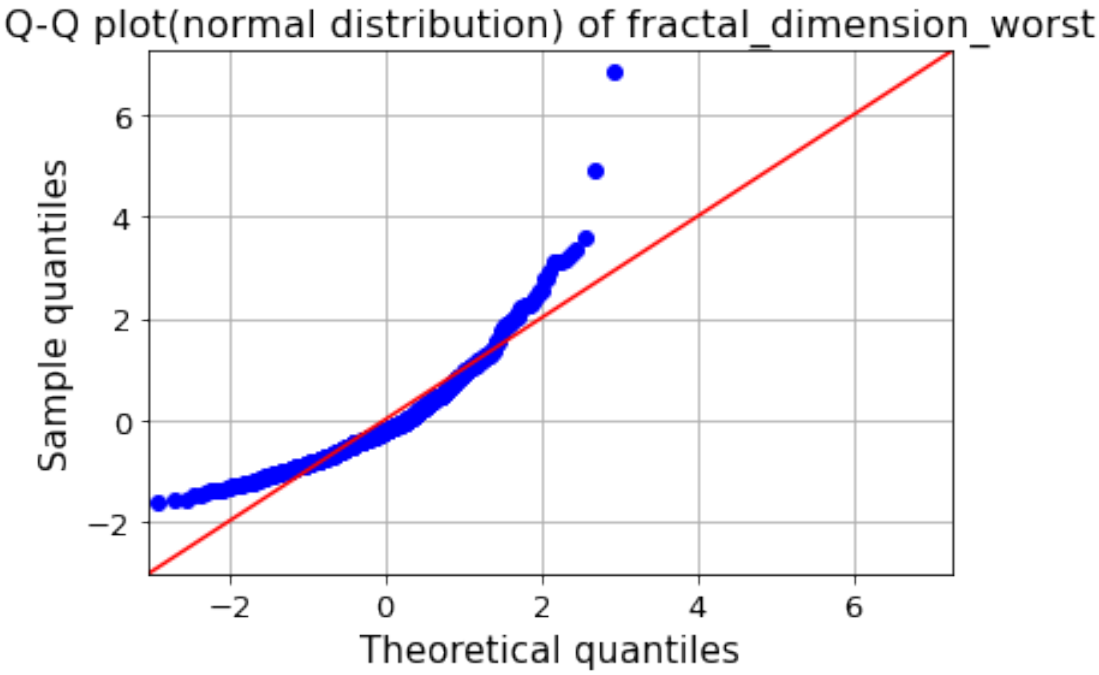
```
<Figure size 720x288 with 0 Axes>
```




```
<Figure size 720x288 with 0 Axes>
```

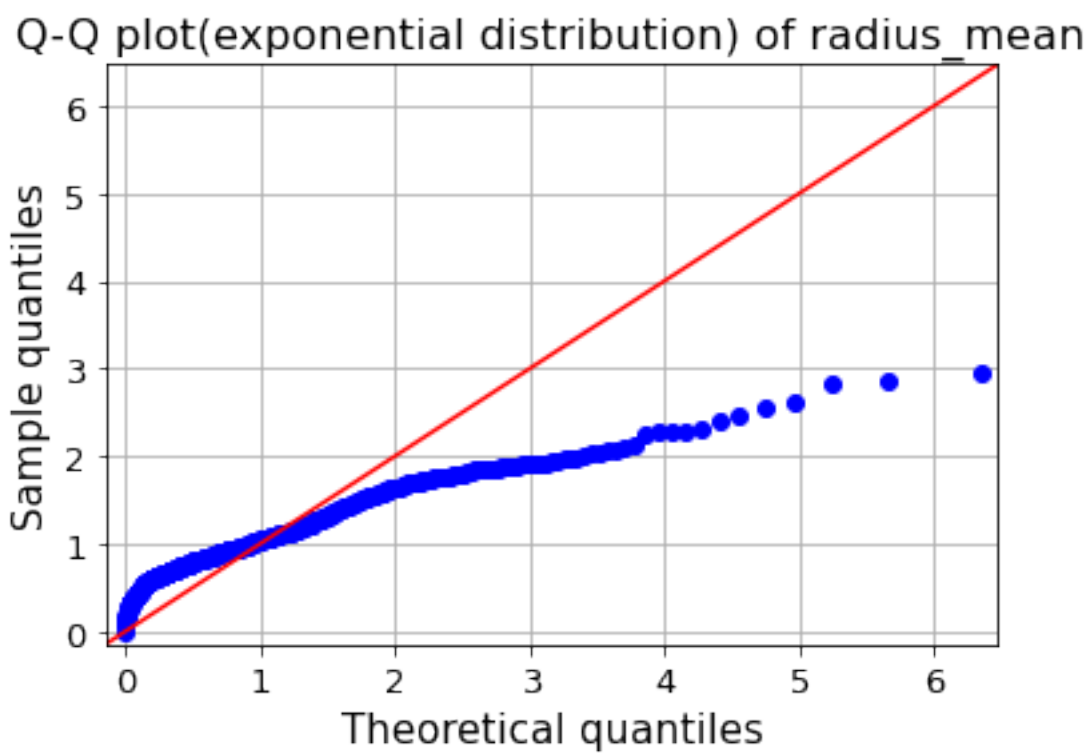



```
<Figure size 720x288 with 0 Axes>
```

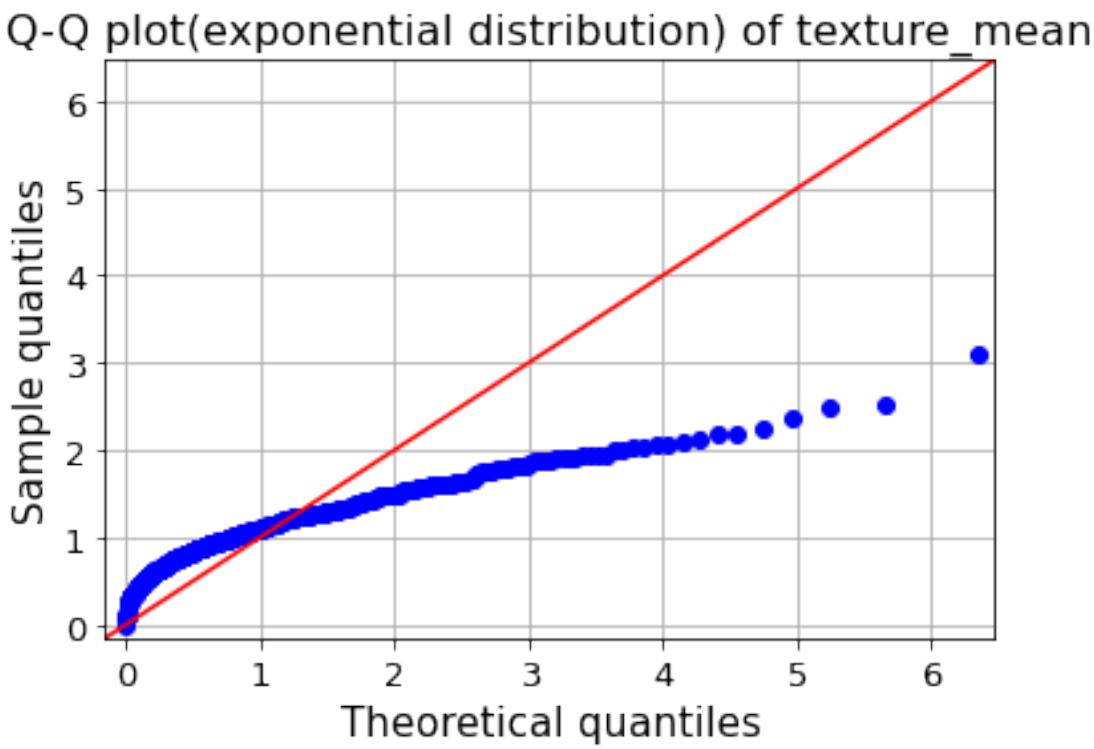


[14]:
```
for c in cols:
        plt.figure(figsize=(8,5))
        fig=qqplot(df[c],dist=sc.stats.expon,line="45",fit='True')
        plt.xticks(fontsize=13)
        plt.yticks(fontsize=13)
        plt.xlabel("Theoretical quantiles",fontsize=15)
        plt.ylabel("Sample quantiles",fontsize=15)
        plt.title("Q-Q plot(exponential distribution) of {}".
 ↪format(c),fontsize=16)
        plt.grid(True)
        plt.show()
```

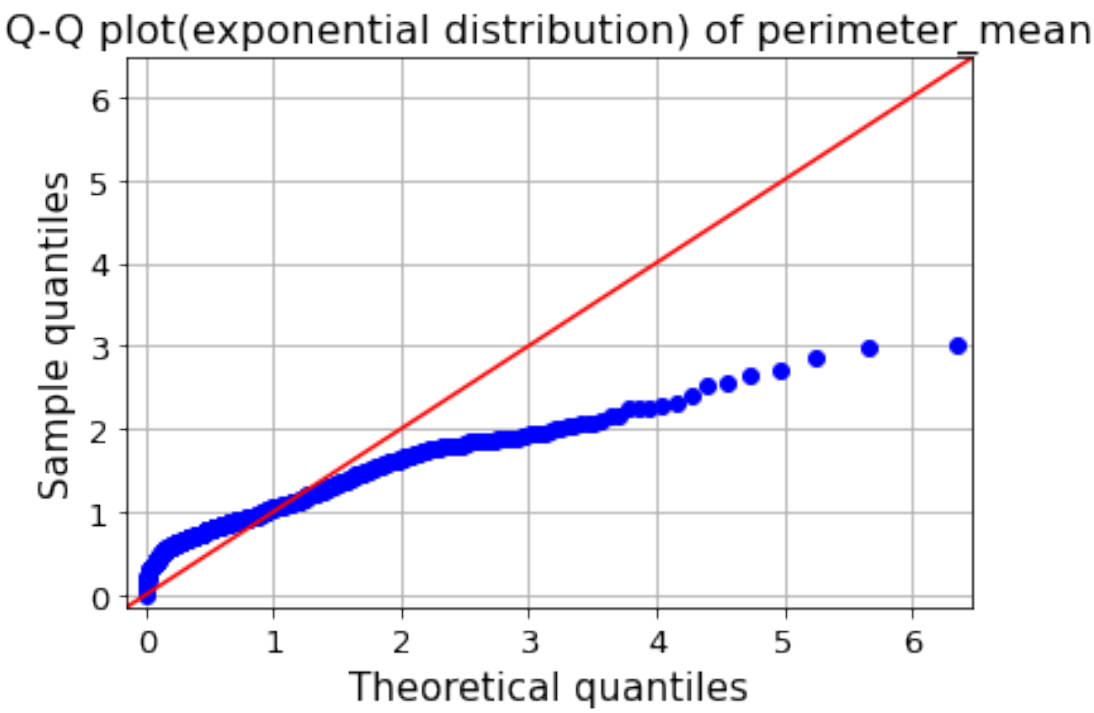
```
<Figure size 576x360 with 0 Axes>
```



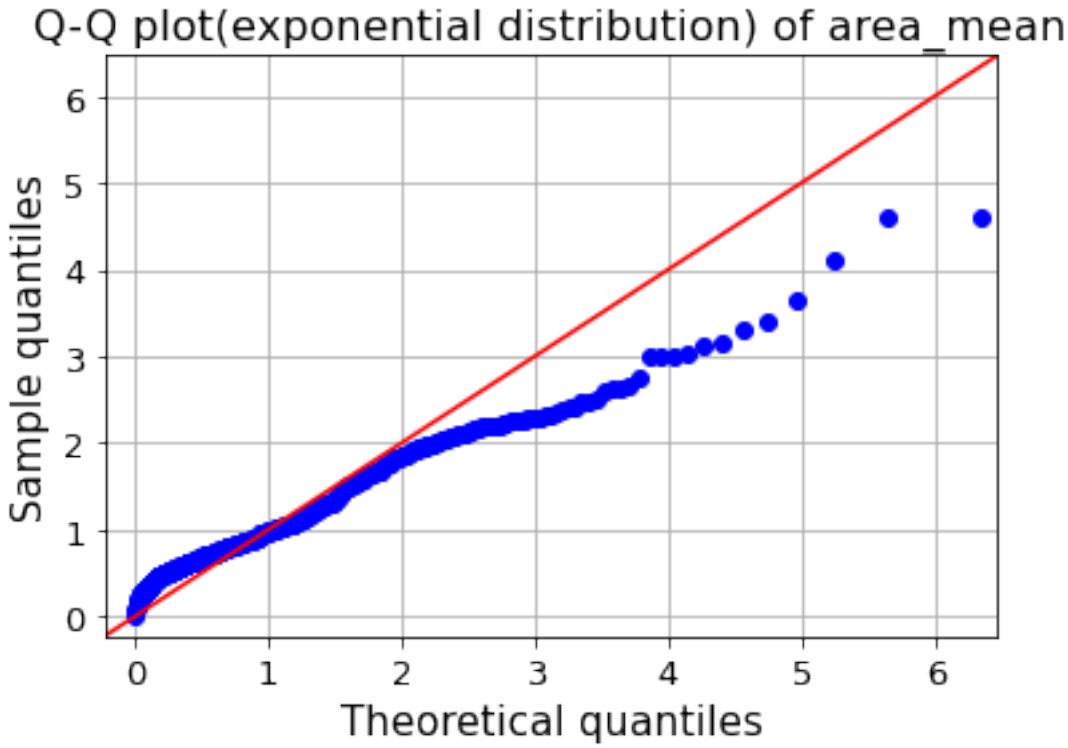

```
<Figure size 576x360 with 0 Axes>
```



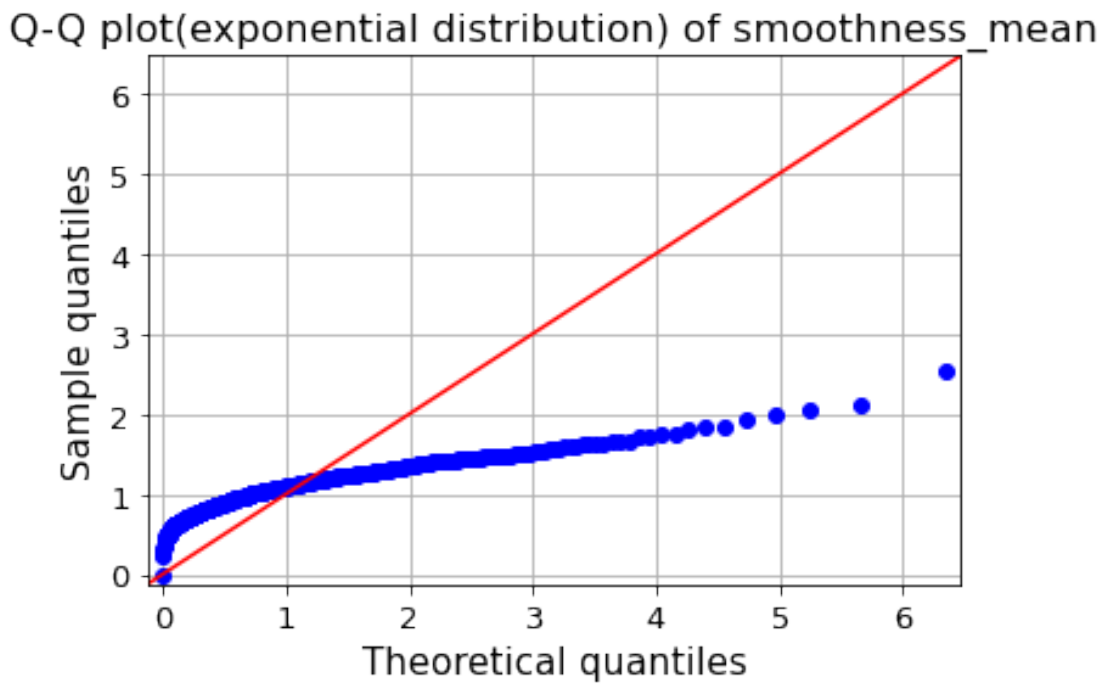

```
<Figure size 576x360 with 0 Axes>
```



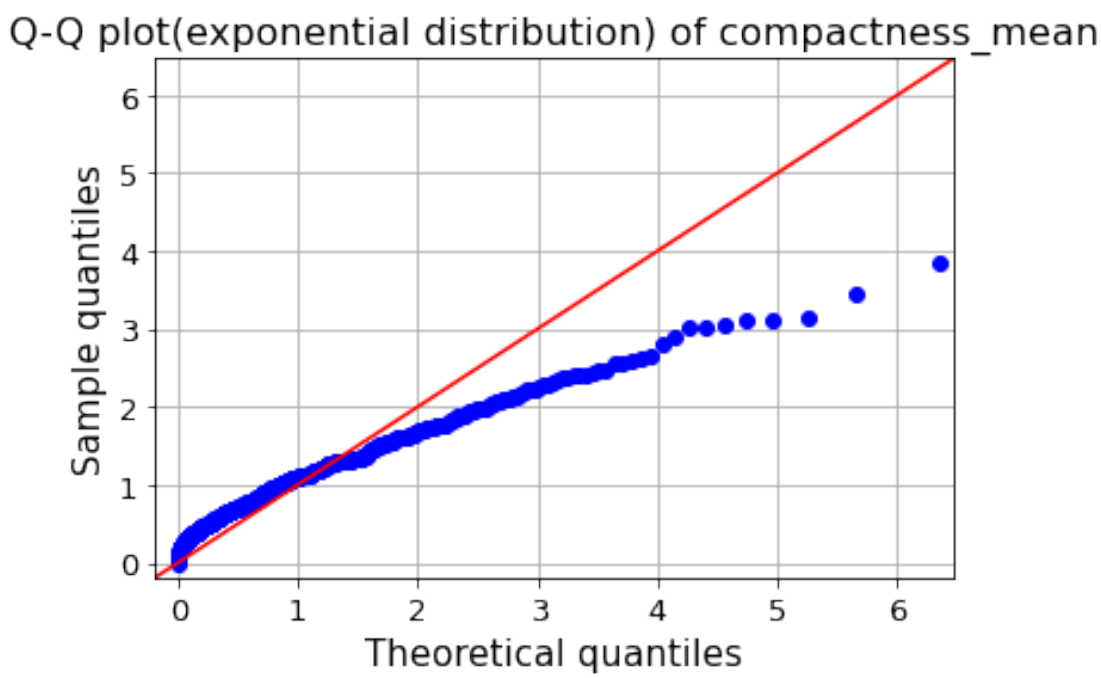
```
<Figure size 576x360 with 0 Axes>
```



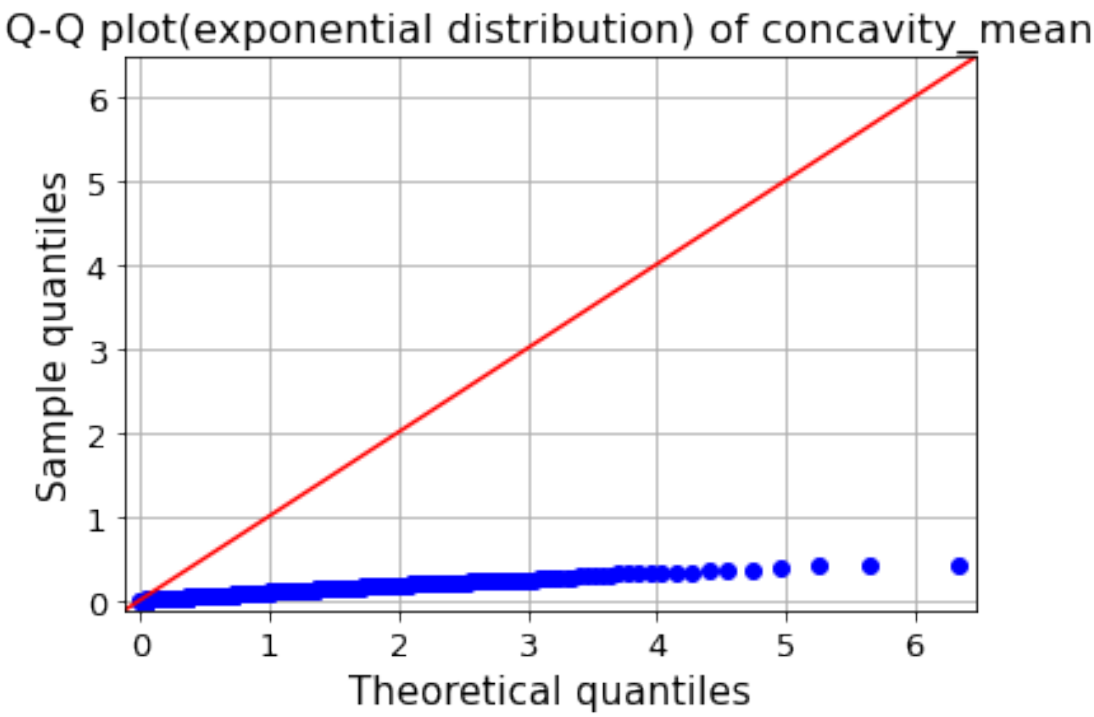

```
<Figure size 576x360 with 0 Axes>
```



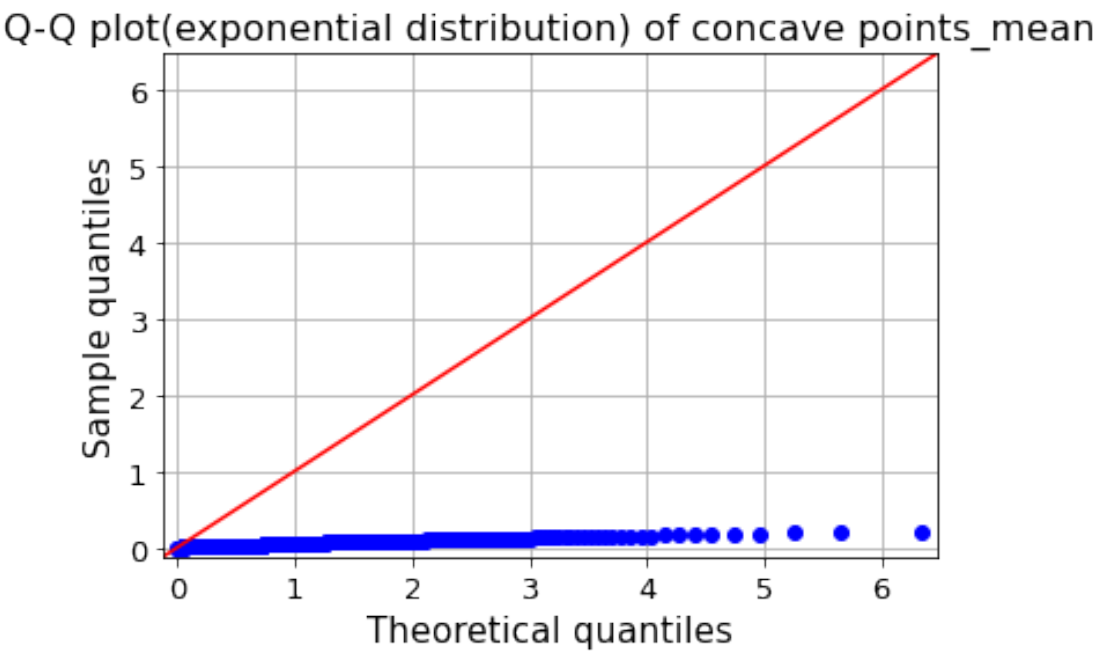

```
<Figure size 576x360 with 0 Axes>
```



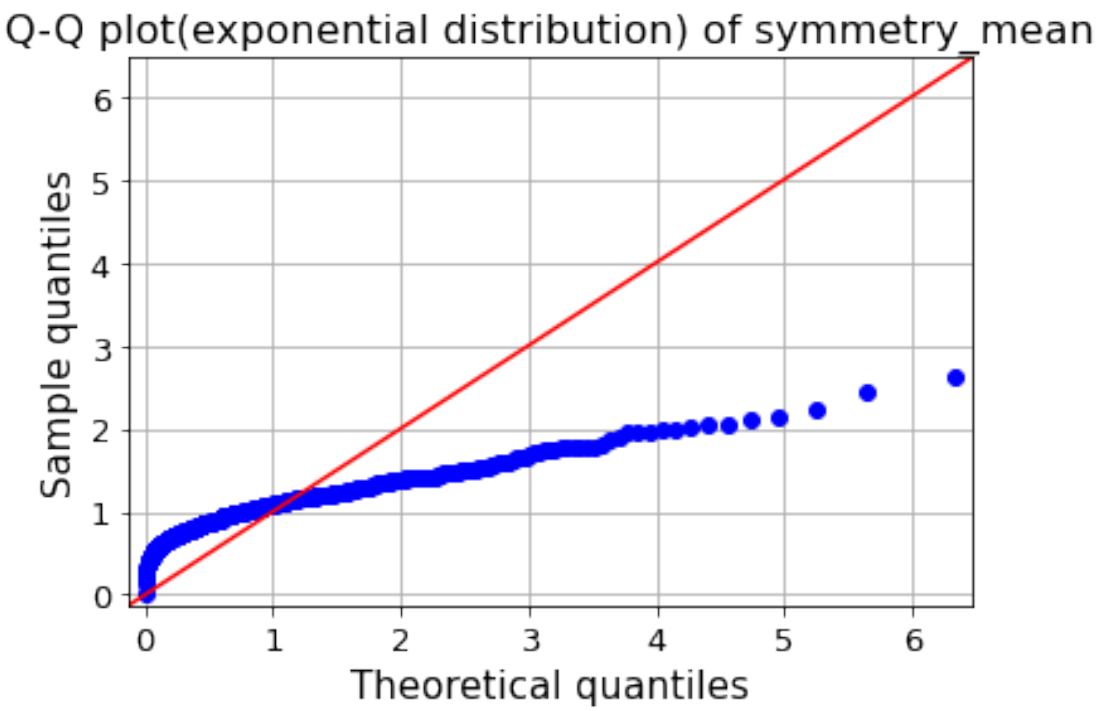

```
<Figure size 576x360 with 0 Axes>
```



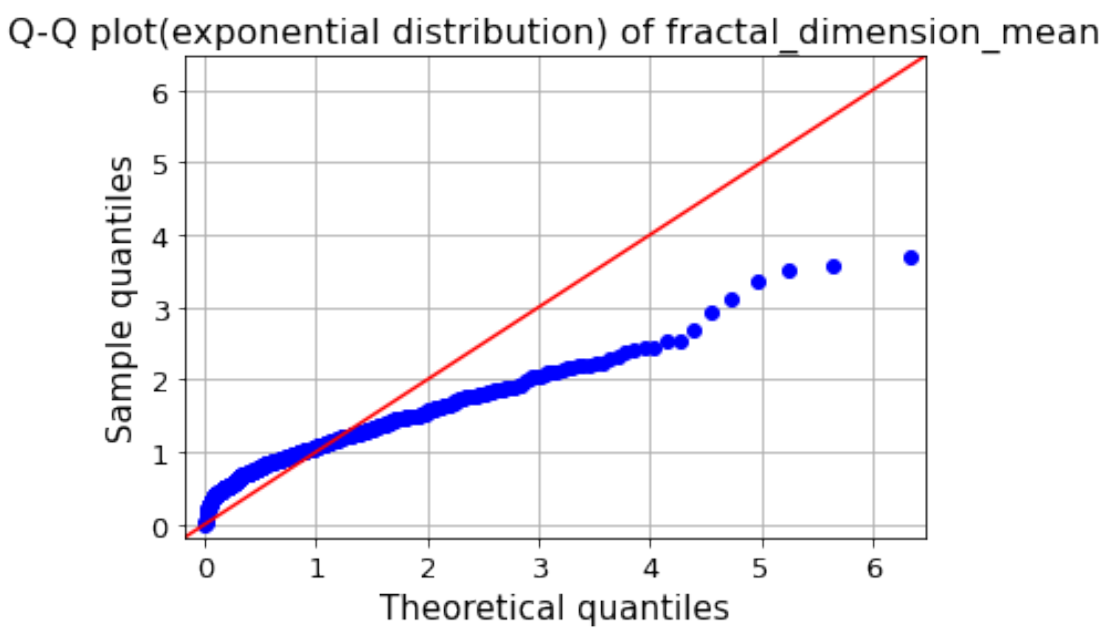

```
<Figure size 576x360 with 0 Axes>
```



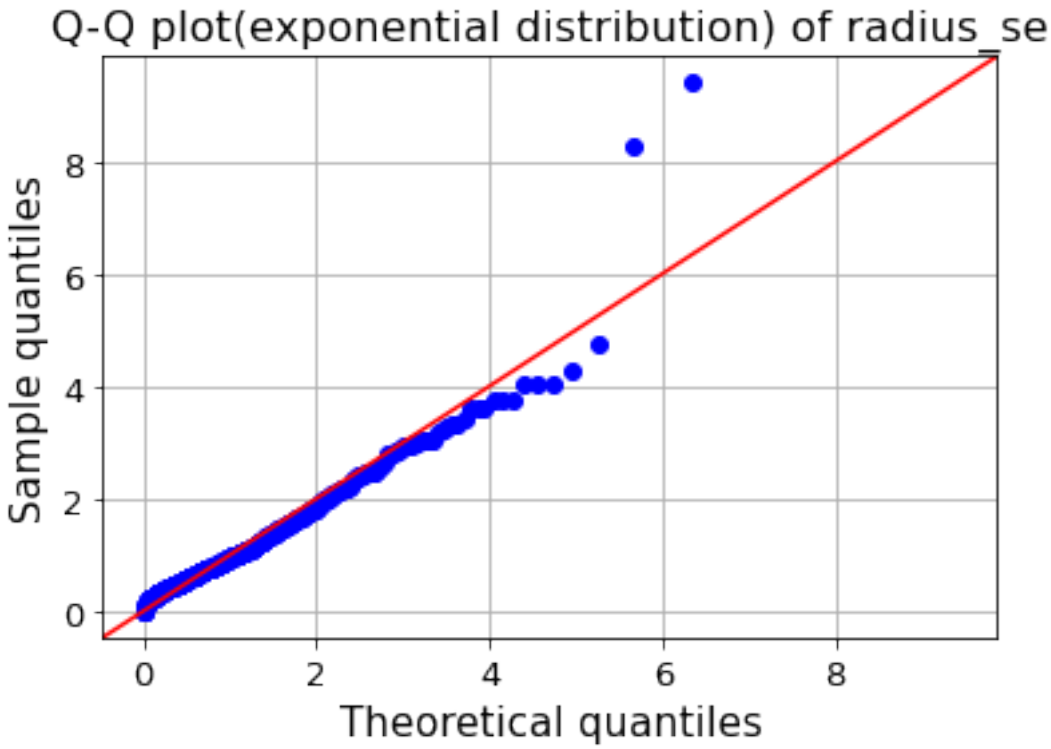

```
<Figure size 576x360 with 0 Axes>
```



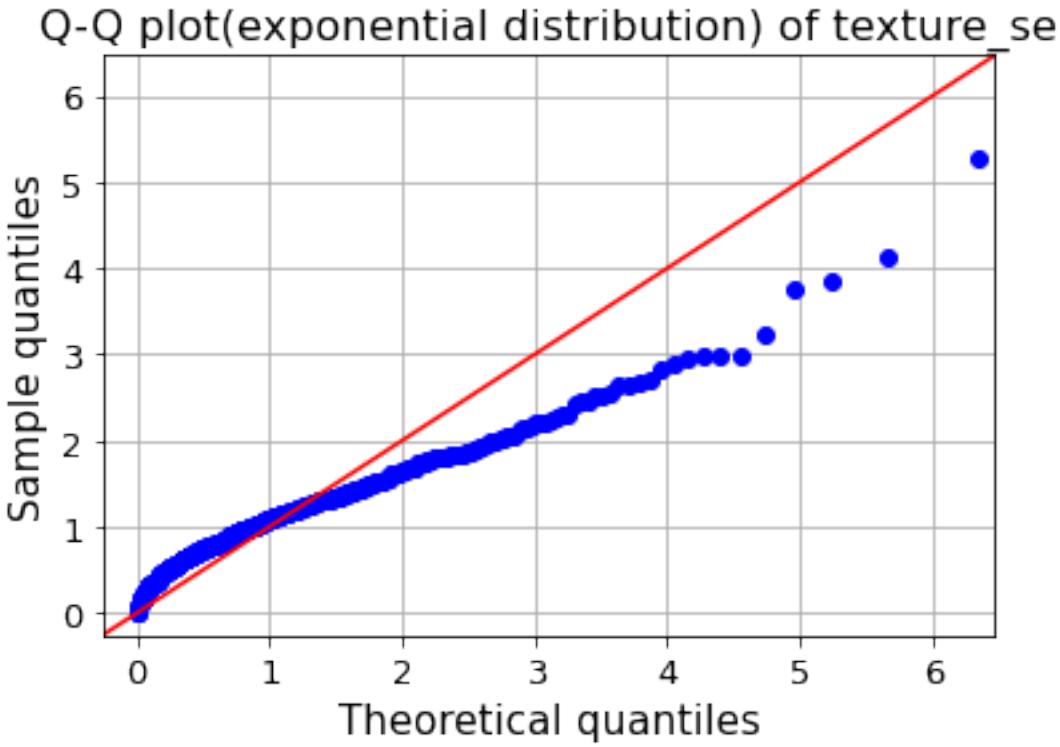

```
<Figure size 576x360 with 0 Axes>
```



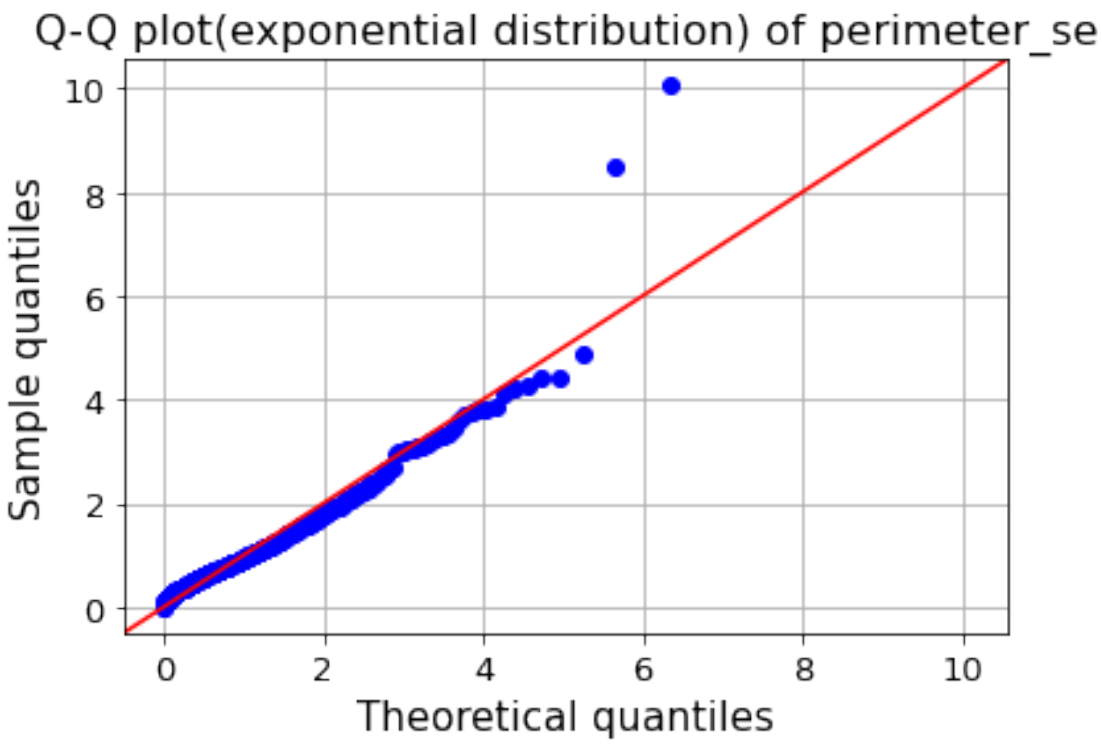

```
<Figure size 576x360 with 0 Axes>
```



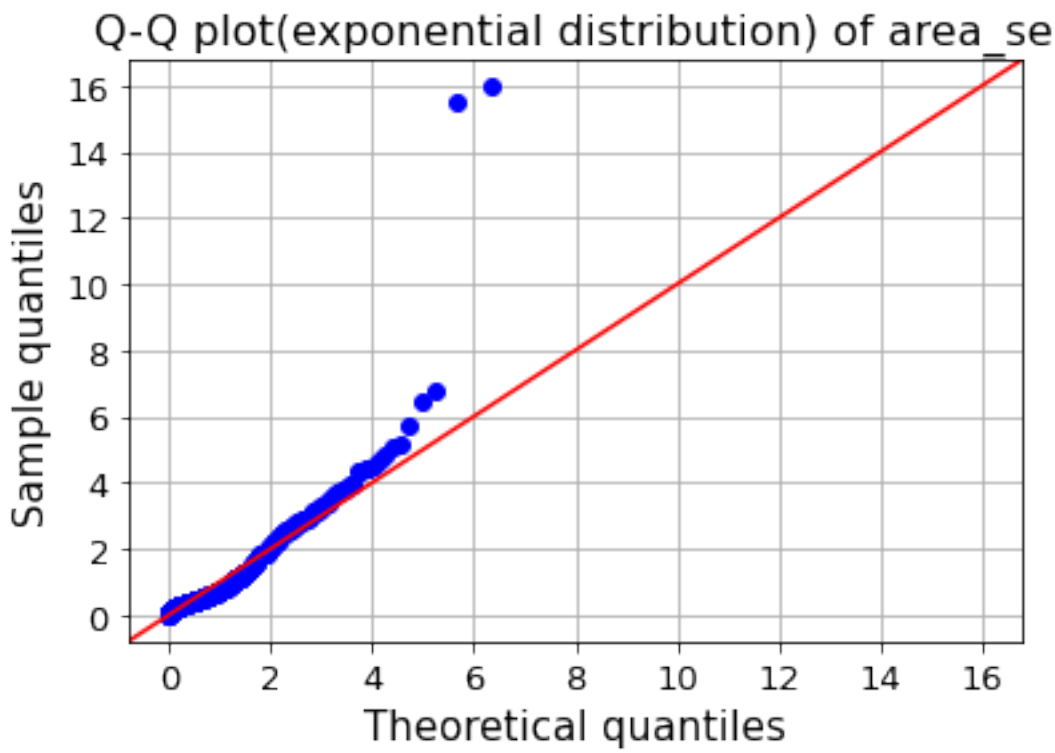

```
<Figure size 576x360 with 0 Axes>
```



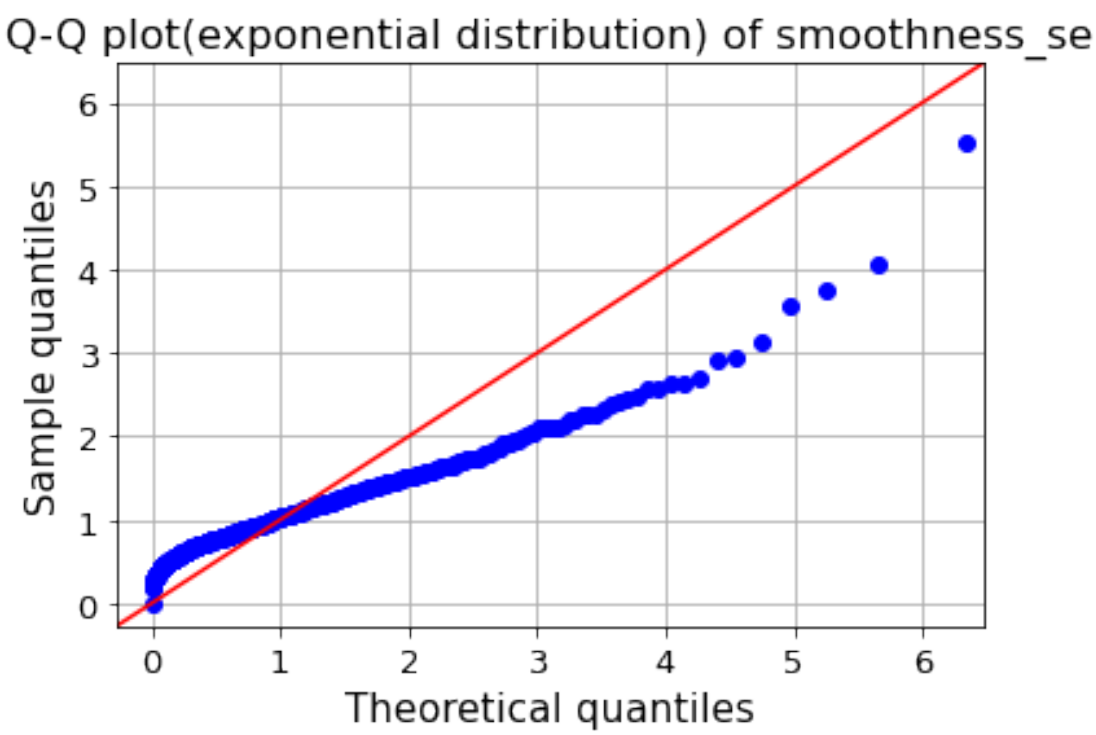
```
<Figure size 576x360 with 0 Axes>
```



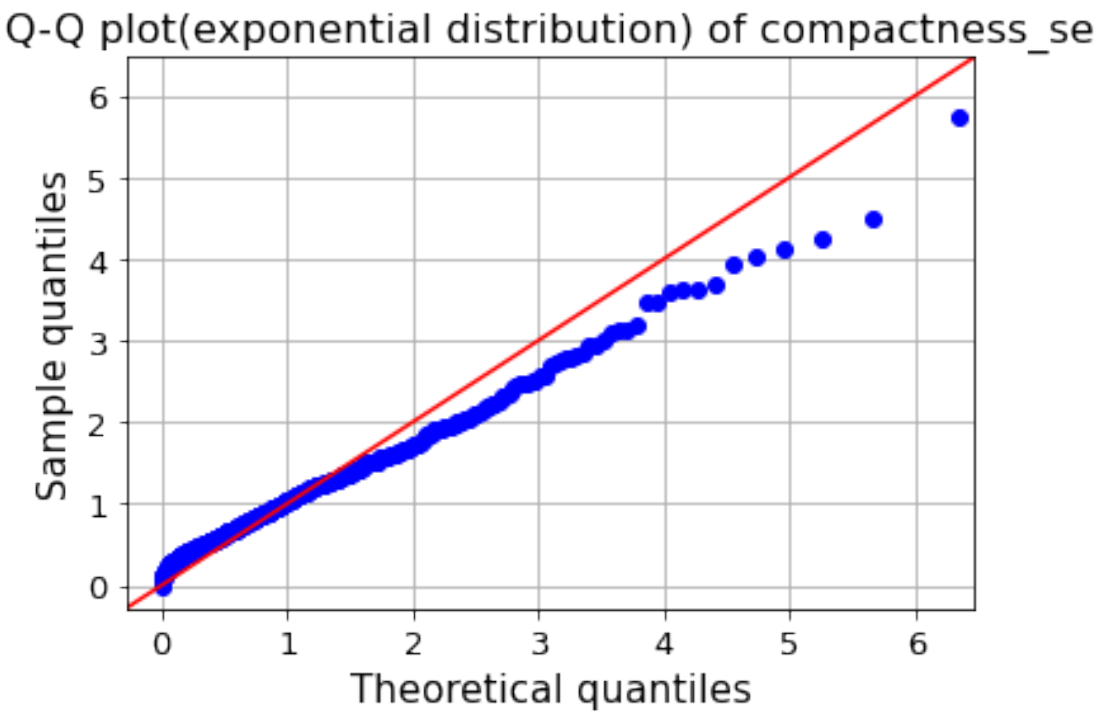

```
<Figure size 576x360 with 0 Axes>
```



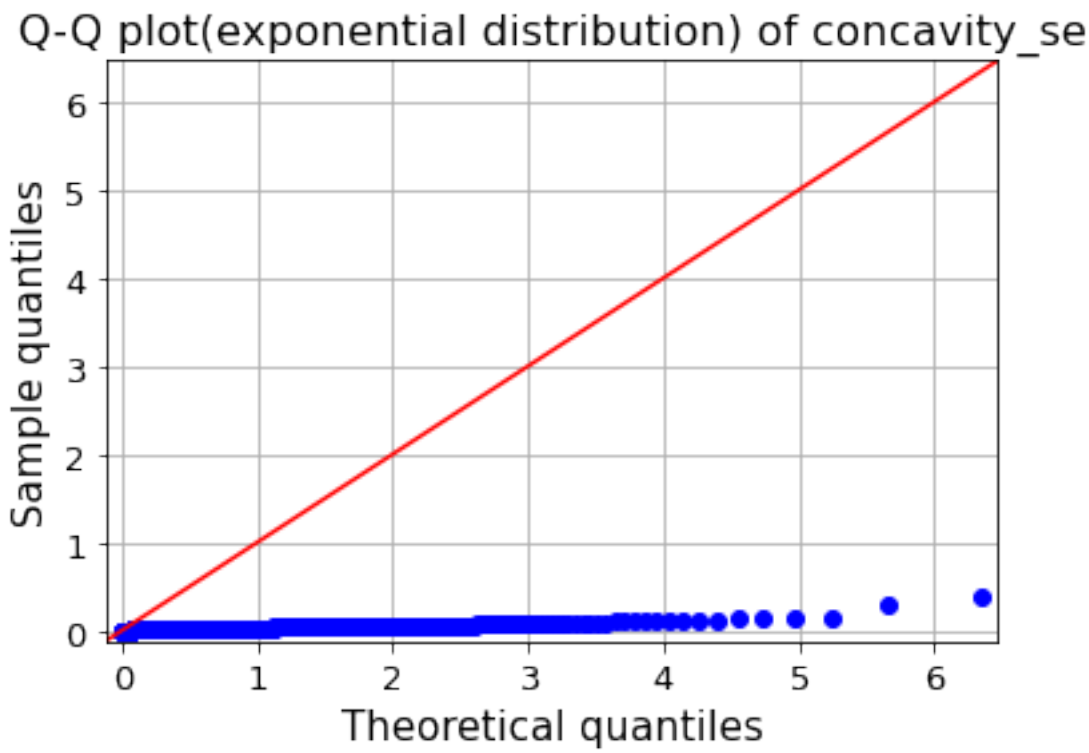

```
<Figure size 576x360 with 0 Axes>
```



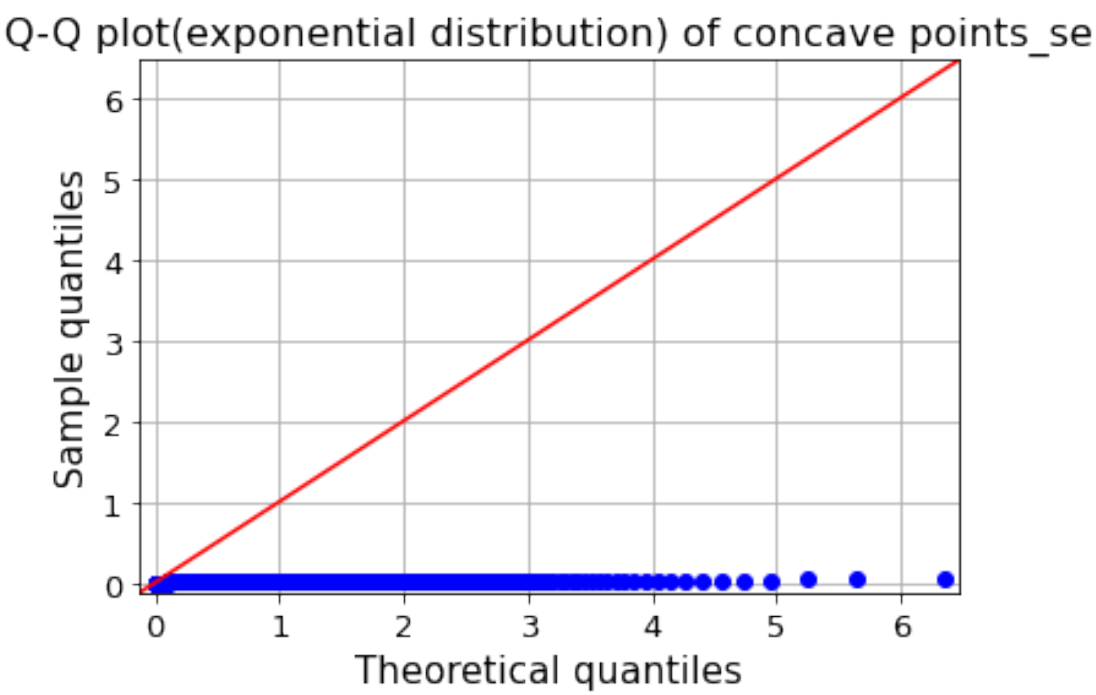
```
<Figure size 576x360 with 0 Axes>
```



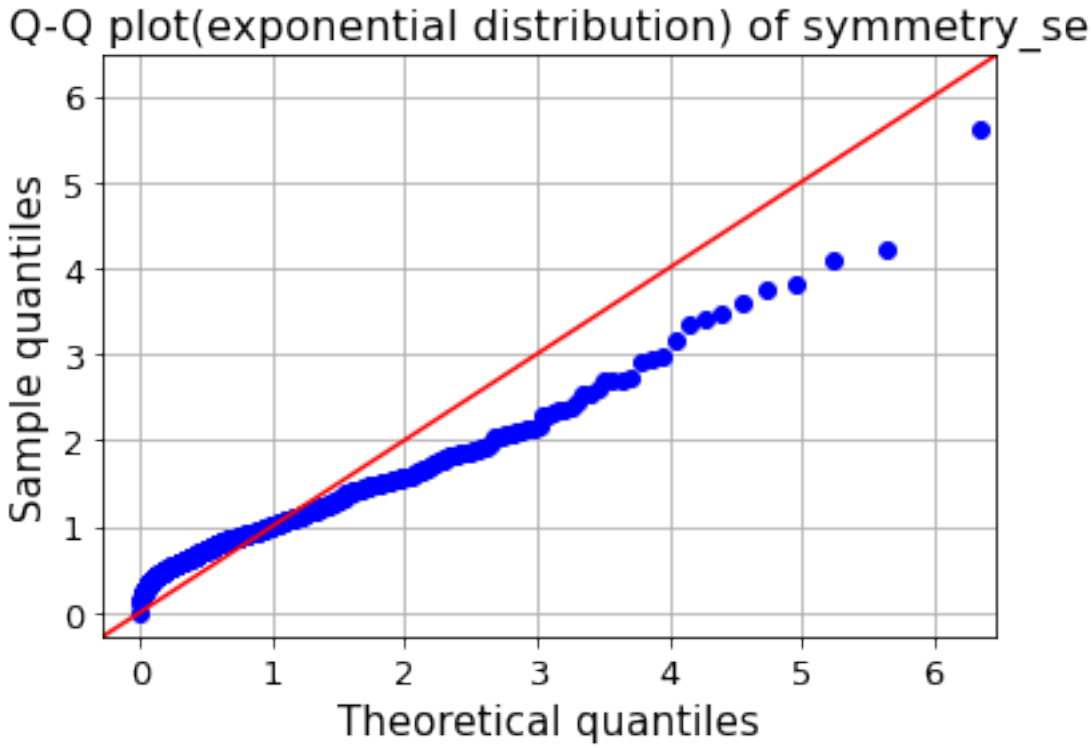

```
<Figure size 576x360 with 0 Axes>
```



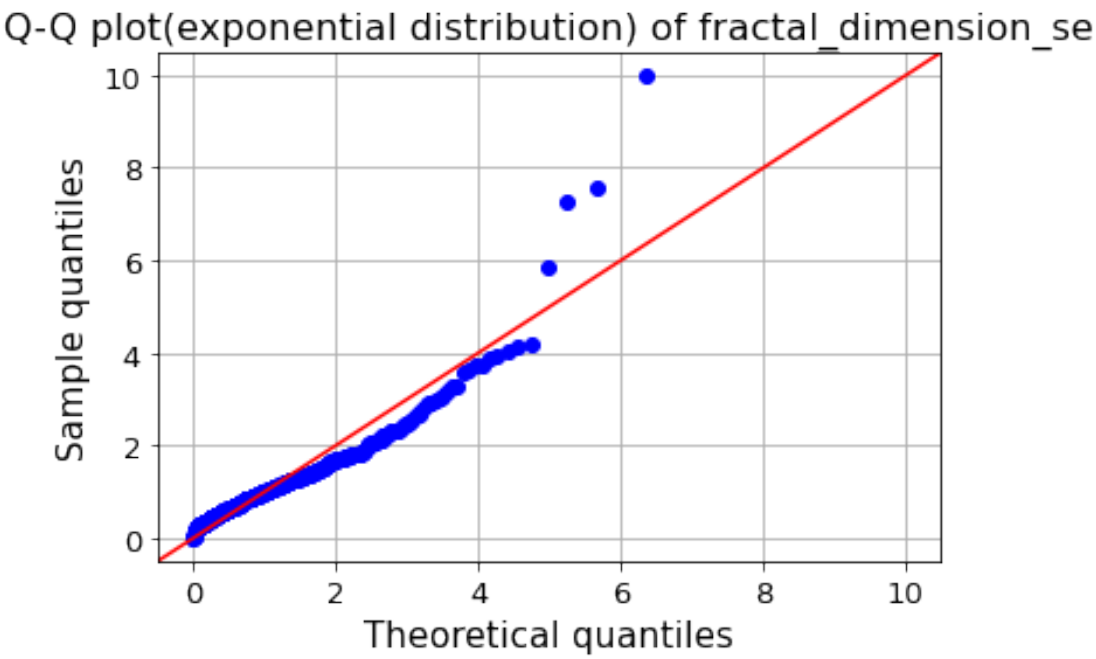

```
<Figure size 576x360 with 0 Axes>
```



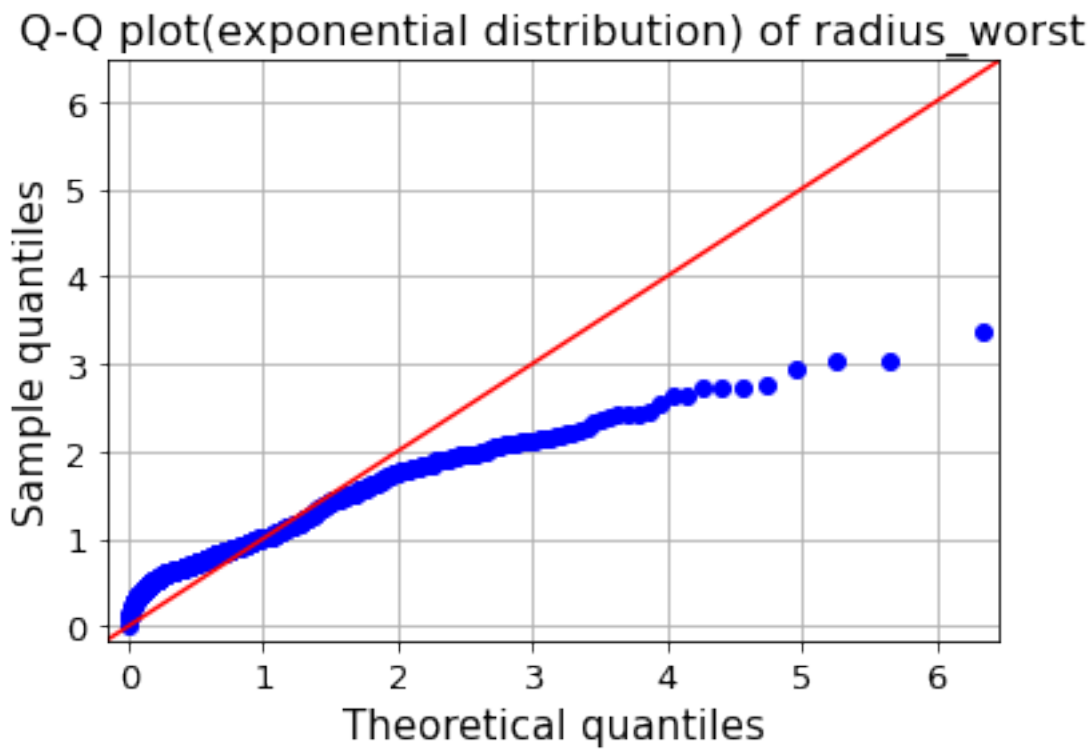
```
<Figure size 576x360 with 0 Axes>
```



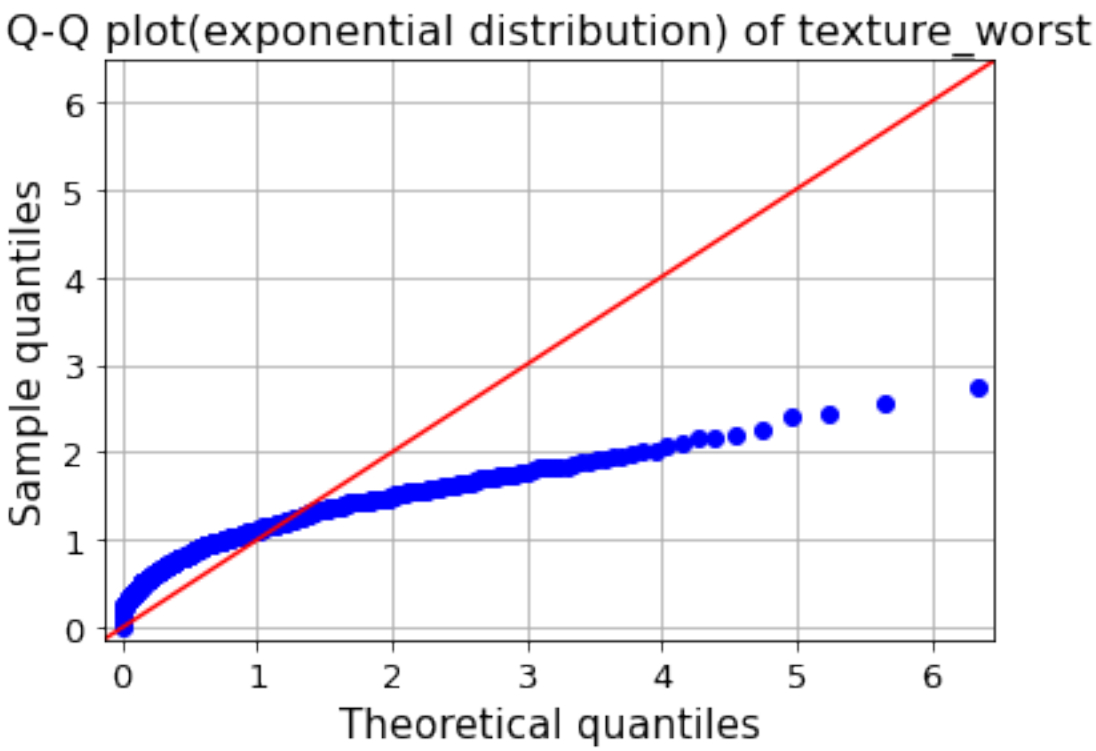

```
<Figure size 576x360 with 0 Axes>
```



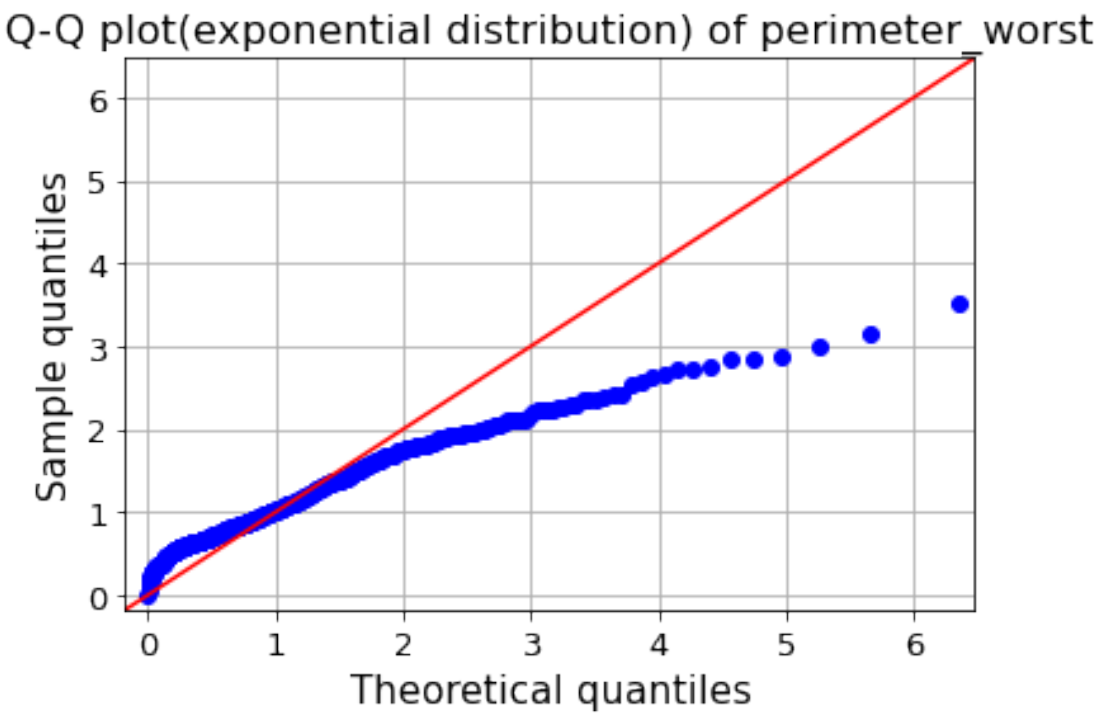

```
<Figure size 576x360 with 0 Axes>
```



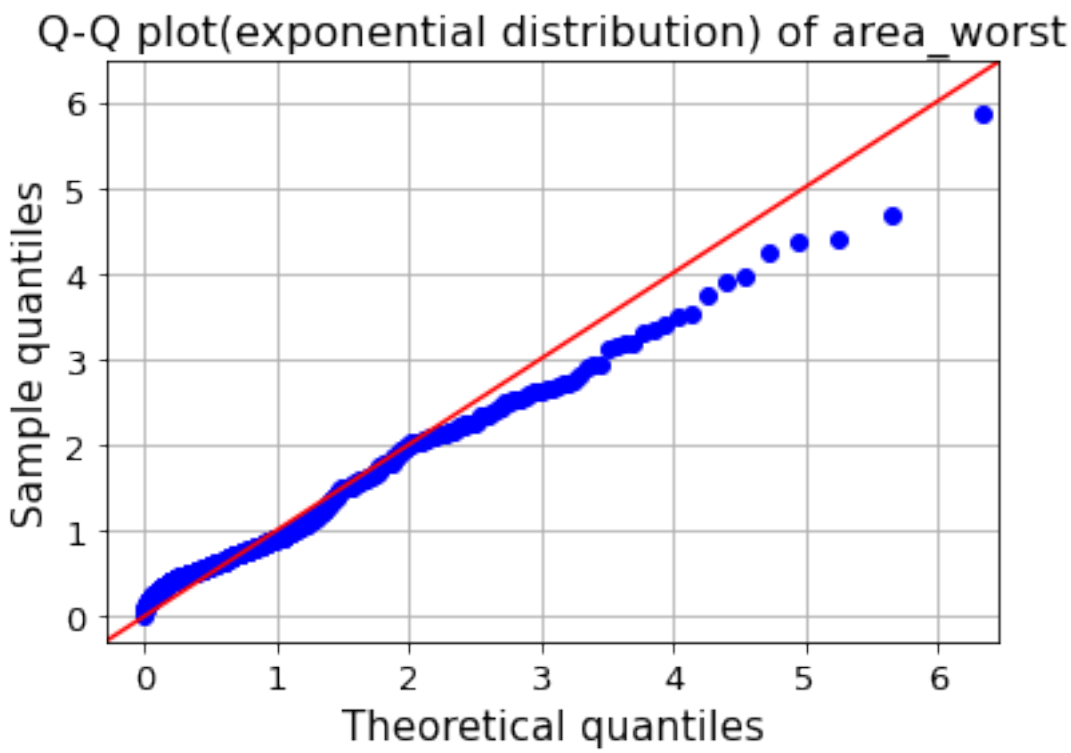
```
<Figure size 576x360 with 0 Axes>
```



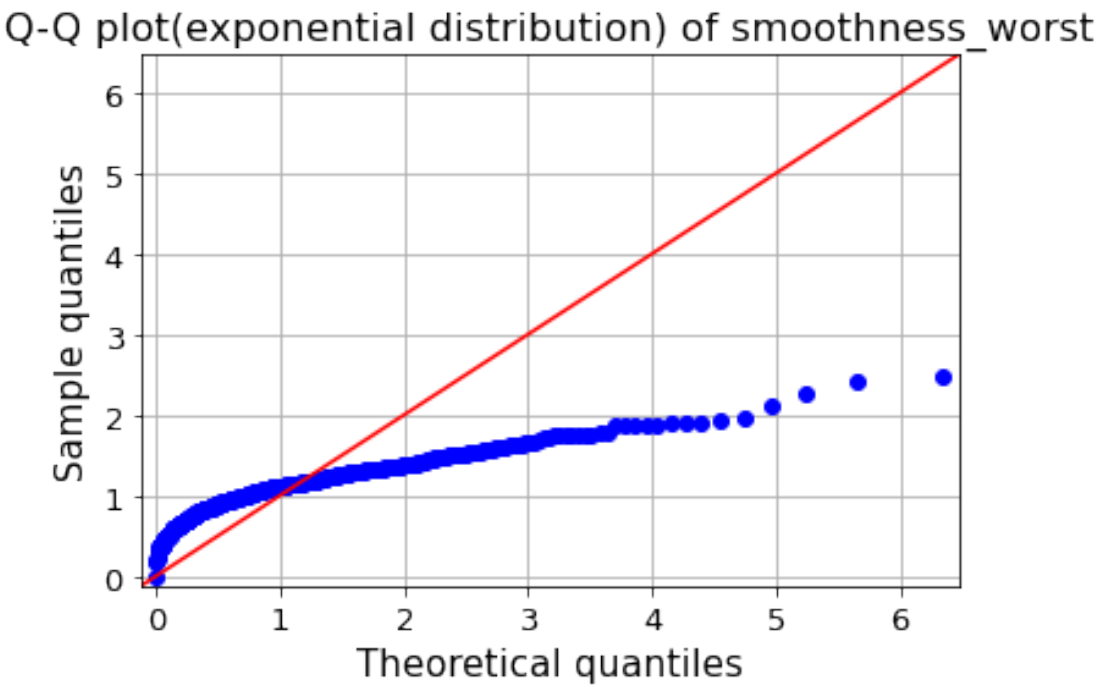

```
<Figure size 576x360 with 0 Axes>
```



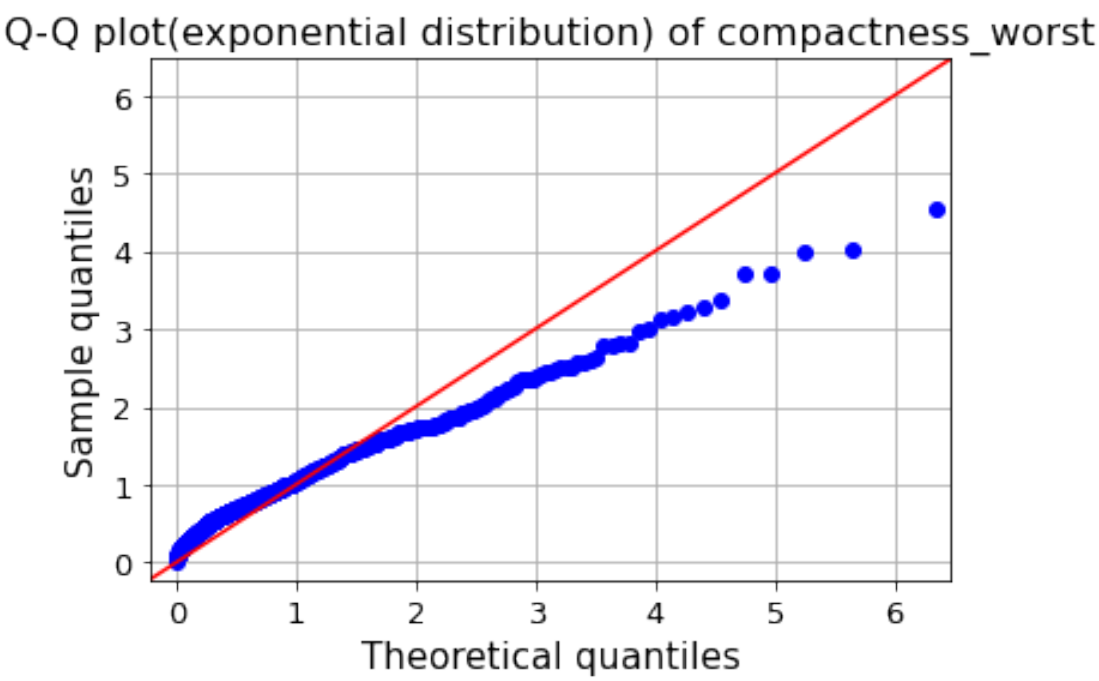

```
<Figure size 576x360 with 0 Axes>
```



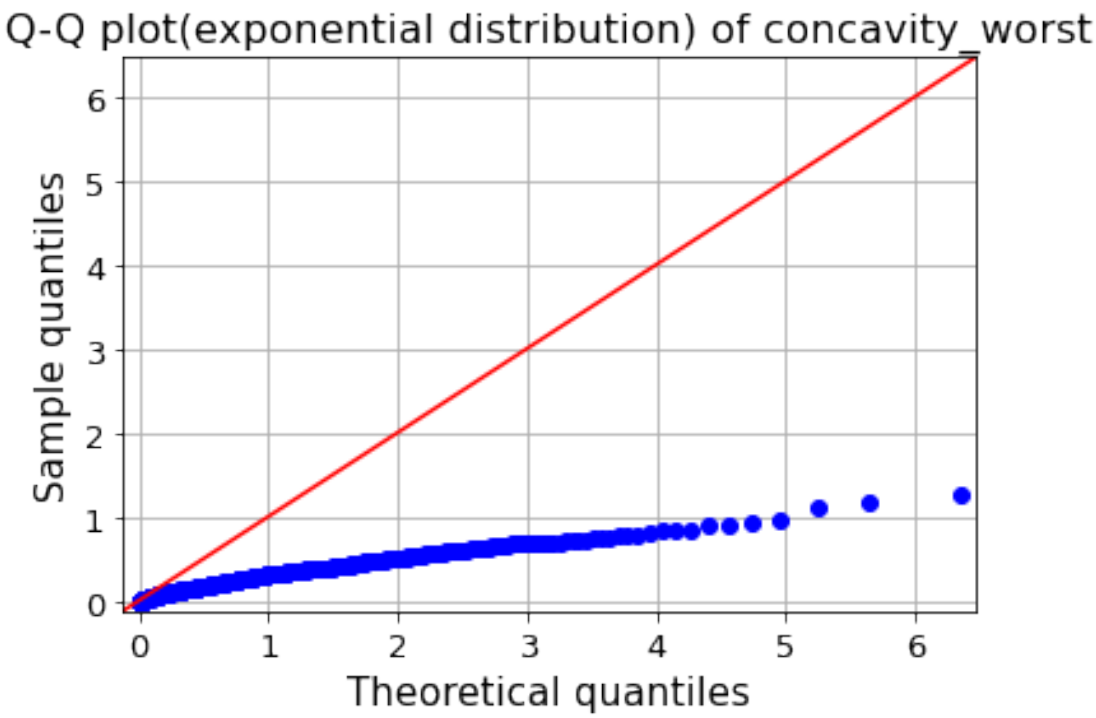
```
<Figure size 576x360 with 0 Axes>
```



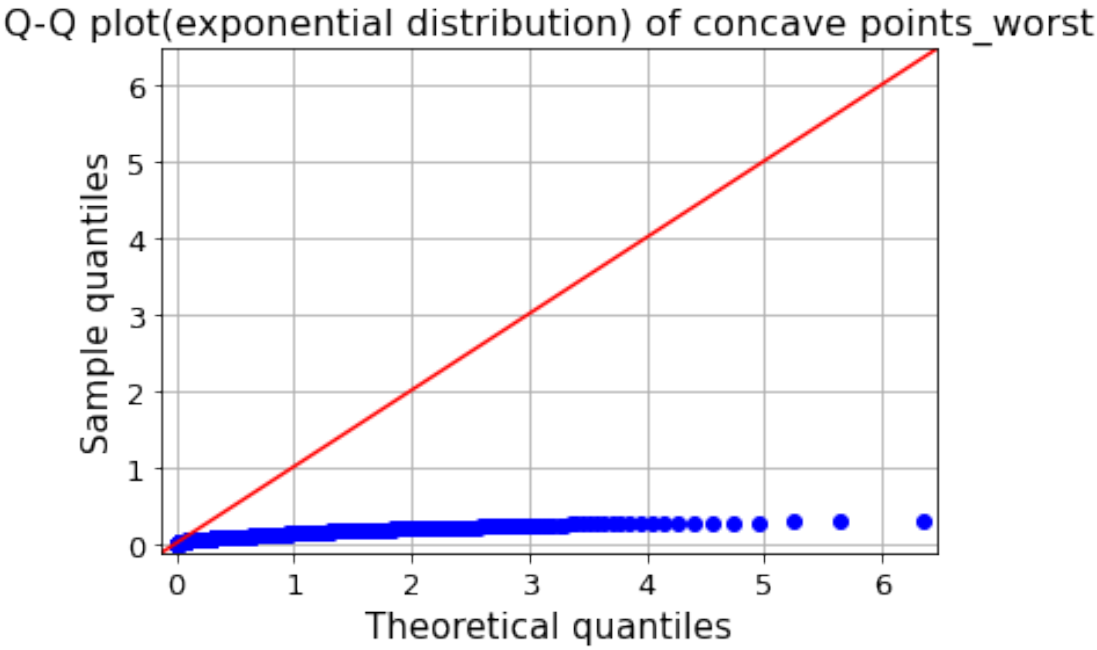

```
<Figure size 576x360 with 0 Axes>
```



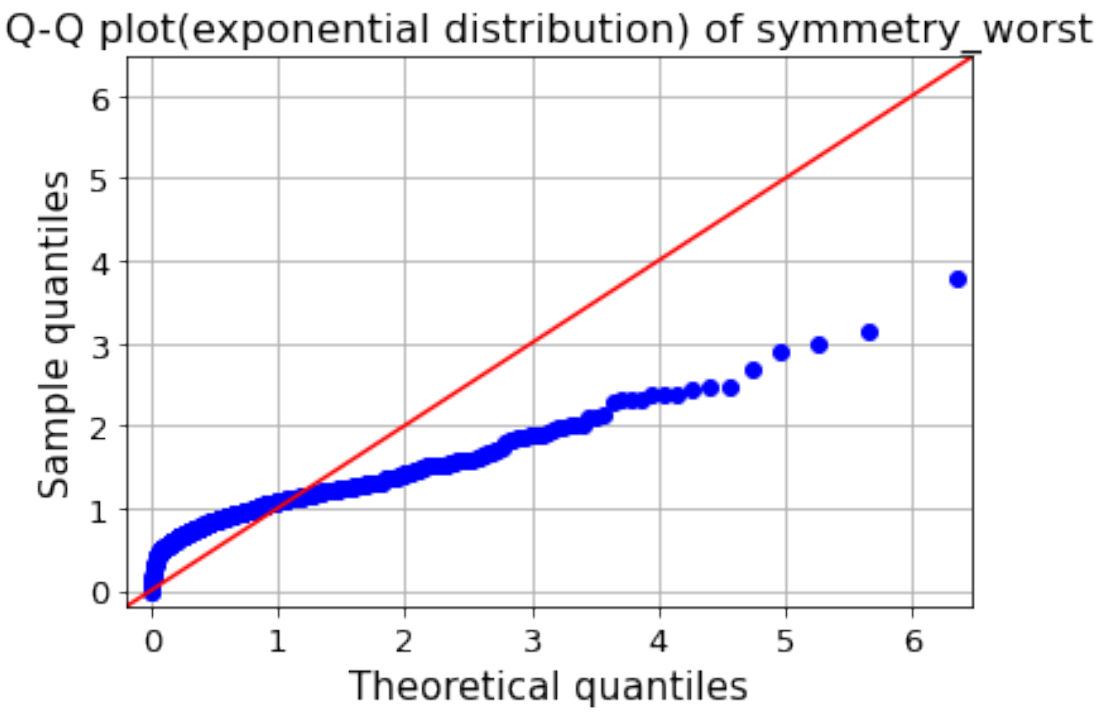
```
<Figure size 576x360 with 0 Axes>
```



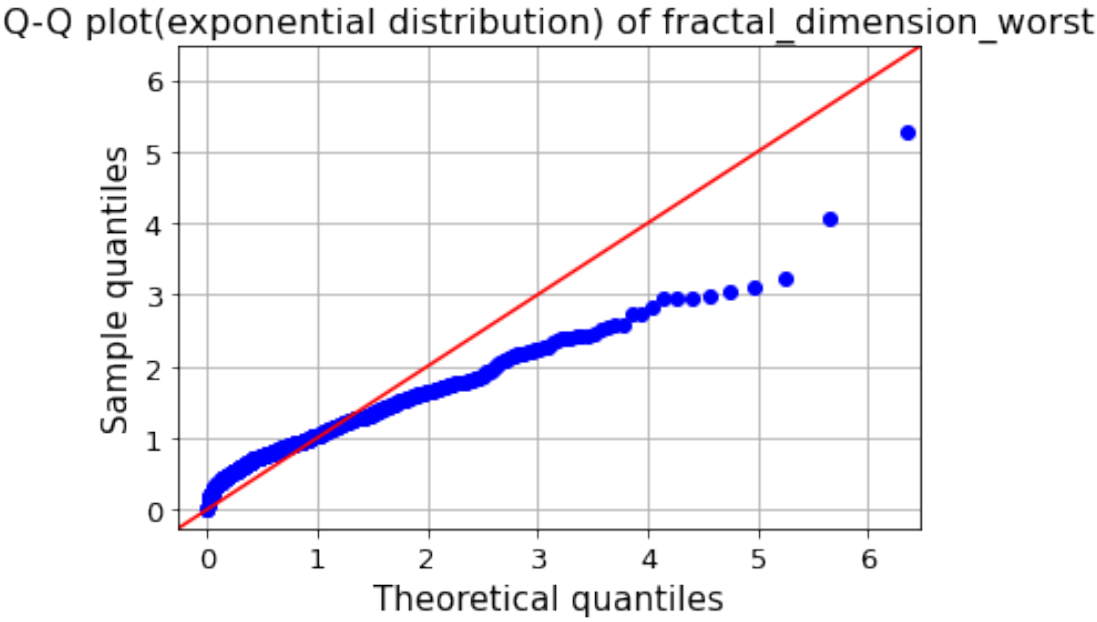

```
<Figure size 576x360 with 0 Axes>
```



```
<Figure size 576x360 with 0 Axes>
```

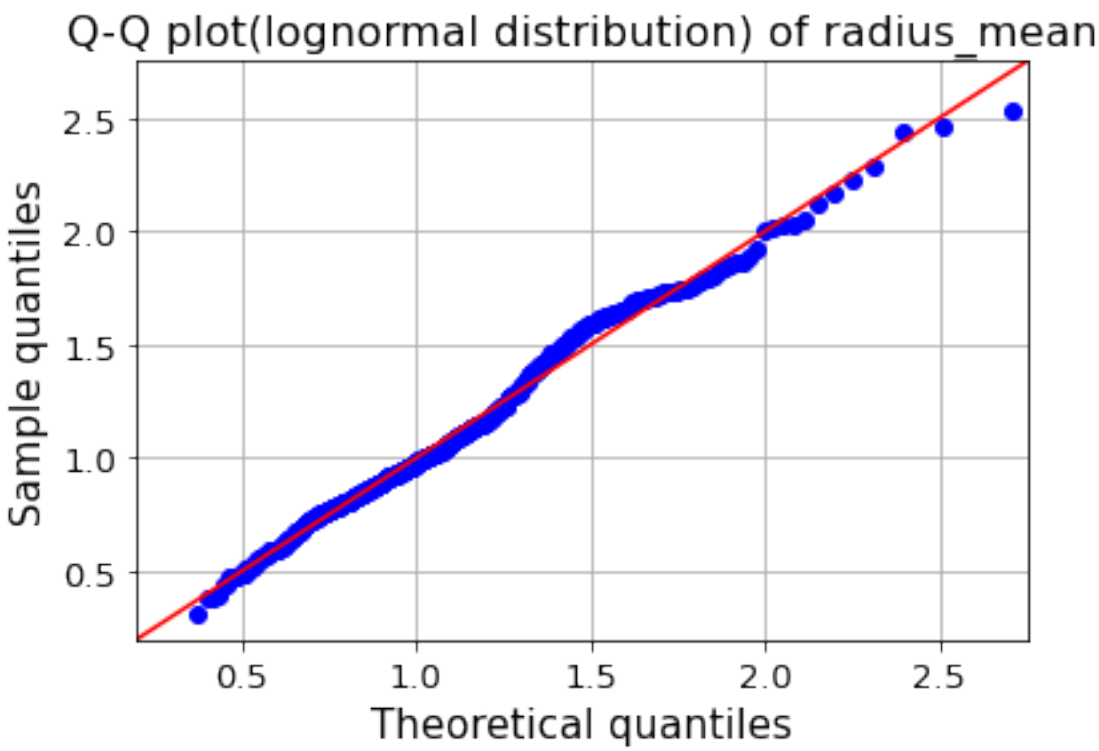



```
<Figure size 576x360 with 0 Axes>
```

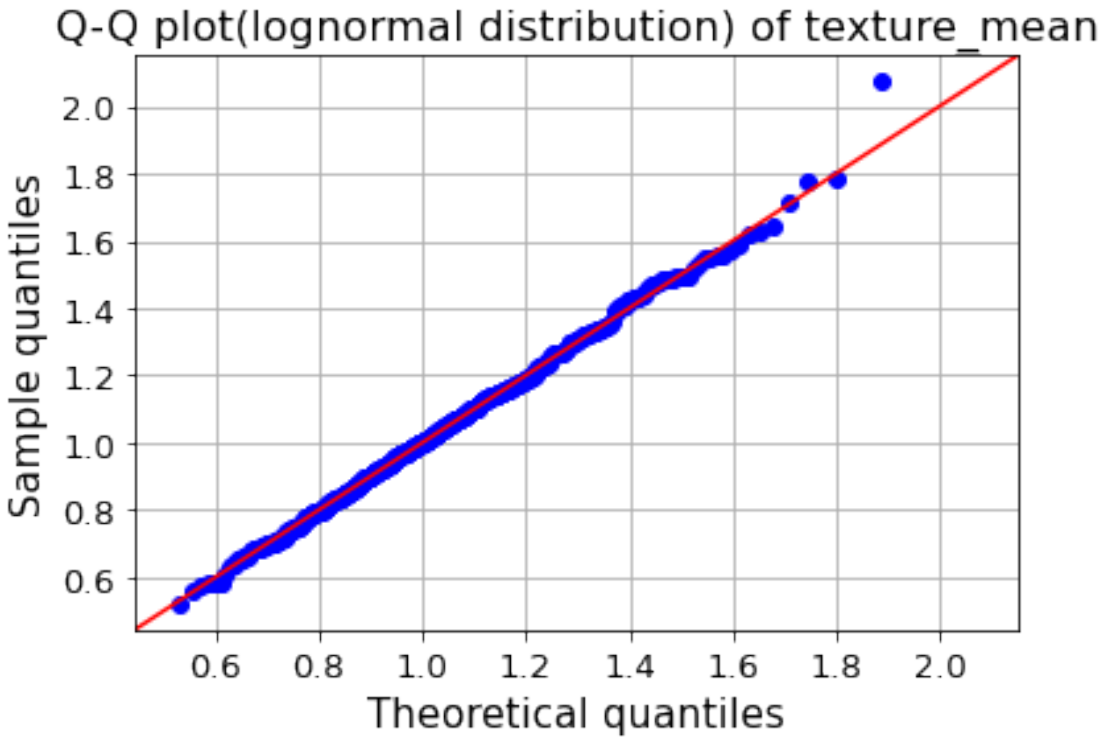


[15]:
```
for c in cols:
        plt.figure(figsize=(8,5))
        fig=qqplot(df[c],dist=sc.stats.lognorm,line="45",fit='True')
        plt.xticks(fontsize=13)
        plt.yticks(fontsize=13)
        plt.xlabel("Theoretical quantiles",fontsize=15)
        plt.ylabel("Sample quantiles",fontsize=15)
        plt.title("Q-Q plot(lognormal distribution) of {}".
 ↪format(c),fontsize=16)
        plt.grid(True)
        plt.show()
```

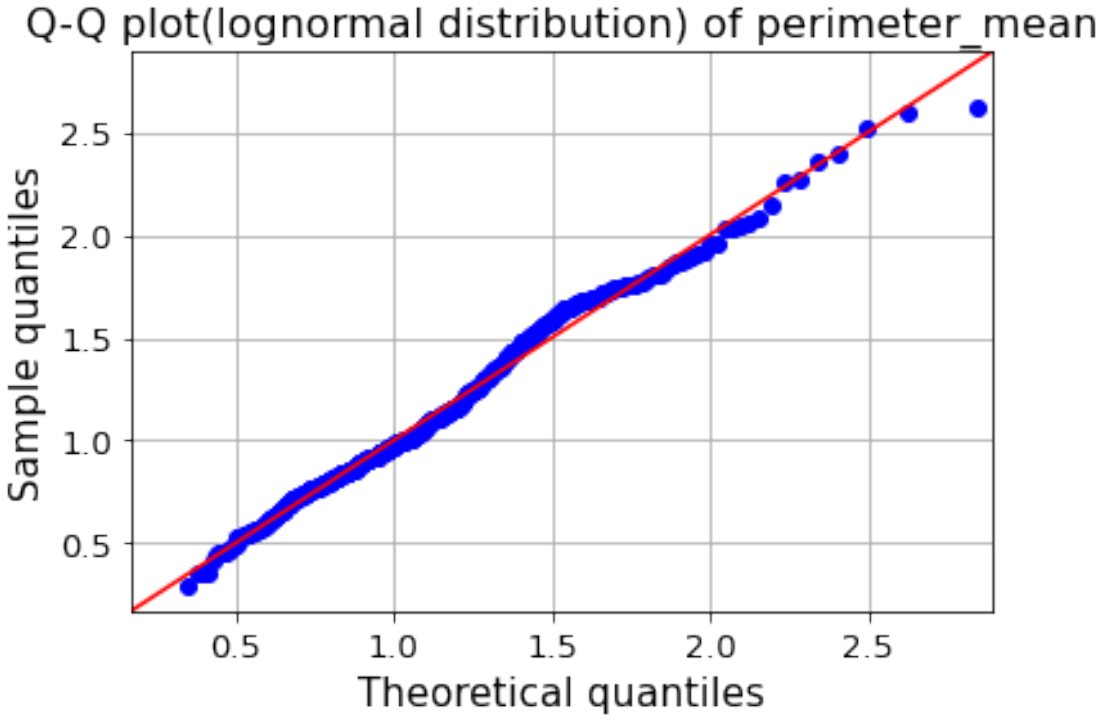
```
<Figure size 576x360 with 0 Axes>
```



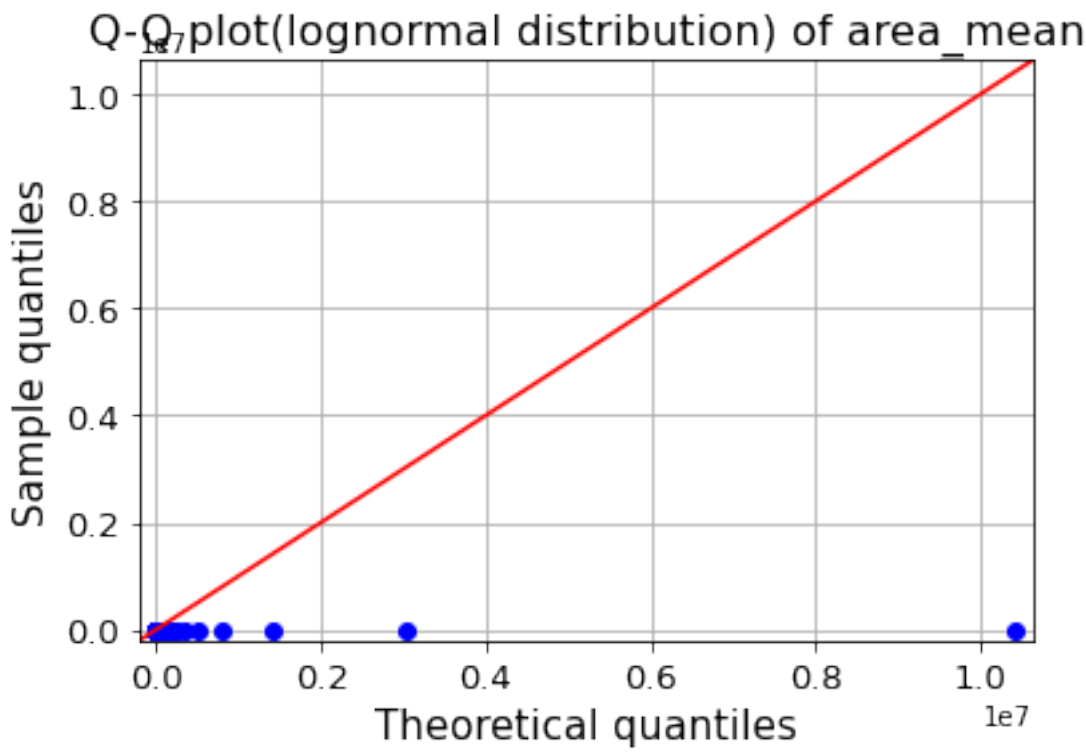

```
<Figure size 576x360 with 0 Axes>
```



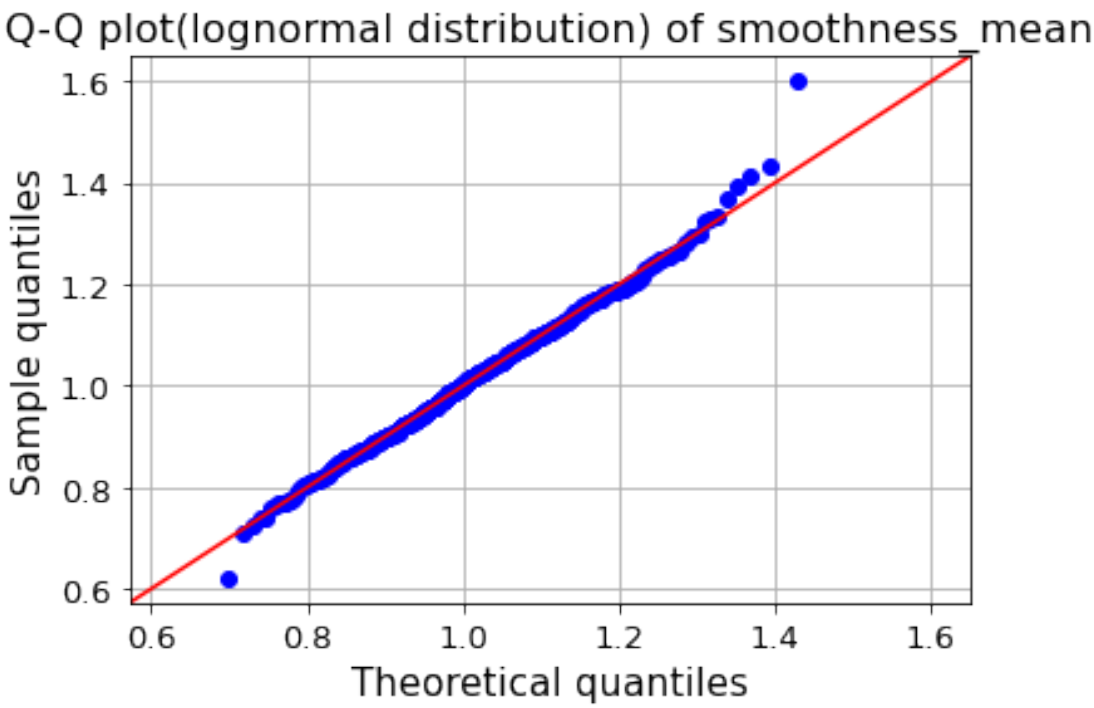

```
<Figure size 576x360 with 0 Axes>
```



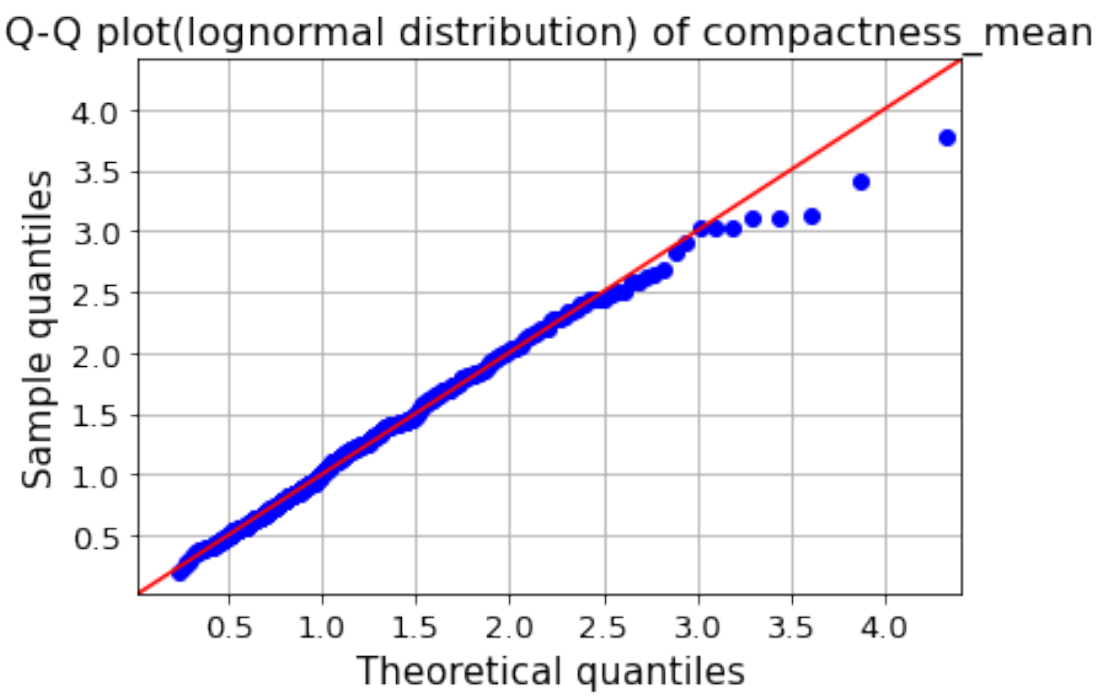
```
<Figure size 576x360 with 0 Axes>
```



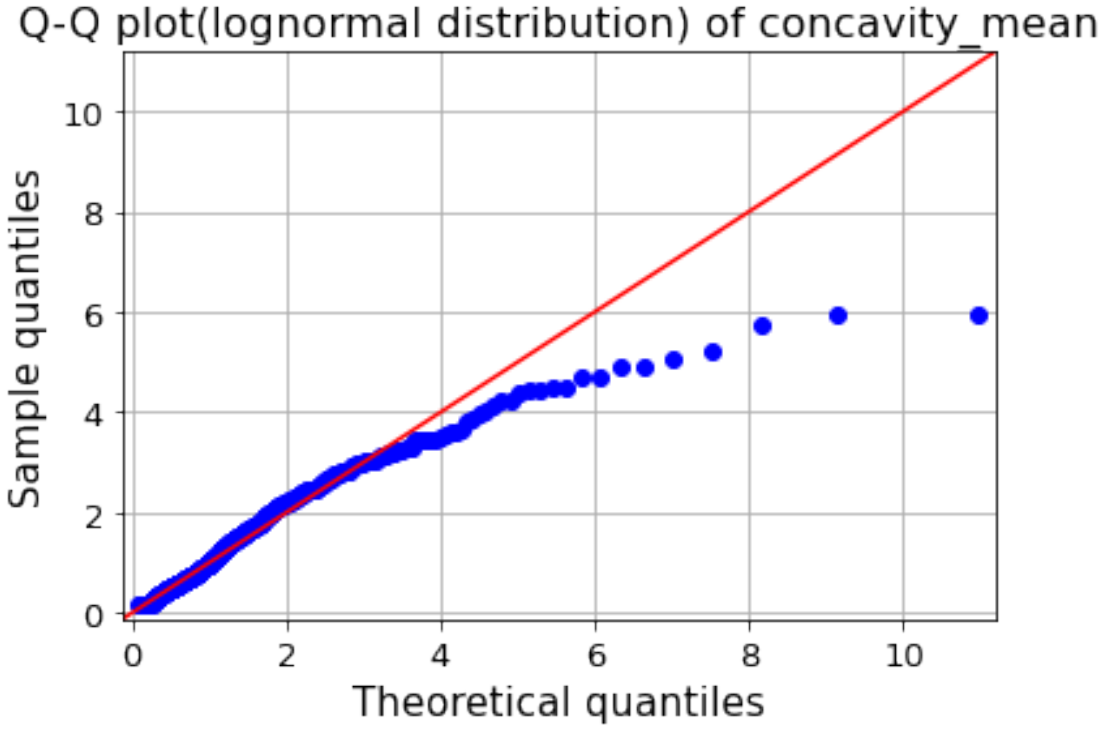

```
<Figure size 576x360 with 0 Axes>
```



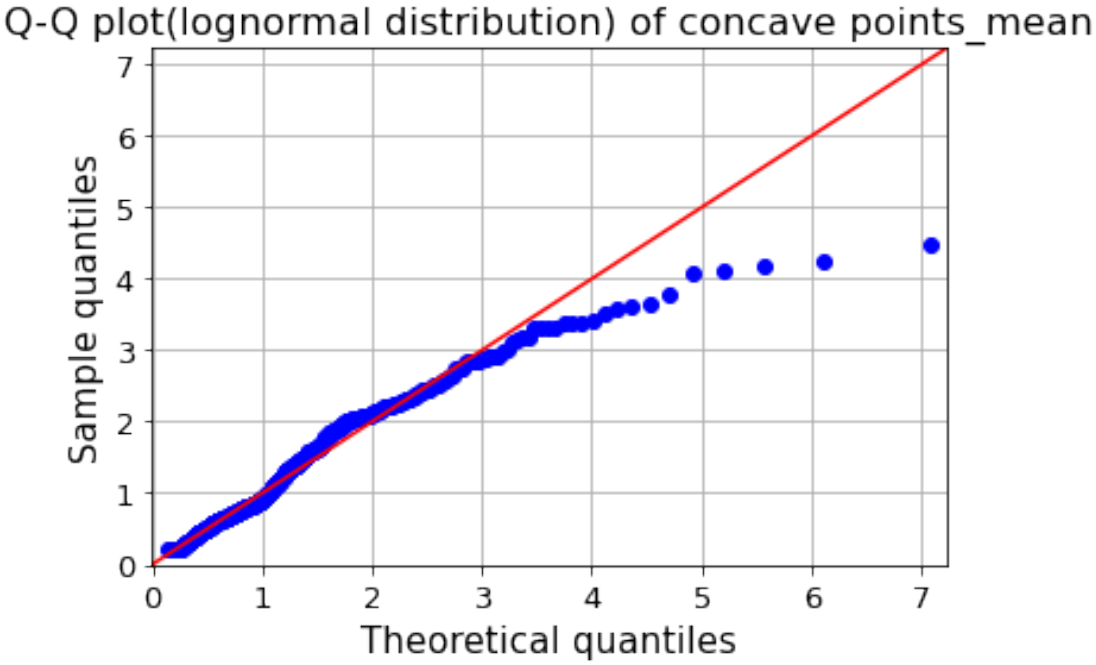

```
<Figure size 576x360 with 0 Axes>
```



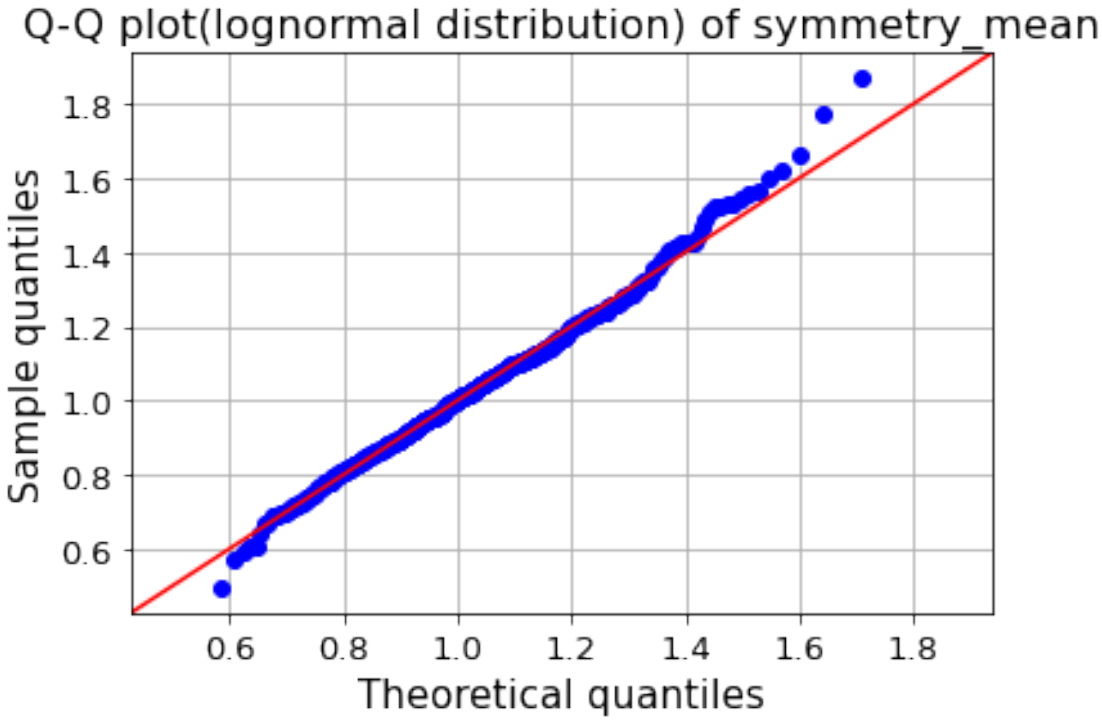
```
<Figure size 576x360 with 0 Axes>
```



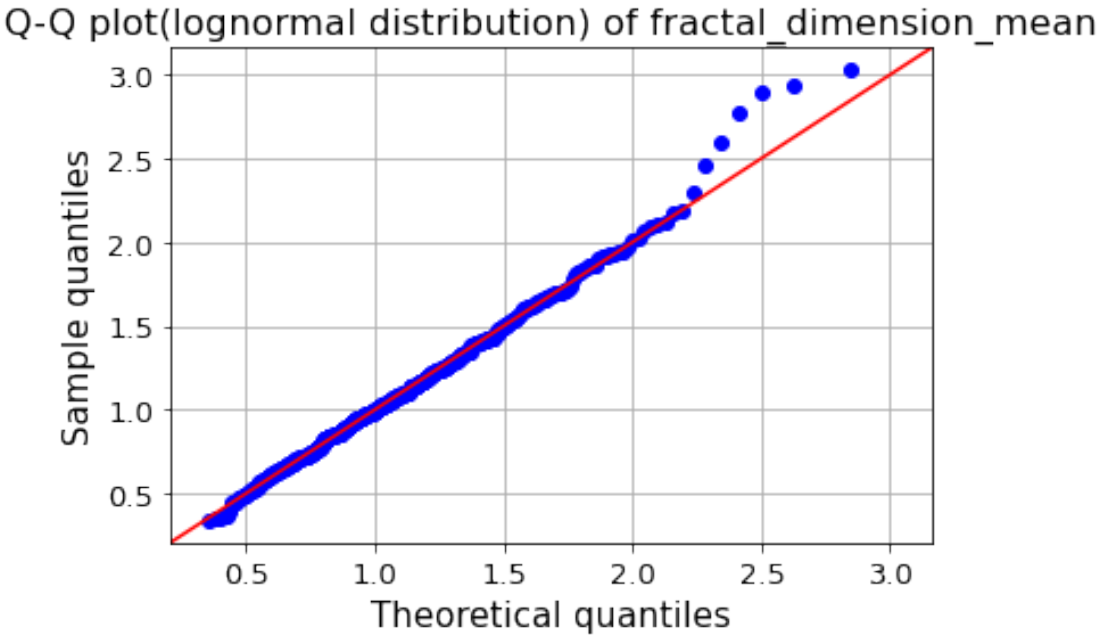

```
<Figure size 576x360 with 0 Axes>
```



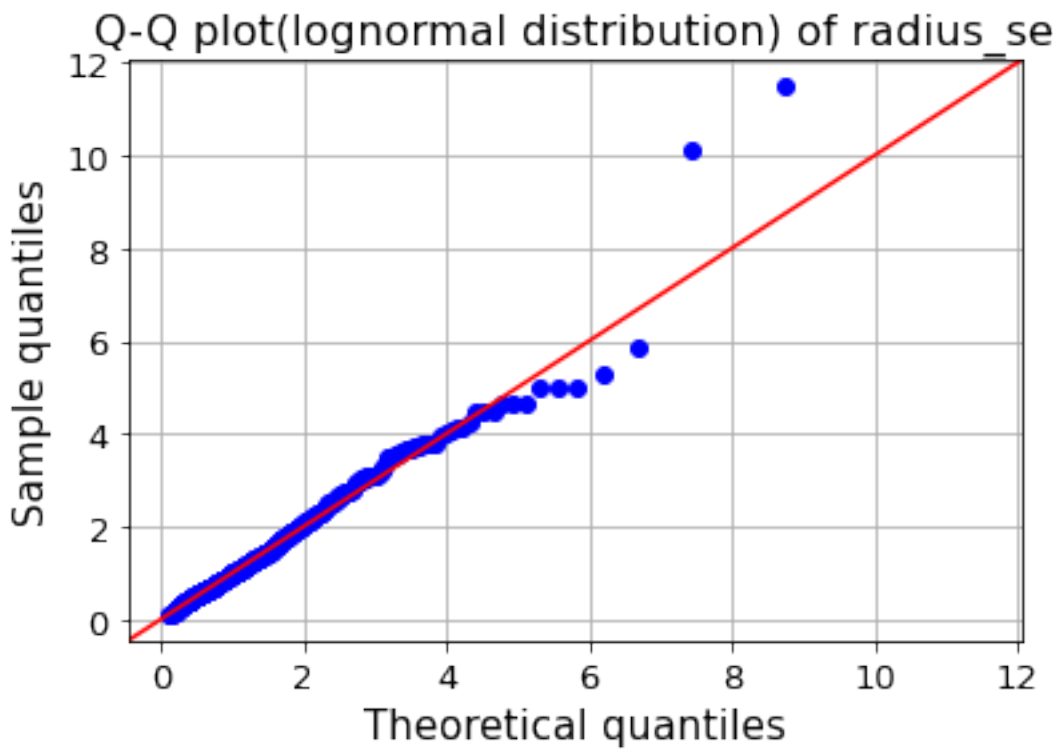
```
<Figure size 576x360 with 0 Axes>
```



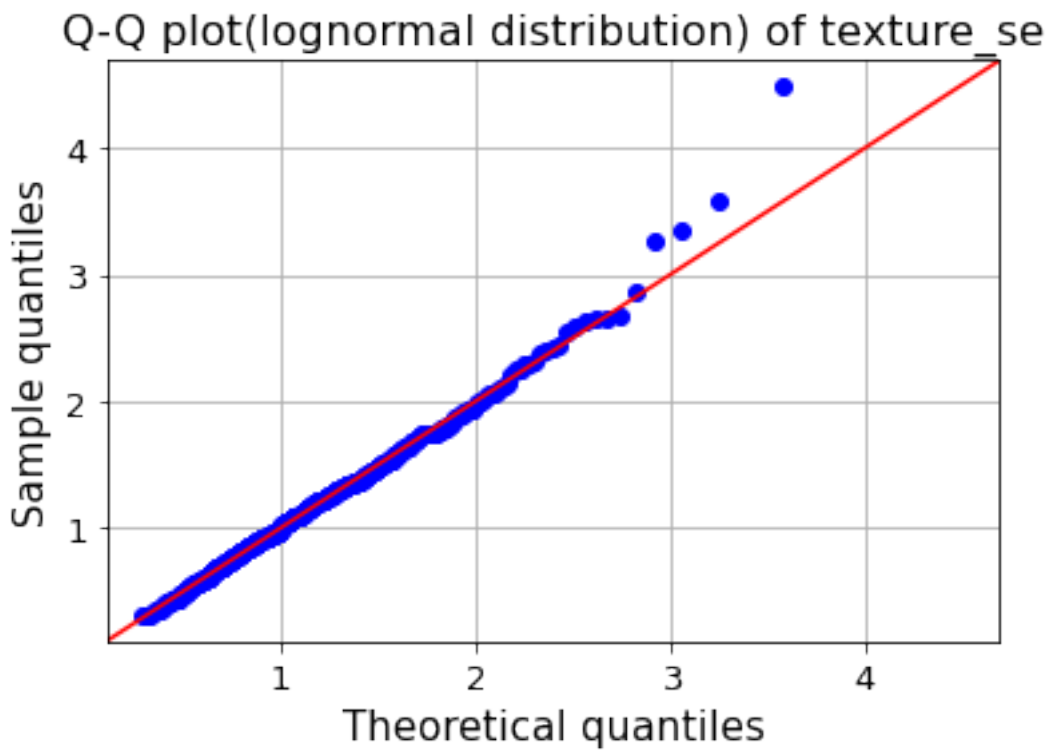

```
<Figure size 576x360 with 0 Axes>
```



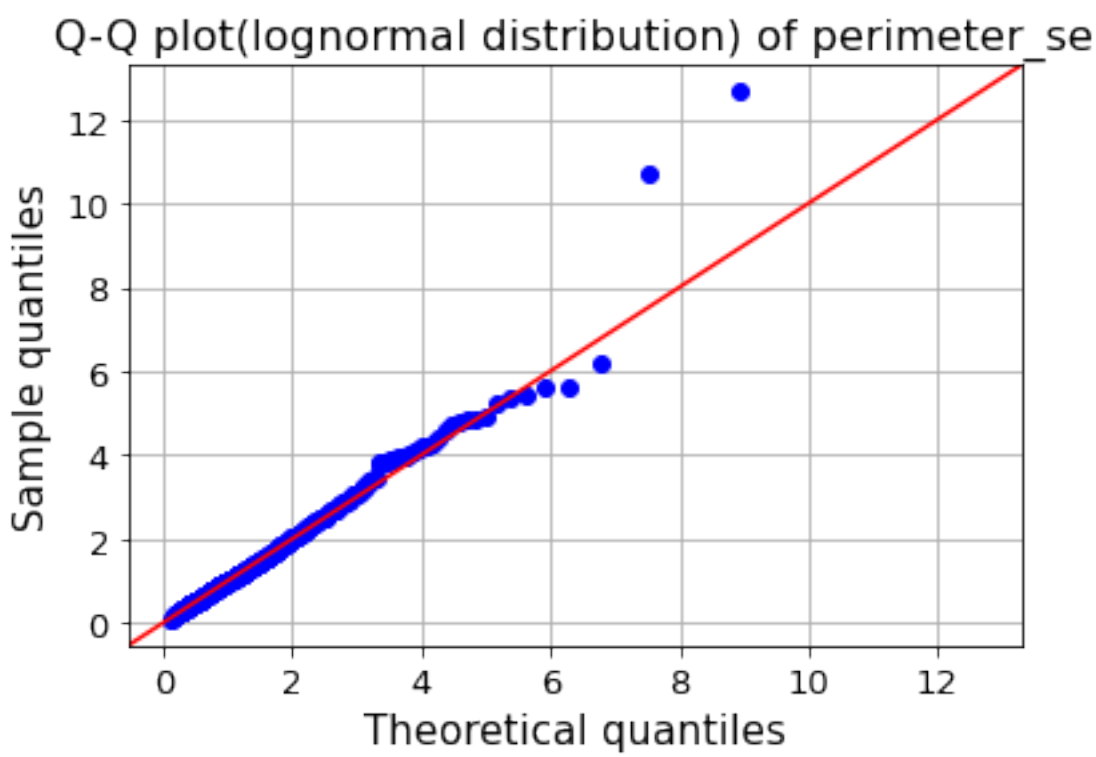
```
<Figure size 576x360 with 0 Axes>
```



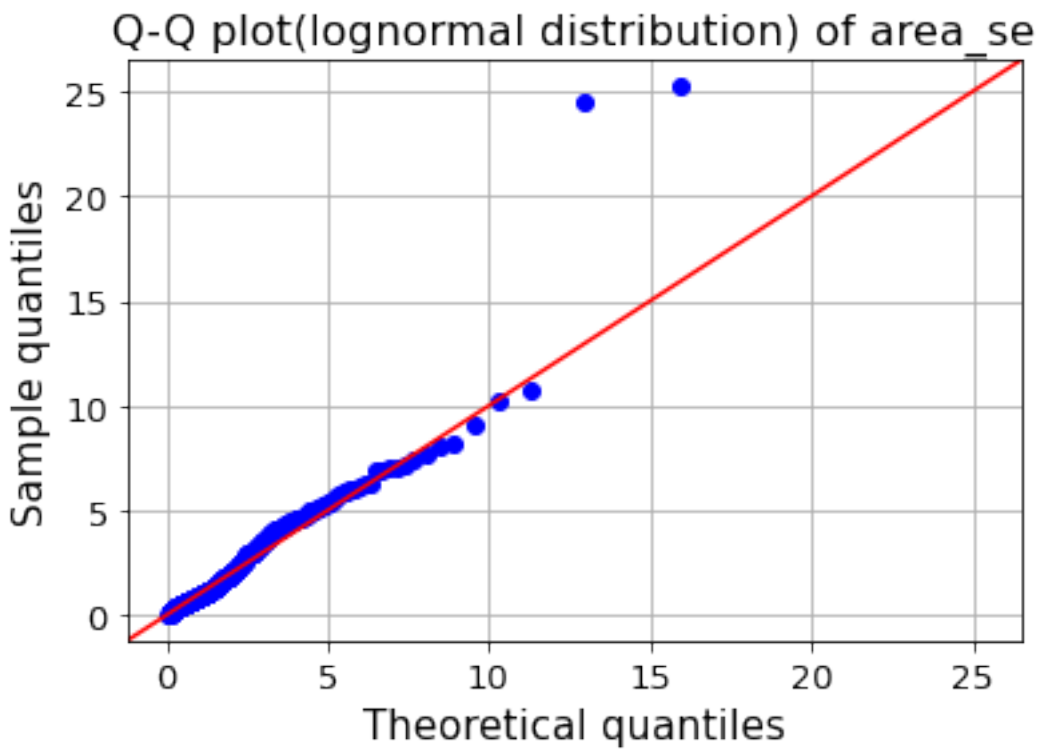

```
<Figure size 576x360 with 0 Axes>
```



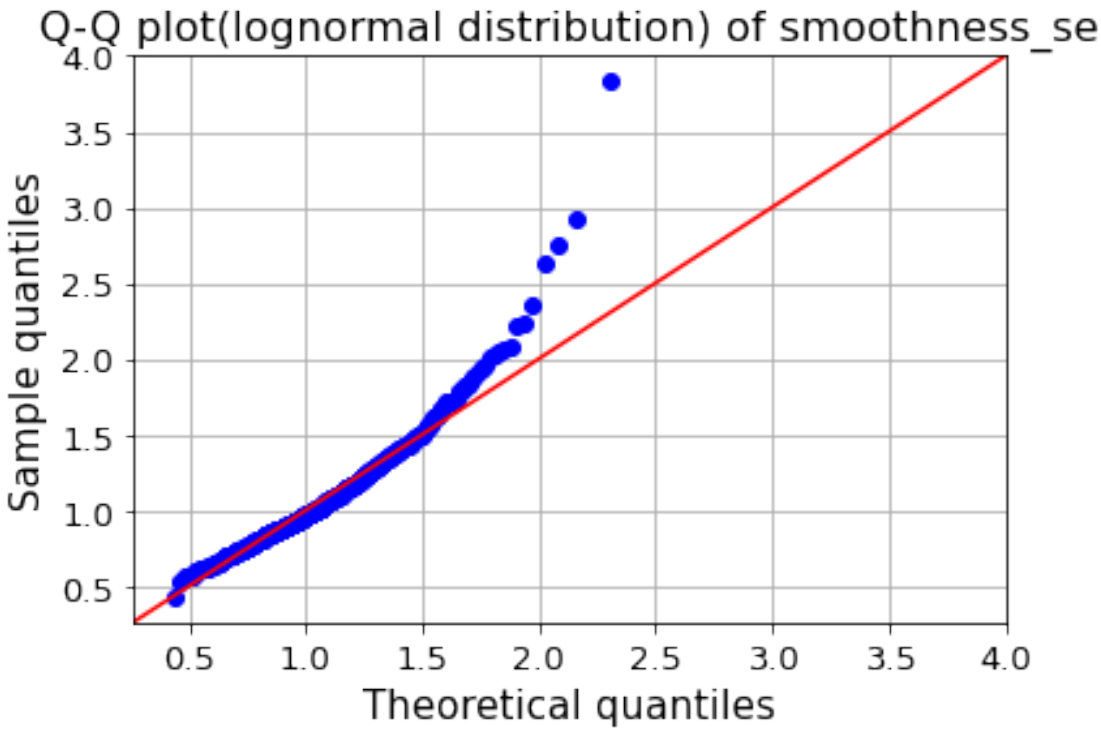

```
<Figure size 576x360 with 0 Axes>
```



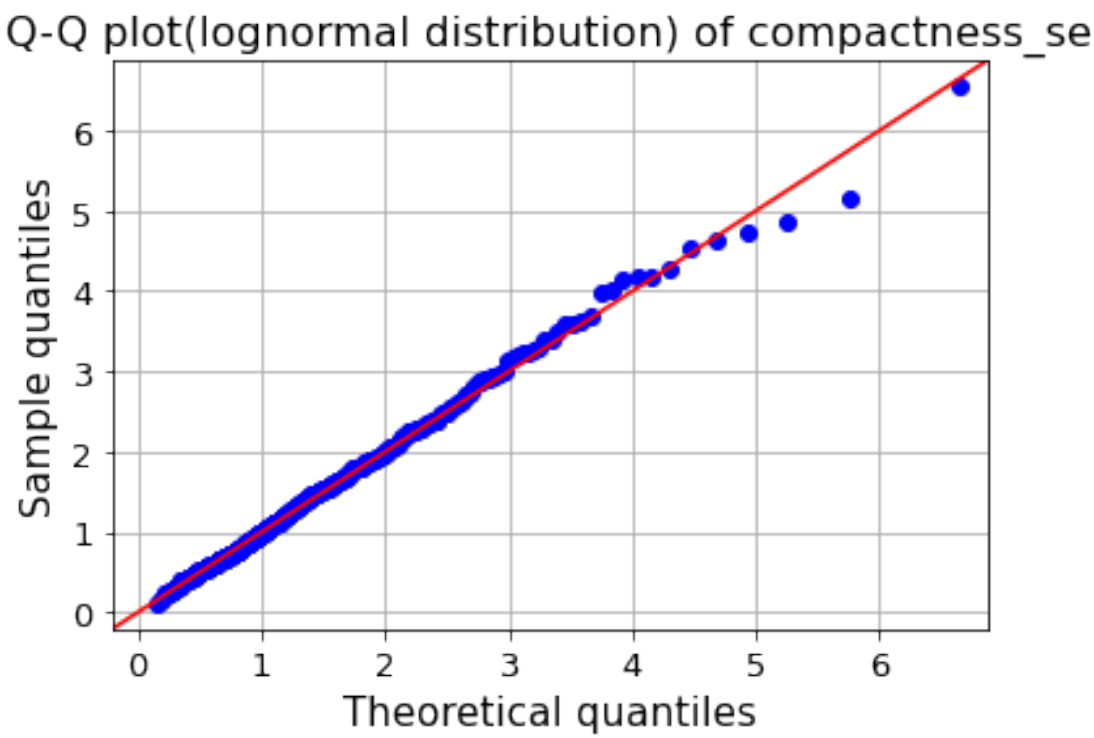
```
<Figure size 576x360 with 0 Axes>
```



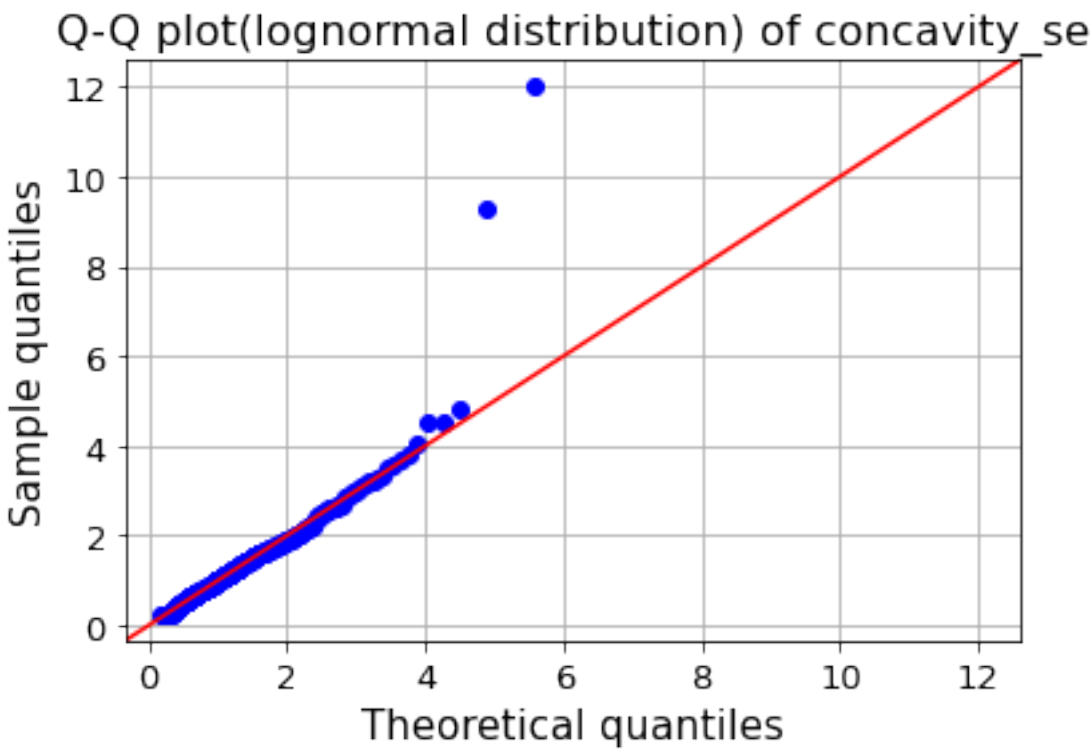

```
<Figure size 576x360 with 0 Axes>
```



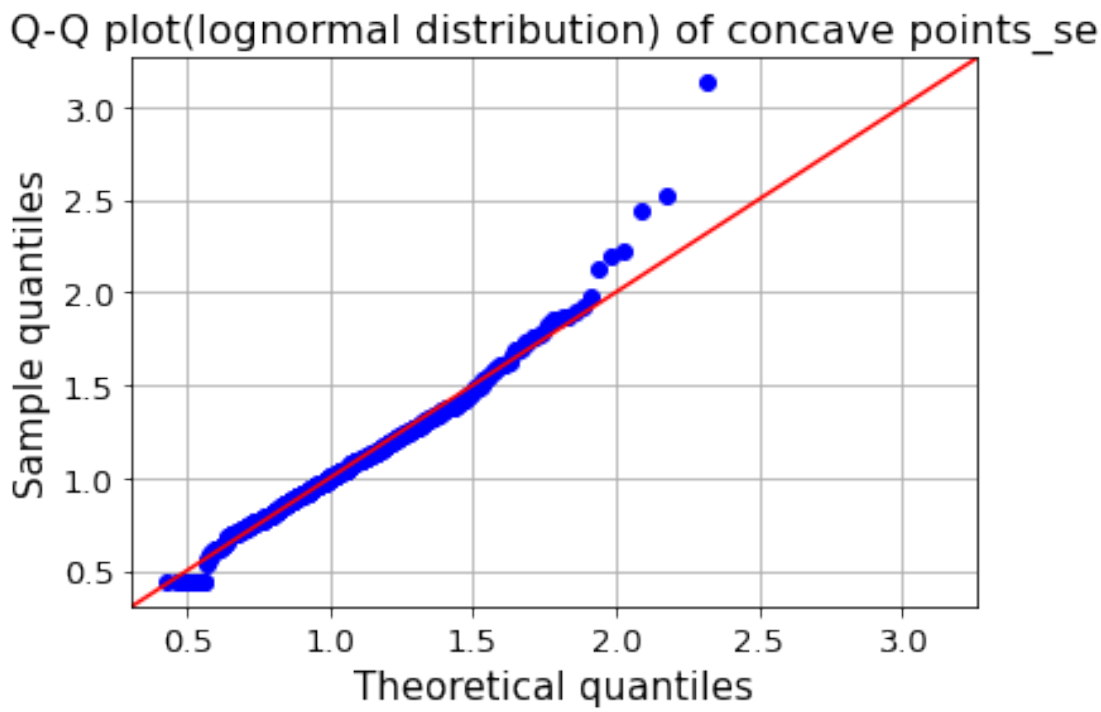

```
<Figure size 576x360 with 0 Axes>
```



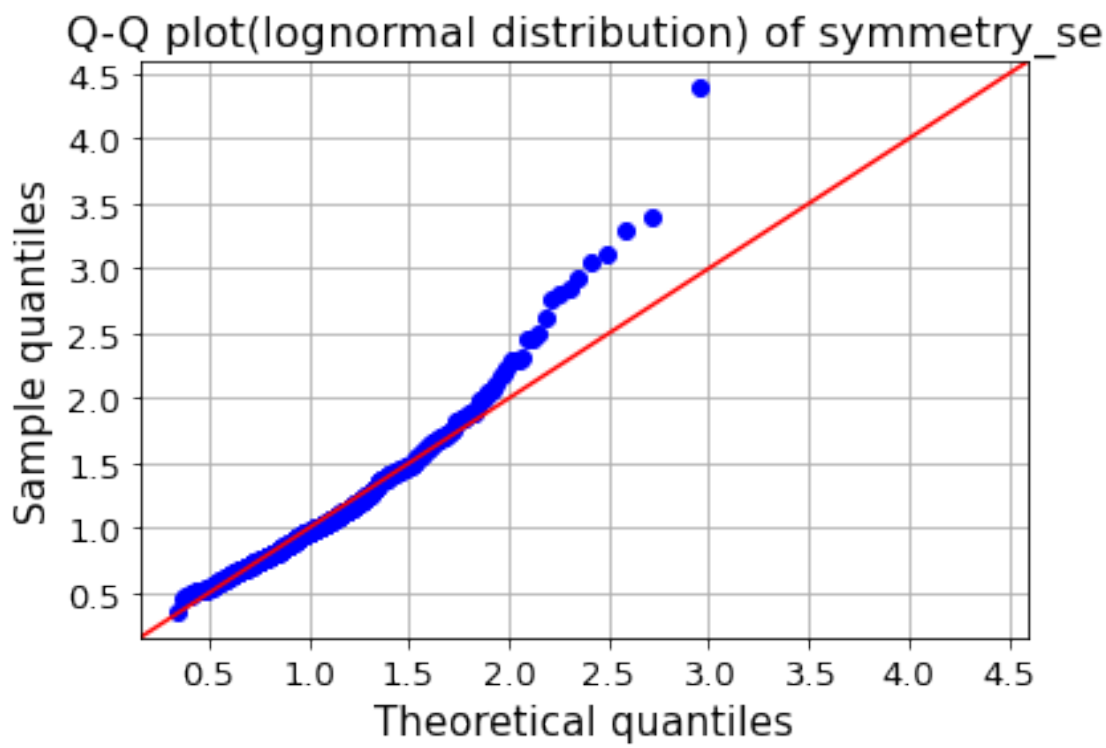
```
<Figure size 576x360 with 0 Axes>
```



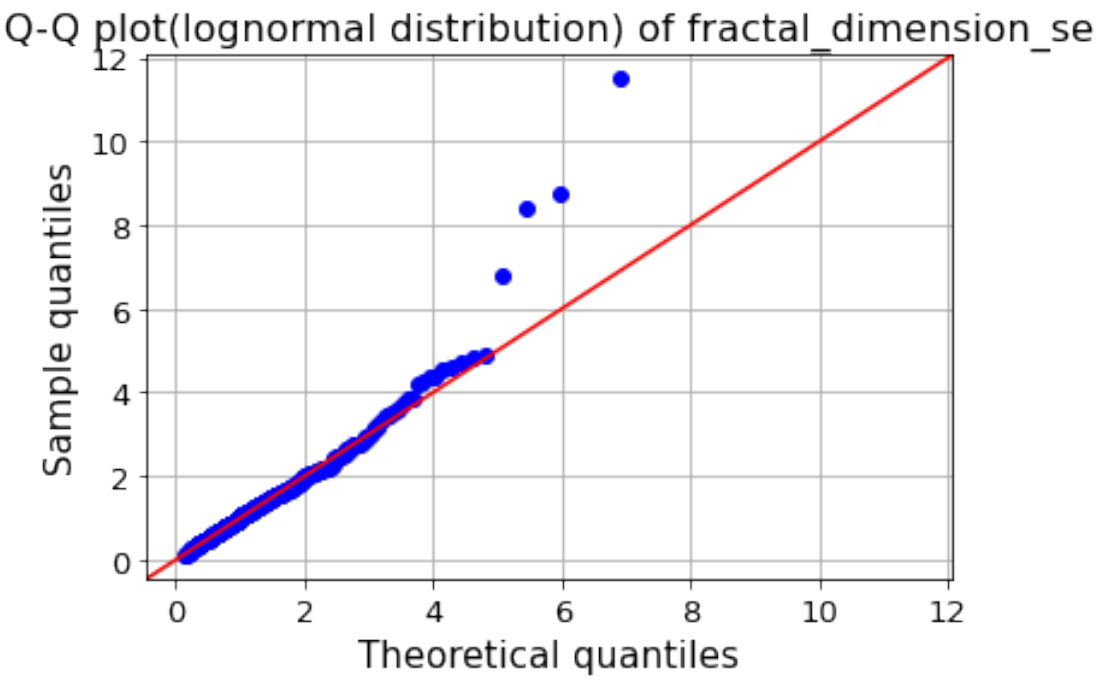

```
<Figure size 576x360 with 0 Axes>
```



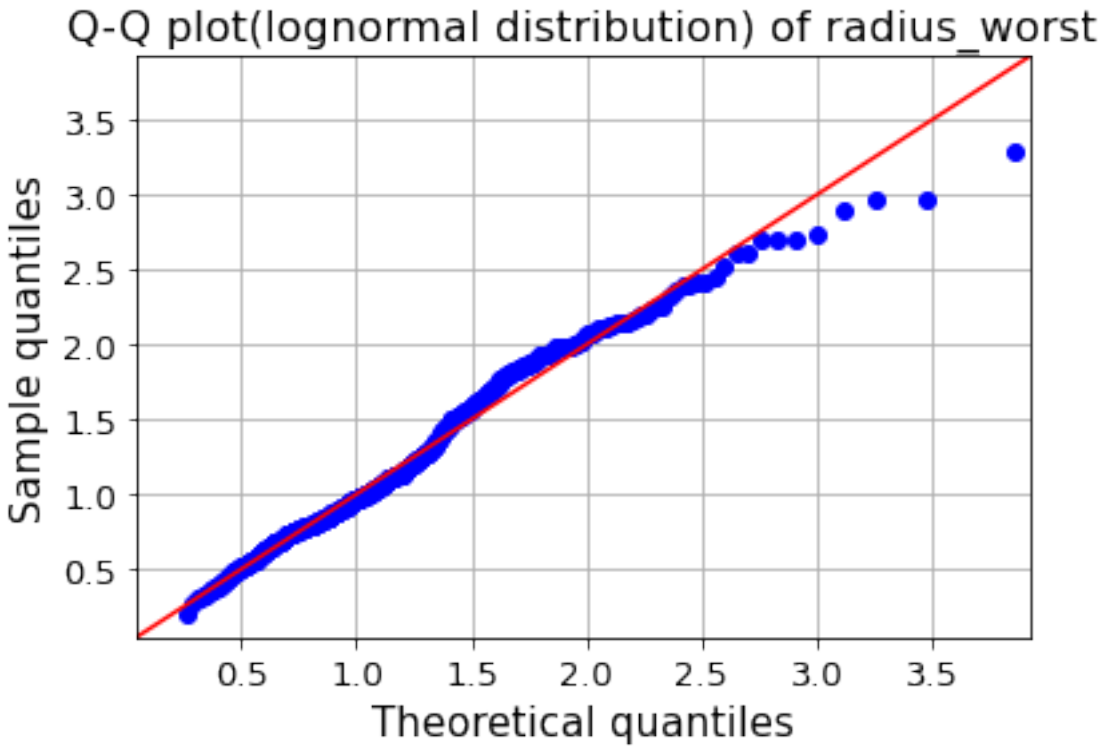

```
<Figure size 576x360 with 0 Axes>
```



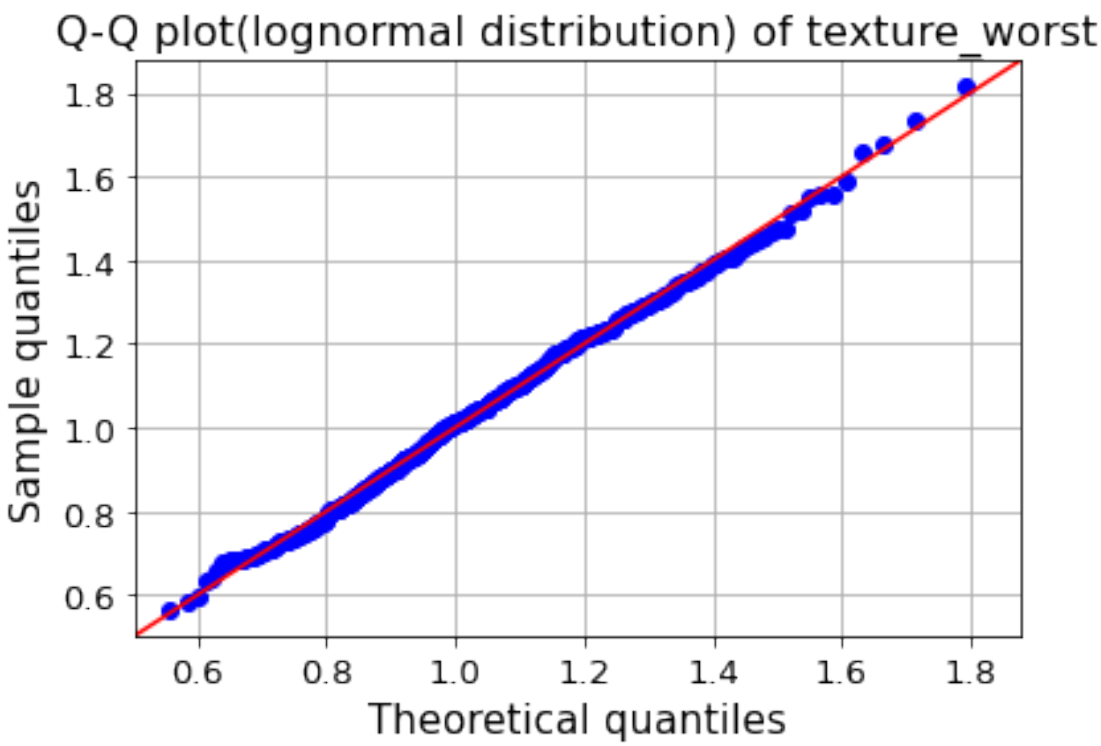
```
<Figure size 576x360 with 0 Axes>
```



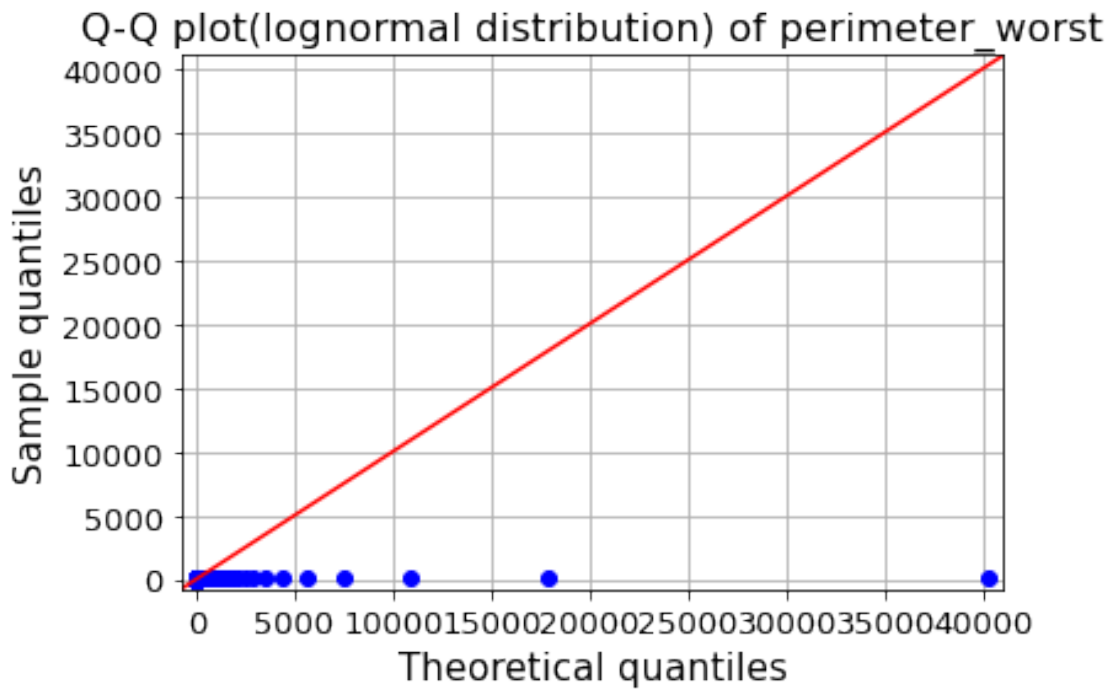

```
<Figure size 576x360 with 0 Axes>
```



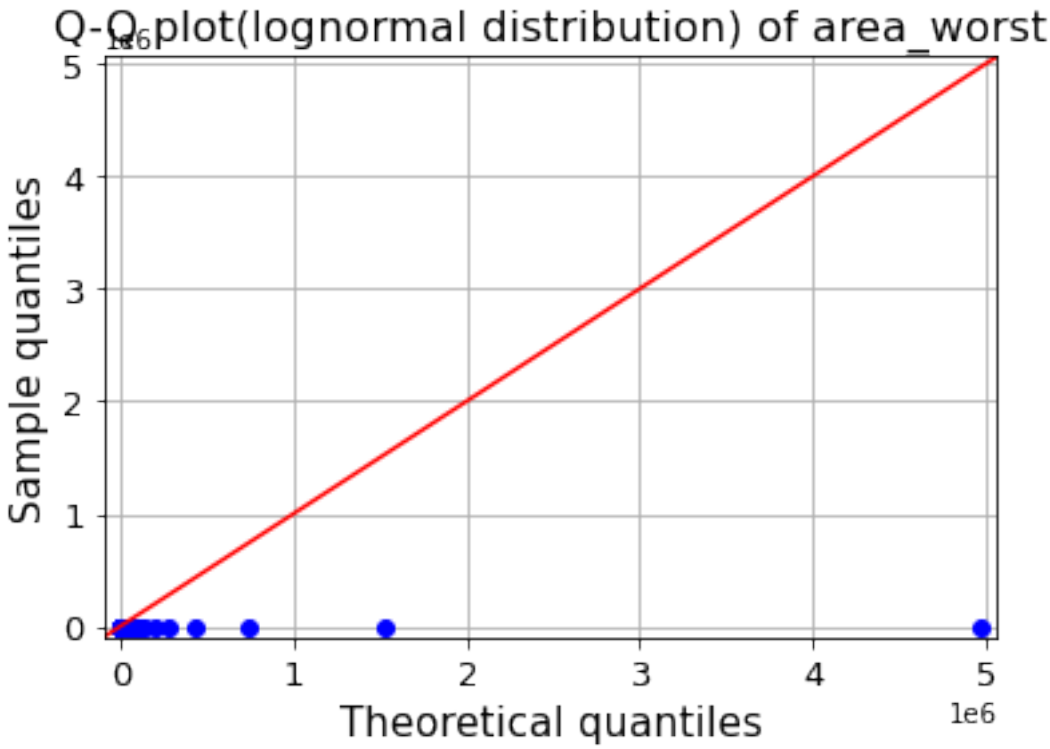

```
<Figure size 576x360 with 0 Axes>
```



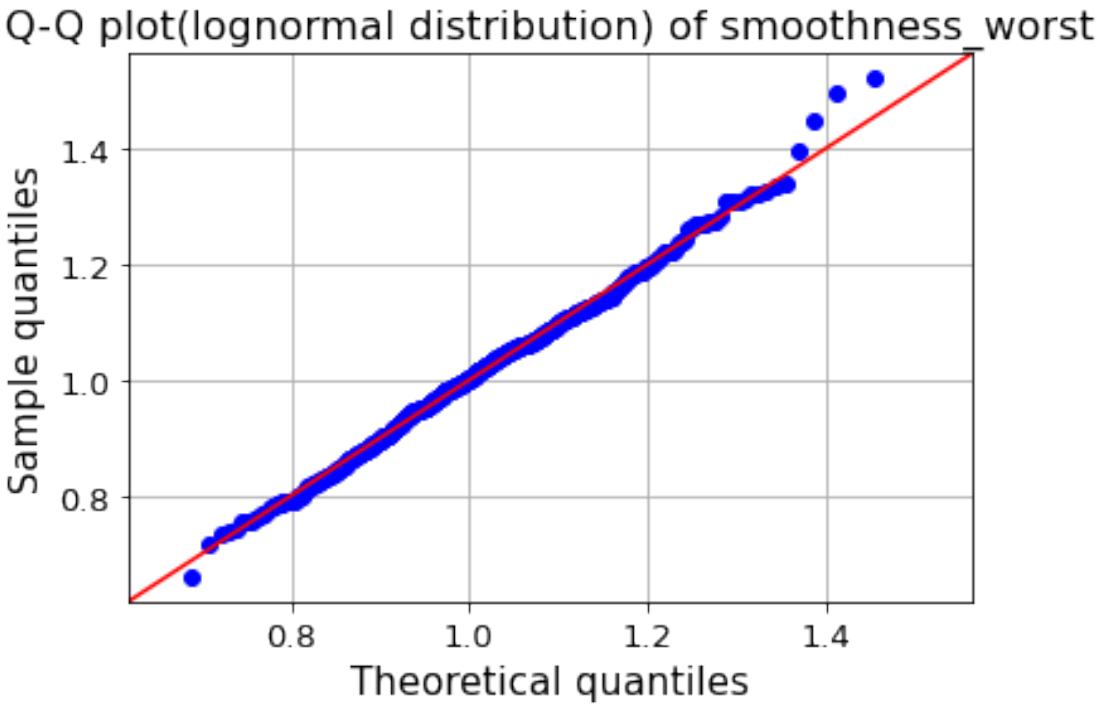
```
<Figure size 576x360 with 0 Axes>
```



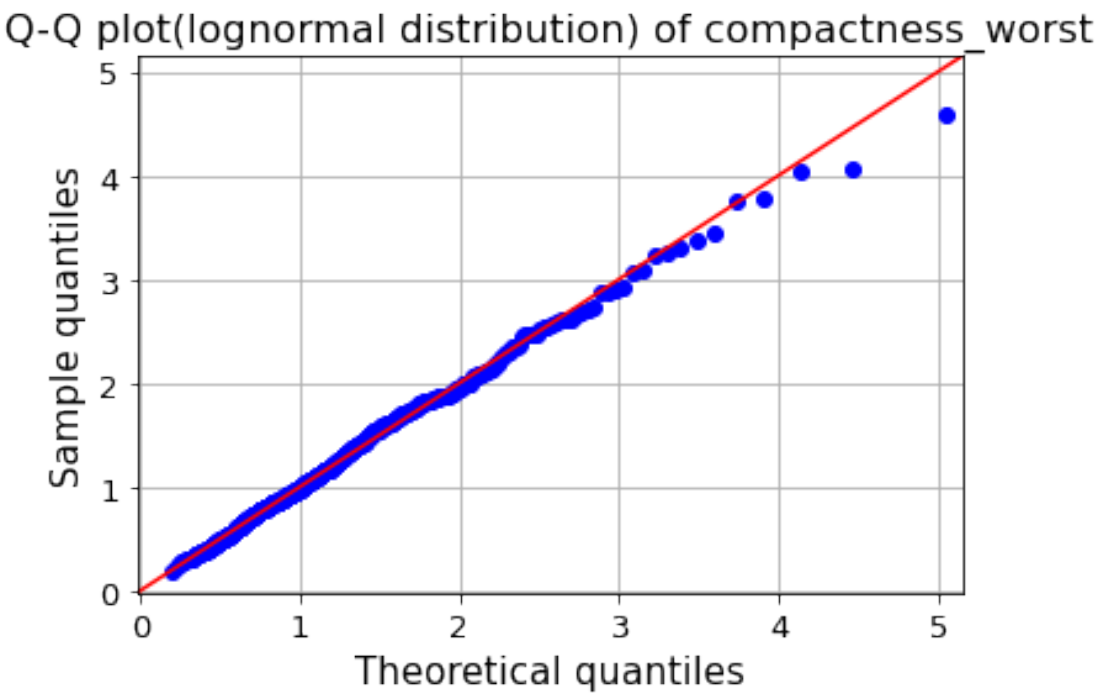

```
<Figure size 576x360 with 0 Axes>
```



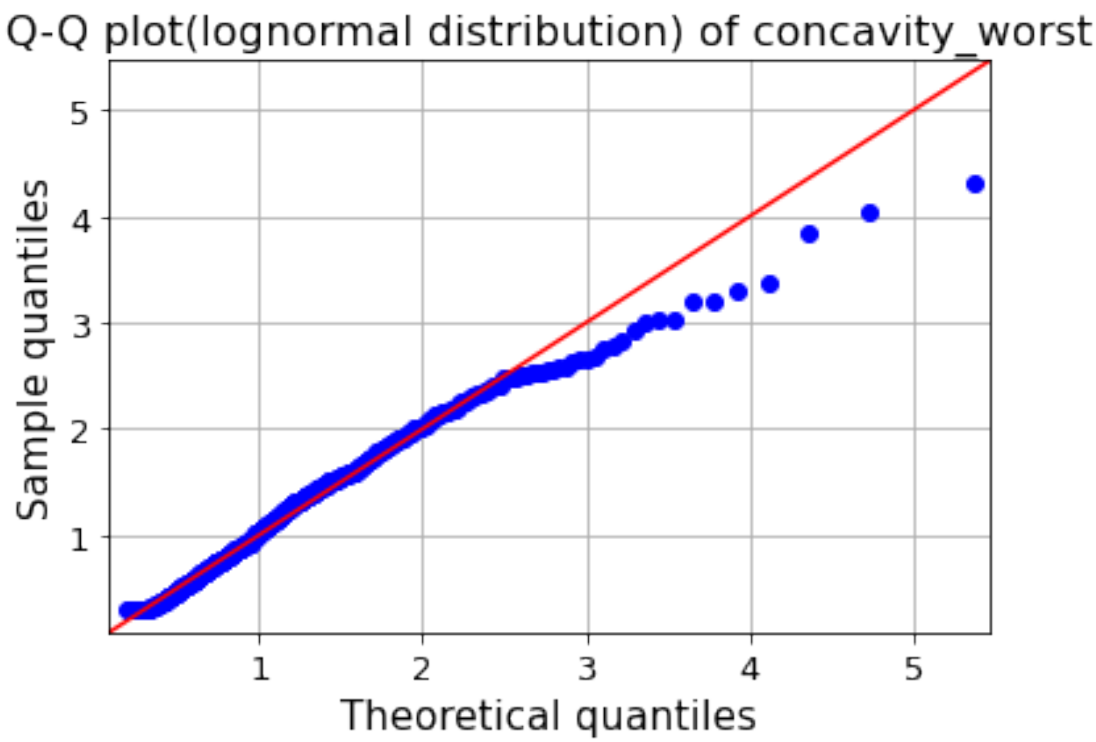

```
<Figure size 576x360 with 0 Axes>
```



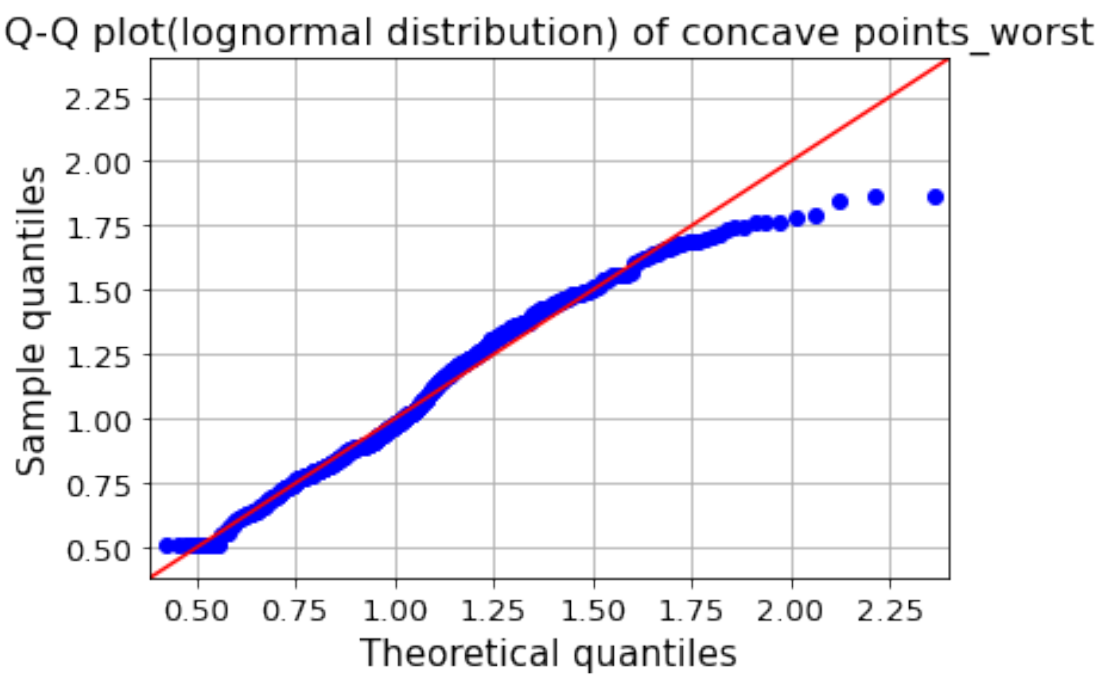
```
<Figure size 576x360 with 0 Axes>
```



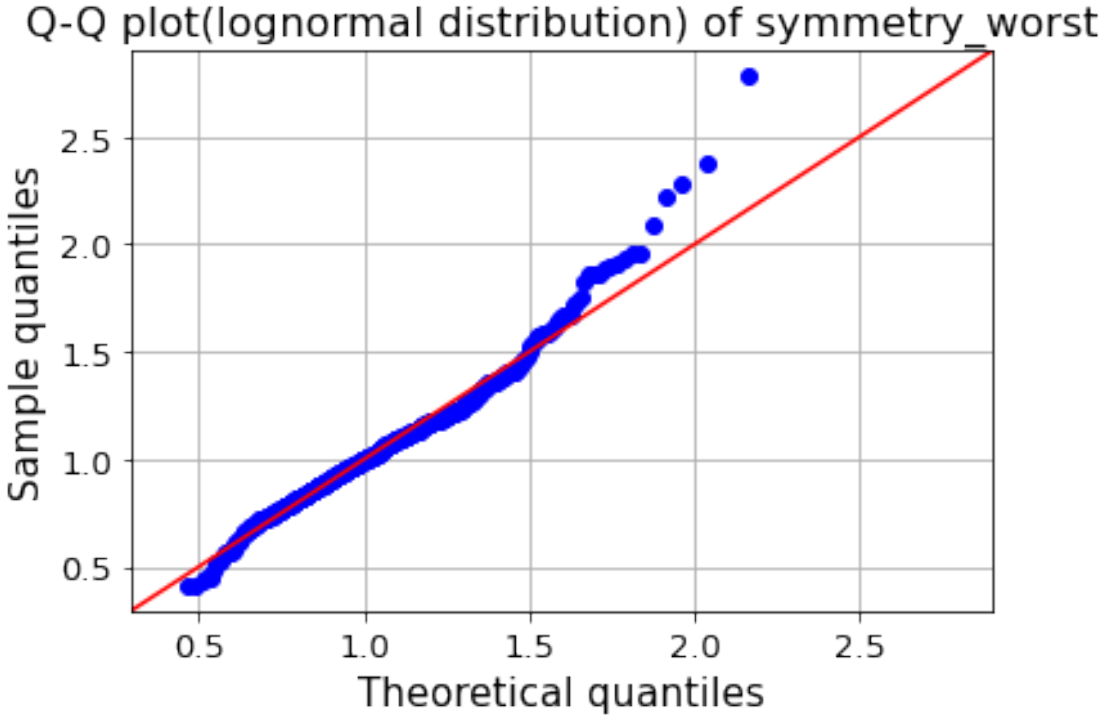

```
<Figure size 576x360 with 0 Axes>
```



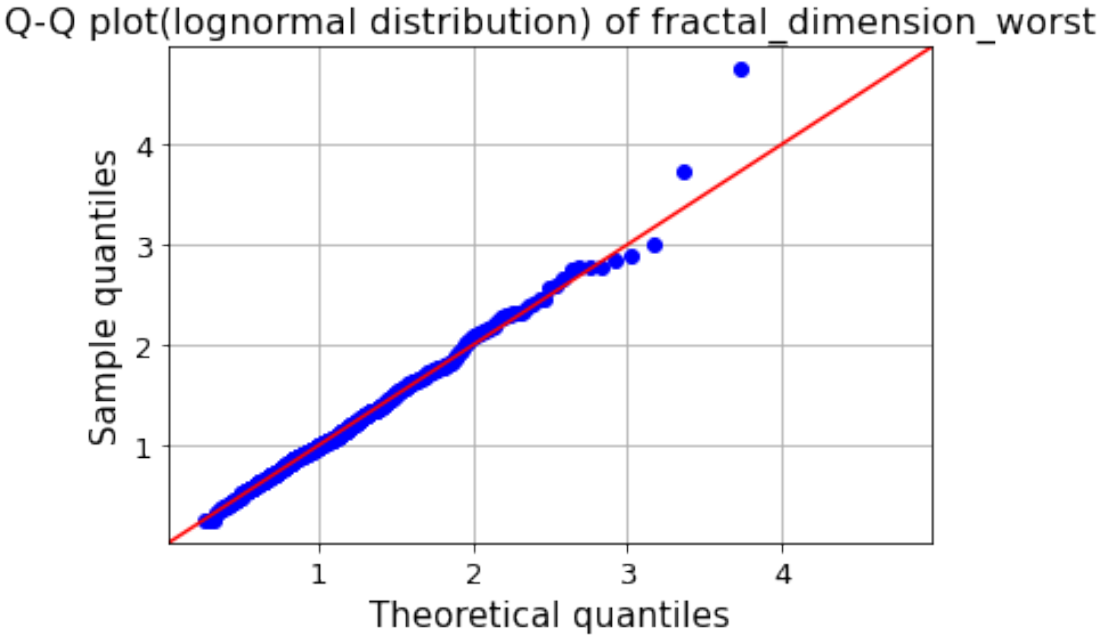

```
<Figure size 576x360 with 0 Axes>
```



```
<Figure size 576x360 with 0 Axes>
```

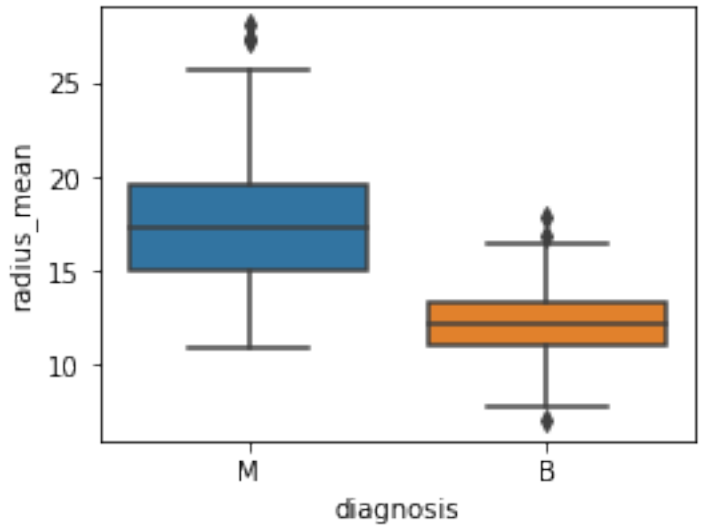


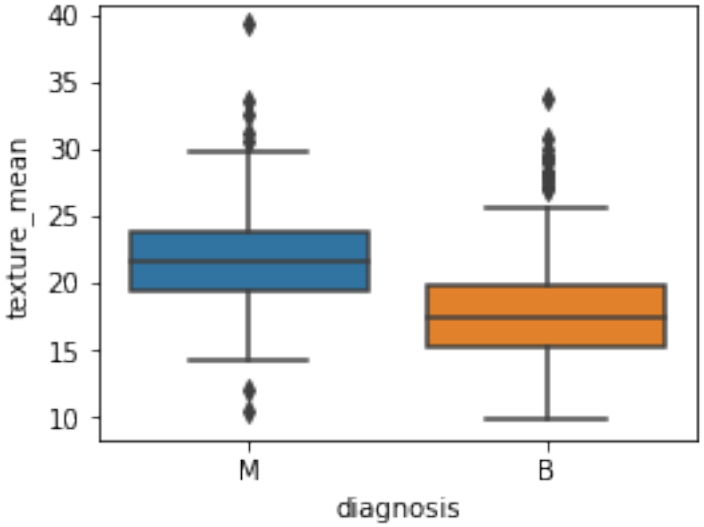

```
<Figure size 576x360 with 0 Axes>
```

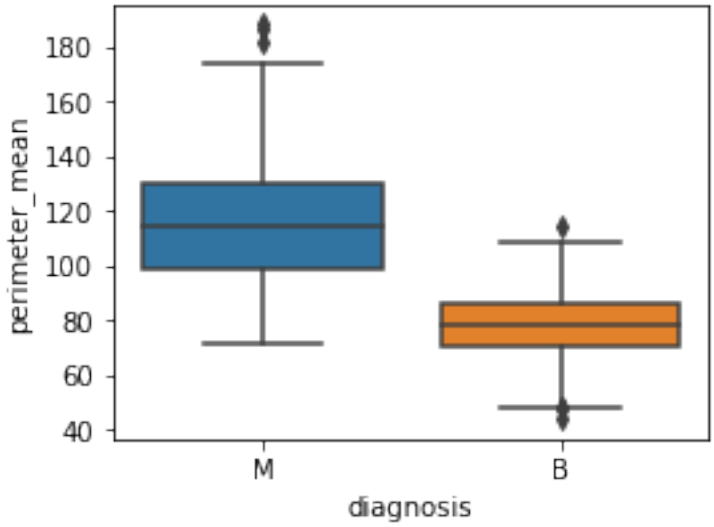


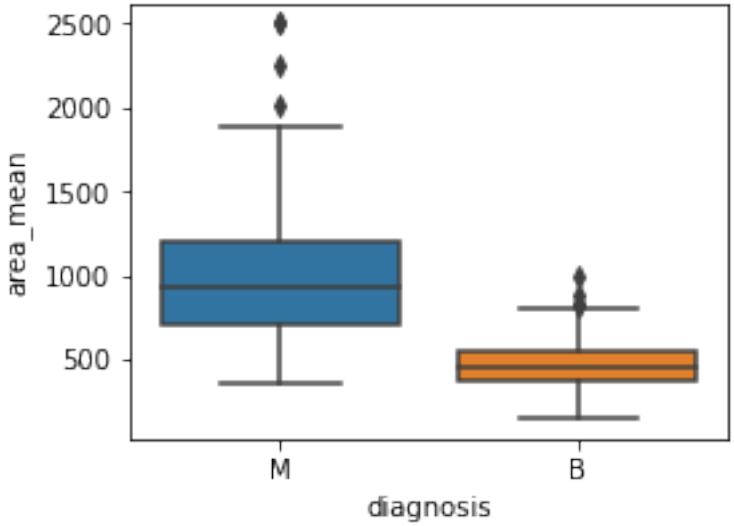
The qqplot is constructed for the independent features, on different distributions.
It is observed that majority of the features follow a *log normal distribution*

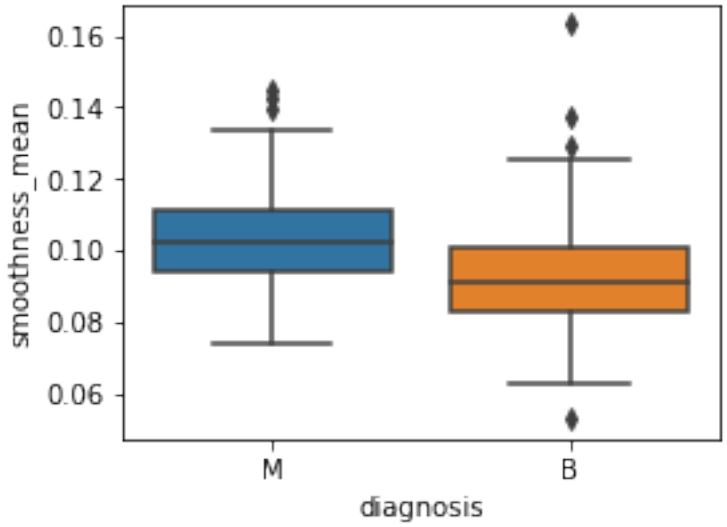
[16]:
```
# Boxplot to show the distribution of dependent variable for different␣
↪independent variables
for a in cols:
  plt.figure(figsize=(4,3))
  fig = sns.boxplot(x=df_target, y=df[a])
```

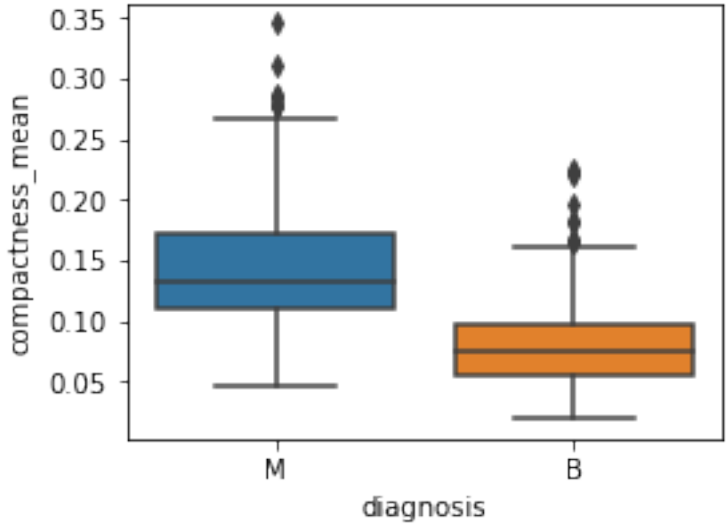


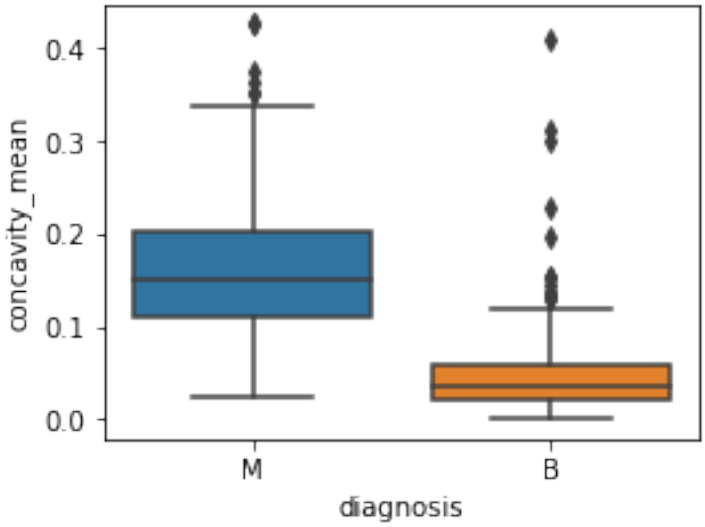


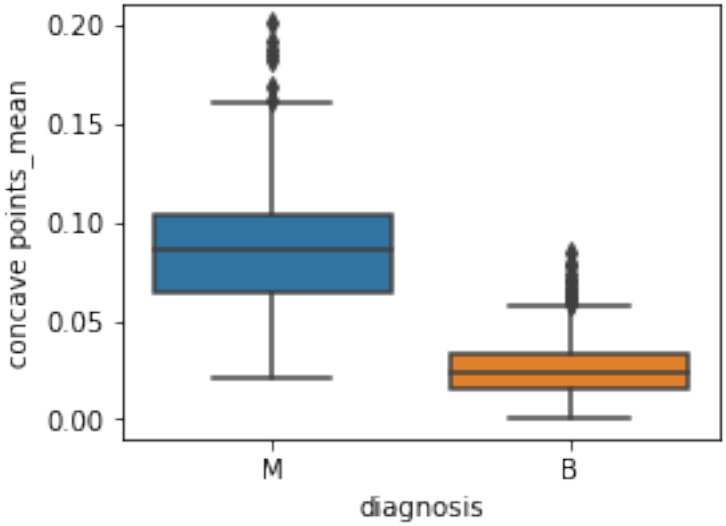

perimeter_mean
180
160
140
120
100
80
60
40
M
B
diagnosis

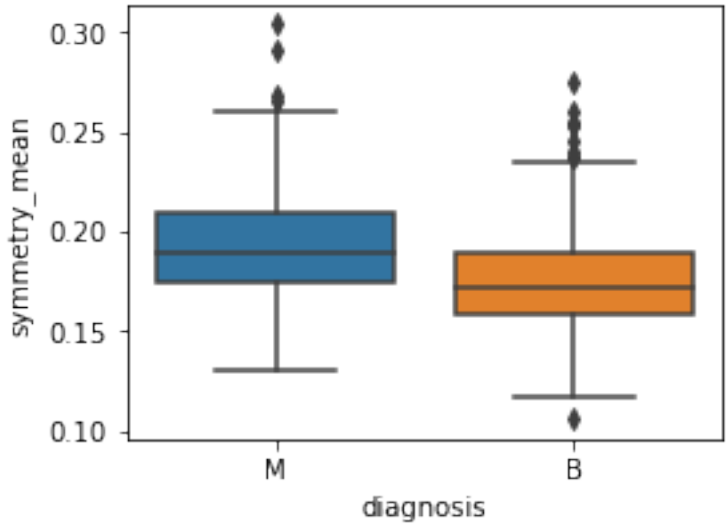

area_mean
2500
2000
1500
1000
500
M
B
diagnosis

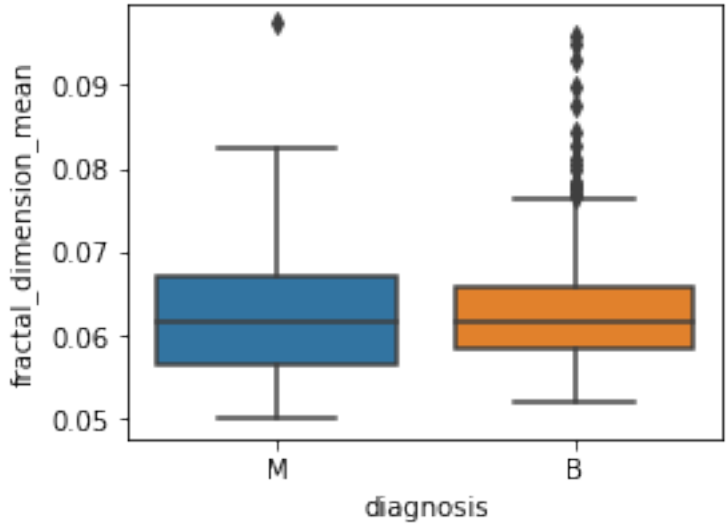

0.16
0.14
0.12
0.10
0.08
0.06
smoothness_mean
M
B
diagnosis

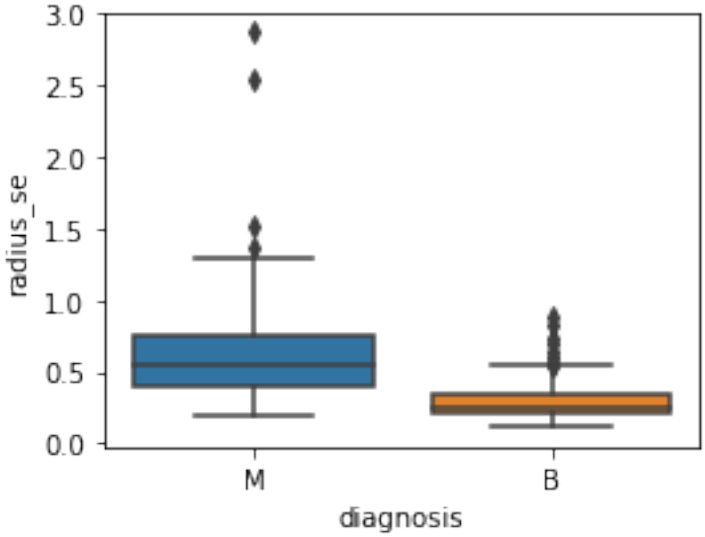

0.35
0.30
0.25
0.20
0.15
0.10
0.05
compactness_mean
M
B
diagnosis

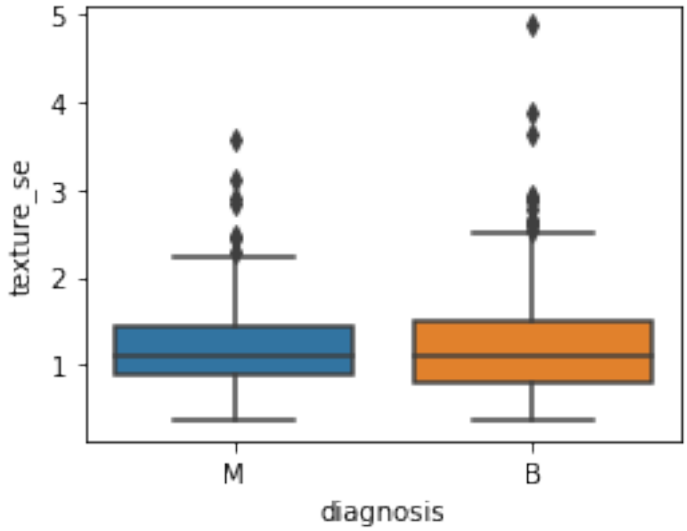

0.4
0.3
0.2
0.1
0.0
concavity_mean
M
B
diagnosis

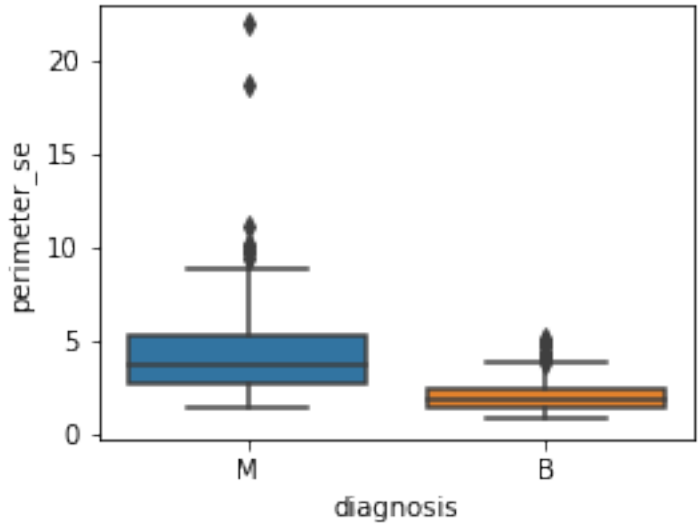

0.20
0.15
0.10
0.05
0.00
concave points_mean
M
B
diagnosis

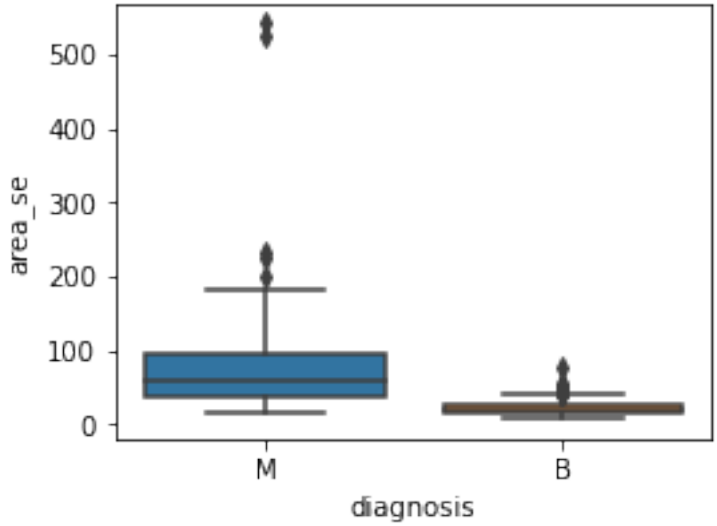

0.30
0.25
0.20
0.15
0.10
symmetry_mean
M
B
diagnosis

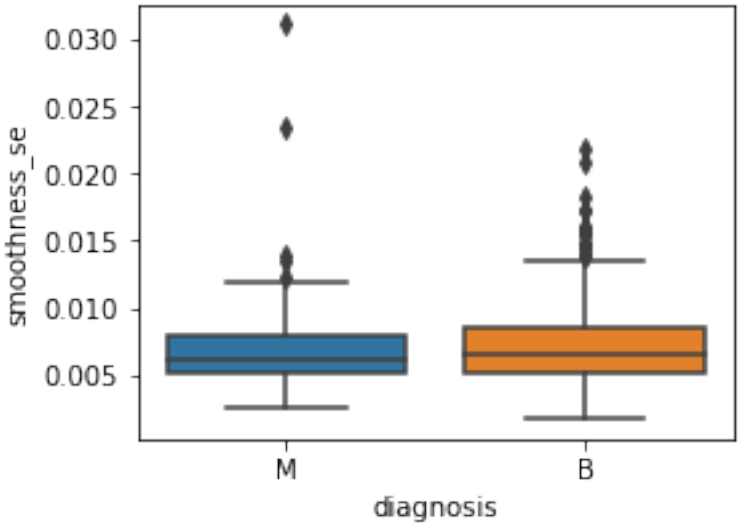

0.09
0.08
0.07
0.06
0.05
fractal_dimension_mean
M
B
diagnosis

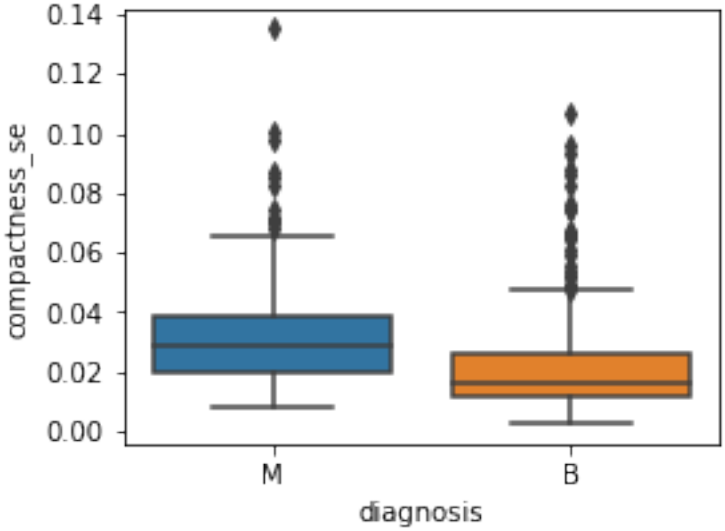

3.0
2.5
2.0
1.5
1.0
0.5
0.0
radius_se
M
B
diagnosis

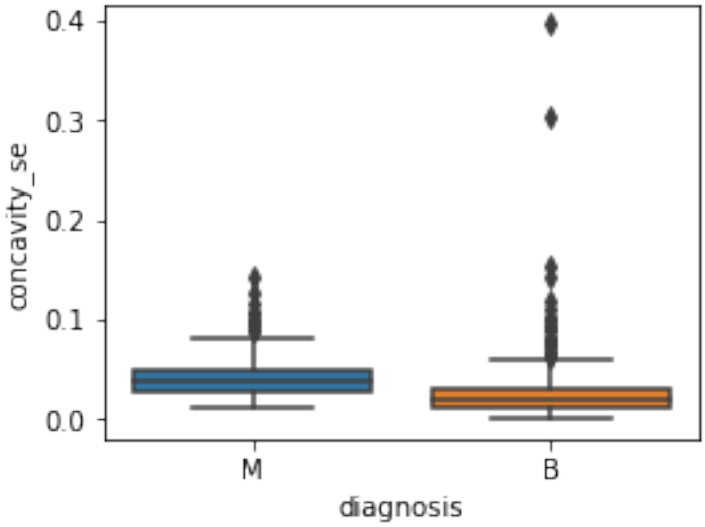

5
4
3
2
1
texture_se
M
B
diagnosis

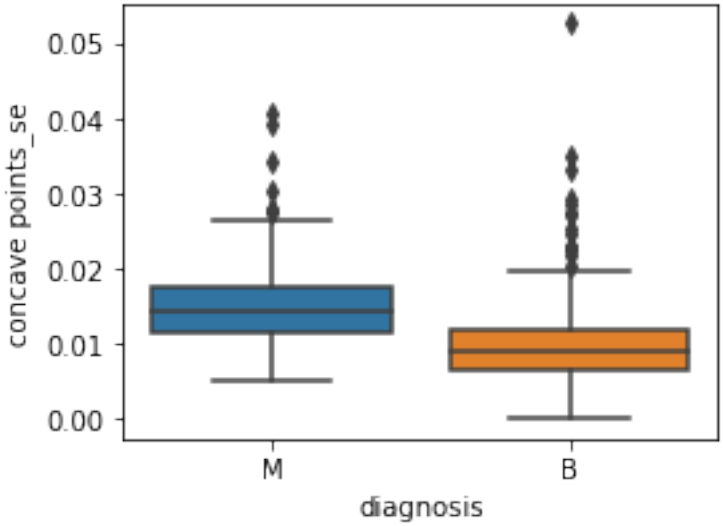

20
15
10
5
0
perimeter_se
M
B
diagnosis

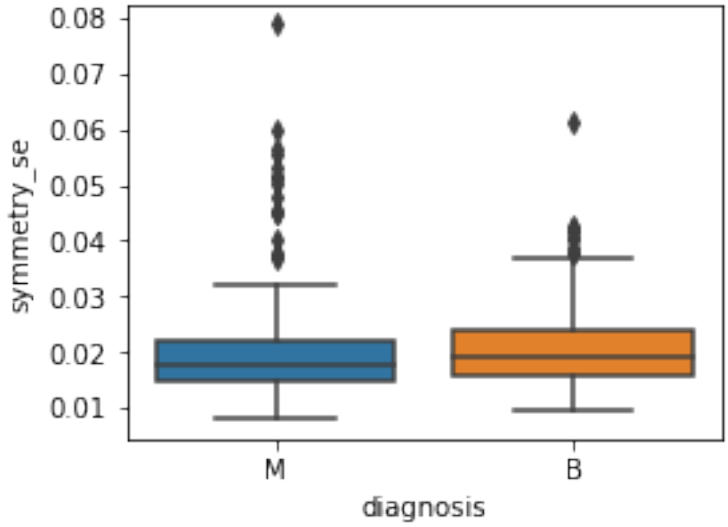

500
400
300
200
100
0
area_se
M
B
diagnosis

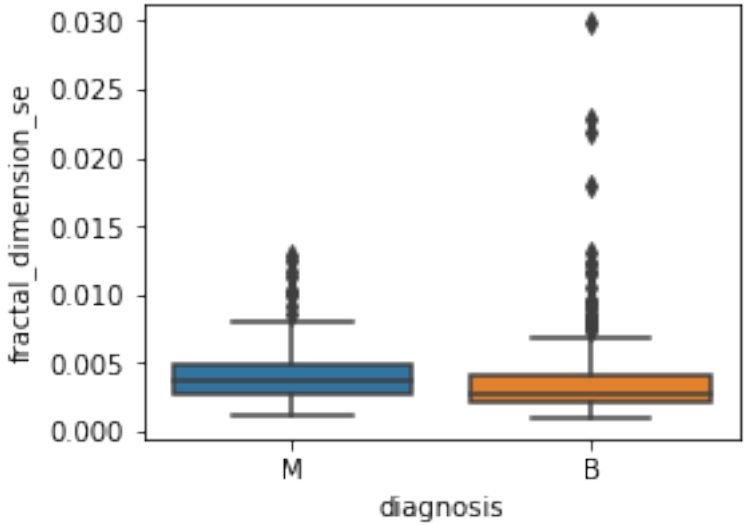

0.030
0.025
0.020
0.015
0.010
0.005
smoothness_se
M
B
diagnosis

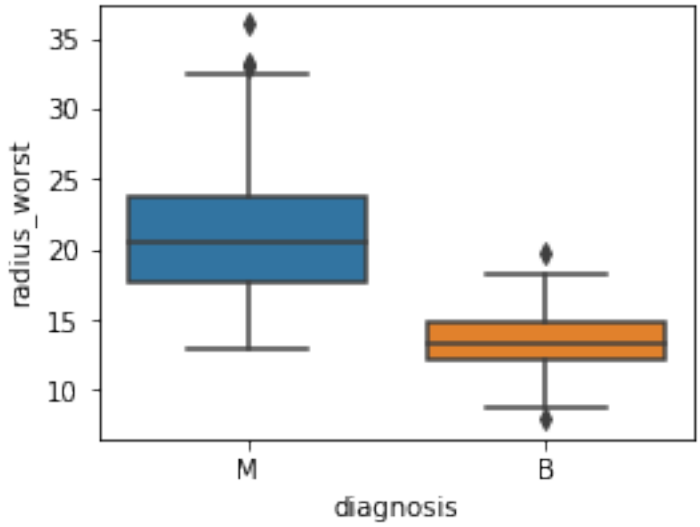

0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
compactness_se
M
B
diagnosis

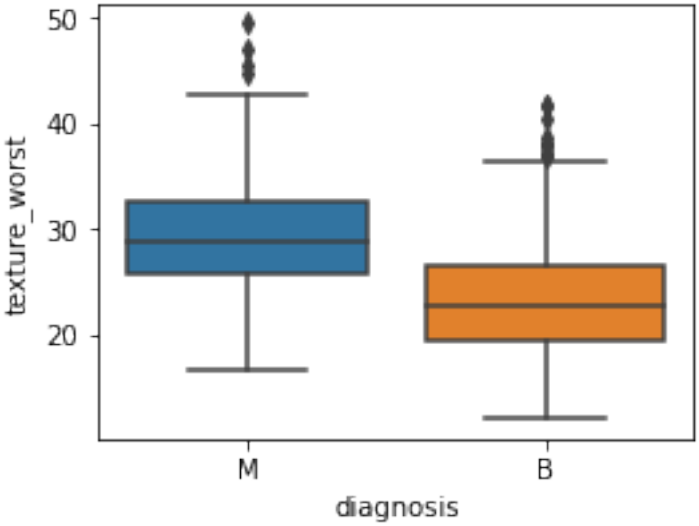

0.4
0.3
0.2
0.1
0.0
concavity_se
M
B
diagnosis

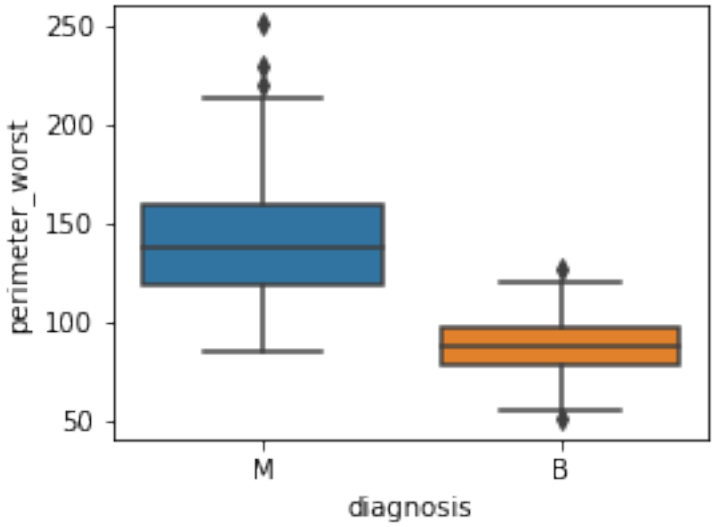

0.05
0.04
0.03
0.02
0.01
0.00
concave points_se
M
B
diagnosis

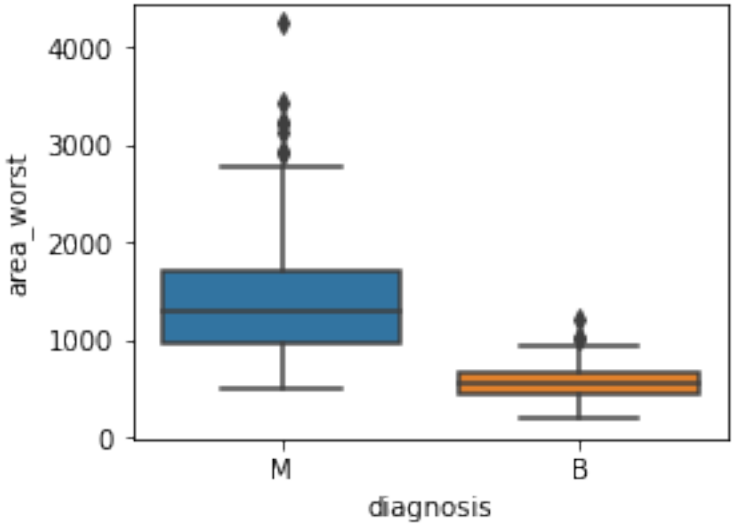

0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
symmetry_se
M
B
diagnosis

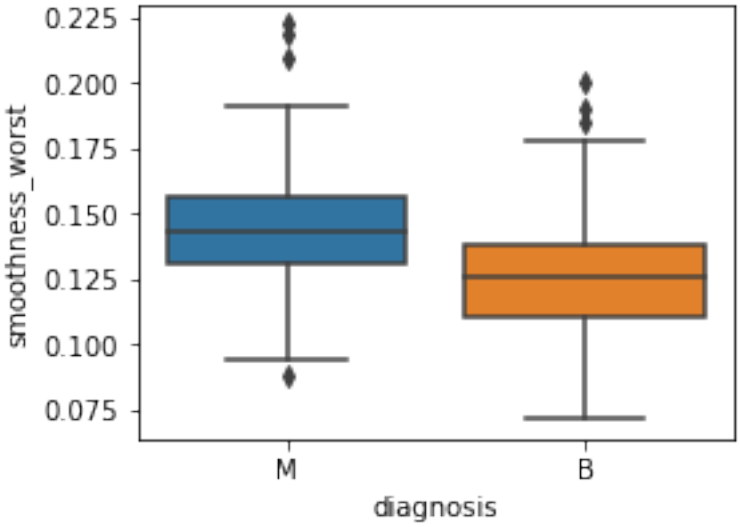

0.030
0.025
0.020
0.015
0.010
0.005
0.000
fractal_dimension_se
M
B
diagnosis

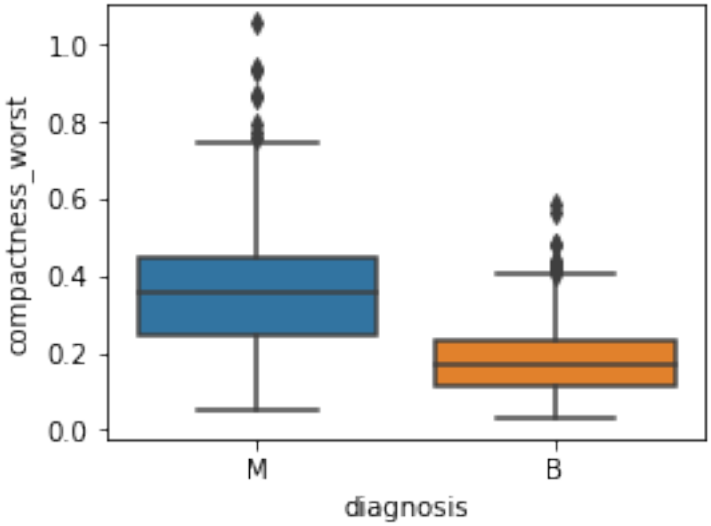

35
30
25
20
15
10
radius_worst
M
B
diagnosis

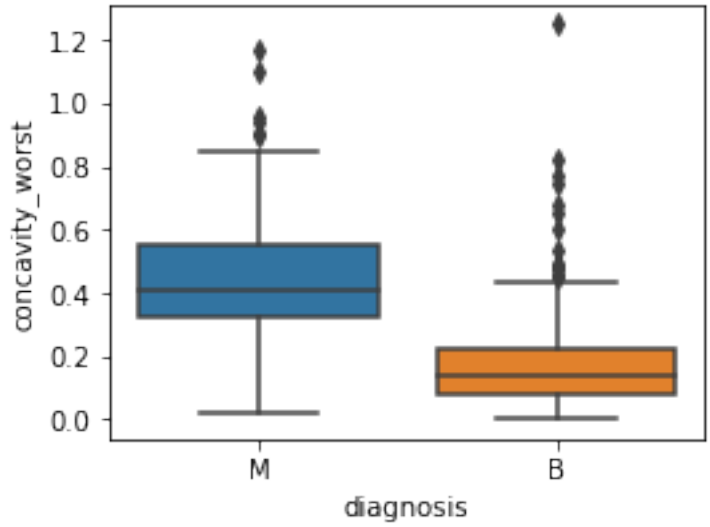

50
40
30
20
texture_worst
M
B
diagnosis

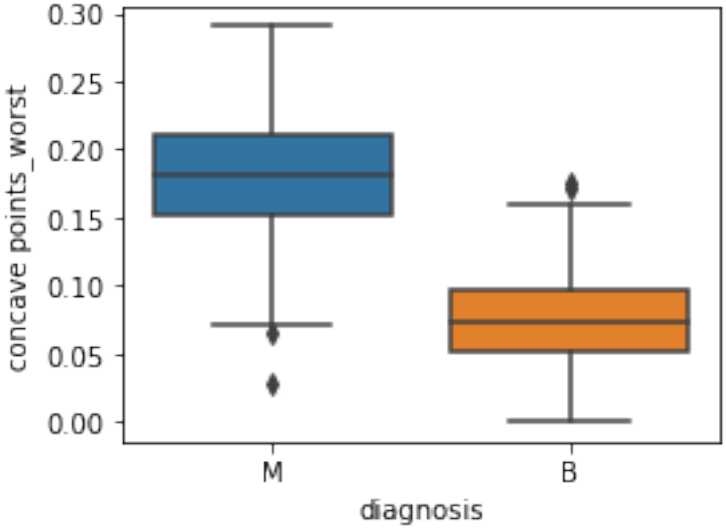

250
200
150
100
50
perimeter_worst
M
B
diagnosis

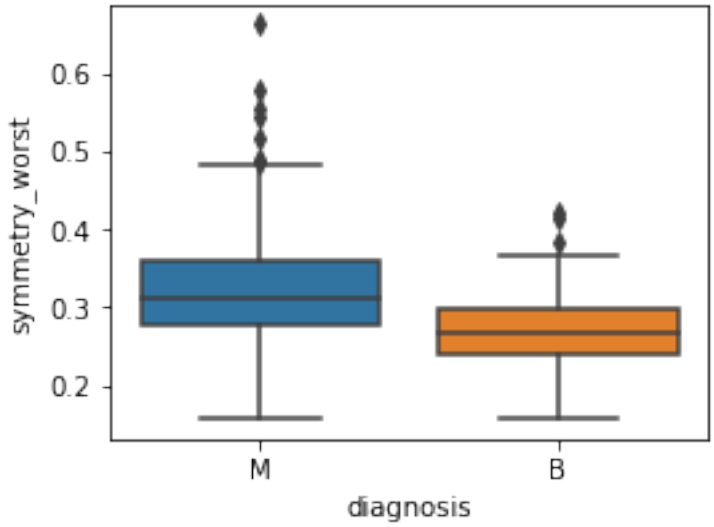

4000
3000
2000
1000
0
area_worst
M
B
diagnosis

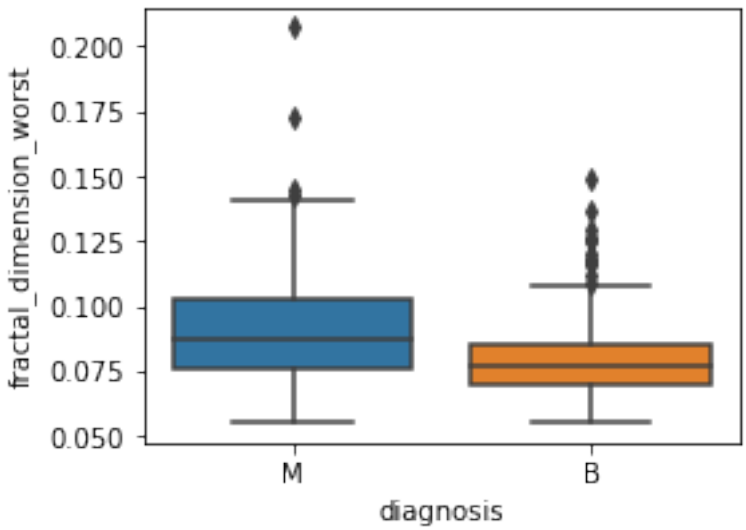

0.225
0.200
0.175
0.150
0.125
0.100
0.075
smoothness_worst
M
B
diagnosis


1.0
0.8
0.6
0.4
0.2
0.0
compactness_worst
M
B
diagnosis


concavity_worst
1.2
1.0
0.8
0.6
0.4
0.2
0.0
M
B
diagnosis


concave points_worst
0.30
0.25
0.20
0.15
0.10
0.05
0.00
M
B
diagnosis





We can observe that for most of the independent features, the malignant variant(M) has exhibited a trend of being associated with higher readings compared to the benign variant(B).
Note: A set of outliers exist in the distribution of benign variant which exhibit values on-par with or greater than malignant variants.

[17]:
```
#Concatenating the response DF to predictor DF, for use in outlier filtering.
df=pd.concat([df,df_target],axis=1)
```

[17]:
```
```

3 Outlier Detection

3.1 Z Score

Outliers in a normal distribution can be detected by using the Z-score statistic. From the above qq plots, we can infer that the features *texture_mean,smoothness_mean,symmetry_mean,texture_worst,smoothness_worst and concave points_worst* closely resemble a normal distribution.

[18]:
```
def boxplot(dataset,column):
    sns.boxplot(data=dataset,x=dataset[column])
    plt.title(column)
    plt.show()
#Creating a list containing tuples of upper and lower limits(3 SD's) of a␣
↪feature column, for use in Z score analysis(Z-score of 3)
limit_list=[]
norm_features=['texture_mean','smoothness_mean','symmetry_mean','texture_worst','smoothness_wo
↪points_worst']
for a in norm_features:
    limit_list.append((df[a].mean() + 2.5*df[a].std(),df[a].mean() - 2.5*df[a].
↪std()))
```

[19]:
```
# Storing dataframes with outliers removed from each normally distributed␣
↪feature
outlier_df=[]
df_filtered=df
for l in enumerate(norm_features):
  dtf=df[(df[l[1]] < limit_list[l[0]][0]) & (df[l[1]] > limit_list[l[0]][1])]
  dtf = dtf.reset_index(drop=True)
  outlier_df.append(dtf)
# Dataframe without outliers in the normally distributed features
for l in enumerate(norm_features):
  df_filtered=df_filtered[(df_filtered[l[1]] < limit_list[l[0]][0]) &␣
↪(df_filtered[l[1]] > limit_list[l[0]][1])]
  df_filtered = df_filtered.reset_index(drop=True)
```

3.2 Detection using Percentile

[20]:
```
# Outliers for rest of the independent features are identified by␣
↪percentile-based filtering
l=[a for a in cols if a not in norm_features]
for d in l:

 lower, upper = np.percentile(a=df_filtered[l],q=[0.5,99.8])
 df_filtered = df_filtered[(df_filtered[d]>lower) & (df_filtered[d]<upper)]
df_filtered.reset_index(drop=True)
```

[20]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	\
0	20.57	17.77	132.90	1326.0	0.08474	
1	19.69	21.25	130.00	1203.0	0.10960	
2	20.29	14.34	135.10	1297.0	0.10030	
3	12.45	15.70	82.57	477.1	0.12780	
4	18.25	19.98	119.60	1040.0	0.09463	
..	…	…	…	…	…	
425	14.05	27.15	91.38	600.4	0.09929	
426	20.92	25.09	143.00	1347.0	0.10990	
427	20.13	28.25	131.20	1261.0	0.09780	
428	16.60	28.08	108.30	858.1	0.08455	
429	20.60	29.33	140.10	1265.0	0.11780	

	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	\
0	0.07864	0.08690	0.07017	0.1812	
1	0.15990	0.19740	0.12790	0.2069	
2	0.13280	0.19800	0.10430	0.1809	
3	0.17000	0.15780	0.08089	0.2087	
4	0.10900	0.11270	0.07400	0.1794	
..	…	…	…	…	
425	0.11260	0.04462	0.04304	0.1537	
426	0.22360	0.31740	0.14740	0.2149	
427	0.10340	0.14400	0.09791	0.1752	
428	0.10230	0.09251	0.05302	0.1590	
429	0.27700	0.35140	0.15200	0.2397	

	fractal_dimension_mean	…	texture_worst	perimeter_worst	area_worst	\
0	0.05667	…	23.41	158.8	1956.0	
1	0.05999	…	25.53	152.5	1709.0	
2	0.05883	…	16.67	152.2	1575.0	
3	0.07613	…	23.75	103.4	741.6	
4	0.05742	…	27.66	153.2	1606.0	
..	…	…	…	…	…	
425	0.06171	…	33.17	100.2	706.7	
426	0.06879	…	29.41	179.1	1819.0	
427	0.05533	…	38.25	155.0	1731.0	
428	0.05648	…	34.12	126.7	1124.0	
429	0.07016	…	39.42	184.6	1821.0	

	smoothness_worst	compactness_worst	concavity_worst	\
0	0.1238	0.1866	0.2416	
1	0.1444	0.4245	0.4504	
2	0.1374	0.2050	0.4000	
3	0.1791	0.5249	0.5355	
4	0.1442	0.2576	0.3784	
..	…	…	…	
425	0.1241	0.2264	0.1326	

```
426            0.1407            0.4186           0.6599
427            0.1166            0.1922           0.3215
428            0.1139            0.3094           0.3403
429            0.1650            0.8681           0.9387

     concave points_worst  symmetry_worst  fractal_dimension_worst  diagnosis
0                  0.1860          0.2750                  0.08902          M
1                  0.2430          0.3613                  0.08758          M
2                  0.1625          0.2364                  0.07678          M
3                  0.1741          0.3985                  0.12440          M
4                  0.1932          0.3063                  0.08368          M
..                    …               …                        …        …
425                0.1048          0.2250                  0.08321          B
426                0.2542          0.2929                  0.09873          M
427                0.1628          0.2572                  0.06637          M
428                0.1418          0.2218                  0.07820          M
429                0.2650          0.4087                  0.12400          M

[430 rows x 31 columns]
```

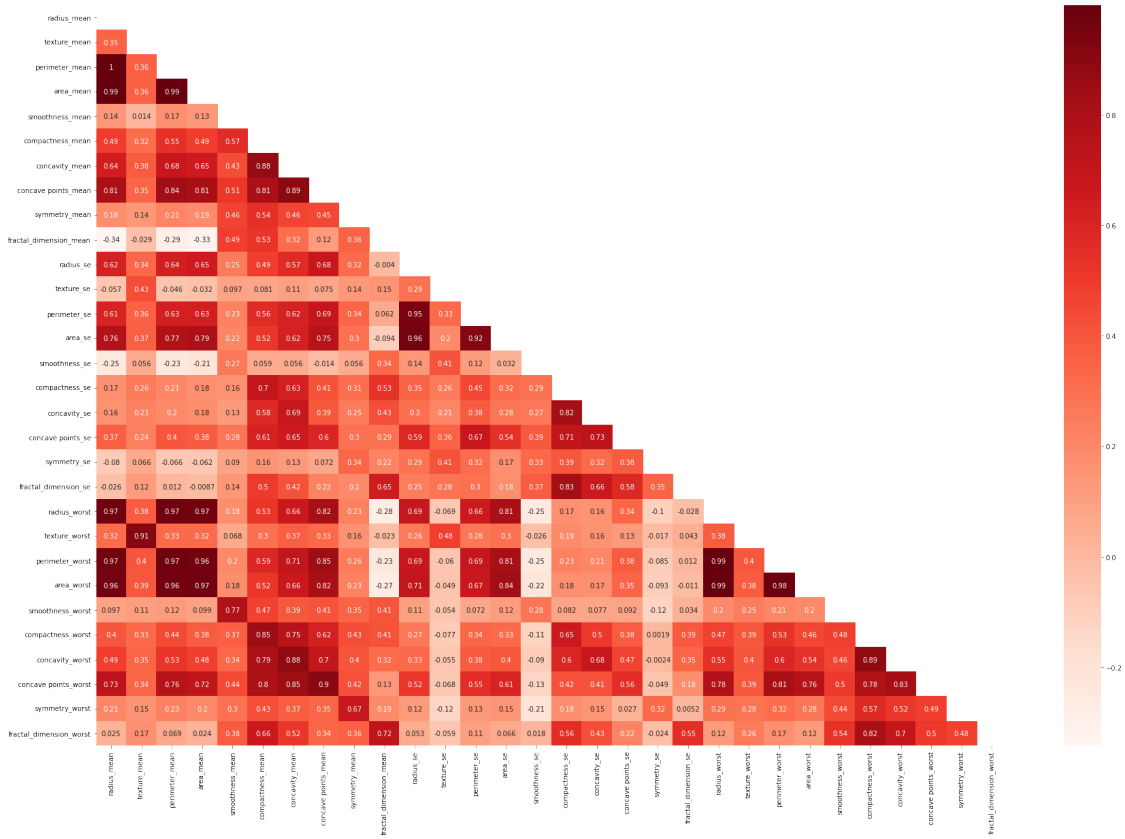
4 Feature Selection

The dataset contains 30 independent features, constructed from indicators such as area,perimeter,symmetry etc. These indicators readings are represented using 3 parameters(_mean,_se,_worst). Hence applying feature selection on such a column collection could provide deeper insights into the indicator-dependent variable relationship.

4.1 Heatmap

[21]:
```
# A correlation heatmap is contructed to visualize the relation between␣
↪predictor and response variables
plt.figure(figsize=(30,20))
df_corr=df_filtered.corr()
mask = np.triu(np.ones_like(df_corr))
sns.heatmap(df_corr,cmap='Reds',annot=True,mask=mask)
```

[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa87d269d90>
```

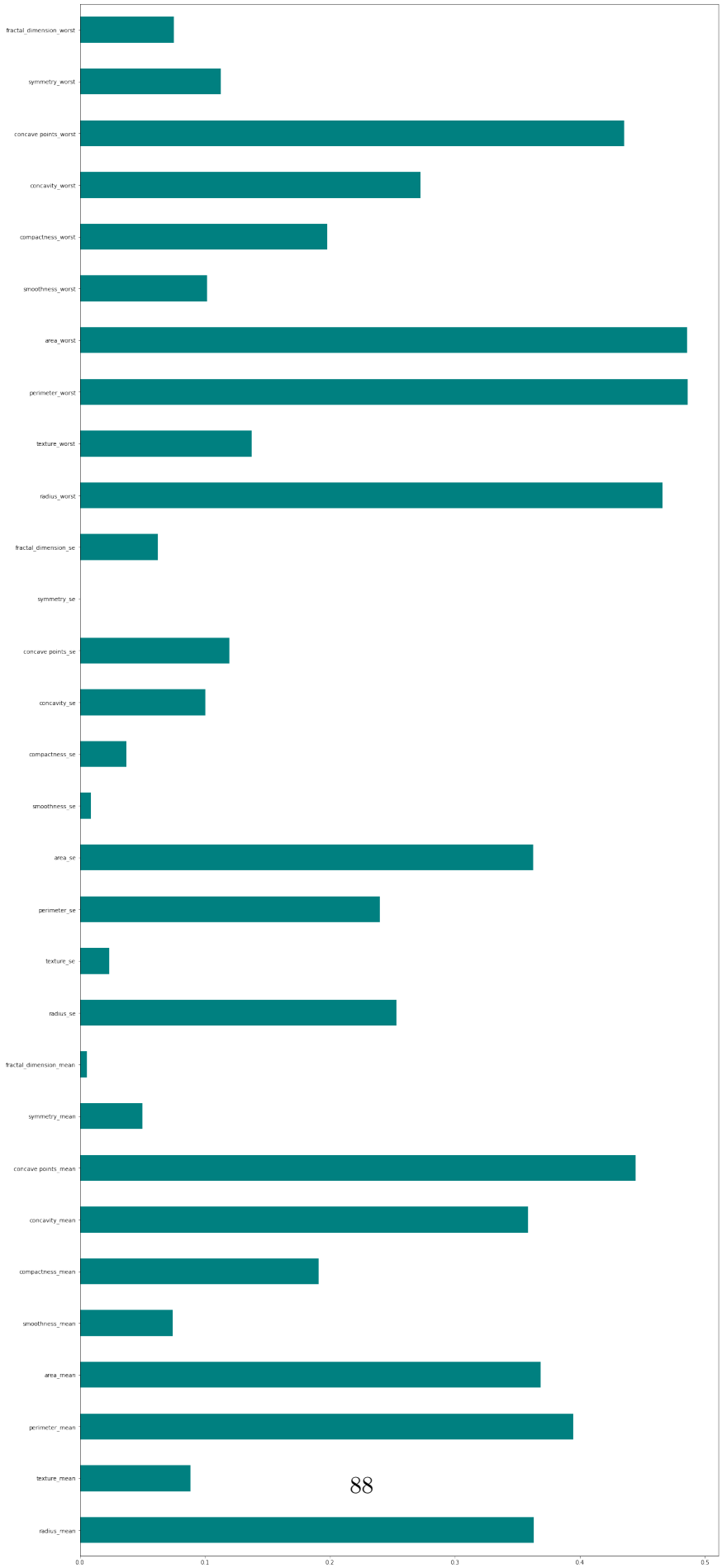


The above plot can help us in inferring that **all** the independent features exhibit considerable correlation with the dependent variable

4.2 Mutual Information

[22]:
```
# plotting the information gain/mutual info data
plt.figure(figsize=(20,50))
X,Y=df_filtered.iloc[:,:-1],df_filtered.iloc[:,-1]
from sklearn.feature_selection import mutual_info_classif
importances=mutual_info_classif(X,Y)
feat_importances=pd.Series(importances,df_filtered.columns[0:len(df_filtered.
 ↪columns)-1])
feat_importances.plot(kind='barh',color='teal')
```

[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa879d014f0>


fractal_dimension_worst
symmetry_worst
concave points_worst
concavity_worst
compactness_worst
smoothness_worst
area_worst
perimeter_worst
texture_worst
radius_worst
fractal_dimension_se
symmetry_se
concave points_se
concavity_se
compactness_se
smoothness_se
area_se
perimeter_se
texture_se
radius_se
fractal_dimension_mean
symmetry_mean
concave points_mean
concavity_mean
compactness_mean
smoothness_mean
area_mean
perimeter_mean
texture_mean
radius_mean
0.0
0.1
0.2
0.3
0.4
0.5

4.3 Feature selection using Gini Impurity

[23]:
```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import ExtraTreesClassifier

estimator = ExtraTreesClassifier(n_estimators = 10)
featureSelection = SelectFromModel(estimator)
featureSelection.fit(X,Y)
selectedFeatures = featureSelection.transform(X)
best_cols=list(X.columns[featureSelection.get_support()])
```

5 Scaling

In view of the significant magnitude variations present in the data, a scaling operation is required to be performed on the dataset

[50]:
```
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler

scale = StandardScaler()
scaled_features=scale.fit_transform(df_filtered[best_cols])
scaled_features_df = pd.DataFrame(scaled_features, index=df_filtered.index,␣
 ↪columns=best_cols)
```

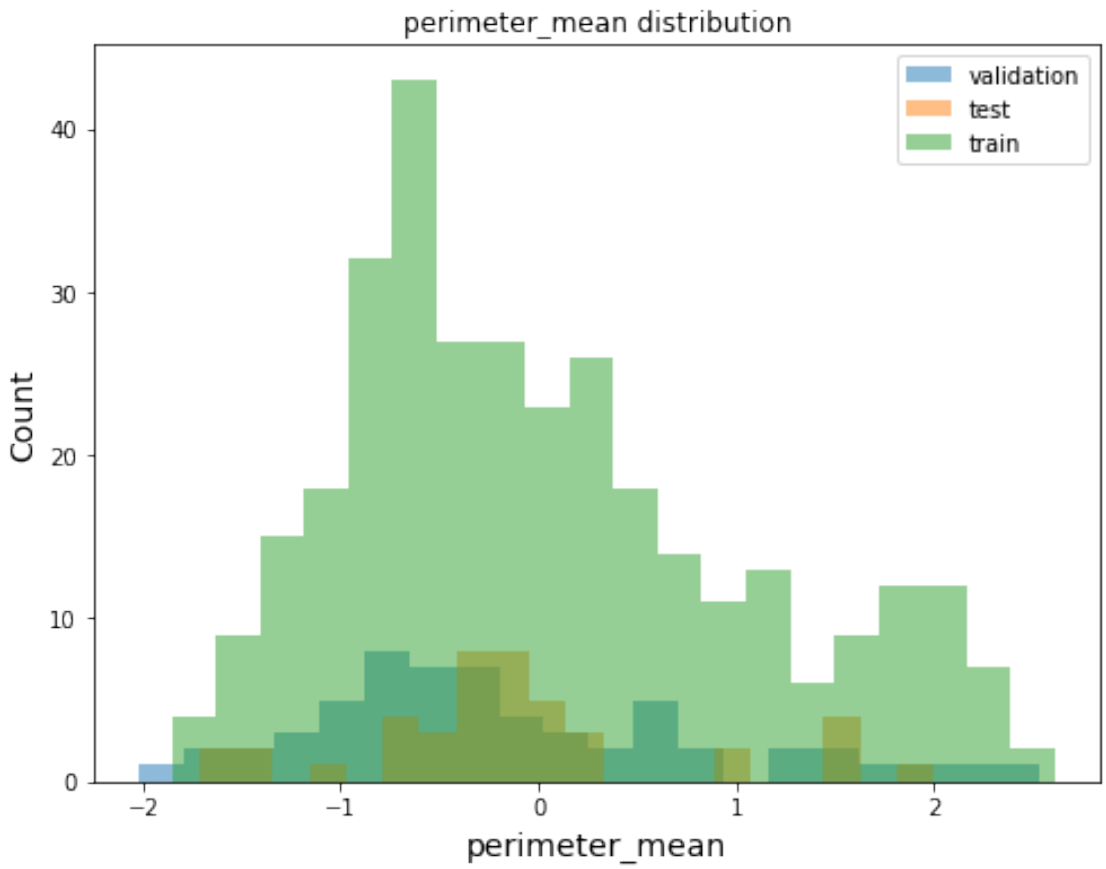
[52]:
```
df_filtered=pd.concat([scaled_features_df,df_filtered['diagnosis']],axis=1)
```

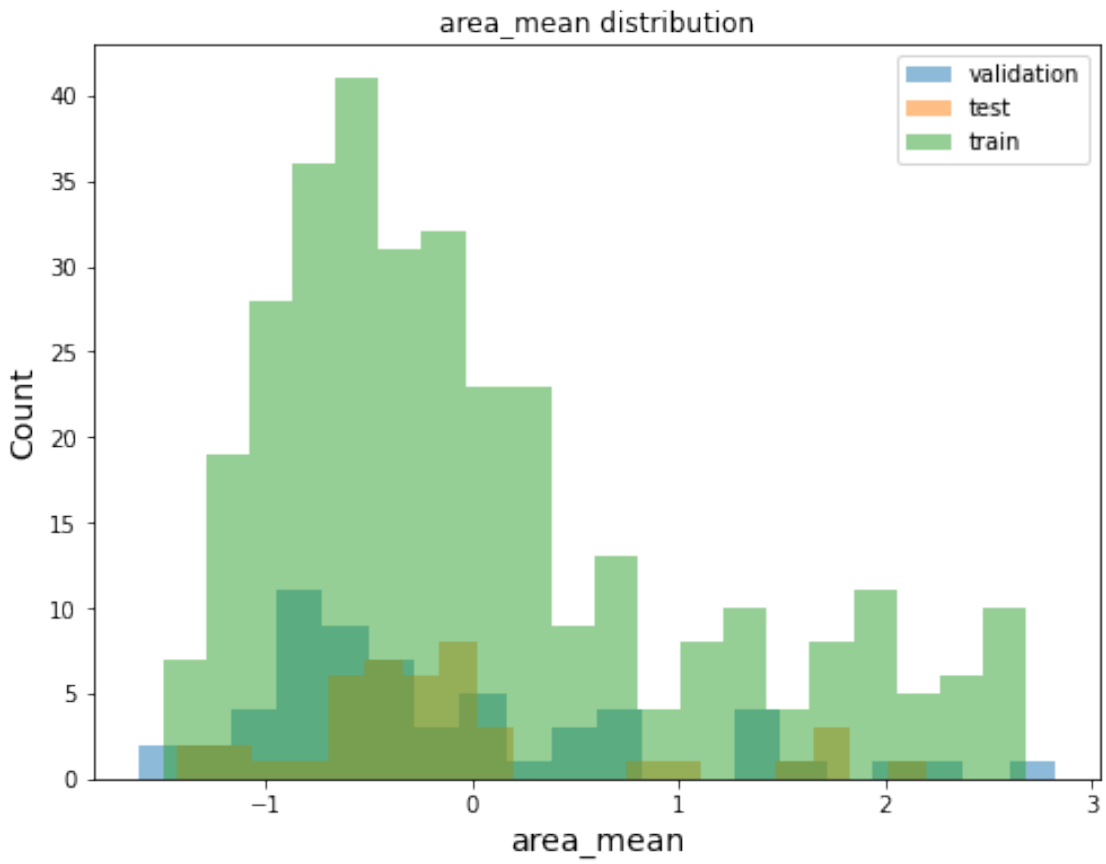
6 Building the model

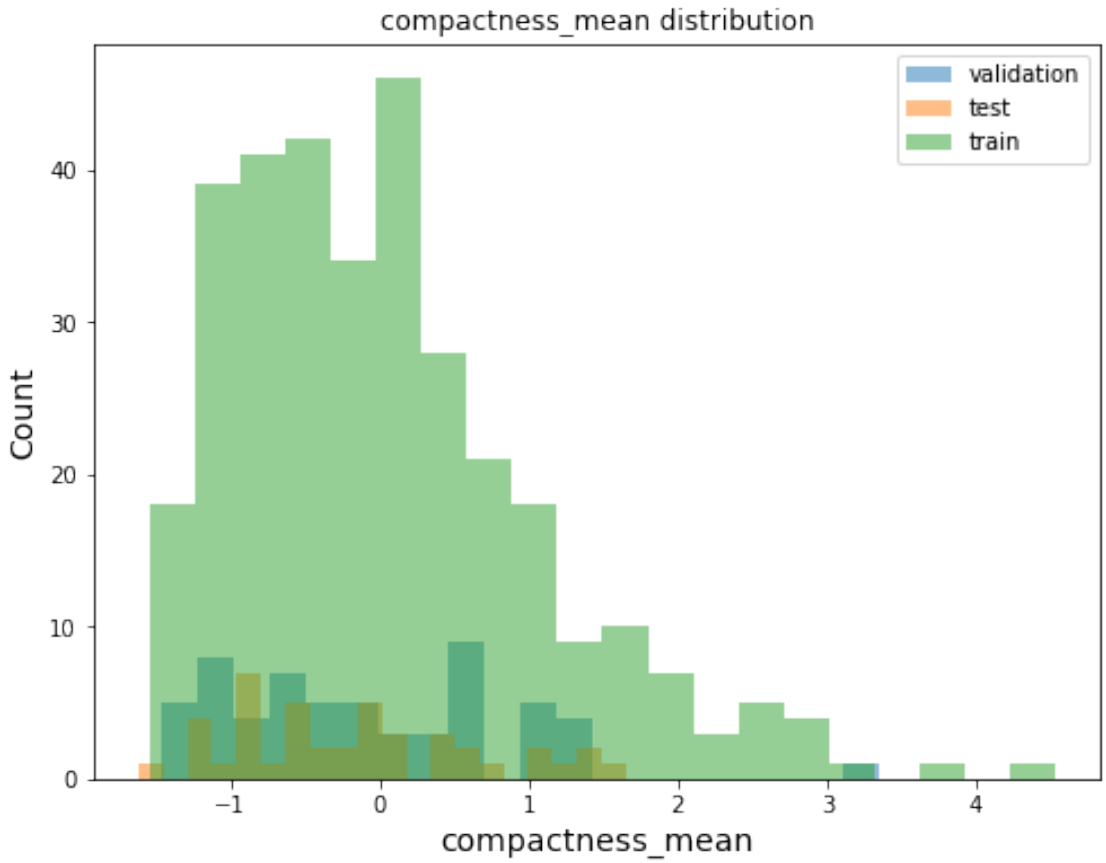
[55]:
```
from sklearn.model_selection import  train_test_split

X = df_filtered[best_cols]

y = df_filtered['diagnosis']

X_t, X_test, y_t, y_test = train_test_split(X, y, test_size=0.1, random_state=1)

X_train, X_val, y_train, y_val = train_test_split(X_t, y_t, test_size=0.15,␣
 ↪random_state=1)
```

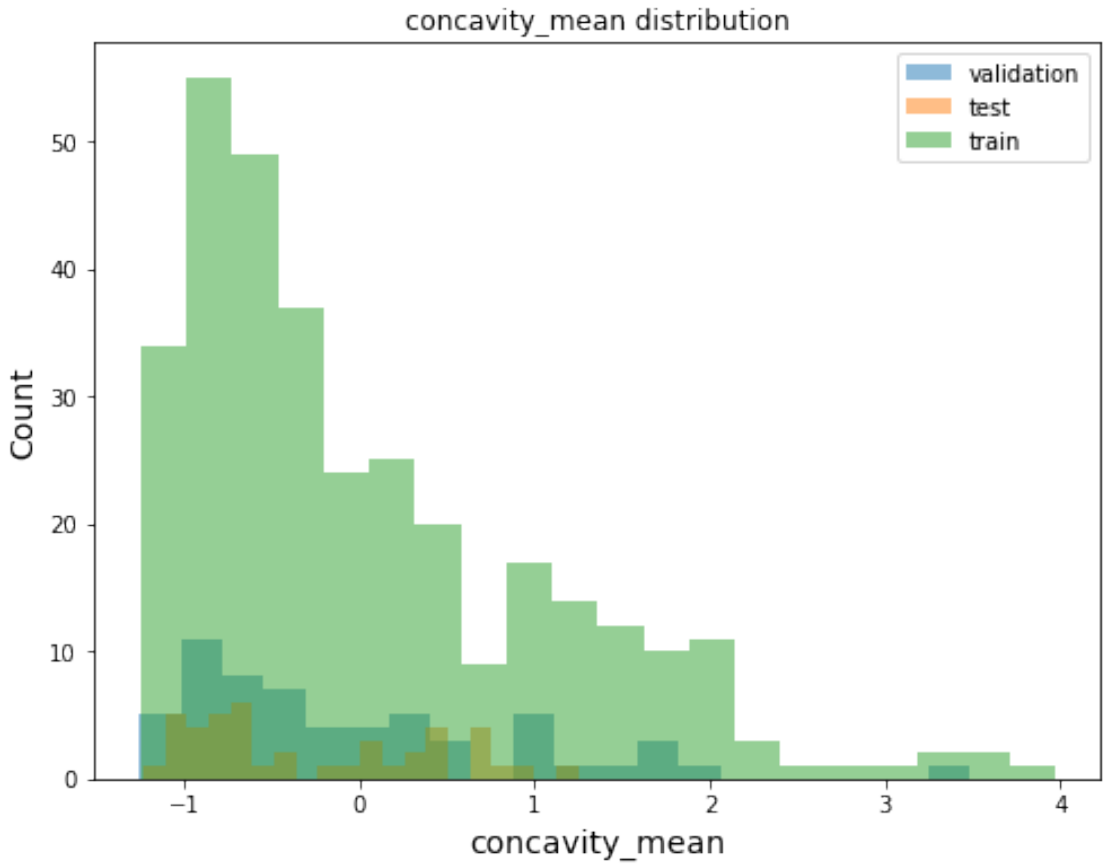
[56]:
```
for c in X_test.columns[:]:
  plt.figure(figsize=(8,6))
  plt.hist(X_val[c], bins=20, alpha=0.5, label="validation")
  plt.hist(X_test[c], bins=20, alpha=0.5, label="test")
```

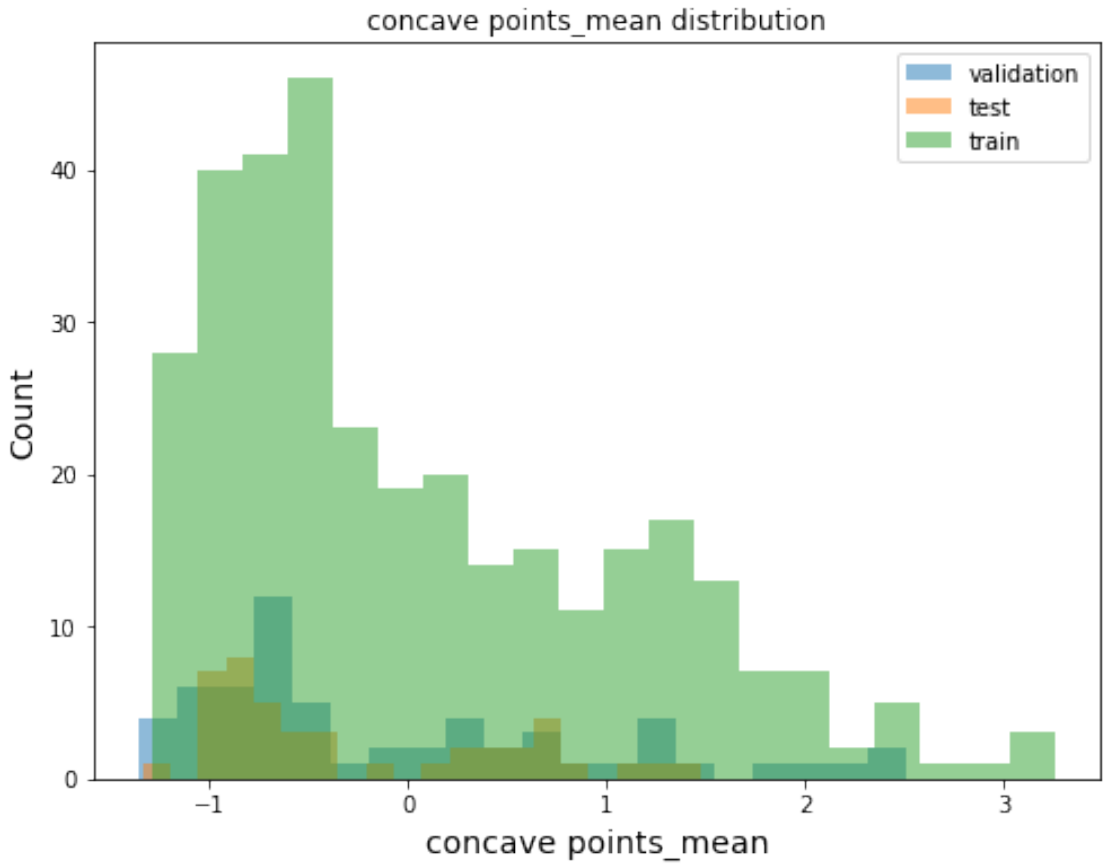
```
plt.hist(X_train[c], bins=20, alpha=0.5, label="train")
plt.xlabel(c, size=14)
plt.ylabel("Count", size=14)
plt.legend(loc='upper right')
plt.title("{} distribution".format(c))
plt.show()
```

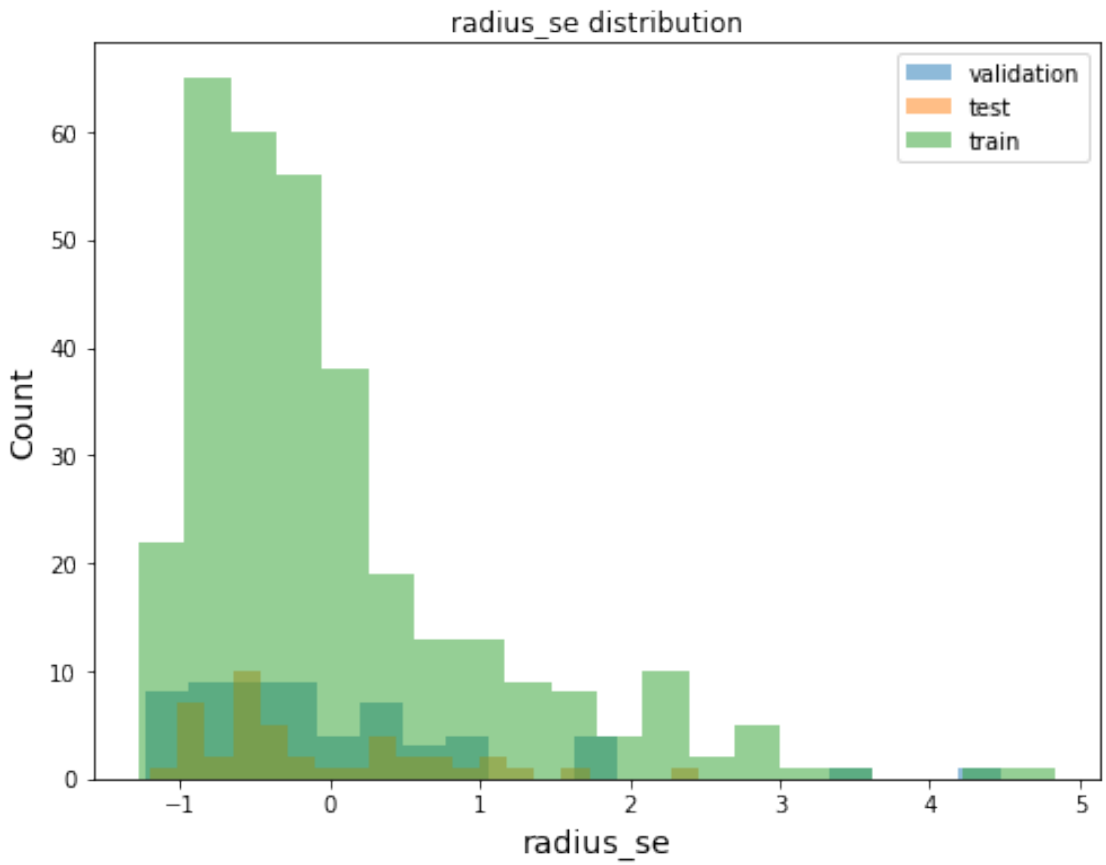


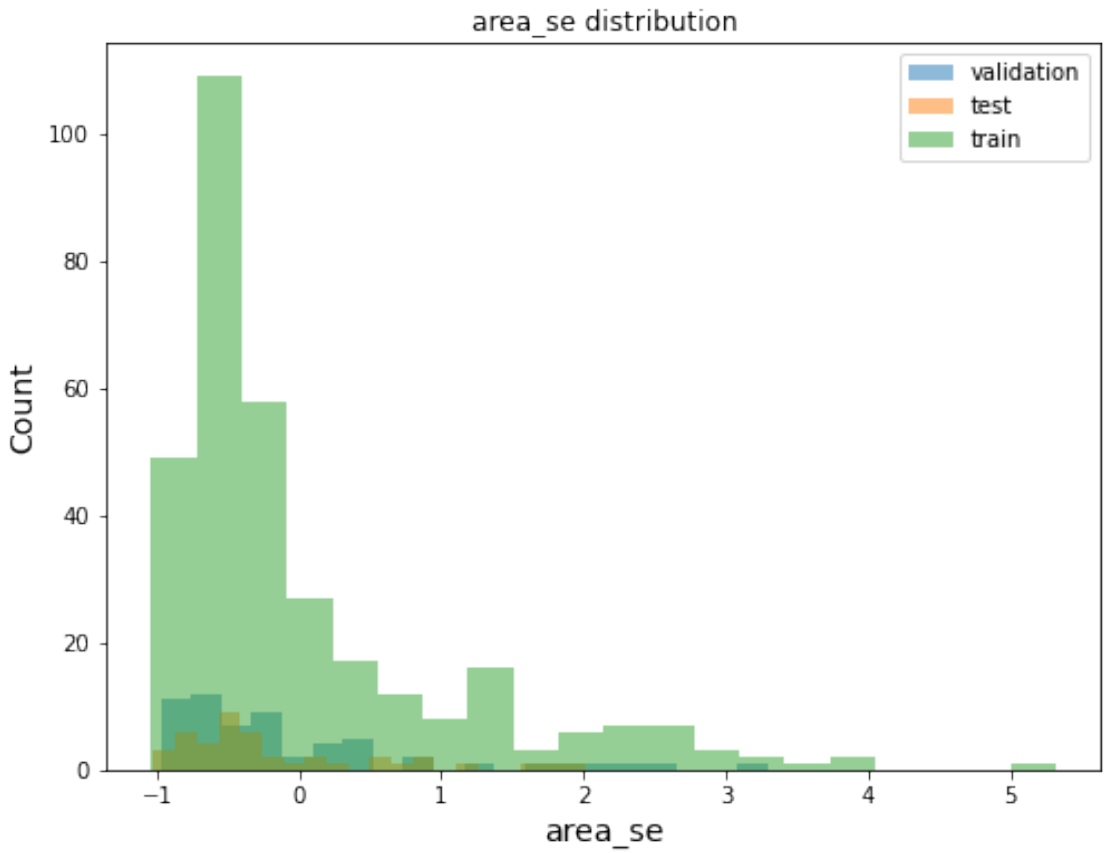

area_mean distribution
validation
test
train
Count
area_mean
0
5
10
15
20
25
30
35
40
−1
0
1
2
3

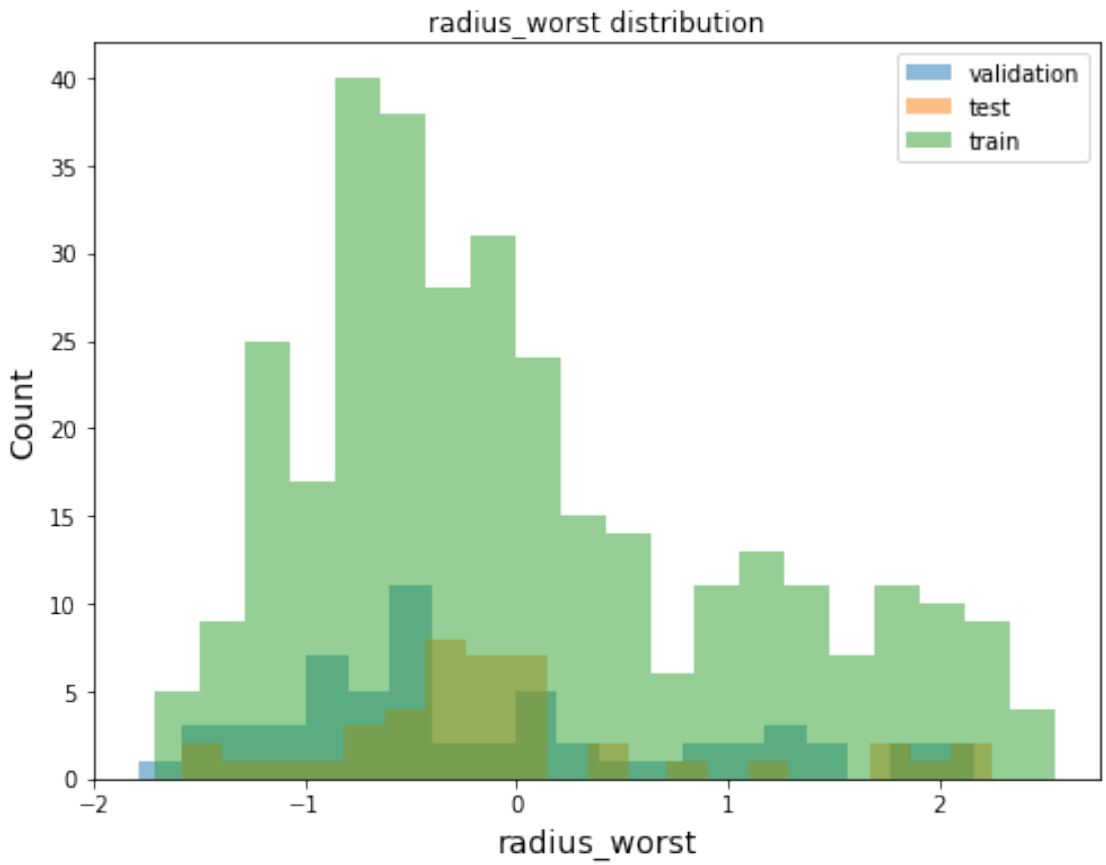

compactness_mean distribution
validation
test
train
Count
0
10
20
30
40
−1
0
1
2
3
4
compactness_mean

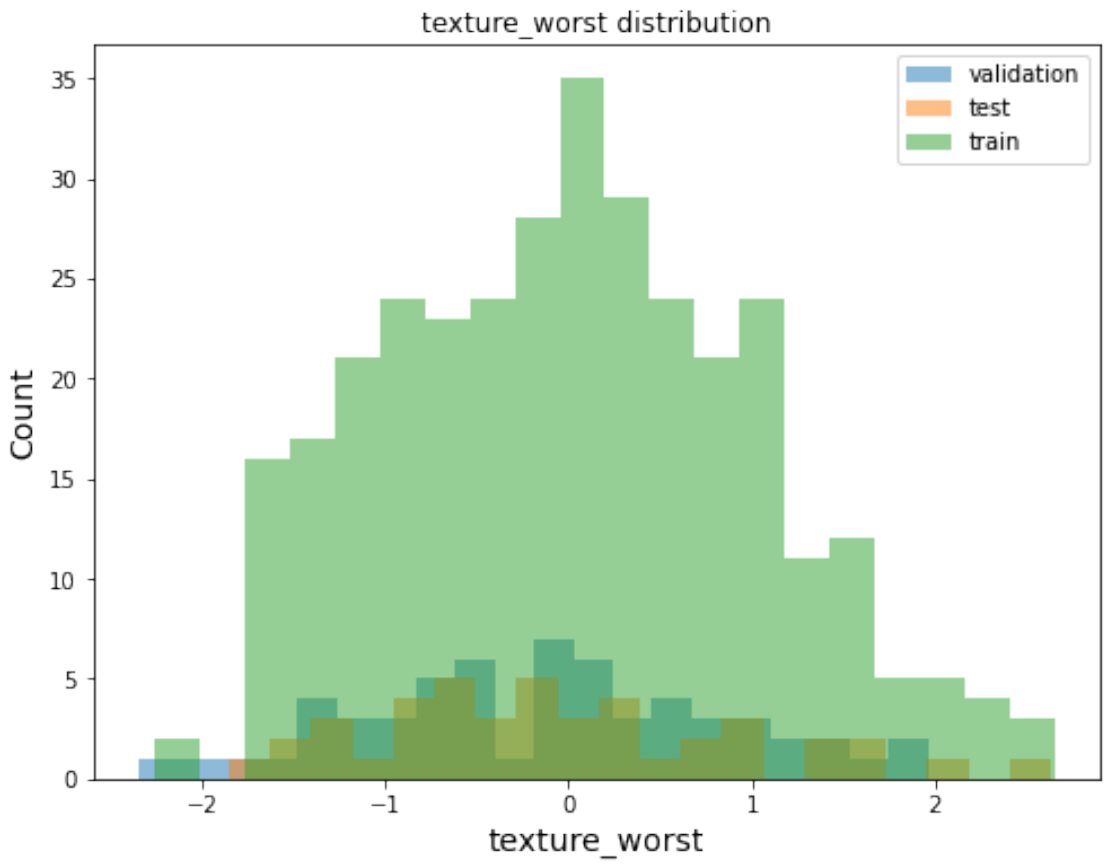

concavity_mean distribution
validation
test
train
Count
concavity_mean
0
10
20
30
40
50
−1
0
1
2
3
4

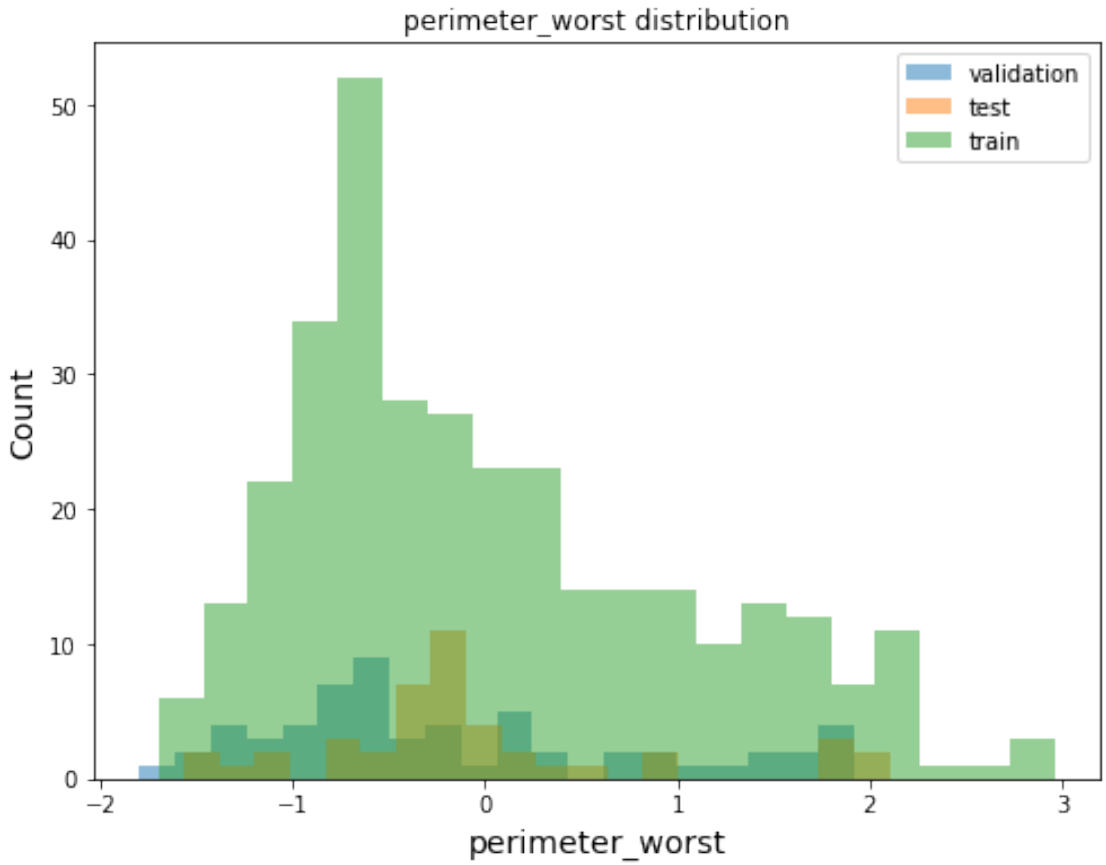

concave points_mean distribution
validation
test
train
Count
0
10
20
30
40
−1
0
1
2
3
concave points_mean

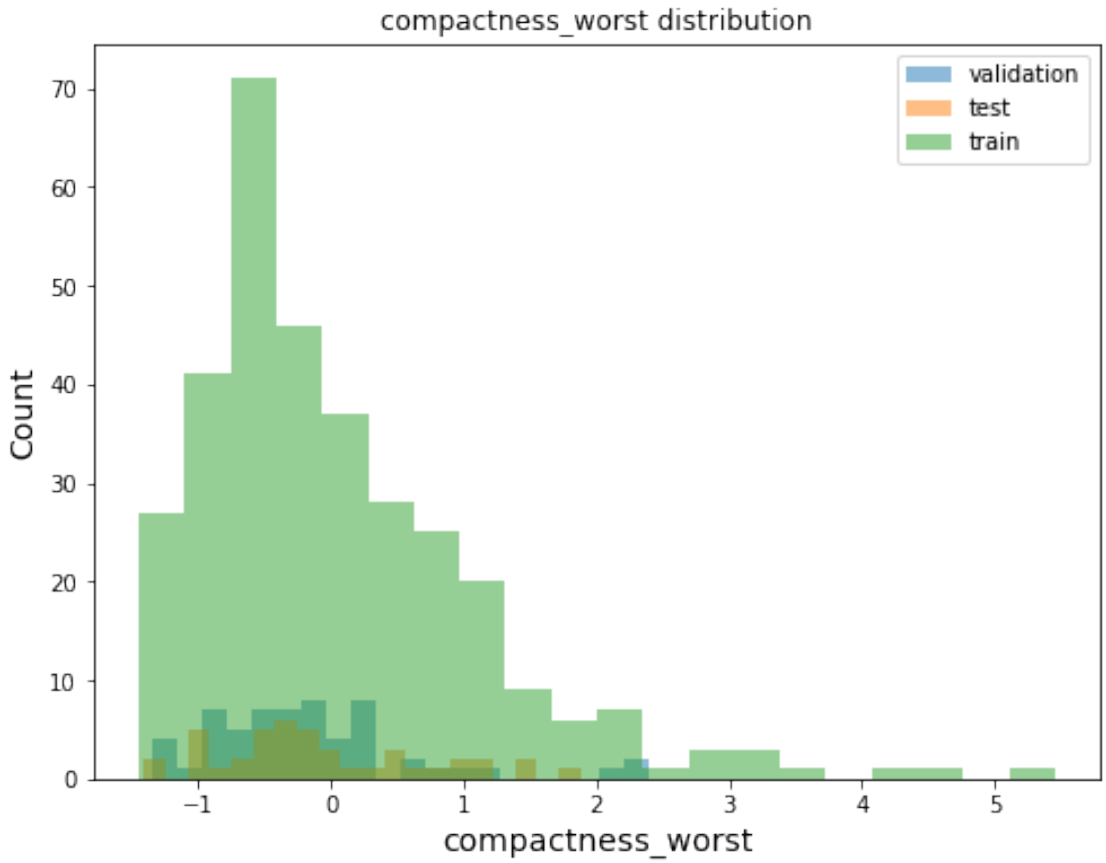

radius_se distribution
validation
test
train
Count
radius_se
0
10
20
30
40
50
60
−1
0
1
2
3
4
5

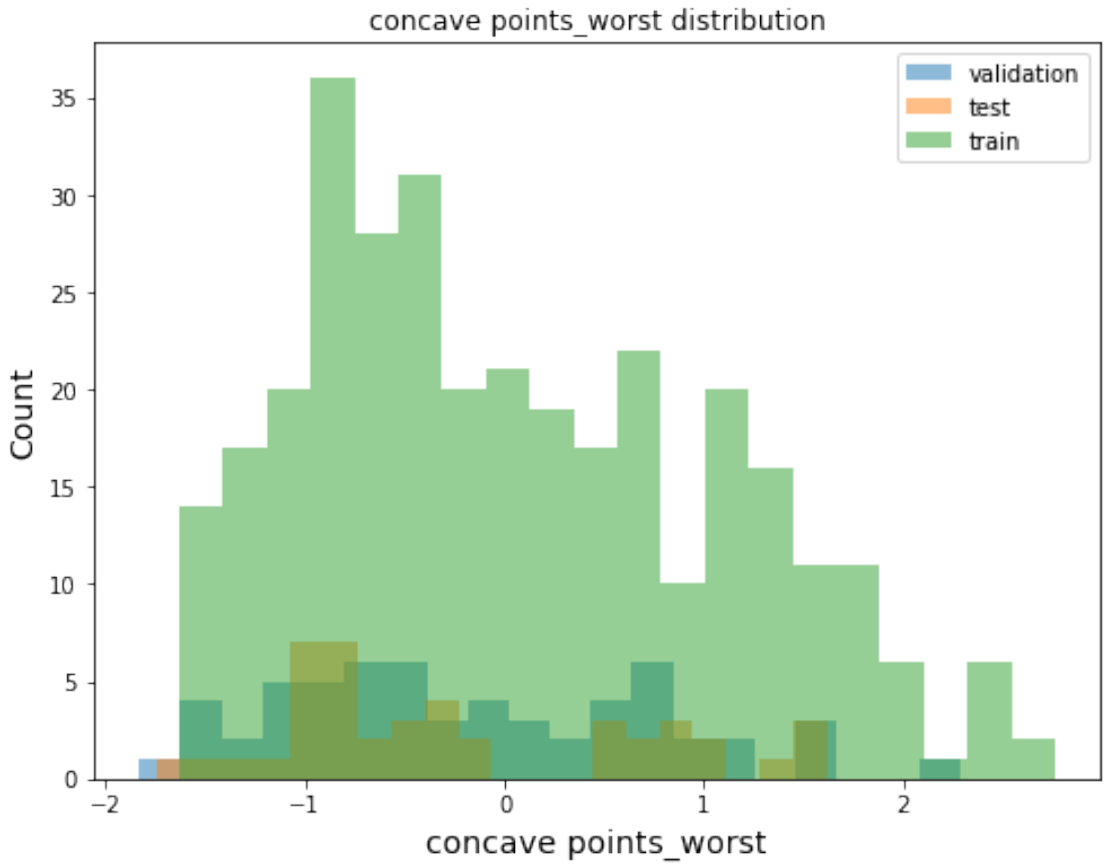

area_se distribution
validation
test
train
Count
100
80
60
40
20
0
−1
0
1
2
3
4
5
area_se


radius_worst distribution
validation
test
train
Count
radius_worst
40
35
30
25
20
15
10
5
0
−2
−1
0
1
2


texture_worst distribution
validation
test
train
Count
texture_worst
35
30
25
20
15
10
5
0
−2
−1
0
1
2


perimeter_worst distribution
validation
test
train
Count
perimeter_worst
0
10
20
30
40
50
−2
−1
0
1
2
3


compactness_worst distribution
validation
test
train
Count
compactness_worst
70
60
50
40
30
20
10
0
−1
0
1
2
3
4
5



7 Logistic Regression

[57]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import linear_model
```

[58]:
```
logisticRegr = linear_model.LogisticRegression()
logisticRegr.fit(X_train,y_train)
```

[58]: LogisticRegression()

7.1 Accuracy Score

[63]:
```
score = logisticRegr.score(X_test, y_test)
y_pred = logisticRegr.predict(X_test)
print("Accuracy of the model is "+str(score))
```

```
Accuracy of the model is 0.9302325581395349
```

7.2 Confusion Matrix

[64]:
```
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[28  2]
 [ 1 12]]
```

[ ]:
```

```