Choose One of the following:

20 Exam Questions Part B

Create 20, 5-point moderately hard exam questions. You will be graded on whether your questions are related to the topics listed on the sample exam page, their correctness, and whether or not they are trivial.

OR

Kaggle Image Dataset with at least 300 images

Join the Computational Art Lab at Northeastern - Experiments in Computational Art to get an account. If you are a student at Northeastern or a recent graduate who wants to be part of the art lab, then e-mail art@coe.northeastern.edu

See https://colab.research.google.com/drive/13Vpzk0bnsUFpzw5OMLv6LH1y4xef8oWk?usp=sharing for a prompt generating script. diffusion_bot.py

Look at "Dogs versus Cats" type datasets and notebooks. https://www.kaggle.com/datasets/shaunthesheep/microsoft-catsvsdogs-dataset

Create an image classification dataset and upload it to Kaggle along with a notebook on Kaggle classifying the images

GPT Generated Quiz Questions/Responses

Sai Kashyap Cheruku
NUID-002756849

1)
Prompt-> Generate a moderately difficult multiple choice question, with answer on the topic of regularization in machine learning.

Question: Which of the following statements about regularization in machine learning is true?

A. Regularization can only be applied to linear models.
B. L1 regularization adds a penalty term proportional to the square of the magnitude of the coefficients.
C. Ridge regression is a type of L2 regularization.
D. Regularization is only used to prevent overfitting in the training data.

Answer: C. Ridge regression is a type of L2 regularization.

Explanation: Regularization is a technique used in machine learning to prevent overfitting by adding a penalty term to the loss function. This penalty term is proportional to the magnitude of the coefficients of the model. Regularization can be applied to both linear and non-linear models.

L1 regularization adds a penalty term proportional to the absolute value of the magnitude of the coefficients. This penalty term is also known as the L1 norm or the Manhattan norm. L2 regularization, on the other hand, adds a penalty term proportional to the square of the magnitude of the coefficients. This penalty term is also known as the L2 norm or the Euclidean norm.

Ridge regression is a linear regression technique that adds an L2 regularization penalty term to the loss function. This penalty term helps to shrink the magnitude of the coefficients towards zero, which helps to reduce the variance in the model and prevent overfitting. Therefore, statement C is true, and the correct answer is C.

2)
Prompt-> Generate a moderately difficult multiple choice question, with answer on the topic of bias and variance tradeoff in machine learning models

Question: Suppose you have a dataset with a large number of features, and you want to train a model to predict a target variable based on these features. Which of the following actions is most likely to improve the performance of your model, in terms of reducing the Bias-Variance Tradeoff?

A) Adding more features to the model
B) Increasing the regularization penalty on the model's parameters
C) Decreasing the amount of training data used to fit the model
D) Decreasing the learning rate used in the model's optimization algorithm

Answer: B) Increasing the regularization penalty on the model's parameters

Explanation: When dealing with a large number of features, there is a risk of overfitting the model to the training data, resulting in high variance and poor generalization to new data. Regularization techniques, such as L1 or L2 regularization, can help to reduce overfitting by adding a penalty term to the model's objective function that discourages large parameter values. This can help to balance the

Bias-Variance Tradeoff by reducing variance without increasing bias too much. Adding more features to the model, decreasing the amount of training data used, or decreasing the learning rate used in the optimization algorithm are less likely to improve the performance of the model in this context.

3)
Prompt-> Generate a moderately difficult question the topic of Encoding in Machine Learning.
Question:Which of the following is NOT an encoding method in machine learning?
A. One-Hot Encoding
B. Label Encoding
C. Target Encoding
D. Decision Tree Encoding
Answer: D. Decision Tree Encoding
Encoding methods are used to convert categorical data into numerical data that can be used in machine learning models. One-hot encoding, label encoding, and target encoding are examples of encoding methods. One-hot encoding creates a binary column for each category in the dataset. Label encoding assigns a numerical label to each category in the dataset. Target encoding uses the target variable to assign a numerical value to each category in the dataset. Decision tree encoding is not an encoding method but rather a tree-based model used in machine learning.

4)
Prompt-> Generate a moderately diffuclt question on the topic of resampling and cross-validation methods in Machine Learning.
Which of the following techniques is used to estimate the performance of a machine learning model on an independent dataset?
A. Resampling
B. Bootstrapping
C. Cross-validation
D. Encoding
Answer: C. Cross-validation
Explanation:Cross-validation is a technique used to estimate the performance of a machine learning model on an independent dataset. It involves partitioning the data into k-folds, training the model on k-1 folds and validating it on the remaining fold. This process is repeated k times with different folds used for validation each time. Resampling and bootstrapping are techniques used for creating multiple datasets from a single dataset. Resampling involves randomly sampling from the dataset with replacement to create multiple datasets. Bootstrapping is a resampling technique that involves creating multiple datasets of the same size as the original dataset by sampling from it with replacement.

5)
Prompt-> Generate a difficult multiple choice question with answers on the topic of Linear regression in Machine Learning
Question: Which of the following assumptions must hold true in order for linear regression to be a valid model for a given dataset?

A. The relationship between the independent and dependent variables is linear.
B. The residuals (the difference between the predicted values and the actual values) are normally distributed.
C. There is no multicollinearity among the independent variables.
D. All of the above.

Answer: D. All of the above.

Explanation: Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. In order for linear regression to be a valid model for a given dataset, several assumptions must hold true:

A. The relationship between the independent and dependent variables is linear. This means that the change in the dependent variable is directly proportional to the change in the independent variable(s).

B. The residuals (the difference between the predicted values and the actual values) are normally distributed. This means that the errors are random and follow a normal distribution.
C. There is no multicollinearity among the independent variables. This means that there is no strong correlation between the independent variables.

If any of these assumptions are violated, the model may not be appropriate for the data and the results may be inaccurate. Therefore, option D is the correct answer, as all of the above assumptions must hold true for linear regression to be a valid model for a given dataset.

6)
Prompt-> Generate a moderately difficult question, with answers on the topic of Boosting Machines in Machine Learning.
Question: Which of the following statements regarding boosting is FALSE?
A) Boosting is an ensemble learning technique that combines several weak learners into a strong learner.
B) In boosting, each weak learner focuses on instances that the previous learners misclassified or predicted poorly.
C) The AdaBoost algorithm assigns equal weights to each instance in the training set at the beginning of the boosting process.
D) Gradient boosting is a boosting algorithm that uses gradient descent to minimize a loss function.

Answer: C) The AdaBoost algorithm assigns equal weights to each instance in the training set at the beginning of the boosting process.
Explanation: This statement is false because in AdaBoost, initially, each instance is assigned a weight that is inversely proportional to the number of instances in the same class. As the boosting process proceeds, the weights are updated based on the performance of the weak learners.

7)
Prompt-> Generate a difficult multiple choice question with answers on the topic of neural networks, CNNs in general.
Question: What is the difference between a convolutional neural network (CNN) and a recurrent neural network (RNN)?

A) CNNs are better suited for sequential data, while RNNs are better suited for image data.
B) CNNs use convolutional layers to extract spatial features, while RNNs use recurrent layers to capture temporal dependencies.
C) CNNs use fully connected layers to combine information from different parts of the input, while RNNs use pooling layers to reduce the dimensionality of the input.
D) CNNs and RNNs are essentially the same type of network, but with different layer configurations.

Answer: B) CNNs use convolutional layers to extract spatial features, while RNNs use recurrent layers to capture temporal dependencies.

Explanation: CNNs and RNNs are both types of neural networks, but they are designed for different types of data. CNNs are commonly used for image and video data, where they use convolutional

layers to extract spatial features from the input. RNNs, on the other hand, are commonly used for sequential data, such as text or speech, where they use recurrent layers to capture temporal dependencies in the data. While both types of networks can be used for classification or regression tasks, they have different layer configurations that are suited to their respective types of data.

8)
Prompt-> Generate a moderately difficult multiple choice question on the topic of LDA in Machine Learning
Question: What is the difference between LDA and PCA?

A) LDA is a supervised learning algorithm, while PCA is unsupervised
B) LDA is used for dimensionality reduction, while PCA is used for feature extraction
C) LDA tries to maximize the class separation, while PCA tries to maximize the variance
D) LDA can only be used with linearly separable data, while PCA can handle non-linearly separable data

Answer: A) and C)

Explanation: LDA is a supervised learning algorithm that tries to maximize the class separation in the data, while PCA is an unsupervised learning algorithm that tries to maximize the variance. Both algorithms can be used for dimensionality reduction, but LDA is specifically designed for classification tasks. While LDA can only be used with linearly separable data, PCA can handle non-linearly separable data. However, this is often addressed by using kernel PCA, which projects the data into a higher-dimensional space where it can be linearly separable.

9)
Prompt-> Generate a multiple choice questions/response on the topic of Neural networks in deep learning.
Question: What is the purpose of the backpropagation algorithm in neural networks?

A) To compute the gradient of the loss function with respect to the network parameters
B) To initialize the weights of the network
C) To adjust the learning rate during training
D) To apply regularization to prevent overfitting

Answer: A) To compute the gradient of the loss function with respect to the network parameters.

Explanation: The backpropagation algorithm is a key component of training neural networks. It involves computing the gradient of the loss function with respect to the network parameters (i.e., the weights and biases) using the chain rule of calculus. This gradient is then used to update the parameters in the direction that minimizes the loss function. By iteratively adjusting the parameters in this way, the network can learn to make more accurate predictions on the training data.

10)
Prompt->Generate a difficult multiple choice question with multiple answers on the topic of Outlier and Leverage Analysis in Machine Learning

Question: Which of the following statements are true about outliers and leverage analysis in machine learning?

A. Outliers can have a significant impact on model performance and should always be removed from the dataset before training a model.

B. Leverage points are observations that have a low value of the independent variable and can have a large impact on the regression line.
C. Outliers can be detected using statistical measures such as the interquartile range (IQR) and z-scores.
D. Leverage analysis is used to identify influential observations that have a large impact on the regression line, regardless of whether they are outliers or not.
E. Outliers can be detected using machine learning algorithms such as Isolation Forest and Local Outlier Factor (LOF).

Options:
a) A and B
b) B and C
c) C and D
d) A and E
e) B and D

Answer: (e) B and D
Explanation: Outliers and leverage points are two important concepts in machine learning that can significantly affect model performance. Outliers are observations that are significantly different from other observations in the dataset, while leverage points are observations that have a low value of the independent variable and can have a large impact on the regression line.

Statement A is incorrect, as outliers should not always be removed from the dataset before training a model. While some outliers may be errors in the data or may have a significant impact on model performance, others may be valid observations that provide important information about the underlying relationship between variables. Therefore, it is important to carefully evaluate the impact of outliers on model performance before deciding whether to remove them from the dataset.

Statement B is true, as leverage points can have a large impact on the regression line and should be carefully considered when fitting a model.

Statement C is true, as statistical measures such as the interquartile range (IQR) and z-scores can be used to detect outliers in a dataset.

Statement D is true, as leverage analysis can be used to identify influential observations that have a large impact on the regression line, regardless of whether they are outliers or not.

Statement E is also true, as machine learning algorithms such as Isolation Forest and Local Outlier Factor (LOF) can be used to detect outliers in a dataset.

Therefore, the correct answer is option (e) B and D.

11)
Prompt-> Generate a moderately difficult question, with answers on the topic of Ensemble Methods in Machine Learning.
Question: Which of the following statements about boosting is true?

A) Boosting combines multiple weak models to create a stronger model.
B) Boosting randomly selects subsets of data for training each weak model.
C) Boosting can only be used with decision trees as weak models.
D) Boosting reduces the risk of overfitting by decreasing the complexity of the model.

Answer: A) Boosting combines multiple weak models to create a stronger model.

Explanation: Boosting is an ensemble learning technique that involves combining multiple weak models to create a stronger model. The weak models are typically decision trees, but they can be any other machine learning model as well. Boosting iteratively trains weak models on different subsets of the data, with each subsequent model focusing on the examples that the previous models got wrong. This way, the final boosted model is able to capture the patterns in the data more accurately and is less prone to overfitting.

12)
Prompt-> Generate a difficult multiple choice question, with response on the topic of bayes classifier in Machine Learning.

Question:Suppose you are working on a binary classification problem with an imbalanced dataset, where 95% of the examples belong to the negative class and 5% belong to the positive class. You decide to use a Bayes classifier to classify new examples. Which of the following statements is true regarding the Bayes classifier's performance?

A) The Bayes classifier will achieve high accuracy because it is not affected by class imbalance.
B) The Bayes classifier will perform poorly because it is not designed to handle imbalanced datasets.
C) The Bayes classifier will have high precision but low recall because it tends to predict the majority class.
D) The Bayes classifier will have high recall but low precision because it tends to predict the minority class.

Answer: C) The Bayes classifier will have high precision but low recall because it tends to predict the majority class.

Explanation:
The Bayes classifier is a probabilistic classifier that uses Bayes' theorem to calculate the probability of a new example belonging to a particular class. It assumes that all features are independent of each other, and it uses the class prior probabilities and the conditional probabilities of the features given the class to compute the posterior probabilities of the classes.

In an imbalanced dataset, where one class has a much larger number of examples than the other, the Bayes classifier tends to predict the majority class because it has a higher prior probability. In this case, since 95% of the examples belong to the negative class, the Bayes classifier will tend to predict the negative class for most new examples.

However, this does not mean that the Bayes classifier will achieve high accuracy. In fact, in an imbalanced dataset, accuracy can be a misleading metric because a classifier that always predicts the majority class can achieve high accuracy even though it is not doing a good job of classifying the minority class.

Precision and recall are more appropriate metrics for evaluating the performance of a classifier on imbalanced datasets. Precision measures the fraction of positive predictions that are correct, while recall measures the fraction of positive examples that are correctly identified by the classifier. Since the Bayes classifier tends to predict the majority class, it will have high precision (i.e., the negative predictions will be mostly correct), but low recall (i.e., it will miss most of the positive examples). Therefore, answer C is correct.

13)
Prompt-> Generate a moderately difficult multiple choice question on the topic of model interpretability in Machine Learning.

Question: Which of the following statements is true about SHAP (SHapley Additive exPlanations) analysis?

A) SHAP analysis is a global interpretability technique that provides an overall understanding of a model's behavior.
B) SHAP analysis can only be applied to linear models.
C) SHAP values represent the importance of a feature for a specific instance relative to a baseline value.
D) SHAP analysis can only be used for classification tasks.

Answer: C) SHAP values represent the importance of a feature for a specific instance relative to a baseline value.

Explanation: SHAP (SHapley Additive exPlanations) analysis is a model-agnostic technique for explaining individual predictions in machine learning. It works by assigning an importance value to each feature for a given instance, based on how much it contributes to the predicted outcome relative to a baseline value. This allows us to understand how each feature affects the model's output for a specific instance, providing valuable insights into the model's behavior. SHAP analysis is not limited to linear models and can be applied to any type of model, including tree-based models and neural networks. Finally, SHAP analysis can be used for both classification and regression tasks.

14)
Prompt-> Generate a moderately difficult multiple choice question, with answers on the topic of quadratic discriminant analysis in Machine Learning.

Question :Which of the following statements is true about quadratic discriminant analysis?

A) Quadratic discriminant analysis assumes that the covariance matrix is the same for all classes.
B) Quadratic discriminant analysis can only be used for binary classification problems.
C) Quadratic discriminant analysis is a linear classifier.
D) Quadratic discriminant analysis is more flexible than linear discriminant analysis in terms of decision boundary shape.

Answer: D) Quadratic discriminant analysis is more flexible than linear discriminant analysis in terms of decision boundary shape.

Explanation: Quadratic discriminant analysis (QDA) is a classification algorithm that makes use of a quadratic decision boundary to separate different classes in the feature space. Unlike linear discriminant analysis (LDA), QDA does not assume that the covariance matrix is the same for all classes, making it more flexible in terms of decision boundary shape. QDA can be used for both binary and multi-class classification problems, and it is a non-linear classifier due to the quadratic decision boundary.

15)
Prompt-> Generate a multiple choice question, with answer on the topic of Batch Learning in Deep Learning
Question: Which of the following statements about batch learning in deep learning is false?

A) Batch learning updates the model's parameters based on a subset of training data at a time, rather than the entire dataset.
B) Batch learning is more computationally efficient than online learning, which updates the model's parameters one sample at a time.
C) The batch size can impact the stability and convergence of the training process in batch learning.
D) In batch learning, the batch size should always be set to the entire training dataset to achieve the best performance.

Answer: D) In batch learning, the batch size should always be set to the entire training dataset to achieve the best performance.

Explanation: In batch learning, the batch size should always be set to the entire training dataset to achieve the best performance." This statement is not true because using the entire training dataset as a single batch for updating the model's parameters can lead to several issues such as high memory consumption, longer training time, and a higher risk of overfitting.

In batch learning, the training dataset is divided into smaller subsets, called batches. The model's parameters are updated based on the gradients computed on each batch of training data. Choosing the batch size is an important hyperparameter in the batch learning process, and it can significantly affect the training process's stability and convergence. A smaller batch size can make the training process more stable and converge faster, while a larger batch size can result in more accurate updates to the model's parameters.

Therefore, statement D is false because the batch size should be chosen based on the specific task, the model architecture, and available resources to achieve the best performance, but it is not always optimal to use the entire training dataset as a single batch.

16)
Prompt-> Generate a multiple choice question answer on the topic of multilayer perceptron model in deep learning

Question: Which of the following best describes the multilayer perceptron (MLP) model in deep learning?

A) A model that uses multiple layers of neurons, where each neuron is connected to every neuron in the next layer.
B) A model that uses a single layer of neurons, where each neuron is connected to every neuron in the next layer.
C) A model that uses multiple layers of neurons, where each neuron is connected only to the neurons in the same layer.
D) A model that uses a single layer of neurons, where each neuron is connected only to the neurons in the same layer.

Answer: A) A model that uses multiple layers of neurons, where each neuron is connected to every neuron in the next layer.

Explanation: The multilayer perceptron (MLP) is a type of neural network used in deep learning. It is called a "multilayer" perceptron because it contains multiple layers of artificial neurons, typically three or more, including an input layer, one or more hidden layers, and an output layer.

In an MLP model, each neuron in a given layer is connected to every neuron in the next layer. This means that information can flow forward from the input layer to the output layer, passing through each hidden layer along the way. This allows the network to learn complex patterns and relationships in the data it is trained on.

In contrast, the other answer options are not accurate descriptions of the MLP model. Option B describes a single-layer perceptron, which is a much simpler model that can only learn linearly separable patterns. Option C describes a type of neural network called a convolutional neural network, where neurons are arranged in 3D grids and connected to local regions of the input. Option D describes a type of neural network called a self-organizing map, which uses unsupervised learning to create a low-dimensional representation of high-dimensional input data.

17)
Prompt-> Generate a multiple choice question, with answers on the topic of multicollinearity in machine learning.
Question: How can multicollinearity affect the coefficients in a linear regression model?

A) It can cause the coefficients to be too large, resulting in overfitting.
B) It can cause the coefficients to be too small, resulting in underfitting.
C) It can cause the coefficients to be unstable and have high variance.
D) It has no effect on the coefficients in a linear regression model.

Answer: C) It can cause the coefficients to be unstable and have high variance.

Explanation: Multicollinearity can cause the coefficients in a linear regression model to become unstable and have high variance. This happens because when two or more input variables are highly correlated, it becomes difficult for the model to distinguish the individual effect of each variable on the target variable. As a result, small changes in the input data can cause large changes in the estimated coefficients, leading to instability and high variance.

Option A is incorrect because multicollinearity doesn't cause the coefficients to be too large. Instead, overfitting occurs when the model is too complex and fits the noise in the training data. Option B is incorrect because multicollinearity doesn't cause the coefficients to be too small. Instead, underfitting occurs when the model is too simple and can't capture the complexity in the data. Option D is incorrect because as explained, multicollinearity can have a significant effect on the coefficients in a linear regression model.

18)
Prompt-> Generate a moderately difficult question on the topic of stochastic gradient descent.
Question: How does the learning rate affect the convergence of stochastic gradient descent?

A) A high learning rate can cause the algorithm to converge faster, but it may overshoot the minimum.
B) A low learning rate can cause the algorithm to converge slower, but it will be more accurate in reaching the minimum.
C) A learning rate that is too high can cause the algorithm to oscillate around the minimum.
D) All of the above.

Answer: D) All of the above.

Explanation: The learning rate is a hyperparameter that controls the step size of the updates made to the model parameters during stochastic gradient descent. A high learning rate can cause the algorithm to converge faster, but it may overshoot the minimum and oscillate around it, leading to

instability. A low learning rate can cause the algorithm to converge slower, but it will be more accurate in reaching the minimum. Therefore, the optimal learning rate depends on the specific problem and dataset.
Option D is correct because all of the statements are true regarding the effect of learning rate on the convergence of stochastic gradient descent.

19)
Prompt-> Generate a difficult multiple choice question, with answers on the topic of residual analysis in Machine Learning.
Question: Which of the following statements about residual analysis in machine learning is true?

A) Residuals are used to assess the goodness-of-fit of a model
B) A residual plot with no discernible pattern indicates that the model fits the data well
C) The normality of residuals is important because it ensures that the model is unbiased
D) Residual plots should be used to detect outliers in the data

Possible answers:
A) Only A is true
B) Only B is true
C) Only C is true
D) Only D is true
E) Both A and C are true
F) Both B and D are true

Answer: E) Both A and C are true

Explanation:
A) Residuals are indeed used to assess the goodness-of-fit of a model, by measuring the difference between the predicted and actual values of the response variable.
C) The normality of residuals is also important, as it ensures that the model is unbiased and that the errors are randomly distributed. If the residuals are not normally distributed, it may indicate that the model is misspecified and needs to be improved.

B) This statement is incorrect because a residual plot with no discernible pattern does not necessarily indicate that the model fits the data well. It could simply mean that the pattern is too subtle to detect, or that there are other underlying factors that are not captured by the model.

D) This statement is also incorrect because residual plots are not the best tool for detecting outliers in the data. Outliers should be identified and handled separately from residual analysis, as they can have a significant impact on the model's performance.

20)
Prompt-> Generate a moderately difficult multiple choice question, with answers on the topic of decision tree model in machine learning.
Question: What is the main advantage of using the Gini index over entropy in a decision tree model?

A) The Gini index is computationally more efficient than entropy.
B) The Gini index is more sensitive to changes in the distribution of class labels.
C) The Gini index is better at handling imbalanced datasets than entropy.
D) The Gini index results in smaller trees compared to entropy.

Answer: C) The Gini index is better at handling imbalanced datasets than entropy.

Explanation: While both Gini index and entropy are commonly used metrics to determine the best split at each node of a decision tree, the Gini index is known to work better with imbalanced datasets. This is because the Gini index puts more emphasis on larger class labels and can result in a more balanced split of the data. However, it should be noted that the Gini index is not always the best metric to use and it ultimately depends on the specific problem at hand.

Take the assignment "Combine Data Cleaning, Feature Selection, Modeling, and Interpretability" and create a Medium article from it or take other work and publish it on Medium (https://medium.com).

The article can have multiple authors but must follow standard Authorship in Scholarly or Scientific Publications. If the article uses notebooks and guidance from this class then Nik Bear Brown should be the last author and contributing authors are listed in order of their contribution in order from first. You can choose to upload an article and not publish it but you are encouraged to publish. If you actually publish on Medium then please also add a reference to the article on the AI Skunks "What We Have Done?" page https://github.com/aiskunks/aiskunks/tree/main/AISkunks/What_We_Have_Done

Guidance on Authorship in Scholarly or Scientific Publications

Advisors and the professor are the last authors of articles and publications.

Contributing authors are listed in order of their contribution in order from first. If an author's contribution is not deemed significant then they are listed in acknowledgments rather than as an author.

An author can choose not to be listed as an author. All authors must agree on the licensing of text, code, and images.

See Guidance on Authorship in Scholarly or Scientific Publications https://provost.yale.edu/policies/academic-integrity/guidance-authorship-scholarly-or-scientific-publications

Tumor Diagnosis by Applied Machine Learning

Sai Kashyap · Follow

5 min read · Apr 27

Listen Share

Authors: Sai Kashyap, Nik Bear Brown

Fine Needle Analysis (FNA) is a minimal invasive biopsy procedure that involves collecting breast tissue or fluid, used to detect the presence of cancerous cells. This article 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 and Leverage Analysis
- Fit Machine Learning models to the data and study their performances
- Examine Feature importance and Feature Selection for each of these models.

Finally, we include the usage of Automated Machine Learning APIs such as AutoML to gain a lucid understanding of the ML process and outcomes.

About the 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 the mean, standard error, and mean of the 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 the *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, the inclusion of which can be ignored in our study.

Exploratory Data Analysis

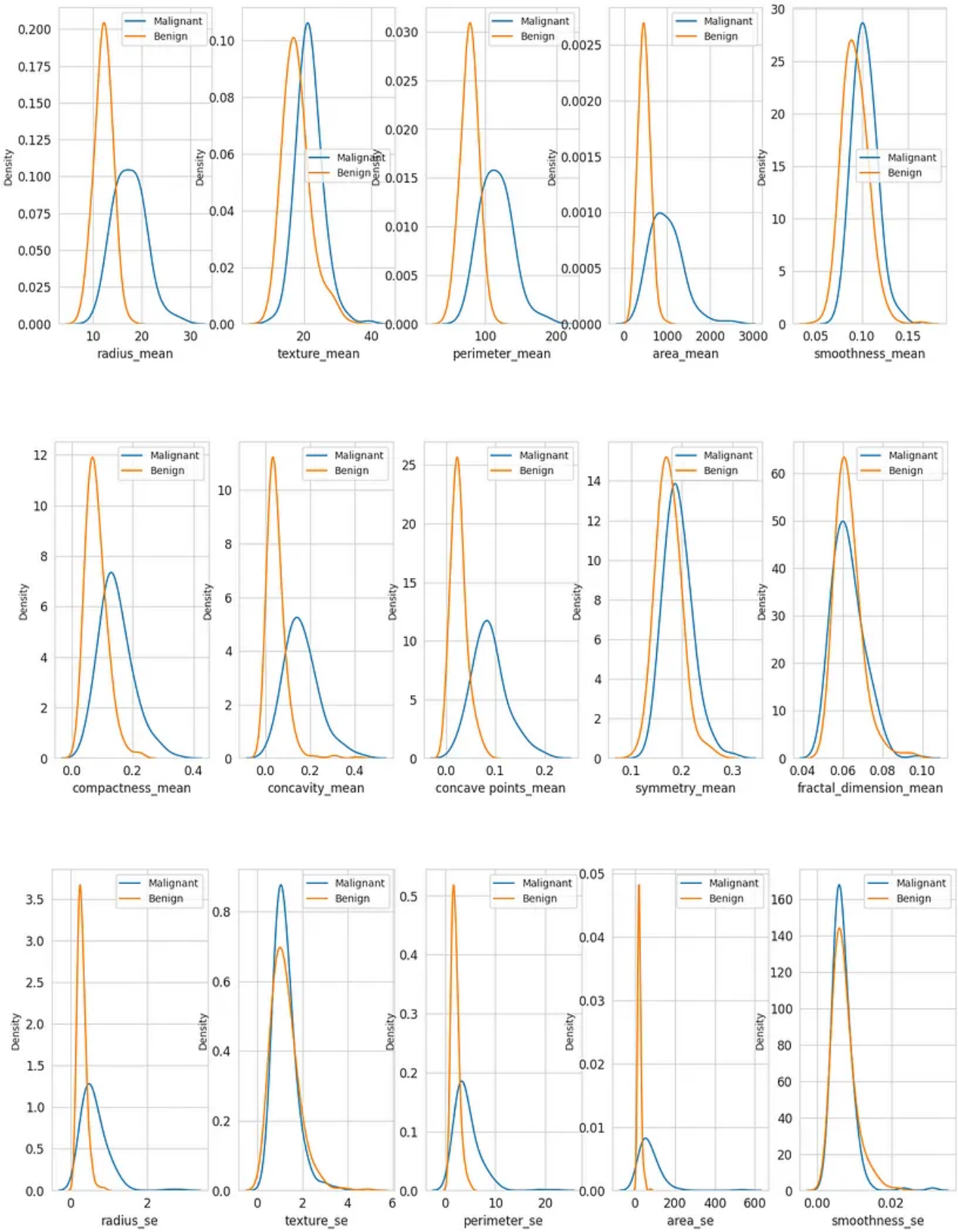
Description of some statistical features pertaining to the dataset

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	radius_worst	texture_worst	perimeter_worst	area_worst
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.000000	569.000000	569.000000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	...	16.269190	25.677223	107.261213	880.583128
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	...	4.833242	6.146258	33.602542	569.356993
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	...	7.930000	12.020000	50.410000	185.200000
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	...	13.010000	21.080000	84.110000	515.300000
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	...	14.970000	25.410000	97.660000	686.500000
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	...	18.790000	29.720000	125.400000	1084.000000
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	...	36.040000	49.540000	251.200000	4254.000000

Observations:

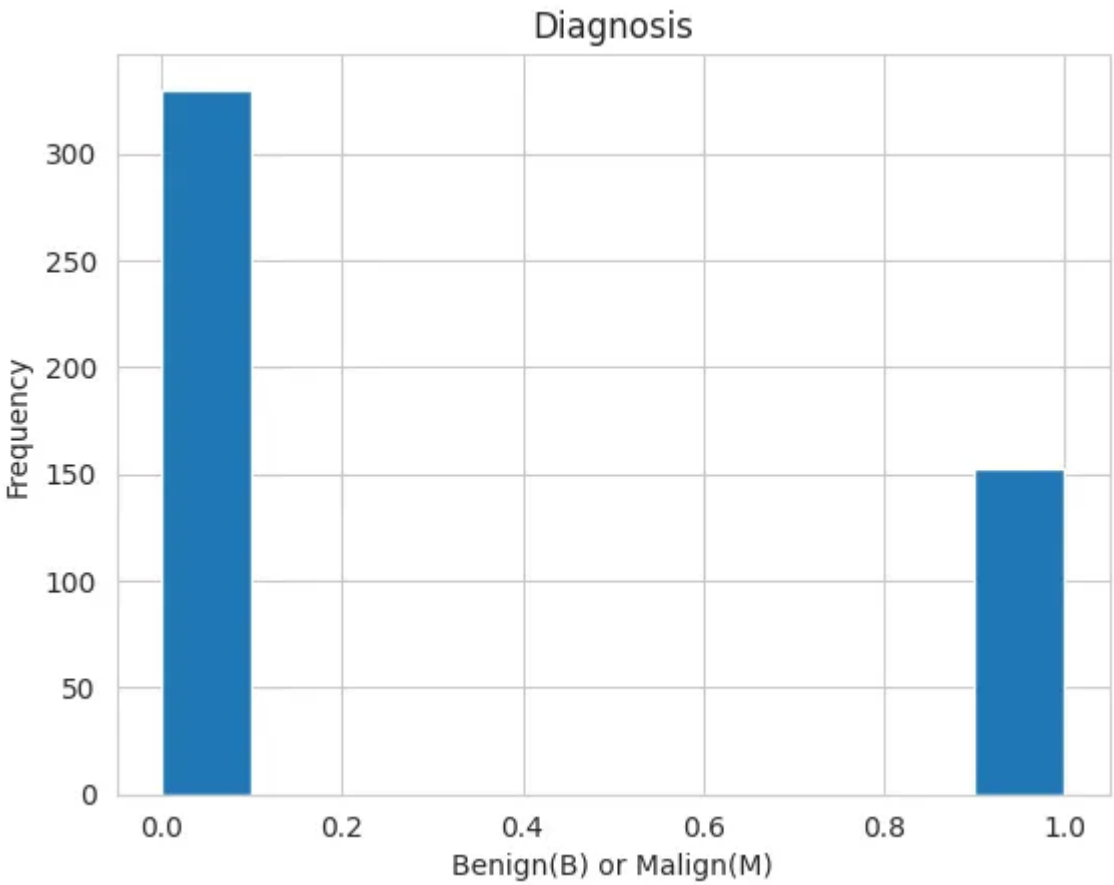
- The ranges of the independent variable is varying significantly.(A scaling operation should be performed, depending on the chosen model to fit)
- Some of the feature values are labeled as 0. (This may or may not be a regular occurrence)

Feature Distributions



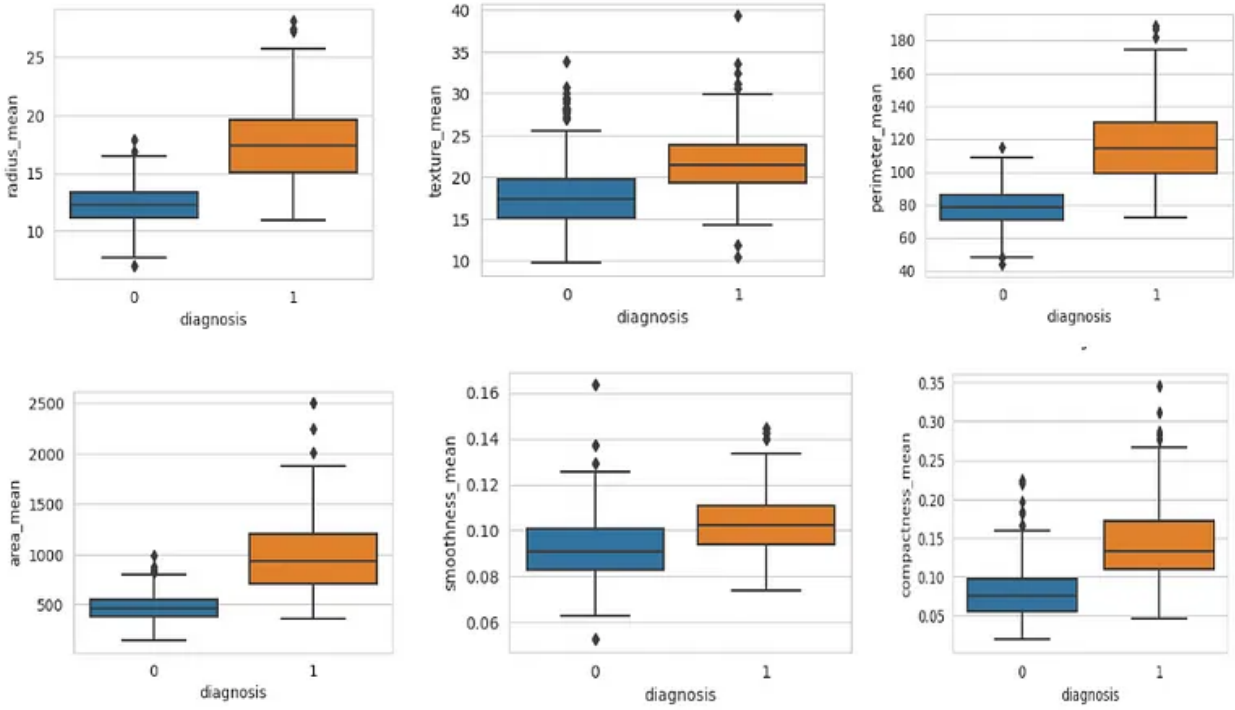
By observing the above kde plots, it is inferred that most of the predictor variables have distributions that are overlapping. The relationship between each of the predictor and target is not evident immediately, hence requiring further analysis.

histogram plot for dependent feature



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

Boxplot to show distributions of the target for different predictors



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 variants which exhibit values on par with or greater than malignant variants.

Anomaly Detection

IsolationForest

A method called “Isolation Forest” is employed to find outliers. The method builds a tree-based structure and measures an observation’s level of abnormality by counting how many splits are necessary to isolate it.

```

from sklearn.ensemble import IsolationForest
rs=np.random.RandomState(100)
clf = IsolationForest(max_samples=100,random_state=rs, contamination=.15)
clf.fit(df)
anomalies=clf.predict(df)
mask=anomalies!=-1
df_array=df.to_numpy()[mask,:]
target_array=df_target.to_numpy()[mask]
df=pd.DataFrame(df_array,columns=df.columns)
df_target=pd.DataFrame(target_array,columns=['diagnosis'])
```

Anomaly Detection hosted with ❤ by GitHub view raw

Collinearity

Variance Inflation Factor

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_data = pd.DataFrame()
dataset=pd.concat([df,df_target],axis=1)
vif_data["feature"] = dataset.columns
vif_data["VIF"] = [variance_inflation_factor(dataset.values, i)
                          for i in range(len(dataset.columns))]
mc_features=[]
for a in zip(vif_data['feature'],vif_data['VIF']):
  if a[1]>10:
    mc_features.append(a[0])
```

VIF hosted with ❤ by GitHub view raw

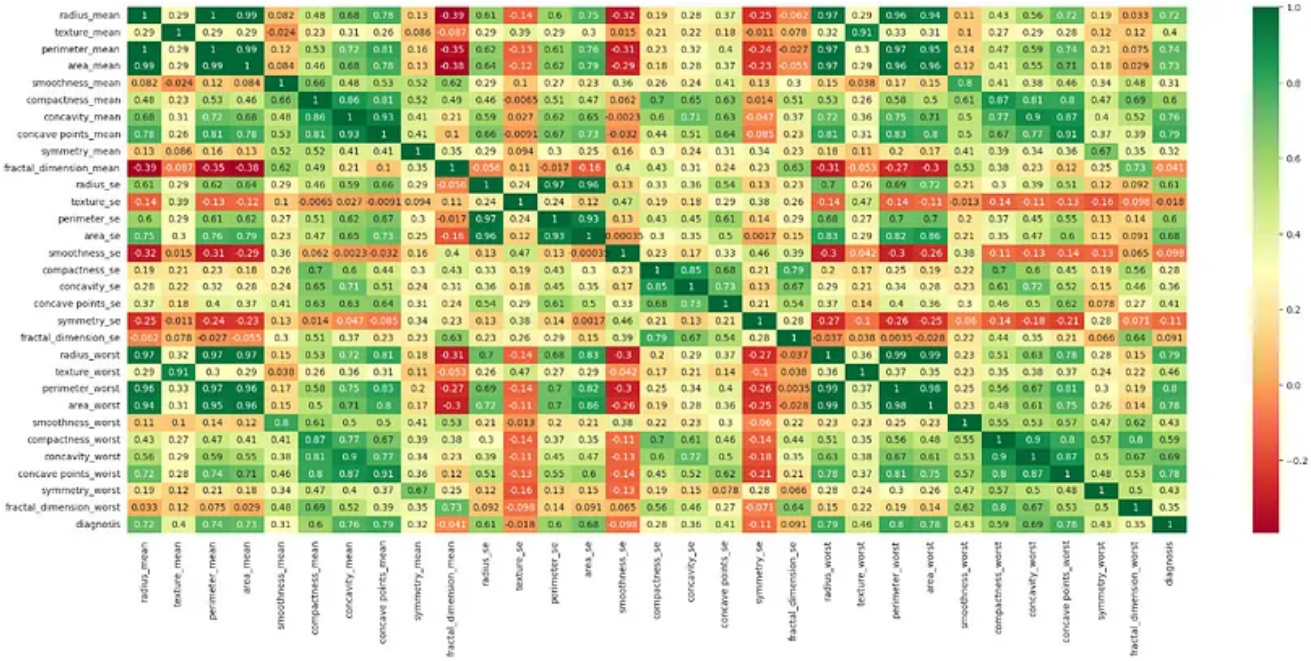
feature	VIF
radius_mean	131777.082403
texture_mean	328.767555
perimeter_mean	114961.546142
area_mean	2204.802594
smoothness_mean	565.497988
compactness_mean	233.662579
concavity_mean	151.605885
concave points_mean	148.635119
symmetry_mean	219.943532
fractal_dimension_mean	891.102647
radius_se	529.068194
texture_se	46.061264

The VIF values show a great number of features exhibiting multicollinearity. The features with high VIF values may be ignored while performing regression analysis, such as in logistic regression.

Correlation

The relationship between features and the target is crucial because it reveals which features are most useful for forecasting the target variable. This is because a feature

with a strong correlation to the target variable would hold crucial information that can be used to precisely predict the target variable.



Feature Significance and Model Fit

Feature significance is arrived at by using feature selection methods. The interpretation of regression coefficients to arrive at feature importance is avoided due to the high collinearity of features.

```
from sklearn.feature_selection import SelectKBest,chi2
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
scaler=MinMaxScaler()
df_scaled=scaler.fit_transform(df)
selector=SelectKBest(score_func=chi2,k=25)
selected=selector.fit_transform(df_scaled,df_target)
support=selector.get_support()
selected_feat=list(df.loc[:,support].columns)
print(selected_feat)
```

KBest FS hosted with ❤ by GitHub view raw

```
['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mear
```

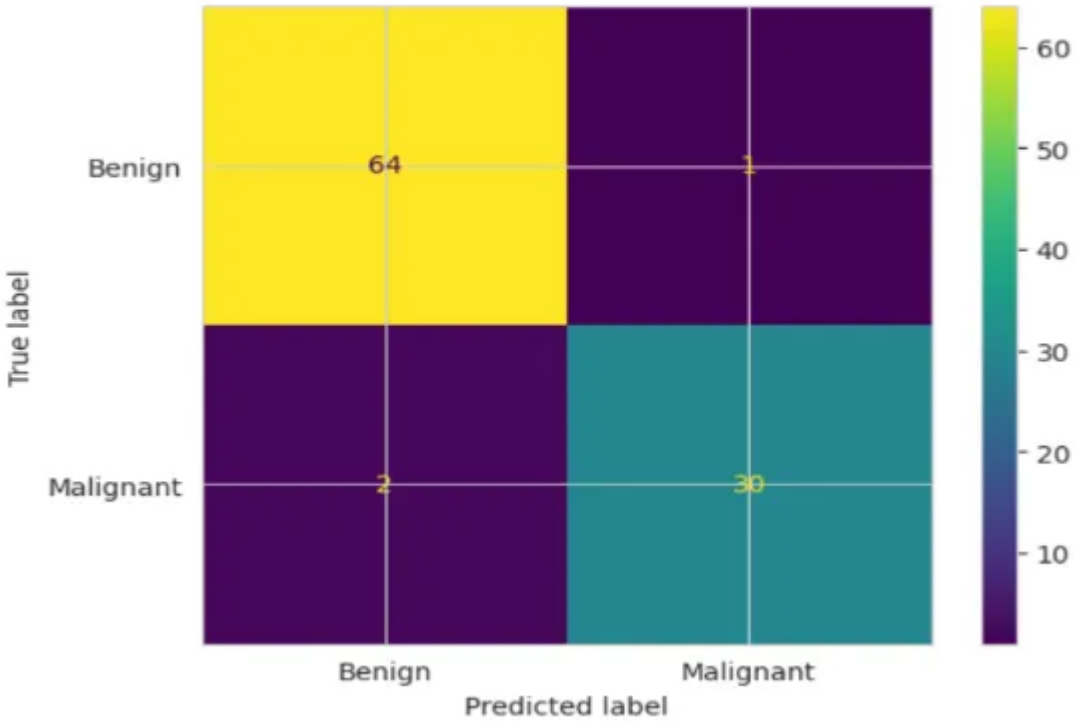
The most important features, computed by using the KBest feature selection method for a random forest model can be observed above.

KBest feature selection is a machine learning technique that is used to pick the most crucial features from a dataset. Using a specified statistical metric, such as mutual information, the ANOVA F-test, or the chi-squared test, the top K features with the highest scores are chosen. The user chooses the K value, which can either be assigned to a specific integer or selected through cross-validation. By deleting unnecessary or redundant features, this method can be used to decrease the dimensionality of a dataset and enhance model performance.

```
df_best=df[selected_feat]
X_train,X_test,y_train,y_test=train_test_split(df_best,df_target,test_size=0.2,random_state=100
train_df_best=pd.concat([X_train,y_train],axis=1)
test_df_best=pd.concat([X_test,y_test],axis=1)

X_train.reset_index(inplace=True,drop=True)
y_train.reset_index(inplace=True,drop=True)
X_test.reset_index(inplace=True,drop=True)
y_test.reset_index(inplace=True,drop=True)
# testing the performance of the KBest features fit to the Random Forest model
rf_fit=rf.fit(X_train,y_train)
y_pred=rf_fit.predict(X_test)
pred_df_best=pd.DataFrame(y_pred,columns=['diagnosis'])
metric_scores(y_test,y_pred)
cf_display(y_test,y_pred)
```

Train and Metric hosted with ❤ by GitHub view raw

```
Accuracy: 0.9690721649484536
Precision:  0.967741935483871
Recall  0.9375
```



AutoML

AutoML is the process of automating the time-intensive practices of machine learning and predictive modeling. An AutoML tool aids the user by automating the 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.

```
#initializing the h2o cluster
h2o.init()
# Train-test split of the h2o frame
df=h2o.H2OFrame(df_initial)
df['diagnosis']=df['diagnosis'].asfactor()
for a in df.columns:
  df[a]=df[a].asnumeric()

train, test = df.split_frame(ratios=[0.8], seed = 1)
#Initializing the H2OAutoML object with necessary hyperparameters and performing the training o

aml = H2OAutoML(max_models =25,exclude_algos=["DeepLearning"],sort_metric='auc',
                seed =1)

train['diagnosis']=train['diagnosis'].asfactor()
aml.train(training_frame=train,y='diagnosis')
#leaderboard visualization
leaders=aml.leaderboard
leaders.head(rows=leaders.nrows)
aml.leader.model_performance(test_data=test)
explain_model = aml.explain(frame = test, figsize = (8,6))
```

h2o hosted with ❤ by GitHub view raw

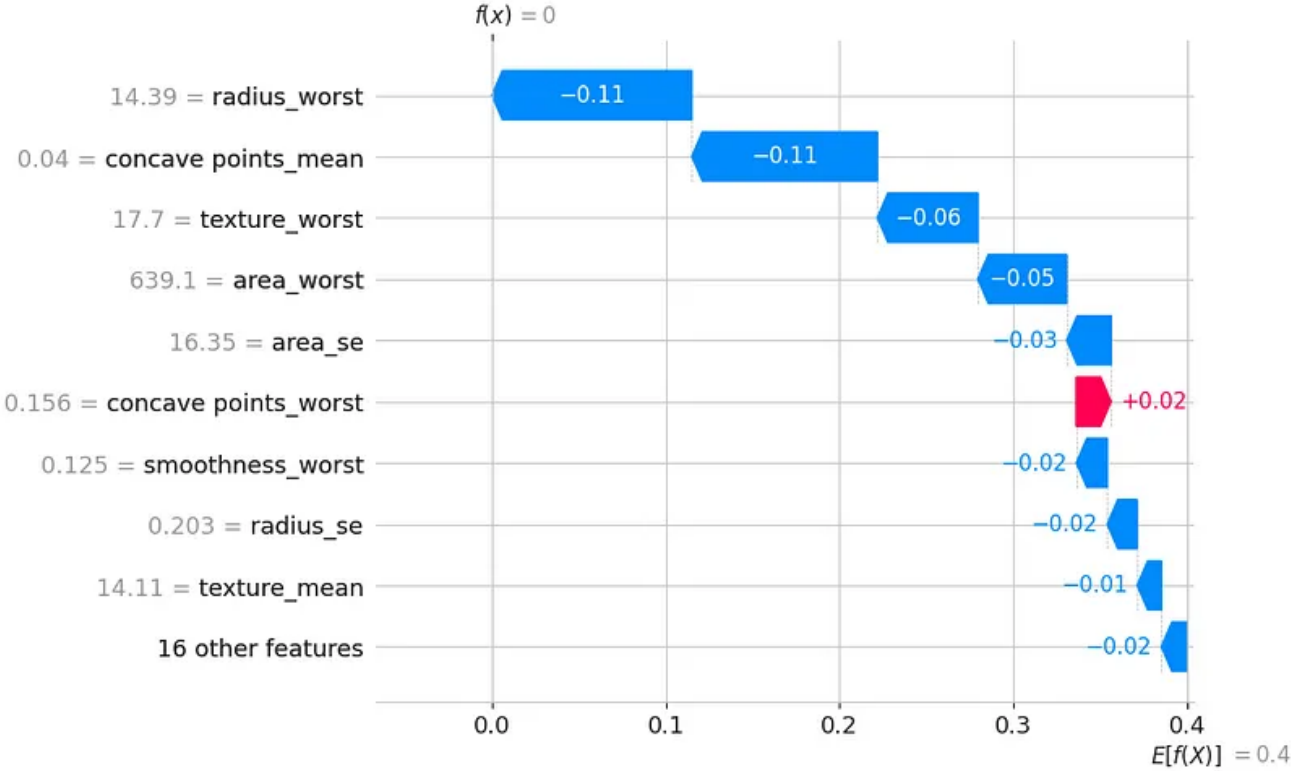
Model Interpretability

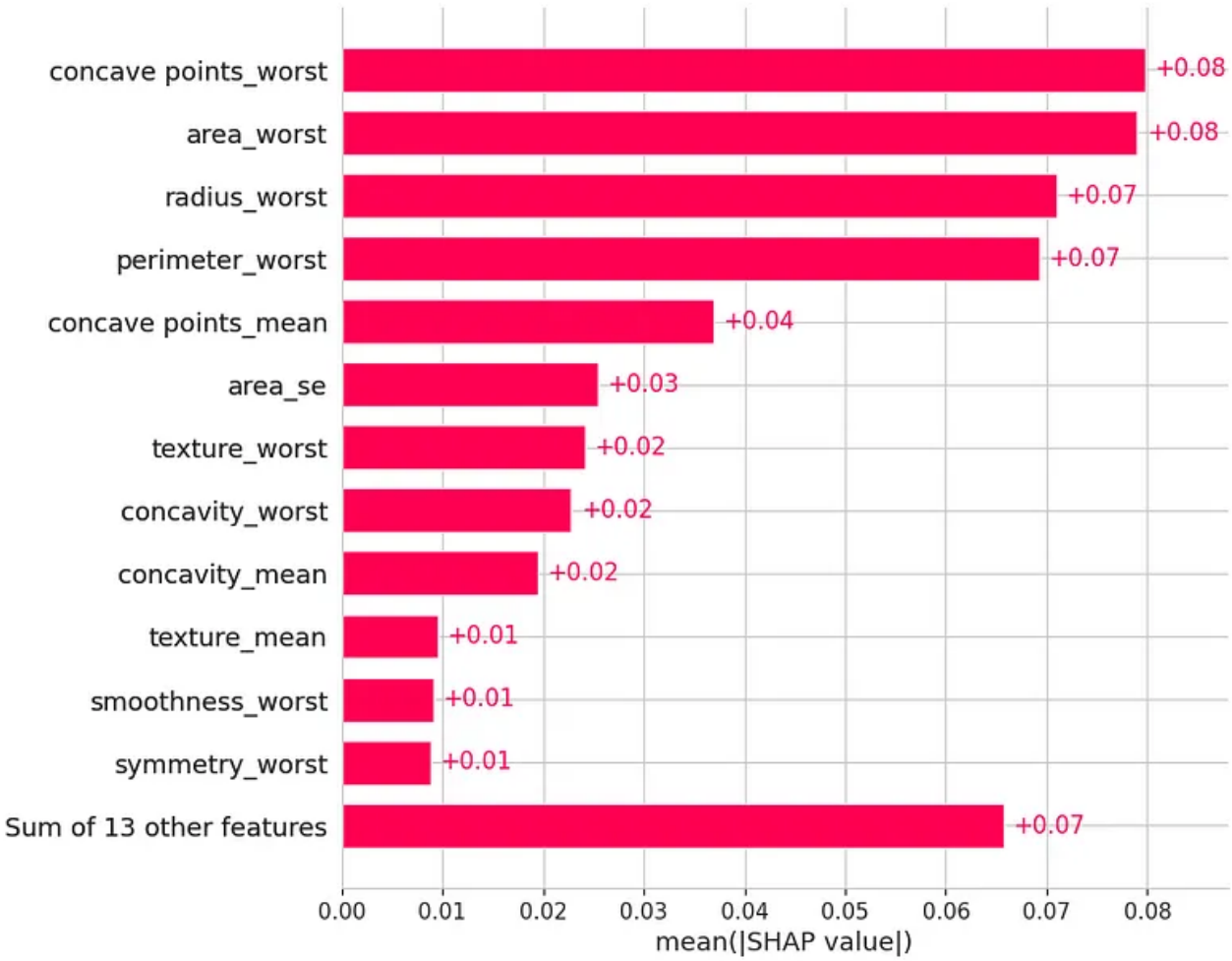
SHAP analysis is performed on selected models to interpret the feature contributions to predictions.

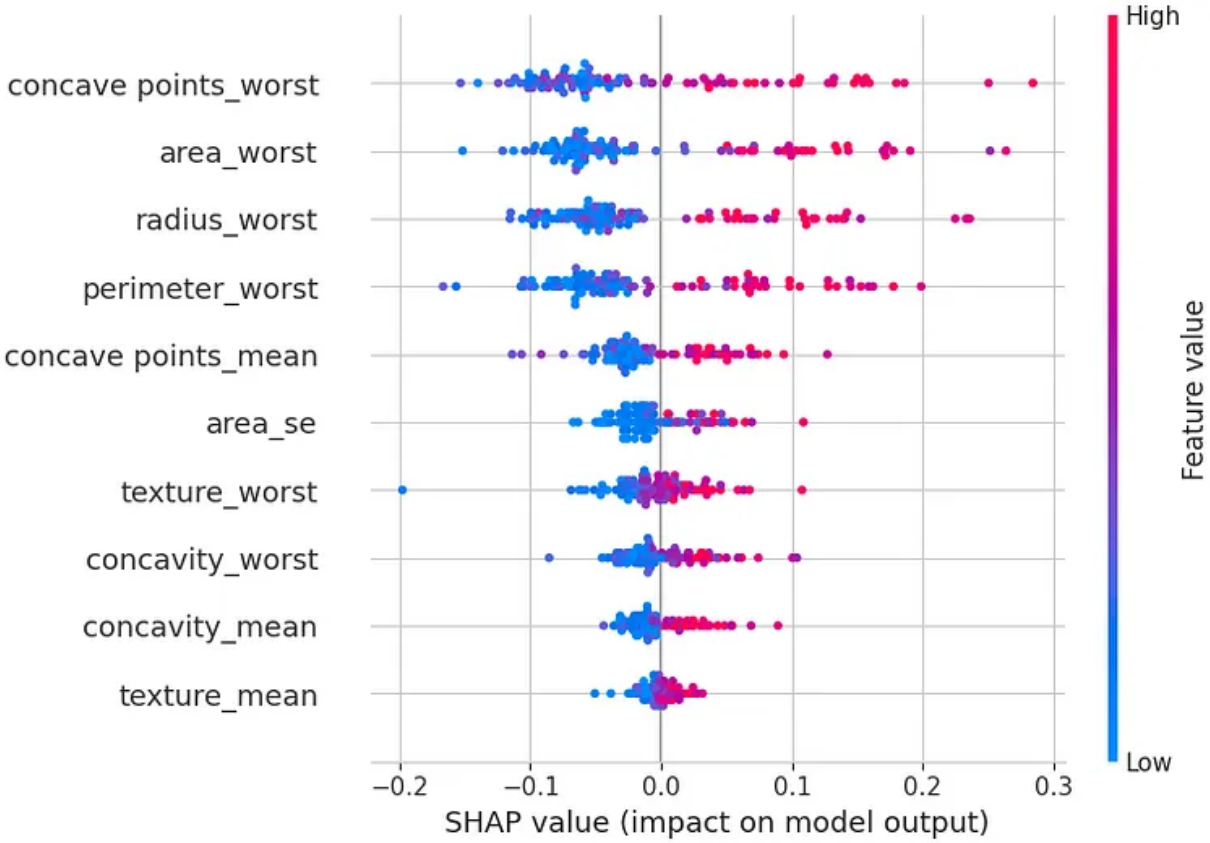
Partial dependence plots and feature importance summary plots are constructed.

Open in app ↗ Sign up Sign in

Medium Search


f(x) = 0
14.39 = radius_worst
0.04 = concave points_mean
17.7 = texture_worst
639.1 = area_worst
16.35 = area_se
0.156 = concave points_worst
0.125 = smoothness_worst
0.203 = radius_se
14.11 = texture_mean
16 other features
−0.11
−0.11
−0.06
−0.05
−0.03
+0.02
−0.02
−0.02
−0.01
−0.02
0.0
0.1
0.2
0.3
0.4
E[f(X)] = 0.4


concave points_worst
area_worst
radius_worst
perimeter_worst
concave points_mean
area_se
texture_worst
concavity_worst
concavity_mean
texture_mean
smoothness_worst
symmetry_worst
Sum of 13 other features
+0.08
+0.08
+0.07
+0.07
+0.04
+0.03
+0.02
+0.02
+0.02
+0.01
+0.01
+0.01
+0.07
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
mean(|SHAP value|)



Some of the best features obtained from KBest method are:-
'concavity_se', 'concave points_se', 'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smoothness_worst', 'compactness_worst', 'concavity_worst', 'concave points_worst', 'symmetry_worst', 'fractal_dimension_worst'.

The 12 best features obtained from the SHAP summary plot are:-
'concave points_worst','area_worst','perimeter worst','radius worst','concavity_worst' ,'concavity mean','concave points_mean','texture_worst', 'texture _mean','area se'.

References

- FeatureImportance
- IsolationForest

Machine Learning

Propose and get approved to write a section for the course Crash Course in Statistical Learning.

The expectation to that you clearly explain something using both theory and code related to understanding and preparing data for analysis in machine learning.

You MUST cite sources and the TAs will check submissions for plagiarism. It MUST you markdown and code so it can easily be added to the GitBook.

Crash_Course_in_Statistical_Learning_Written_Section (1)

December 16, 2023

1 Linear Discriminant Analysis

LDA is a linear methodology for classification problems.
The classification is done based on the value of the *discriminant function* $\delta(x)$. Similar to logistic regression, LDA tries to model the conditional distribution of the target(Y) given the predictors(X).

$$P(Y = k|X = x)$$

where k represents one of the target classes and x is a predictor observation
The conditional distribution for each class is obtained from the prior probability of the class and density function of predictor variable pertaining to that class.
By virtue of Bayes Theorem, we can obtain the conditional distributions of the target variables in the following manner:

$$P(Y = k|X = x) = \frac{\pi_k * f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

where π_k is the prior probability of class k and $f_k(x)$ is the probability of class k occuring when X=x.
For sake of brevity assume $P(Y = k|X = x) = p_k(x)$

1.1 Breast Tumor detection using LDA

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.
We try to diagnose unseen observation by employing a LDA model.
To move forward with LDA, we need to perform few checks on our data and its distribution.

[2]:
```
# importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split,StratifiedKFold
from sklearn.metrics import␣
↪accuracy_score,confusion_matrix,ConfusionMatrixDisplay,precision_score,recall_score,f1_scor
```

[3]:
```
# Obtaining the data in a pandas DataFrame format and dropping the id column
dataset=pd.read_csv("https://raw.githubusercontent.com/SKashyap1999/DataScience/
↪main/Breast_Cancer-Wisconsin.csv")
dataset=dataset.drop(['id'],axis=1)
```

[4]:
```
# Observing some statistical attributes of the features
dataset.describe()
```

[4]:
```
       radius_mean  texture_mean  perimeter_mean    area_mean  \
count   569.000000    569.000000      569.000000   569.000000
mean     14.127292     19.289649       91.969033   654.889104
std       3.524049      4.301036       24.298981   351.914129
min       6.981000      9.710000       43.790000   143.500000
25%      11.700000     16.170000       75.170000   420.300000
50%      13.370000     18.840000       86.240000   551.100000
75%      15.780000     21.800000      104.100000   782.700000
max      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  fractal_dimension_mean  …  radius_worst  \
count     569.000000              569.000000  …    569.000000
mean        0.181162                0.062798  …     16.269190
std         0.027414                0.007060  …      4.833242
min         0.106000                0.049960  …      7.930000
25%         0.161900                0.057700  …     13.010000
50%         0.179200                0.061540  …     14.970000
75%         0.195700                0.066120  …     18.790000
max         0.304000                0.097440  …     36.040000

       texture_worst  perimeter_worst   area_worst  smoothness_worst  \
count     569.000000       569.000000   569.000000        569.000000
mean       25.677223       107.261213   880.583128          0.132369
std         6.146258        33.602542   569.356993          0.022832
min        12.020000        50.410000   185.200000          0.071170
25%        21.080000        84.110000   515.300000          0.116600
50%        25.410000        97.660000   686.500000          0.131300
75%        29.720000       125.400000  1084.000000          0.146000
max        49.540000       251.200000  4254.000000          0.222600
```

```
       compactness_worst  concavity_worst  concave points_worst  \
count         569.000000       569.000000            569.000000
mean            0.254265         0.272188              0.114606
std             0.157336         0.208624              0.065732
min             0.027290         0.000000              0.000000
25%             0.147200         0.114500              0.064930
50%             0.211900         0.226700              0.099930
75%             0.339100         0.382900              0.161400
max             1.058000         1.252000              0.291000

       symmetry_worst  fractal_dimension_worst
count      569.000000               569.000000
mean         0.290076                 0.083946
std          0.061867                 0.018061
min          0.156500                 0.055040
25%          0.250400                 0.071460
50%          0.282200                 0.080040
75%          0.317900                 0.092080
max          0.663800                 0.207500

[8 rows x 30 columns]
```

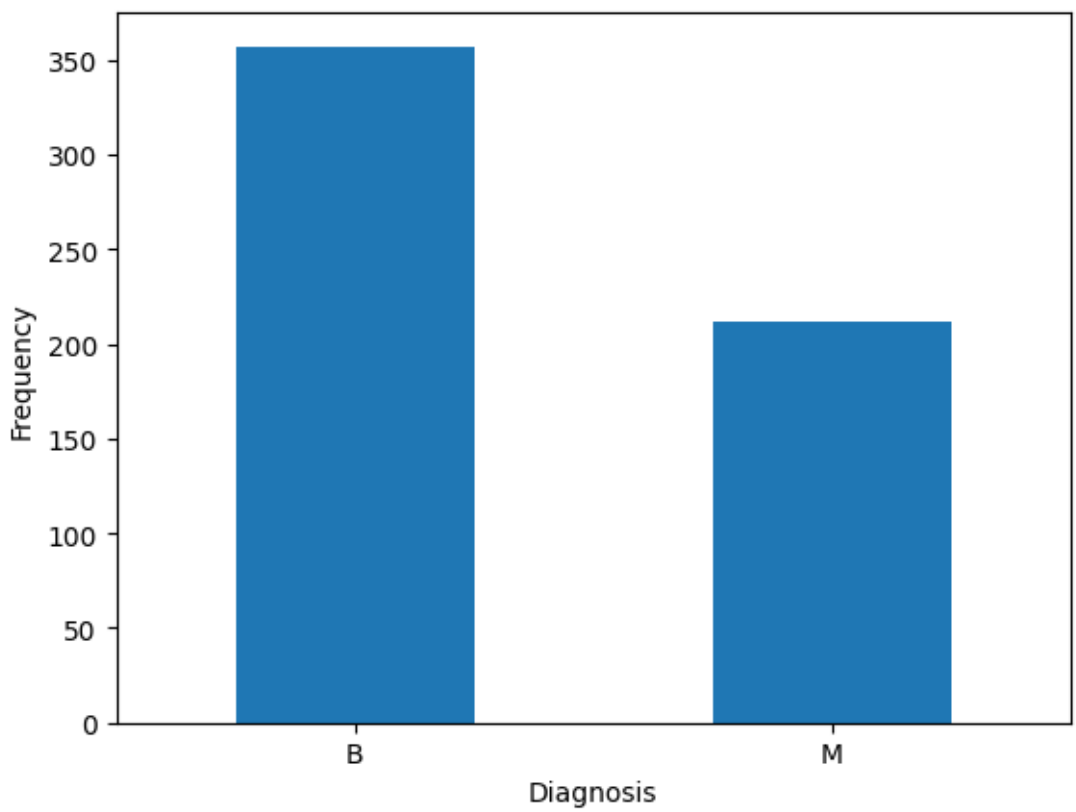
[5]:
```
#Checking for null instances
dataset.isnull().sum()
```

[5]:
```
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
```

[6]:
```
#Observing the target variable distribution
dataset["diagnosis"].value_counts().
 ↪plot(kind='bar',xlabel="Diagnosis",ylabel="Frequency",rot=0)
```

[6]:
```
<Axes: xlabel='Diagnosis', ylabel='Frequency'>
```



The target variable seems to have an uneven distribution of classes.

In LDA, the density function $f_k(x)$ is generally assumed to be a simple distribution, such as a Gaussian.

This means that the predictor(X) is drawn from a multivariate normal distribution, with each of the individual predictors following a one-dimensional normal distribution.
We represent this as: $X \sim N(\mu_k, \Sigma)$
where μ_k is the mean vector of class k and Σ is the covariance matrix of X,which is assumed to be common to all classes.
After assuming $f_k(x)$ to be a multivariate Gaussian, we get:

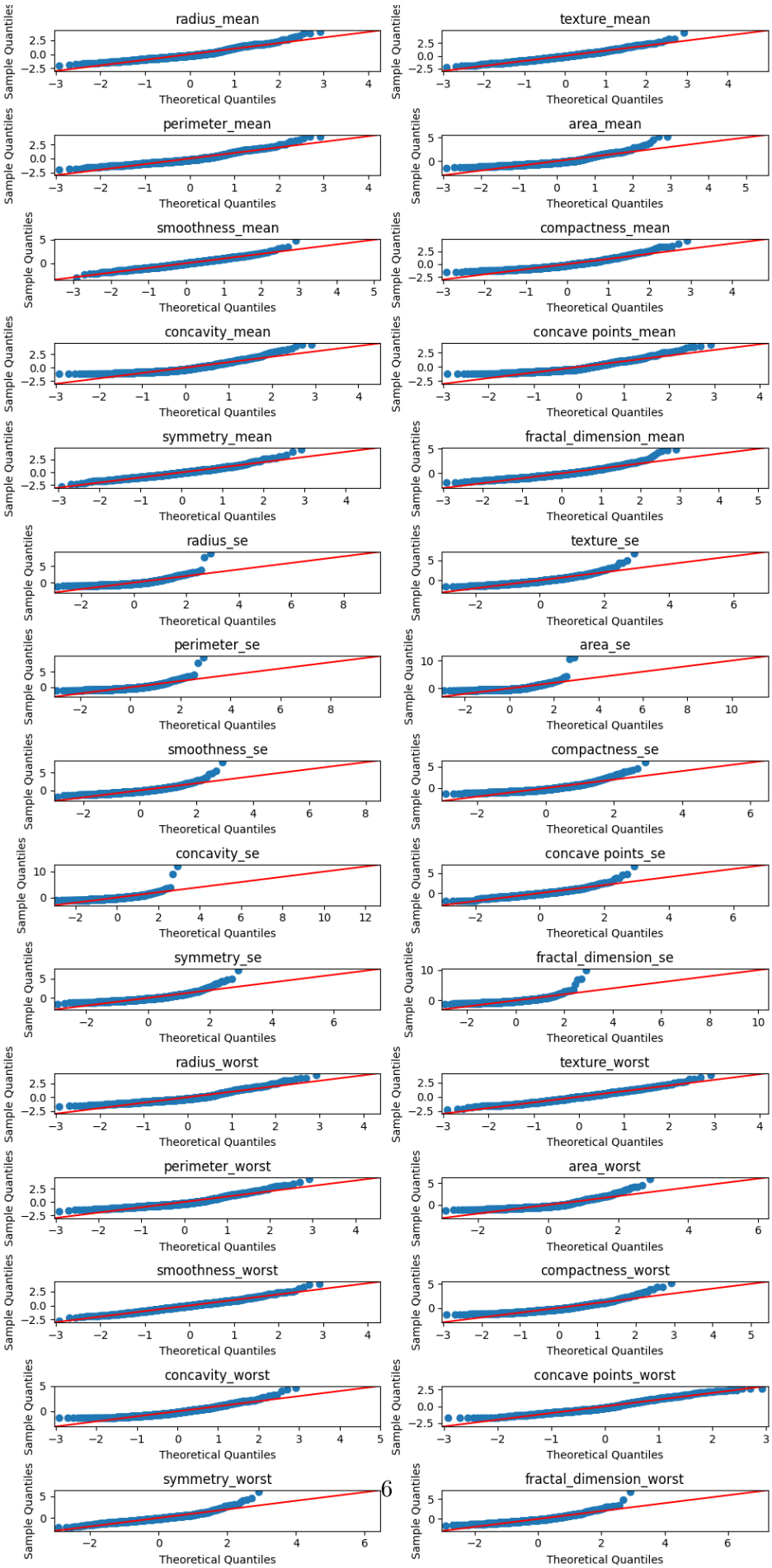
$$f_k(x) = \frac{1}{2\pi^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k))$$

Another assumption of LDA is the equality of class-specific variances $\sigma_1^2 = \sigma_2^2 = .. = \sigma_k^2$.
Applying log on both sides, we get:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log(\pi_k)$$

An observation x is classified to the class for which (x) is the highest.

[7]:
```
# Observing the normality of predictor variables
dataset_predictors=dataset.drop(['diagnosis'],axis=1)
dataset_target=dataset[["diagnosis"]]
fig = plt.figure(figsize=(10,20))

for c in range(30):
 ax = fig.add_subplot(15, 2, c+1)
 sm.qqplot(dataset[dataset_predictors.columns[c]],fit=True, ax=ax,line='45')
 ax.set(title=dataset_predictors.columns[c])
fig.tight_layout()
```


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
Sample Quantiles
Theoretical Quantiles

From the above graphical representation, we can assume that the distribution of feature variables is fairly normal.

1.2 Outlier removal and Scaling

Outlier presence can pose a skewed analysis by the LDA model. The shared variance assumption of LDA may require the data to be standardized.
Outliers are removed using the z-score method and scaling is performed using StandardScaler.

[8]:
```
#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=[]
for a in dataset_predictors.columns:
    limit_list.append((dataset_predictors[a].mean() + 3*dataset_predictors[a].
↪std(),dataset_predictors[a].mean() - 3*dataset_predictors[a].std()))

#Filterig the dataset based on individual feature z-scores
df_filtered=dataset_predictors
for l in enumerate(dataset_predictors.columns):
  df_filtered=df_filtered[(df_filtered[l[1]] < limit_list[l[0]][0]) &␣
↪(df_filtered[l[1]] > limit_list[l[0]][1])]
dataset_filtered = pd.merge(df_filtered, dataset_target, left_index=True,␣
↪right_index=True)
dataset_filtered=dataset_filtered.reset_index(drop=True)
```

[10]:
```
#helper function
def dummy(a):
  if a=="M":
    return 1
  else:
    return 0
def cf_display(test,pred):
  cf =␣
↪ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(test,pred),display_labels=['Benign
  cf.plot()
  plt.show()

def metric_scores(test,pred):
  accuracy = accuracy_score(test,pred)
  precision=precision_score(test,pred)
  recall=recall_score(test,pred)
  f1=f1_score(test,pred)
  print("Accuracy:", accuracy)
  print("Precision: ",precision)
  print("Recall ",recall)
```

```
  print("f1 ",f1)
```

[11]:
```
#Applying the Standard Scaling operation
scaler=StandardScaler()
dataset_filtered_predictor=dataset_filtered.drop(["diagnosis"],axis=1)
filtered_scaled_predictor_array=scaler.fit_transform(dataset_filtered_predictor)
target_array=dataset_filtered['diagnosis'].apply(lambda x:dummy(x)).to_numpy()
```

[12]:
```
#Splitting into train and test samples
X_train,X_test,y_train,y_test=train_test_split(filtered_scaled_predictor_array,target_array,te
 ↪2)
```

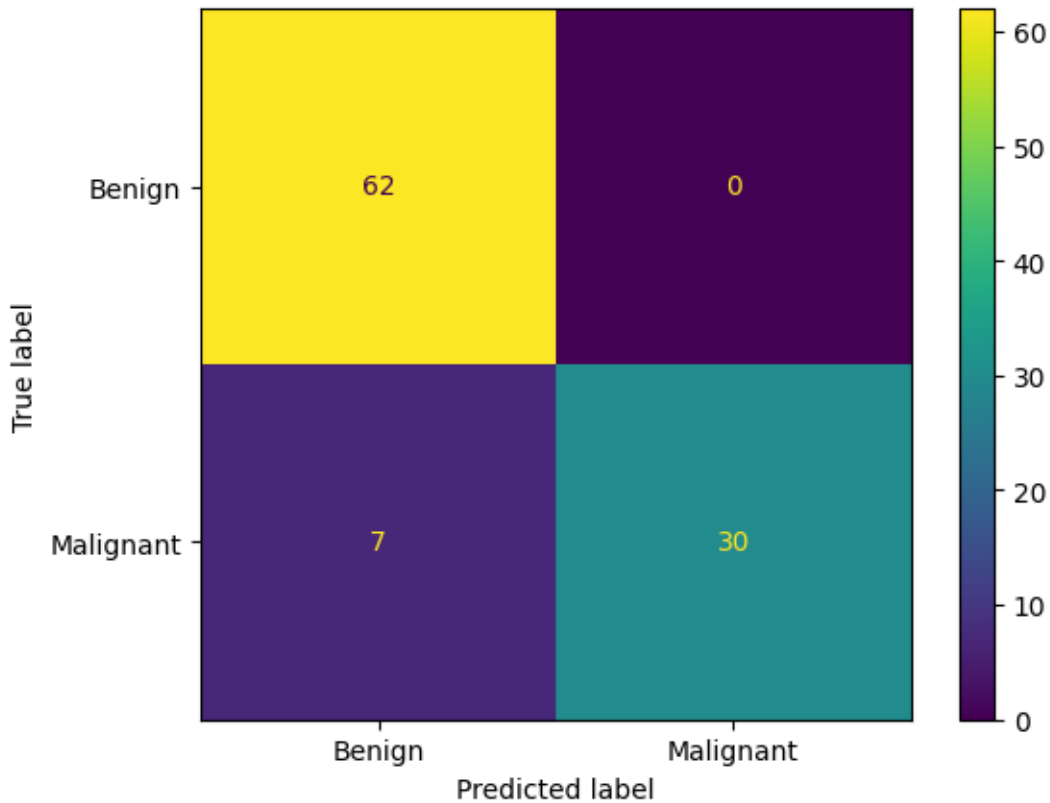
[13]:
```
model=LinearDiscriminantAnalysis()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
```

1.3 Model Performance Metrics

As our problem involves dignosing a medical condition, a classification metric such as precision and recall should be given greater emphasis inorder to judge the true-positive classification rate of the classifier.

[14]:
```
metric_scores(y_test,y_pred)
cf_display(y_test,y_pred)
```

```
Accuracy: 0.9292929292929293
Precision:  1.0
Recall  0.8108108108108109
f1  0.8955223880597014
```

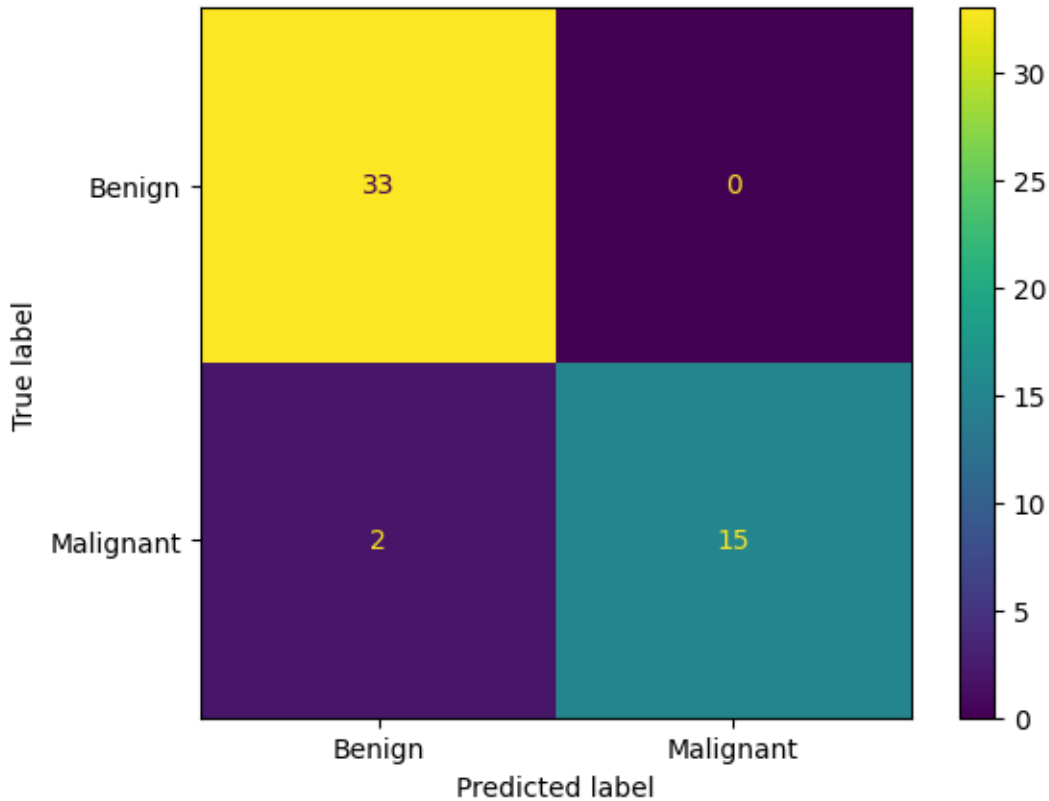


1.3.1 Cross-Validation Results

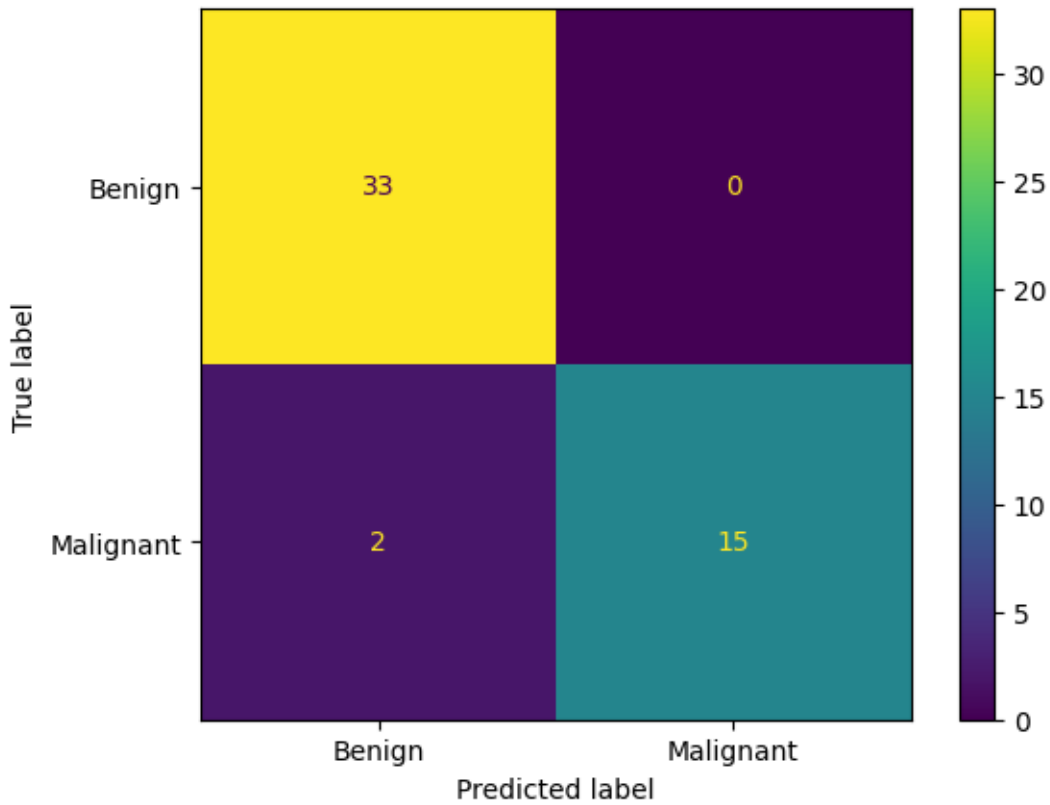
[15]:
```
#Performing StratifiedKFold CV
skf = StratifiedKFold(n_splits=10, shuffle=True)
split=1
for train_index, test_index in skf.split(filtered_scaled_predictor_array,␣
↪target_array):
    x_train_fold, x_test_fold = filtered_scaled_predictor_array[train_index],␣
↪filtered_scaled_predictor_array[test_index]
    y_train_fold, y_test_fold = target_array[train_index],␣
↪target_array[test_index]
    model.fit(x_train_fold, y_train_fold)
    print("____"+"Metrics for split "+str(split)+"____")
    split+=1
    metric_scores(y_test_fold,model.predict(x_test_fold))
    cf_display(y_test_fold,model.predict(x_test_fold))
    print("\n\n")
```

```
____Metrics for split 1____
Accuracy: 0.96
```

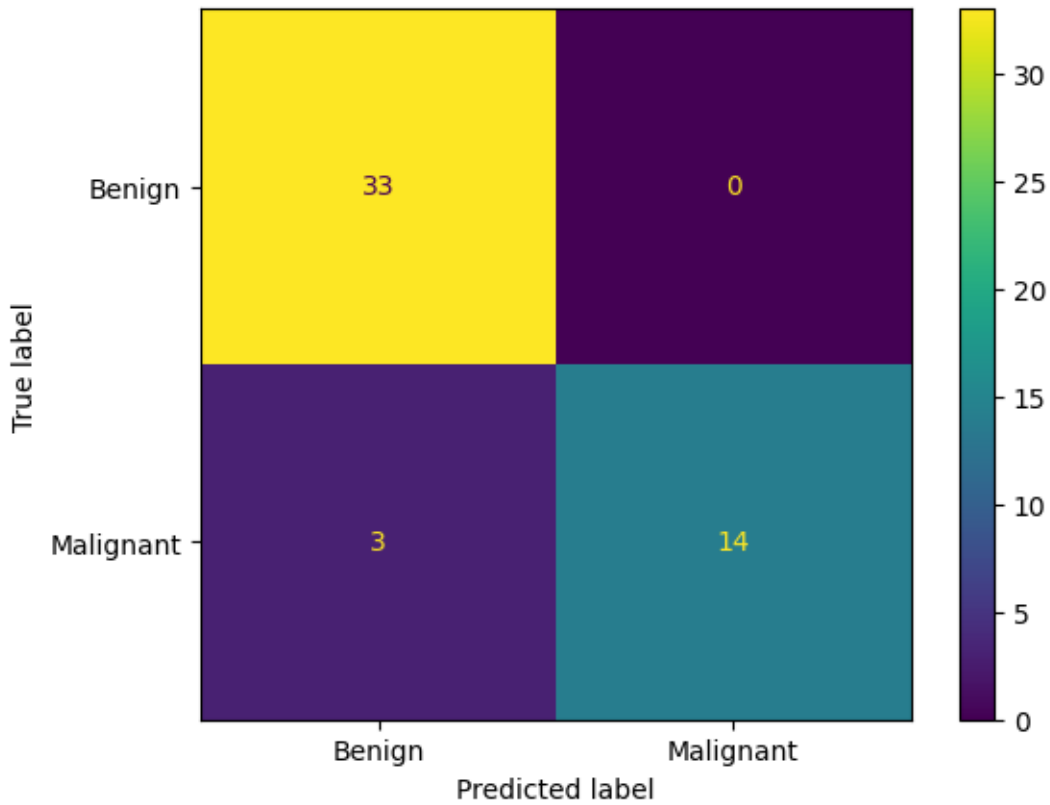
```
Precision:  1.0
Recall  0.8823529411764706
f1  0.9375
```



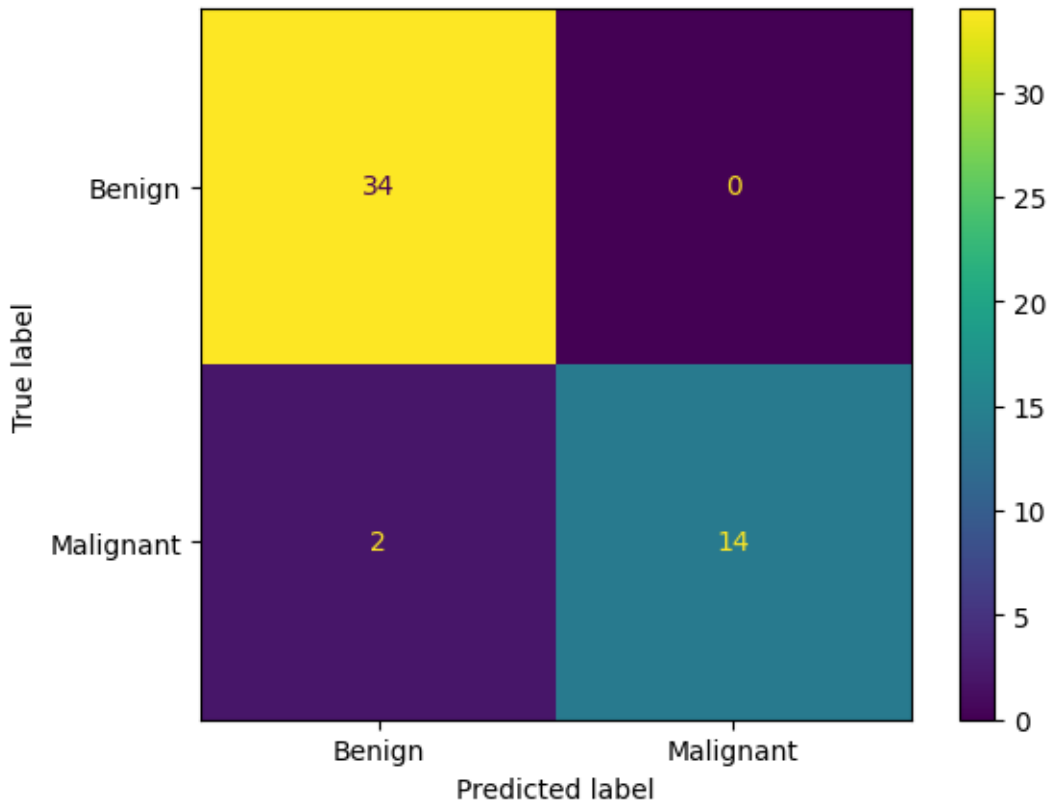

```
____Metrics for split 2____
Accuracy: 0.96
Precision:  1.0
Recall  0.8823529411764706
f1  0.9375
```



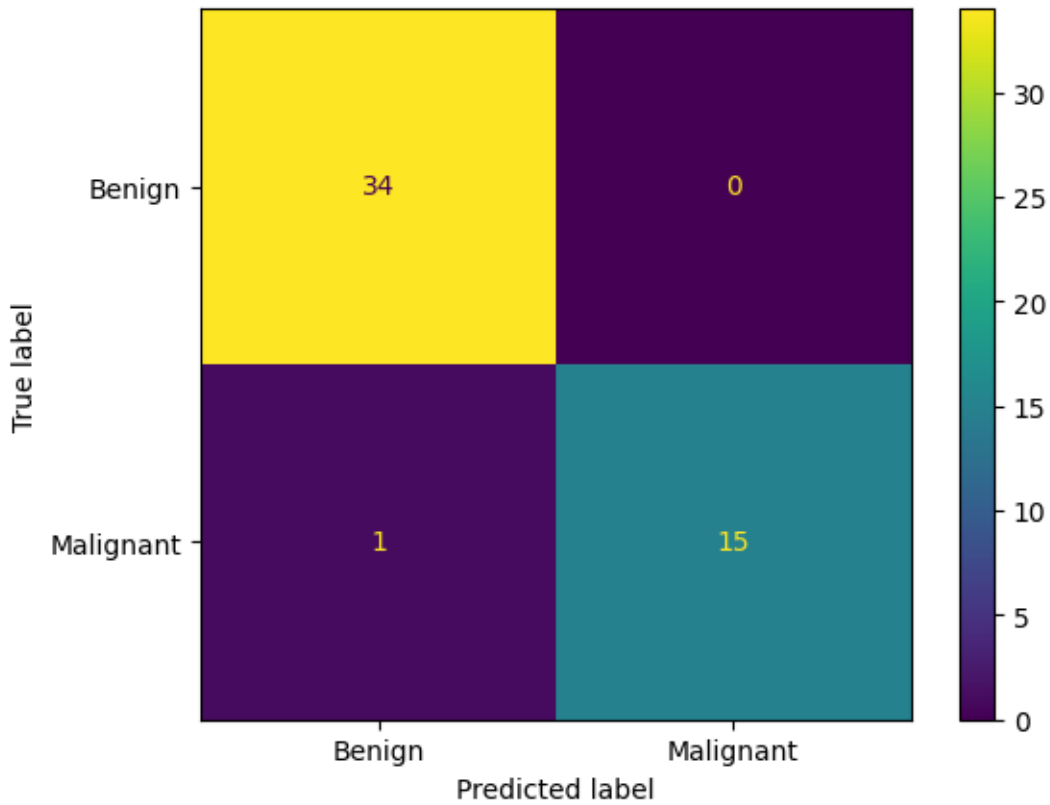

```
____Metrics for split 3____
Accuracy: 0.94
Precision:  1.0
Recall  0.8235294117647058
f1  0.9032258064516129
```



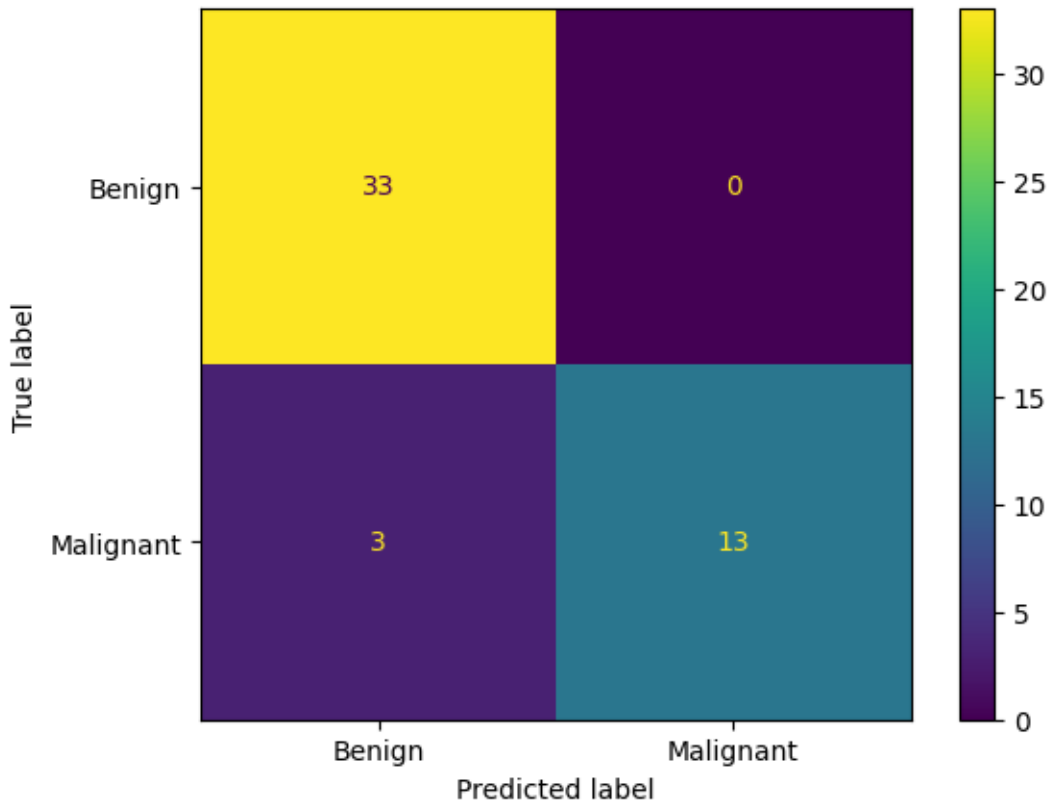

```
____Metrics for split 4____
Accuracy: 0.96
Precision:  1.0
Recall  0.875
f1  0.9333333333333333
```



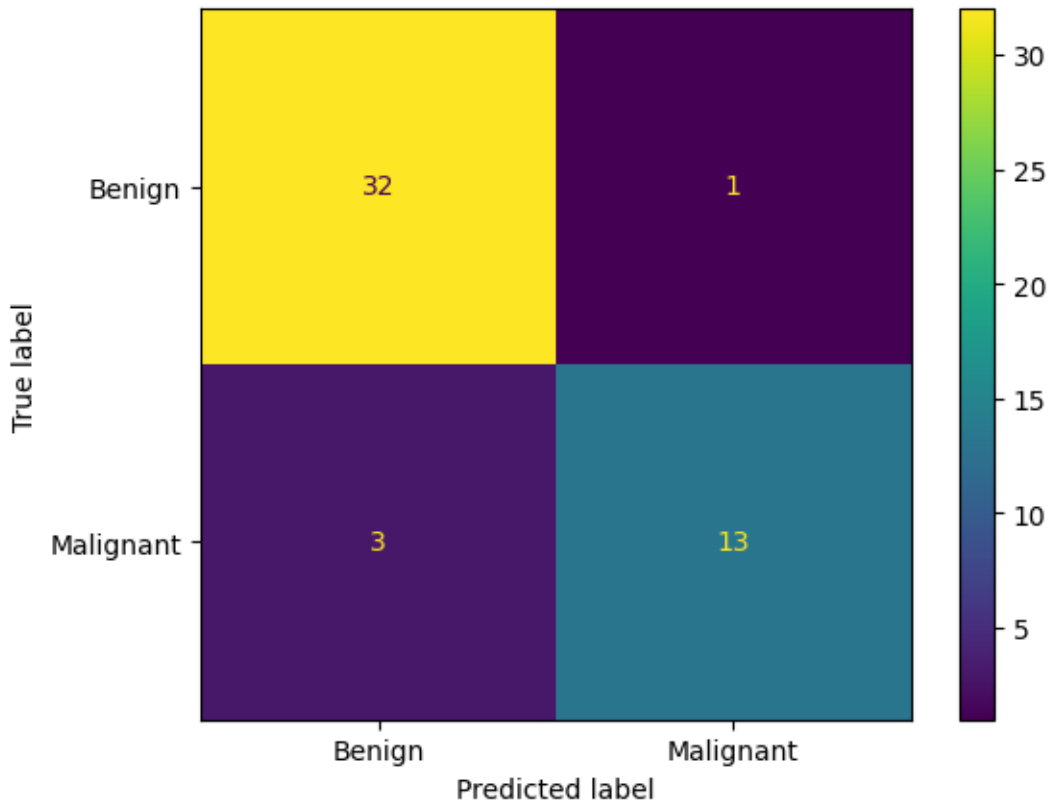

```
____Metrics for split 5____
Accuracy: 0.98
Precision:  1.0
Recall  0.9375
f1  0.967741935483871
```



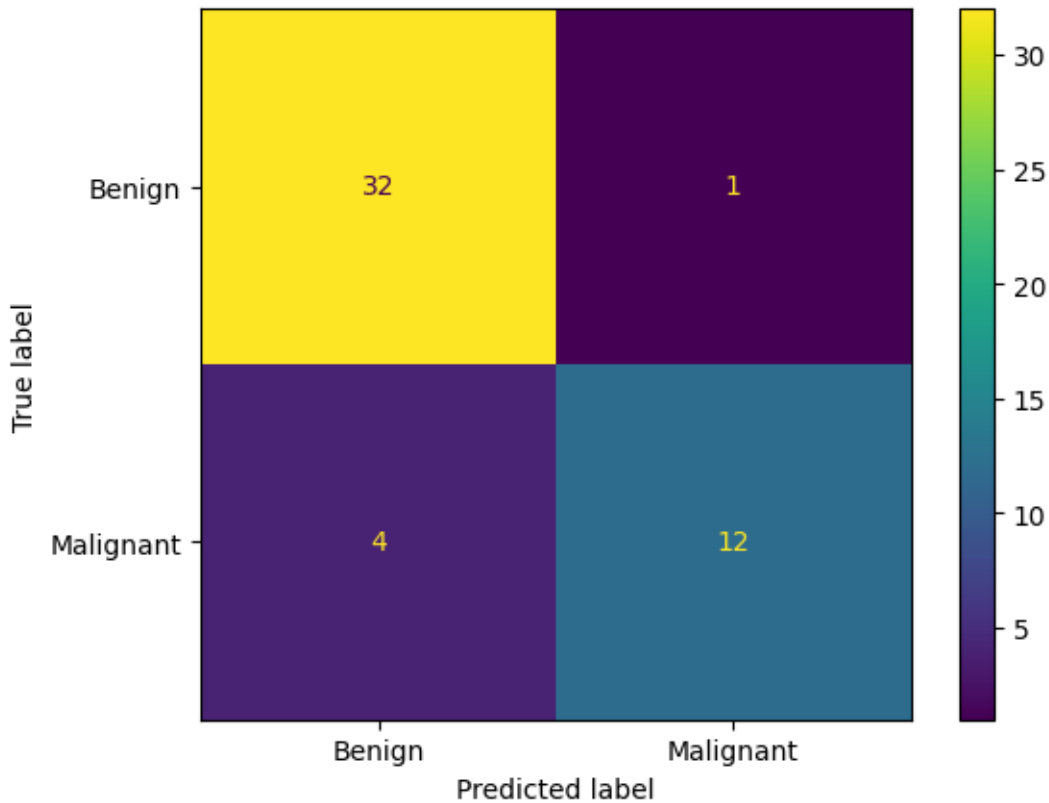

```
____Metrics for split 6____
Accuracy: 0.9387755102040817
Precision:  1.0
Recall  0.8125
f1  0.896551724137931
```



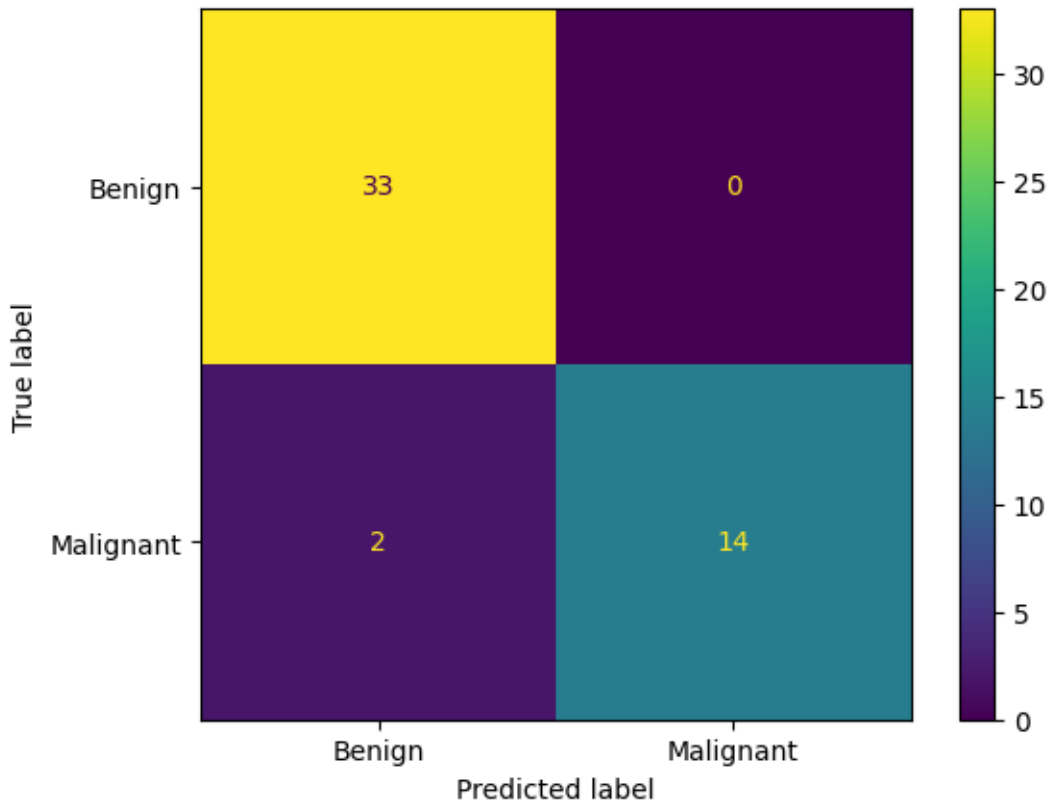

```
____Metrics for split 7____
Accuracy: 0.9183673469387755
Precision:  0.9285714285714286
Recall  0.8125
f1  0.8666666666666666
```



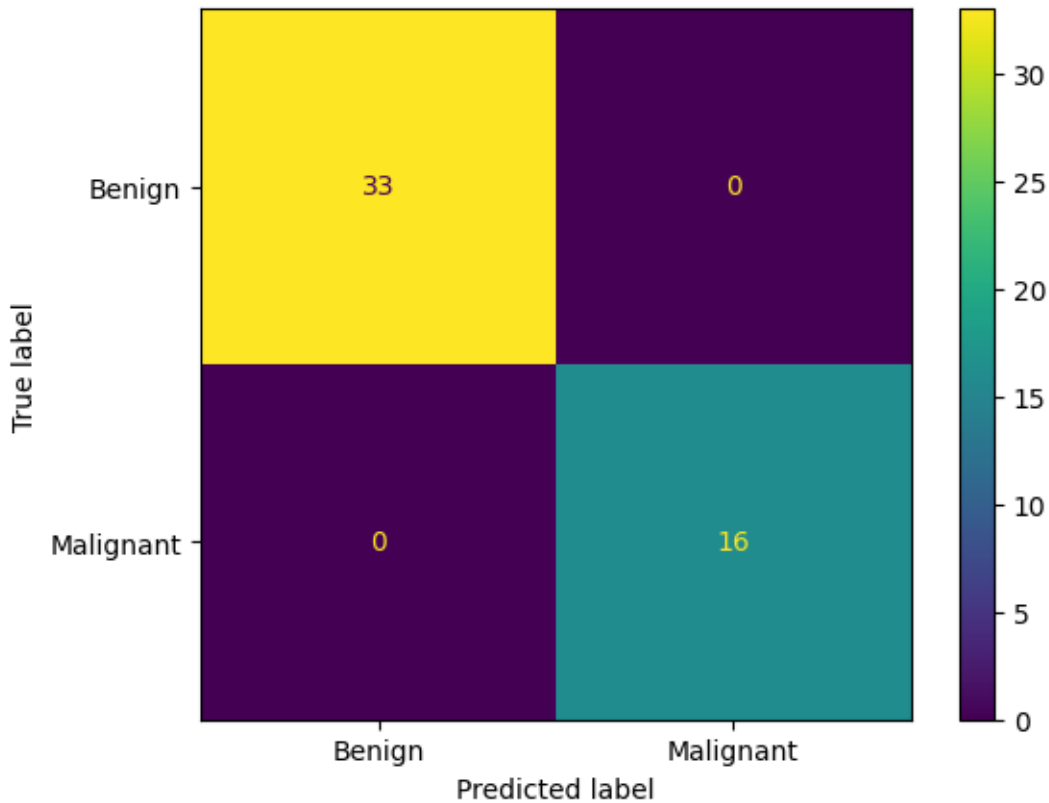

```
____Metrics for split 8____
Accuracy: 0.8979591836734694
Precision:  0.9230769230769231
Recall  0.75
f1  0.8275862068965517
```




```
____Metrics for split 9____
Accuracy: 0.9591836734693877
Precision:  1.0
Recall  0.875
f1  0.9333333333333333
```




```
____Metrics for split 10____
Accuracy: 1.0
Precision:  1.0
Recall  1.0
f1  1.0
```



The cross-validation results hint at the best expected performance of the LDA model with recall score of 1.0 and worst performance of 0.58

[15]:

```
```

Logistic Regression and LDA produce linear decision boundaries. Logistic Regression uses maximum-likelihood to estimate regression coefficients, whereas LDA assumes the data to be derived from a Gaussian distribution thereby utilizing mean and variances to calculate the value of the discriminant function.
LDA performs better when the above assumptions and the common covariance matrix condition hold well for the observations.

The normality of data was observed by using qq-plots. There are other statistical test such as the D'Agostino-Pearson and Kolmogorov-Smirnov null-hypothesis tests.

2 References

- James,Gareth(2014) An Introduction to Statistical Learning *Normality Tests
- LDA for Machine Learning

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.

[ ]:

```
```

You must at least have two worked examples, one in your section on a data set and one for exercises related to your section on a different dataset.

Crash_Course_in_Statistical_Learning_Worked_Examples (1) (1)

December 16, 2023

#Abstract The main motive of this notebook is to address two classification problems with the application of Linear Discriminant Analysis.
This notebook contains worked examples on two different datasets to interpret the workings and performance of the LDA model. Performance of the model is observed and measured using metrics such as precision-recall and accuracy scores.

0.1 Linear Discriminant Analysis

LDA is a supervised learning algorithm used for classification purposes. It assumes that the data is normally distributed and that the covariance matrices of the different classes are equal. It also assumes that the features are independent and have equal variances within each class.

0.2 Predicting Employee Satisfaction using LDA

0.3 About Dataset

This is a fictional Kaggle dataset created by generating various parameters on employee job satisfaction index.

It has the following attributes.

emp_id - Unique ID
age - Age
Dept - Department
location - Employee Location
education - Educational Qualification
recruitment_type - Mode of recruitment
job_level - Designation.
rating - Performance Rating.
onsite - Onsite Work ex awards - No. of awards
certifications - Is the employee certified?
salary - Net Salary
satisfied - Is the employee satisfied with his/her job?

The dataset contains 14 columns and 499 observations.

[643]:
```
#importing libraries
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelBinarizer,StandardScaler
from sklearn.model_selection import␣
↪train_test_split,cross_val_score,RepeatedStratifiedKFold,StratifiedKFold
from sklearn.discriminant_analysis import␣
↪LinearDiscriminantAnalysis,QuadraticDiscriminantAnalysis
from sklearn.metrics import␣
↪accuracy_score,recall_score,precision_score,ConfusionMatrixDisplay,confusion_matrix
import statsmodels.api as sm
```

[644]:
```
#Helper Methods
def cf_display(test,pred):
  cf =␣
↪ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(test,pred),display_labels=['Unsati
  cf.plot()
  plt.show()

def metric_scores(test,pred):
  accuracy = accuracy_score(test,pred)
  precision=precision_score(test,pred)
  recall=recall_score(test,pred)
  print("Accuracy:", accuracy)
  print("Precision: ",precision)
  print("Recall ",recall)
```

[645]:
```
# Importing the data in a pandas DataFrame format
dataset=pd.read_csv("https://raw.githubusercontent.com/SKashyap1999/DataScience/
↪main/Employee%20Satisfaction%20Index.csv")
dataset=dataset.iloc[:,2:]
```

[646]:
```
dataset.head(10)
```

[646]:
```
   age        Dept location education    recruitment_type  job_level  rating  \
0   28          HR   Suburb        PG            Referral          5       2
1   50  Technology   Suburb        PG             Walk-in          3       5
2   43  Technology   Suburb        UG            Referral          4       1
3   44       Sales     City        PG           On-Campus          2       3
4   33          HR     City        UG  Recruitment Agency          2       1
5   40  Purchasing   Suburb        UG            Referral          3       3
6   26  Purchasing   Suburb        UG            Referral          5       5
7   25  Technology     City        UG  Recruitment Agency          1       1
8   35          HR     City        PG            Referral          3       4
9   45  Technology     City        PG            Referral          3       3

   onsite  awards  certifications  salary  satisfied
0       0       1               0   86750          1
```

```
1        1        2                1   42419          0
2        0        2                0   65715          0
3        1        0                0   29805          1
4        0        5                0   29805          1
5        0        7                1   42419          1
6        0        2                0   86750          0
7        0        4                0   24076          0
8        0        0                0   42419          1
9        0        9                0   42419          0
```

[647]:
```
dataset.describe()
```

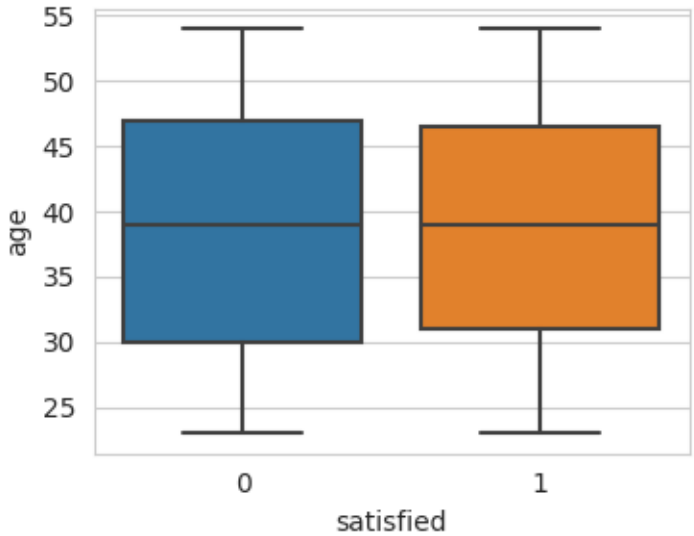
[647]:
```
              age   job_level      rating      onsite      awards  \
count  500.000000  500.000000  500.000000  500.000000  500.000000
mean    38.610000    3.032000    3.072000    0.466000    4.538000
std      9.438648    1.423738    1.425089    0.499342    2.853215
min     23.000000    1.000000    1.000000    0.000000    0.000000
25%     30.000000    2.000000    2.000000    0.000000    2.000000
50%     39.000000    3.000000    3.000000    0.000000    5.000000
75%     47.000000    4.000000    4.000000    1.000000    7.000000
max     54.000000    5.000000    5.000000    1.000000    9.000000

       certifications        salary   satisfied
count      500.000000    500.000000  500.000000
mean         0.496000  50416.056000    0.526000
std          0.500485  23671.392661    0.499824
min          0.000000  24076.000000    0.000000
25%          0.000000  29805.000000    0.000000
50%          0.000000  42419.000000    1.000000
75%          1.000000  65715.000000    1.000000
max          1.000000  86750.000000    1.000000
```

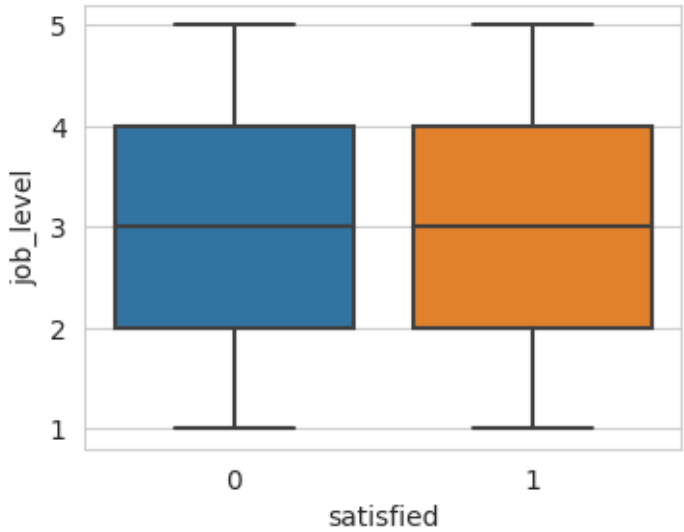
[648]:
```
dataset.isnull().sum()
```

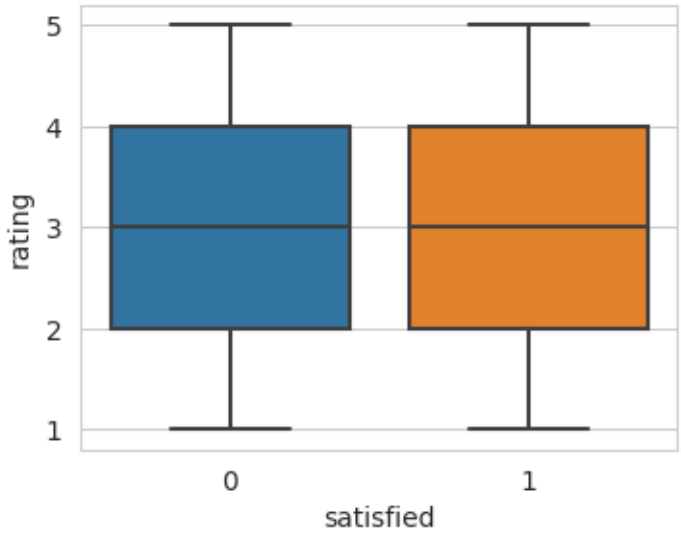
[648]:
```
age                 0
Dept                0
location            0
education           0
recruitment_type    0
job_level           0
rating              0
onsite              0
awards              0
certifications      0
salary              0
satisfied           0
dtype: int64
```

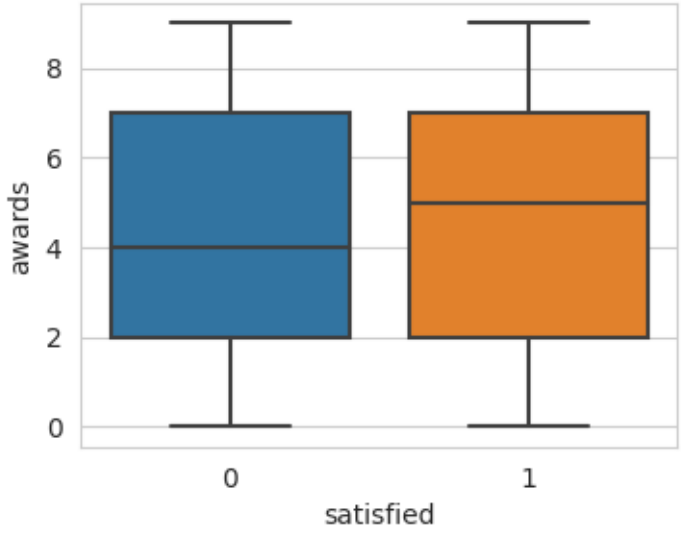
[649]:
```
pred_cols=[c for c in dataset.columns if c!='satisfied']
cat_cols=["Dept","location","education","recruitment_type","certifications","onsite"]
num_cols=[d for d in pred_cols if d not in cat_cols]
```

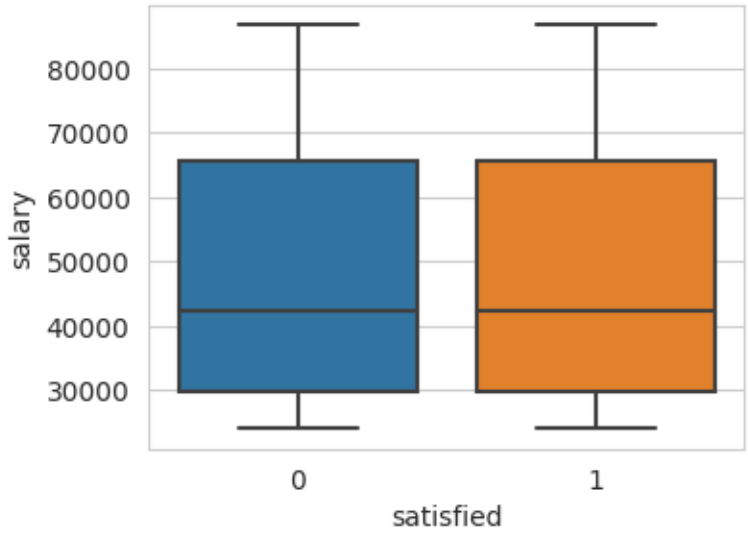
[650]:
```
# Boxplot to show the distribution of dependent variable for different␣
↪independent variables
for a in num_cols:
  plt.figure(figsize=(4,3))
  fig = sns.boxplot(x=dataset['satisfied'], y=dataset[a])
```






5
4
3
2
1
rating
0
1
satisfied

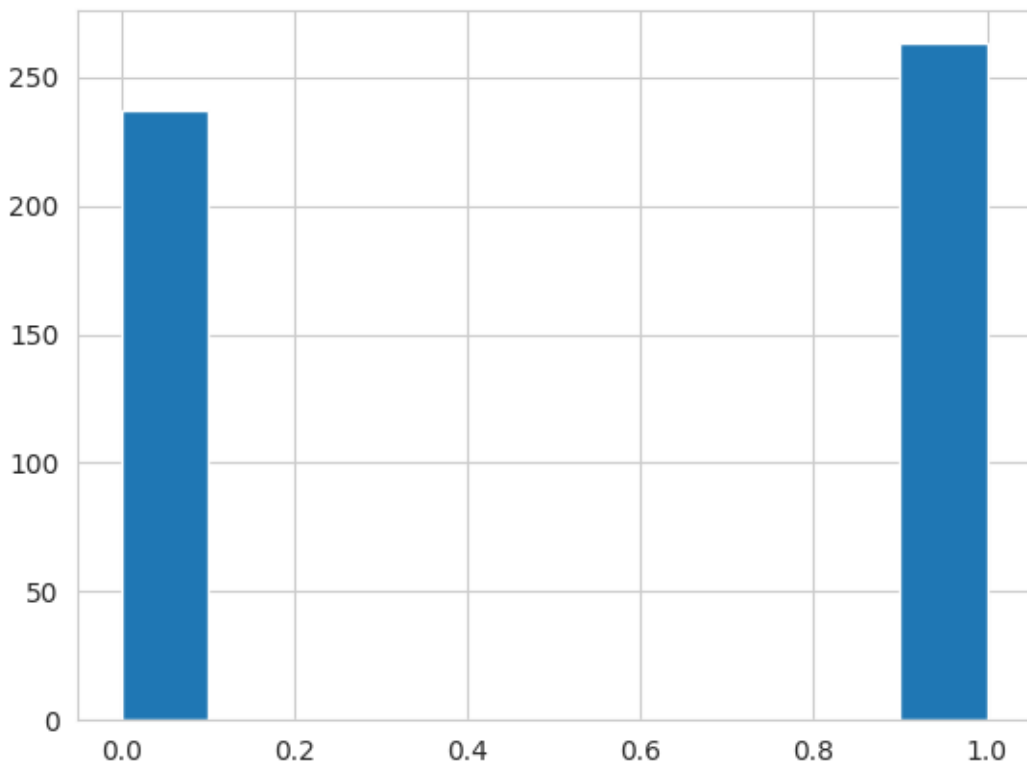

8
6
4
2
0
awards
0
1
satisfied



From observing the above plot, there seems to be a discernable difference in the awards feature, with high award count being prevalent in satisfied than unsatisfied.

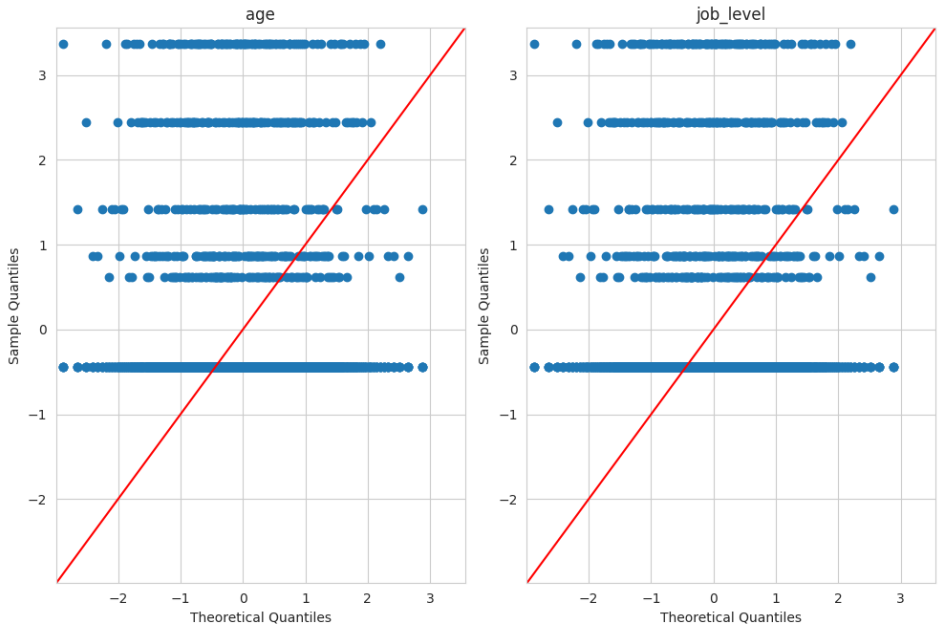
[651]:
```
#Observing the frequency of target classes
plt.hist(x=dataset['satisfied'])
```

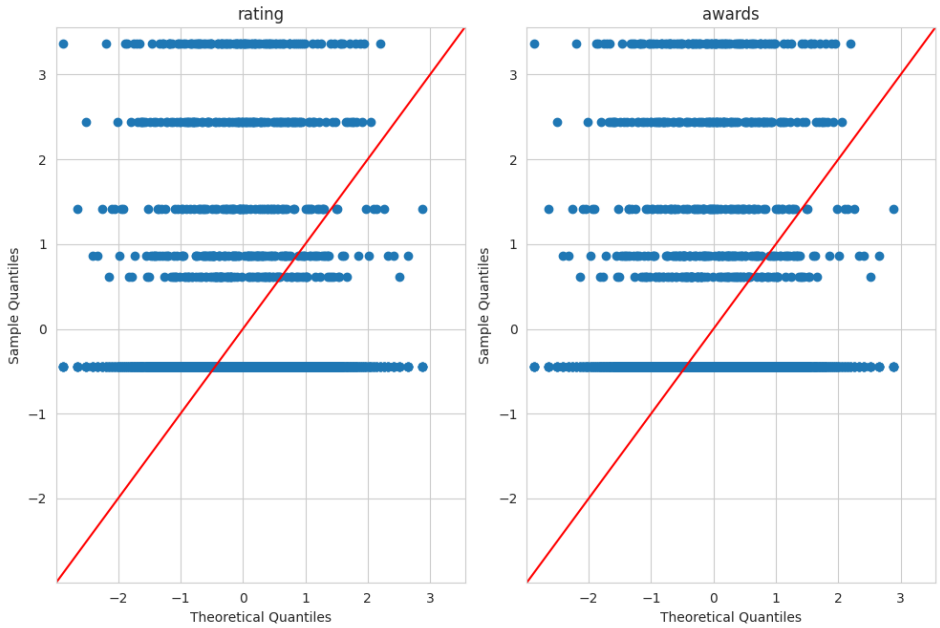
[651]:
```
(array([237.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 263.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```

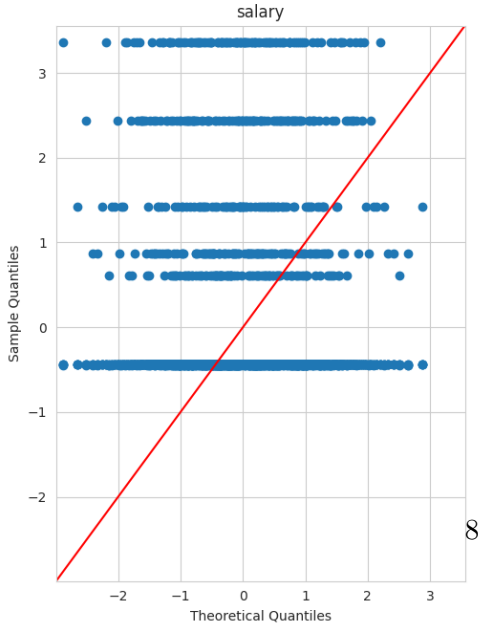


The target class distribution hints at almost balanced compostion.

[652]:
```
# Observing the normality of predictor variables
fig = plt.figure(figsize=(10,20))

for c in range(len(num_cols)):
 ax = fig.add_subplot(3, 2, c+1)
 sm.qqplot(dataset[num_cols],fit=True, ax=ax,line='45')
 ax.set(title=num_cols[c])
fig.tight_layout()
```


age
job_level
Sample Quantiles
Theoretical Quantiles


rating
awards
Sample Quantiles
Theoretical Quantiles


salary
Sample Quantiles
Theoretical Quantiles

The qq-plots for numerical features suggest that the features do not follow a normal distribution. Hence, using LDA might be a drawback in such a scenario.

0.3.1 Encoding Categorical Variable

[653]:
```
lb=LabelBinarizer()
for c in cat_cols:
  if(len(dataset[c].unique())>2):
   df= pd.DataFrame(lb.fit_transform(dataset[[c]]),columns=list(lb.classes_))
  else:
   df=pd.DataFrame(lb.fit_transform(dataset[[c]]),columns=[c+"_"])
  dataset=pd.concat([dataset,df],axis=1)
```

[654]:
```
dataset=dataset.drop(cat_cols,axis=1)
dataset_predictor=dataset.drop(['satisfied'],axis=1)
dataset_target=dataset[['satisfied']]
```

0.3.2 Training Split

[655]:
```
#Performing train-test split
dataset_predictor.columns = dataset_predictor.columns.astype(str)
X_train,X_test,y_train,y_test=train_test_split(dataset_predictor,dataset_target,test_size=0.
 ↪1)
```

0.3.3 Scaling

[656]:
```
# Scaling numerical features for making the columns have same variance
scaler=StandardScaler()
X_train[num_cols]=scaler.fit_transform(X_train[num_cols])
X_test[num_cols]=scaler.transform(X_test[num_cols])
X_scaled=pd.concat([X_train,X_test],axis=0)
y_scaled=pd.concat([y_train,y_test],axis=0)
```

[657]:
```
X_scaled
```

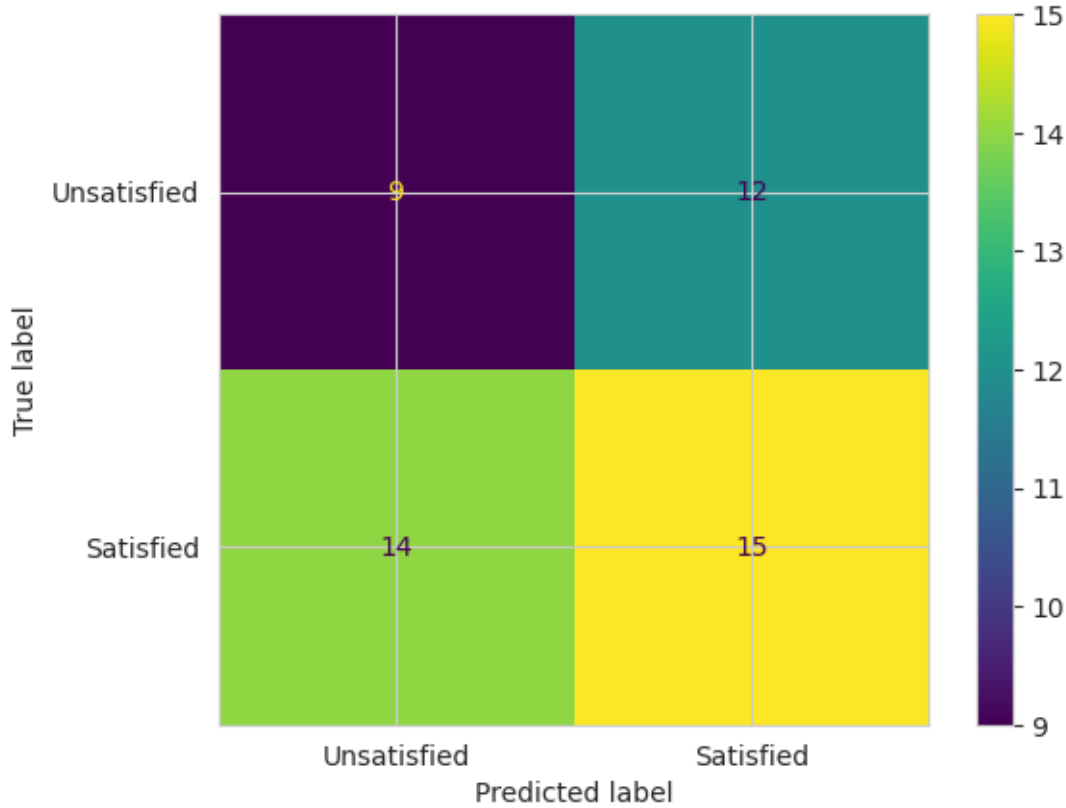
[657]:
```
          age  job_level    rating    awards    salary  HR  Marketing  \
188  0.246007  -0.736788  1.372083 -1.625137 -0.883466   0          1
293  1.623174  -1.434798  0.669652  1.557554 -1.123534   1          0
172 -0.389609  -1.434798  0.669652 -0.564240 -1.123534   0          0
33   0.881623   0.659232 -0.032780 -1.625137  0.621308   0          1
451  0.140071  -0.736788 -0.735212 -0.564240 -0.883466   0          1
..        …         …        …        …        …  ..         …
240 -0.601481  -0.038778 -0.032780  0.850290 -0.354888   0          0
4   -0.601481  -0.736788 -1.437643  0.143025 -0.883466   1          0
373 -0.071801  -0.736788 -0.032780 -0.564240 -0.883466   1          0
```

```
237  0.987559  -0.736788  0.669652  0.496657 -0.883466   0          0
396  0.669751   1.357242 -1.437643  0.143025  1.502759   1          0

     Purchasing  Sales  Technology  location_  education_  On-Campus  \
188           0      0           0          0           1          0
293           0      0           0          1           0          0
172           0      0           1          1           1          0
33            0      0           0          1           0          1
451           0      0           0          0           0          0
..          …    …          …         …          …         …
240           0      0           1          1           0          0
4             0      0           0          0           1          0
373           0      0           0          0           0          0
237           1      0           0          1           0          1
396           0      0           0          0           1          0

     Recruitment Agency  Referral  Walk-in  certifications_  onsite_
188                   0         1        0                1        0
293                   1         0        0                0        0
172                   1         0        0                1        1
33                    0         0        0                0        0
451                   0         1        0                1        0
..                  …       …      …              …       …
240                   0         1        0                1        1
4                     1         0        0                0        0
373                   0         0        1                1        0
237                   0         0        0                0        0
396                   0         1        0                0        0

[500 rows x 18 columns]
```

[658]:
```
ld = LinearDiscriminantAnalysis()
ld.fit(X_train, y_train)
y_pred = ld.predict(X_test)
metric_scores(y_test,y_pred)
cf_display(y_test,y_pred)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

```
Accuracy: 0.48
Precision:  0.5555555555555556
Recall  0.5172413793103449
```



0.3.4 Cross-Validation Results

[659]:
```
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(ld, X_scaled.to_numpy(), y_scaled.to_numpy(),␣
↪scoring='recall', cv=cv, n_jobs=-1)
```

[660]:
```
scores
```

[660]:
```
array([0.59259259, 0.40740741, 0.44444444, 0.57692308, 0.61538462,
       0.53846154, 0.61538462, 0.65384615, 0.65384615, 0.61538462,
       0.44444444, 0.66666667, 0.7037037 , 0.5       , 0.65384615,
       0.46153846, 0.65384615, 0.57692308, 0.42307692, 0.53846154,
       0.62962963, 0.55555556, 0.59259259, 0.5       , 0.61538462,
       0.80769231, 0.5       , 0.46153846, 0.61538462, 0.65384615])
```

[661]:
```
np.mean(scores)
```

[661]:
```
0.575593542260209
```

From cross validation results, the best performance of the model was 74% recall, the mean performance was at 60.3%. The results were subpar, and hence more complex classifiers such as QDA, ensemble methods or Neural Networks may be required to get desirable results.

[661]:

0.4 Mobile Price Range Prediction using LDA

0.4.1 About Dataset

The dataset in question is related to mobile phone sales and prices. Sales data from various companies is collected to better understand the relationship between different features of a mobile phone (such as RAM, internal memory, etc.) and its selling price. The price is mentioned in terms of range(0 being the minimum and 3 being the maximum price categories).

[662]:
```
dataset=pd.read_csv("https://raw.githubusercontent.com/SKashyap1999/DataScience/
↪main/train.csv")
```

[663]:
```
dataset.head(10)
```

[663]:
```
   battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  m_dep  \
0            842     0          2.2         0   1       0           7    0.6
1           1021     1          0.5         1   0       1          53    0.7
2            563     1          0.5         1   2       1          41    0.9
3            615     1          2.5         0   0       0          10    0.8
4           1821     1          1.2         0  13       1          44    0.6
5           1859     0          0.5         1   3       0          22    0.7
6           1821     0          1.7         0   4       1          10    0.8
7           1954     0          0.5         1   0       0          24    0.8
8           1445     1          0.5         0   0       0          53    0.7
9            509     1          0.6         1   2       1           9    0.1

   mobile_wt  n_cores  …  px_height  px_width   ram  sc_h  sc_w  talk_time  \
0        188        2  …         20       756  2549     9     7         19
1        136        3  …        905      1988  2631    17     3          7
2        145        5  …       1263      1716  2603    11     2          9
3        131        6  …       1216      1786  2769    16     8         11
4        141        2  …       1208      1212  1411     8     2         15
5        164        1  …       1004      1654  1067    17     1         10
6        139        8  …        381      1018  3220    13     8         18
7        187        4  …        512      1149   700    16     3          5
8        174        7  …        386       836  1099    17     1         20
9         93        5  …       1137      1224   513    19    10         12

   three_g  touch_screen  wifi  price_range
0        0             0     1            1
1        1             1     0            2
2        1             1     0            2
3        1             0     0            2
4        1             1     0            1
5        1             0     0            1
6        1             0     1            3
```

```
7        1             1     1            0
8        1             0     0            0
9        1             0     0            0

[10 rows x 21 columns]
```

[664]:
```
dataset.describe()
```

[664]:
```
       battery_power       blue  clock_speed     dual_sim           fc  \
count    2000.000000  2000.0000  2000.000000  2000.000000  2000.000000
mean     1238.518500     0.4950     1.522250     0.509500     4.309500
std       439.418206     0.5001     0.816004     0.500035     4.341444
min       501.000000     0.0000     0.500000     0.000000     0.000000
25%       851.750000     0.0000     0.700000     0.000000     1.000000
50%      1226.000000     0.0000     1.500000     1.000000     3.000000
75%      1615.250000     1.0000     2.200000     1.000000     7.000000
max      1998.000000     1.0000     3.000000     1.000000    19.000000

            four_g   int_memory        m_dep    mobile_wt      n_cores  …  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000  …
mean      0.521500    32.046500     0.501750   140.249000     4.520500  …
std       0.499662    18.145715     0.288416    35.399655     2.287837  …
min       0.000000     2.000000     0.100000    80.000000     1.000000  …
25%       0.000000    16.000000     0.200000   109.000000     3.000000  …
50%       1.000000    32.000000     0.500000   141.000000     4.000000  …
75%       1.000000    48.000000     0.800000   170.000000     7.000000  …
max       1.000000    64.000000     1.000000   200.000000     8.000000  …

         px_height     px_width          ram         sc_h         sc_w  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000
mean    645.108000  1251.515500  2124.213000    12.306500     5.767000
std     443.780811   432.199447  1084.732044     4.213245     4.356398
min       0.000000   500.000000   256.000000     5.000000     0.000000
25%     282.750000   874.750000  1207.500000     9.000000     2.000000
50%     564.000000  1247.000000  2146.500000    12.000000     5.000000
75%     947.250000  1633.000000  3064.500000    16.000000     9.000000
max    1960.000000  1998.000000  3998.000000    19.000000    18.000000

         talk_time      three_g  touch_screen         wifi  price_range
count  2000.000000  2000.000000   2000.000000  2000.000000  2000.000000
mean     11.011000     0.761500      0.503000     0.507000     1.500000
std       5.463955     0.426273      0.500116     0.500076     1.118314
min       2.000000     0.000000      0.000000     0.000000     0.000000
25%       6.000000     1.000000      0.000000     0.000000     0.750000
50%      11.000000     1.000000      1.000000     1.000000     1.500000
75%      16.000000     1.000000      1.000000     1.000000     2.250000
max      20.000000     1.000000      1.000000     1.000000     3.000000
```

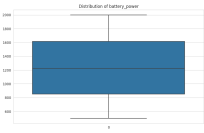
```
[8 rows x 21 columns]
```

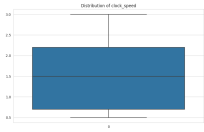
[665]:
```
dataset.isnull().sum()
```

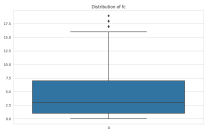
[665]:
```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

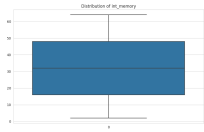
[666]:
```
dataset_predictors=dataset.iloc[:,:-1]
dataset_target=dataset["price_range"]
dataset_predictors.columns=dataset_predictors.columns.astype('object')
cat_cols=["blue","dual_sim","four_g","touch_screen","wifi","three_g"]
num_cols=[c for c in dataset_predictors.columns if c not in cat_cols]
```

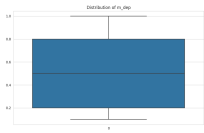
[667]:
```
#Boxplot for every independent feature

plt.rcParams.update({'figure.max_open_warning': 0})
plt.figure(figsize=(10,100))
for a in range(1,len(num_cols)+1):
    plt.subplot(len(num_cols),1,a)
    sns.boxplot(dataset_predictors[num_cols[a-1]])
    plt.title("Distribution of "+num_cols[a-1])
```

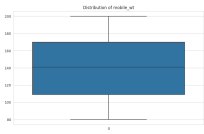

Distribution of battery_power

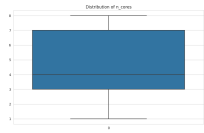

Distribution of clock_speed

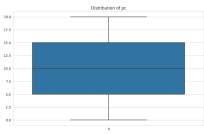

Distribution of fc

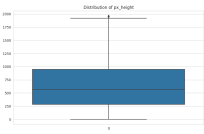

Distribution of int_memory

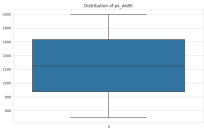

Distribution of m_dep

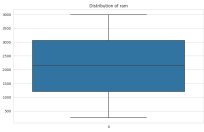

Distribution of mobile_wt

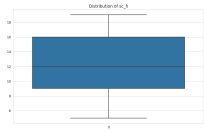

Distribution of n_cores

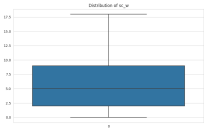

Distribution of pc

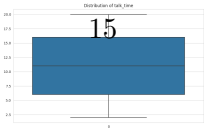

Distribution of px_height


Distribution of px_width


Distribution of ram


Distribution of sc_h

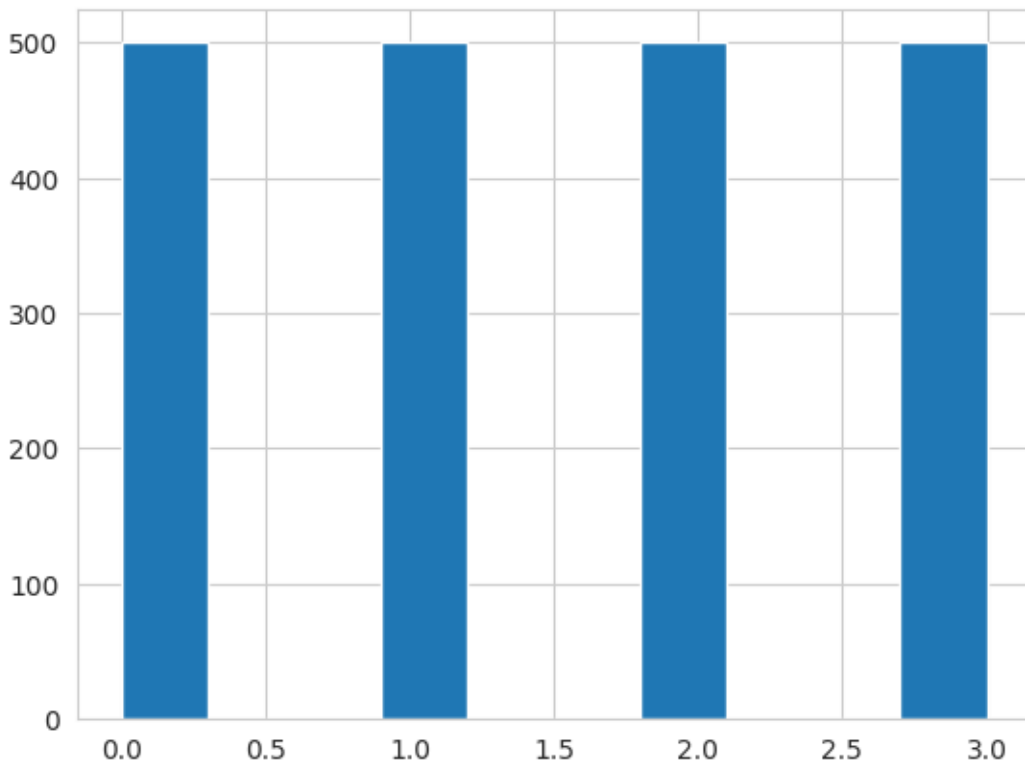

Distribution of sc_w


Distribution of talk_time

We can observe that there is no strong presence of outliers in most of the numerical features.

[668]:
```
#Observing the frequency of target classes
plt.hist(x=dataset_target)
```

[668]:
```
(array([500.,   0.,   0., 500.,   0.,   0., 500.,   0.,   0., 500.]),
 array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3. ]),
 <BarContainer object of 10 artists>)
```

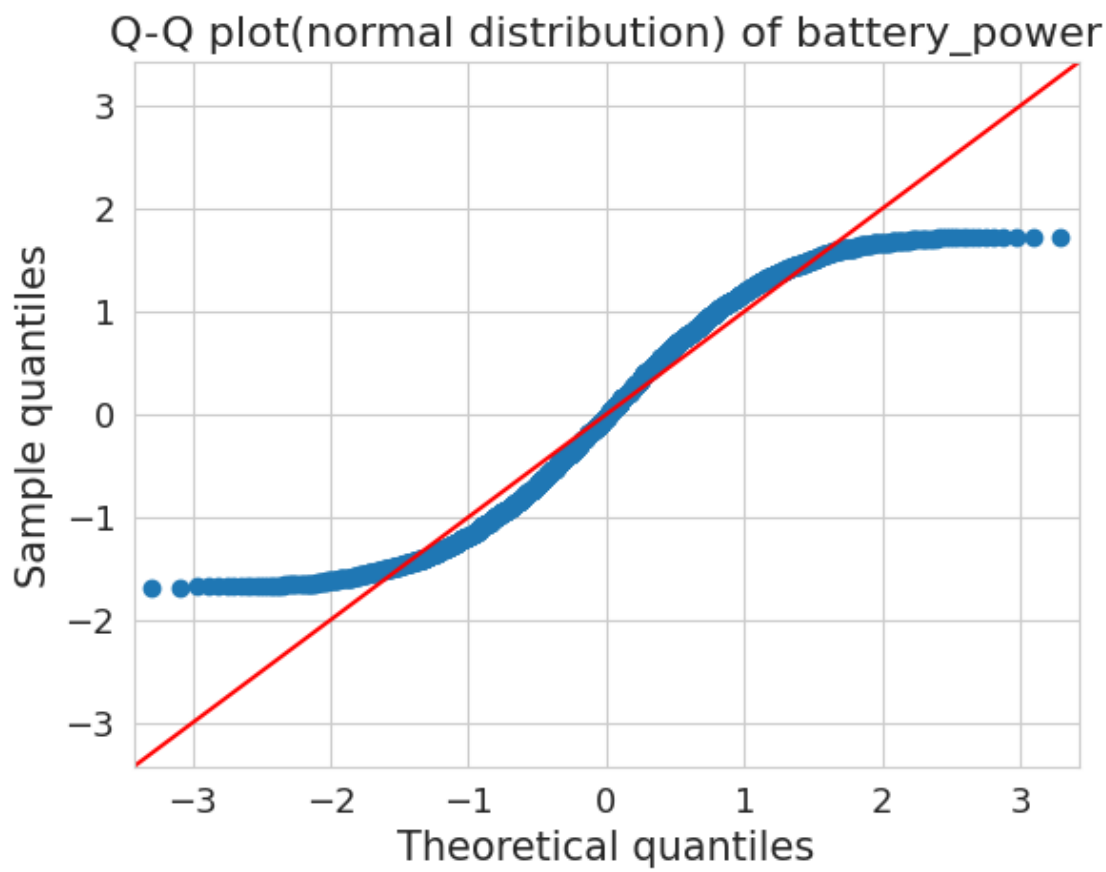


There appears to be a categorically balanced target variable.

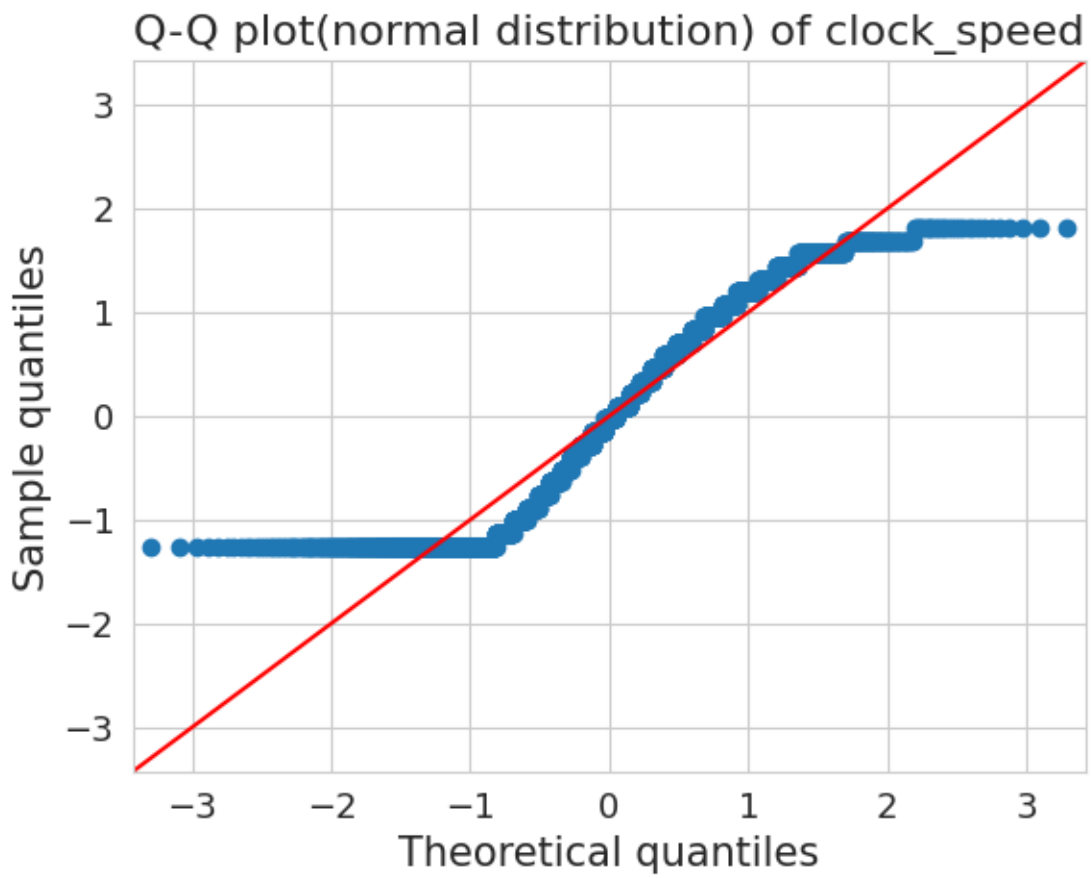
[669]:
```
#QQ-Plot to check for Normal distribution of numerical features
from statsmodels.graphics.gofplots import qqplot
for c in num_cols:
            plt.figure(figsize=(8,4))
            fig=qqplot(dataset_predictors[c],line="45",fit='True')
            plt.xticks(fontsize=13)
            plt.yticks(fontsize=13)
            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()
```

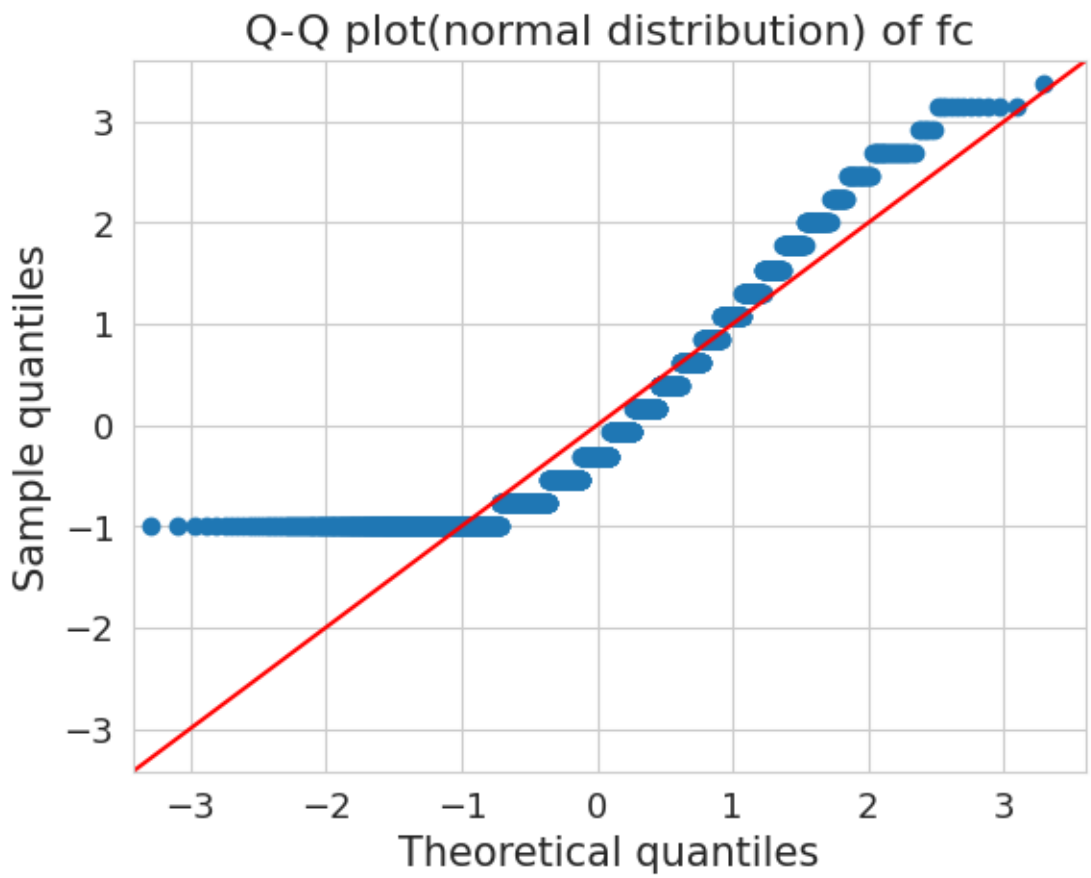
```
<Figure size 800x400 with 0 Axes>
```



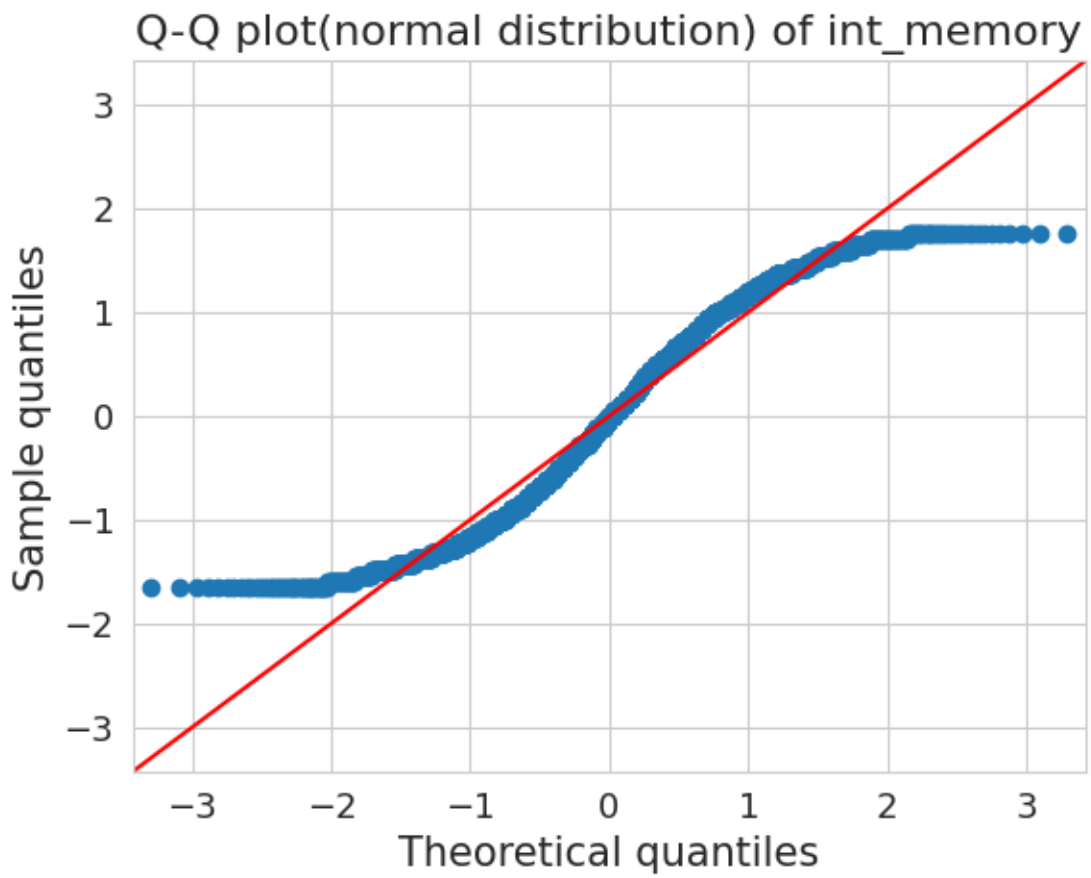

```
<Figure size 800x400 with 0 Axes>
```



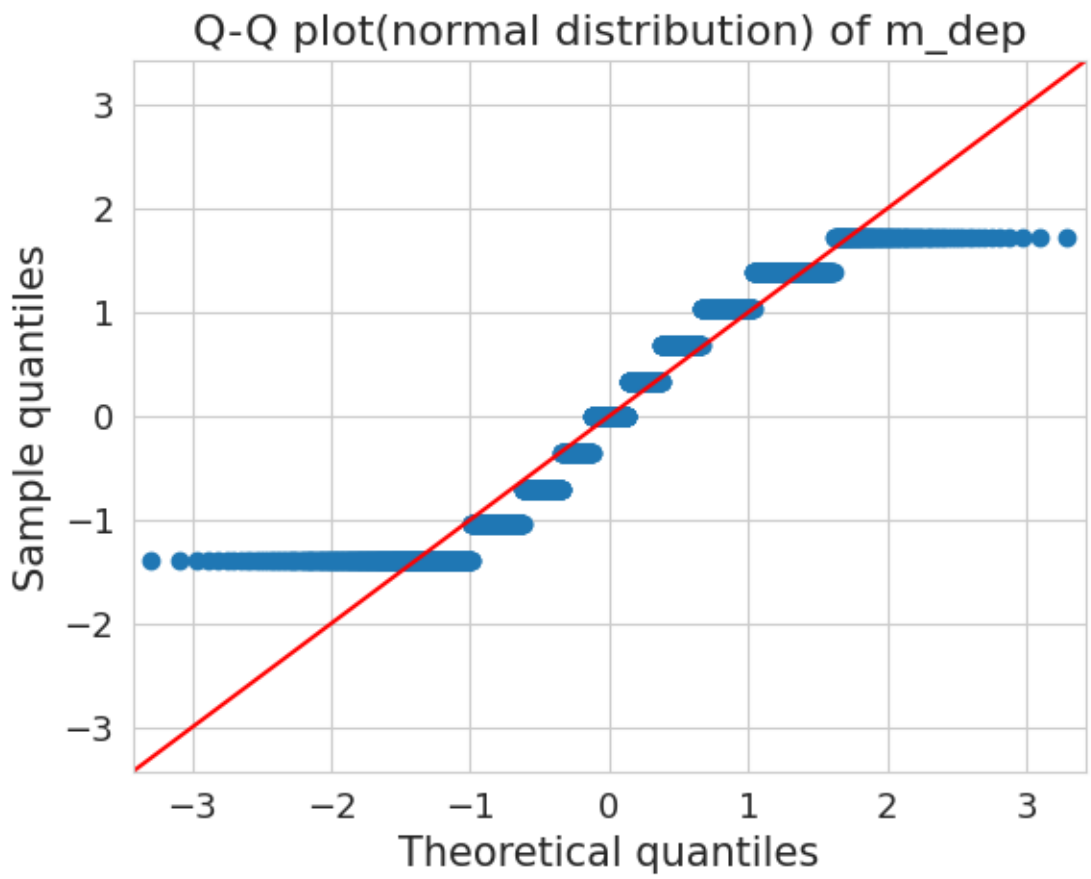

```
<Figure size 800x400 with 0 Axes>
```



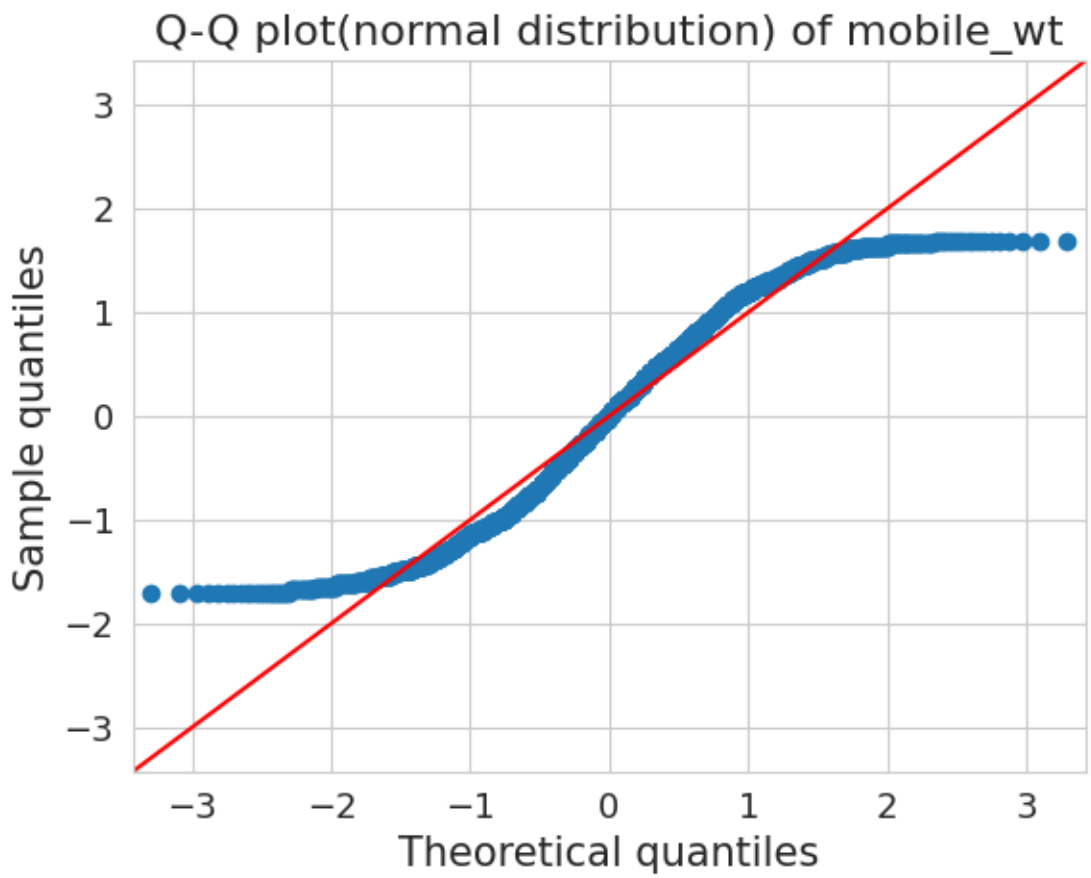

```
<Figure size 800x400 with 0 Axes>
```



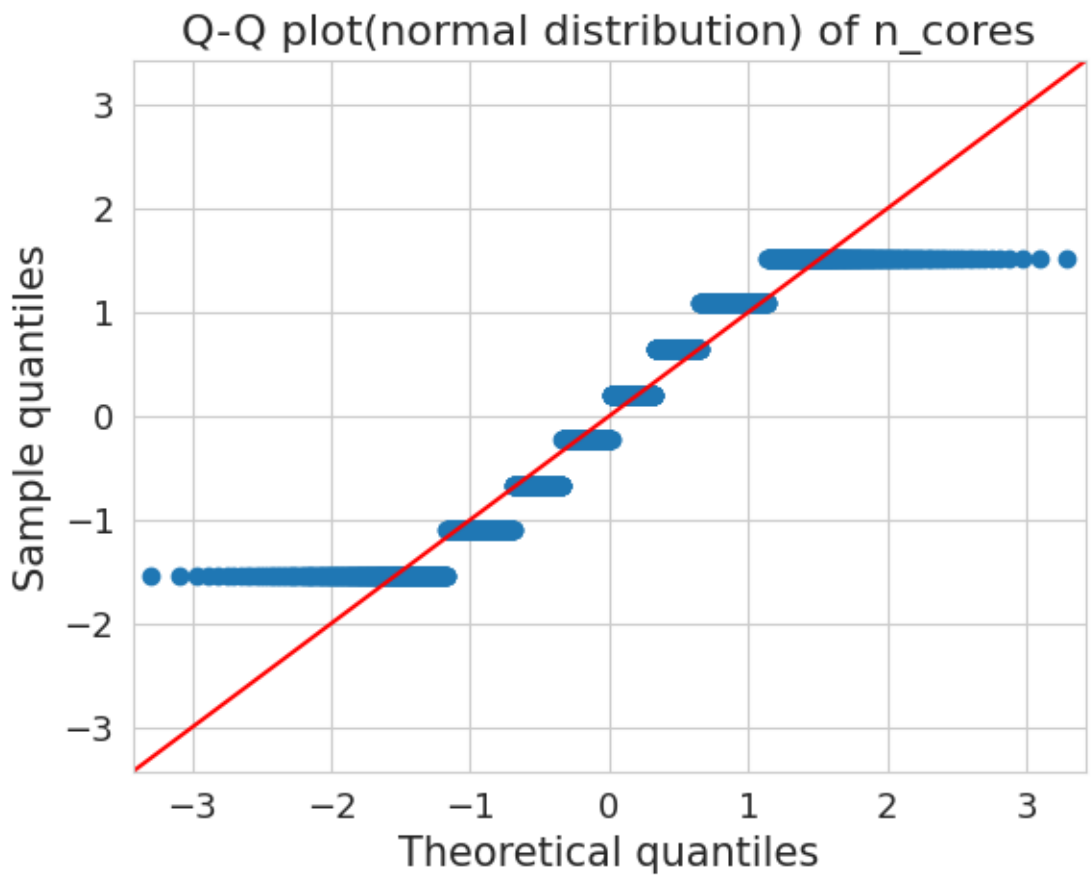

```
<Figure size 800x400 with 0 Axes>
```



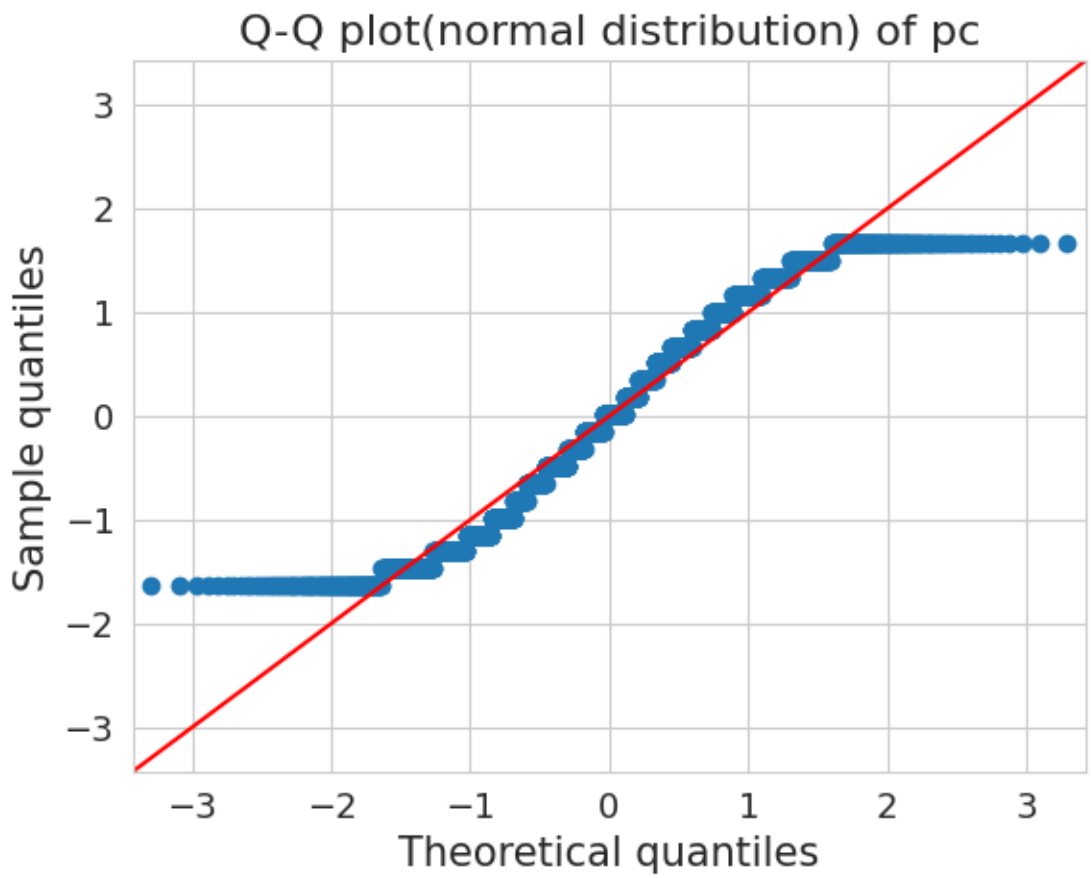

```
<Figure size 800x400 with 0 Axes>
```



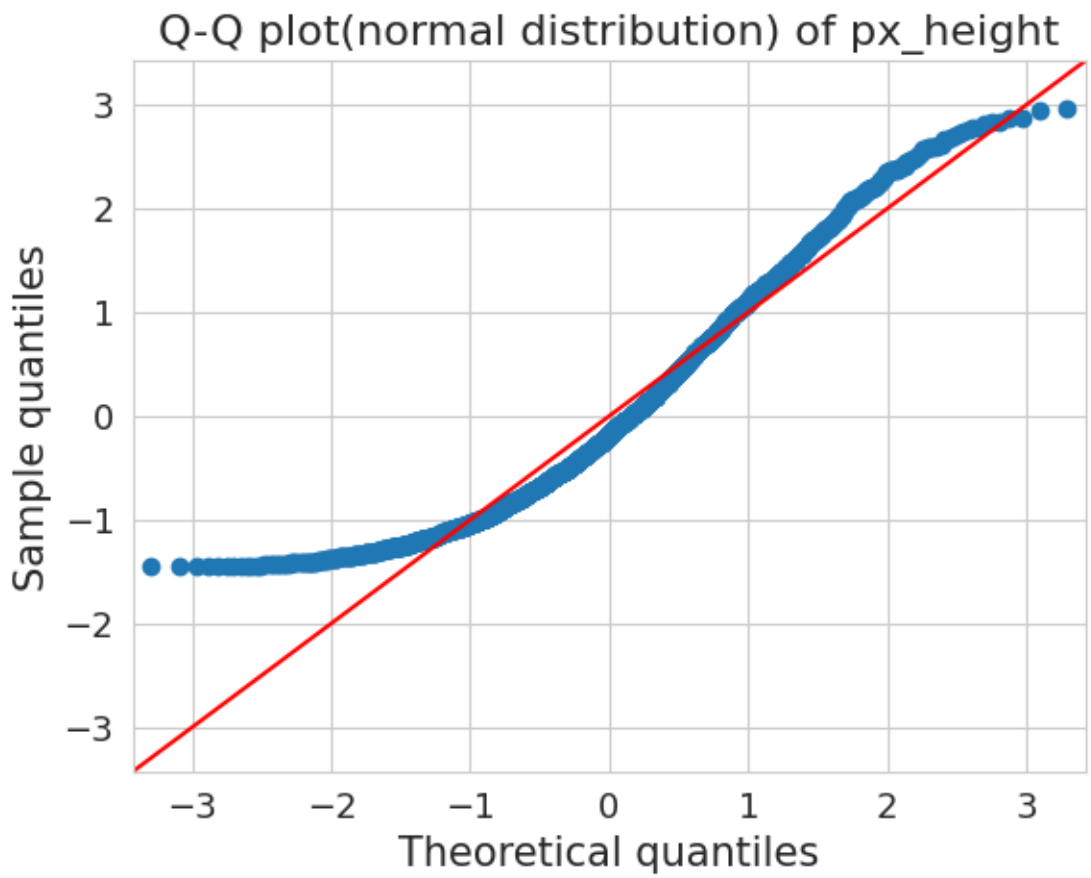

```
<Figure size 800x400 with 0 Axes>
```



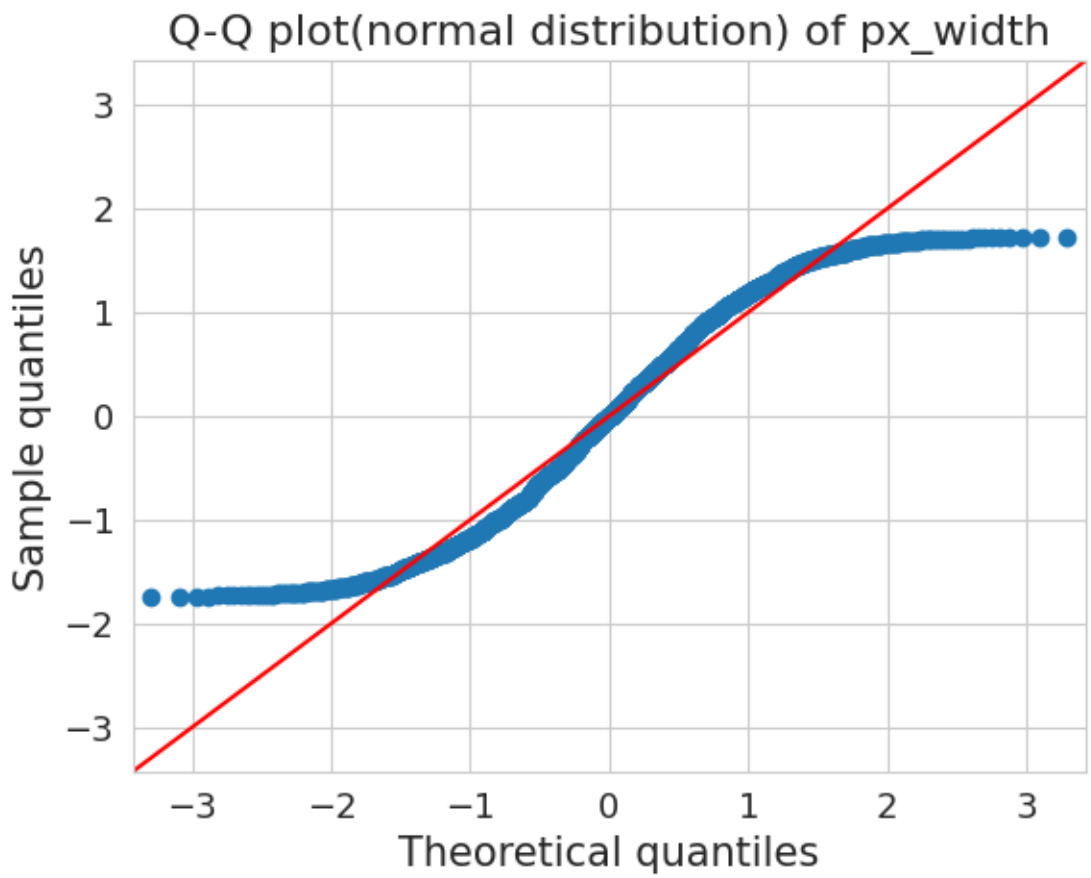

```
<Figure size 800x400 with 0 Axes>
```



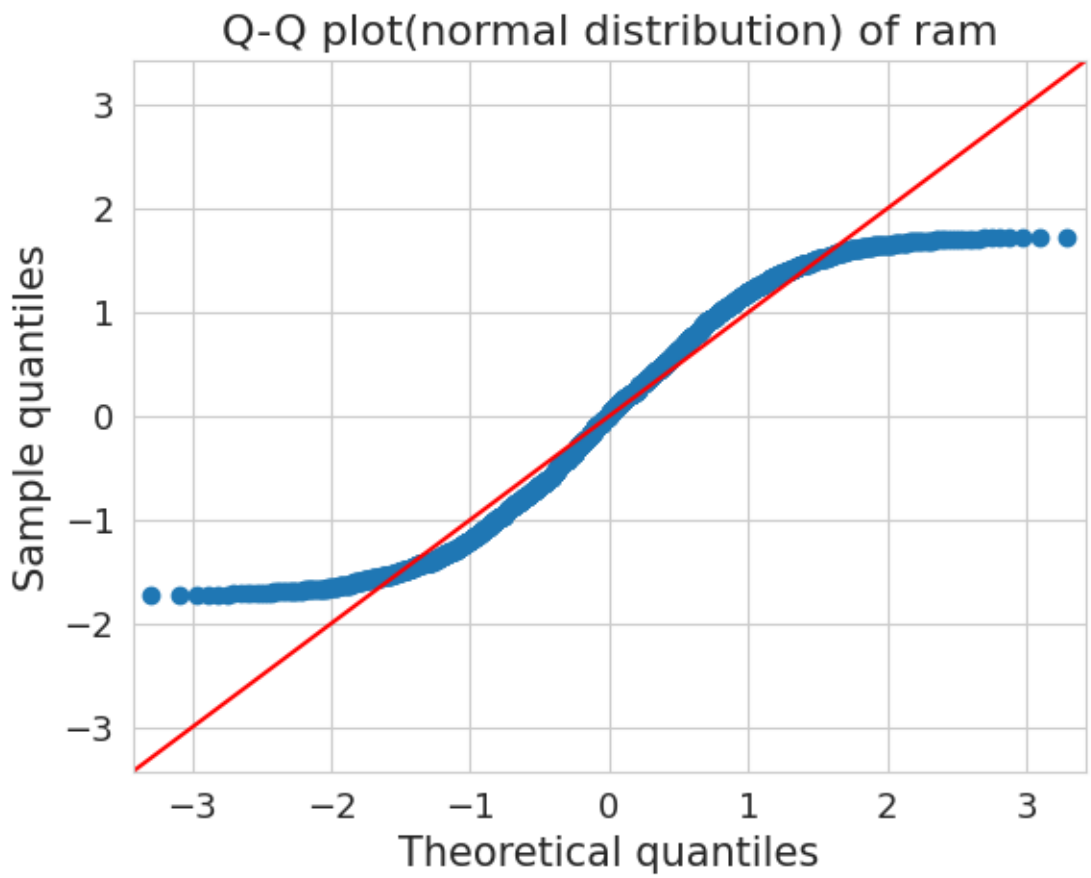

```
<Figure size 800x400 with 0 Axes>
```



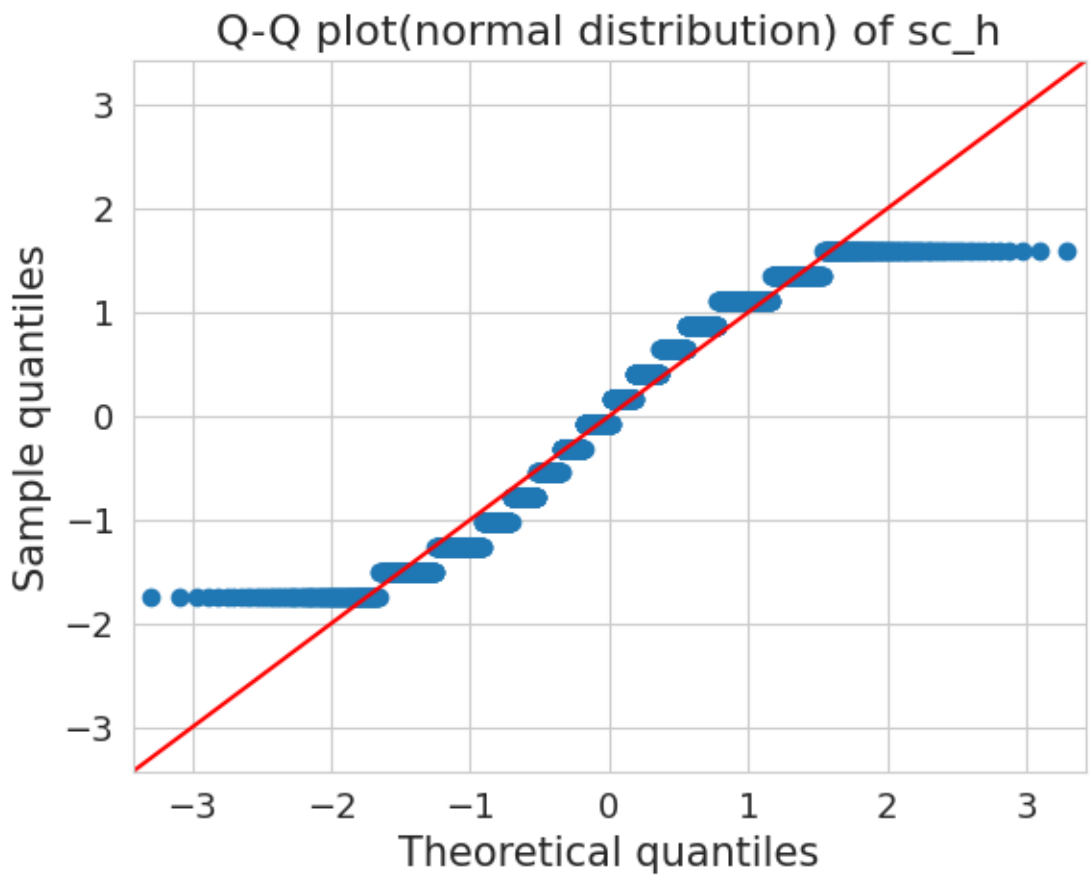

```
<Figure size 800x400 with 0 Axes>
```



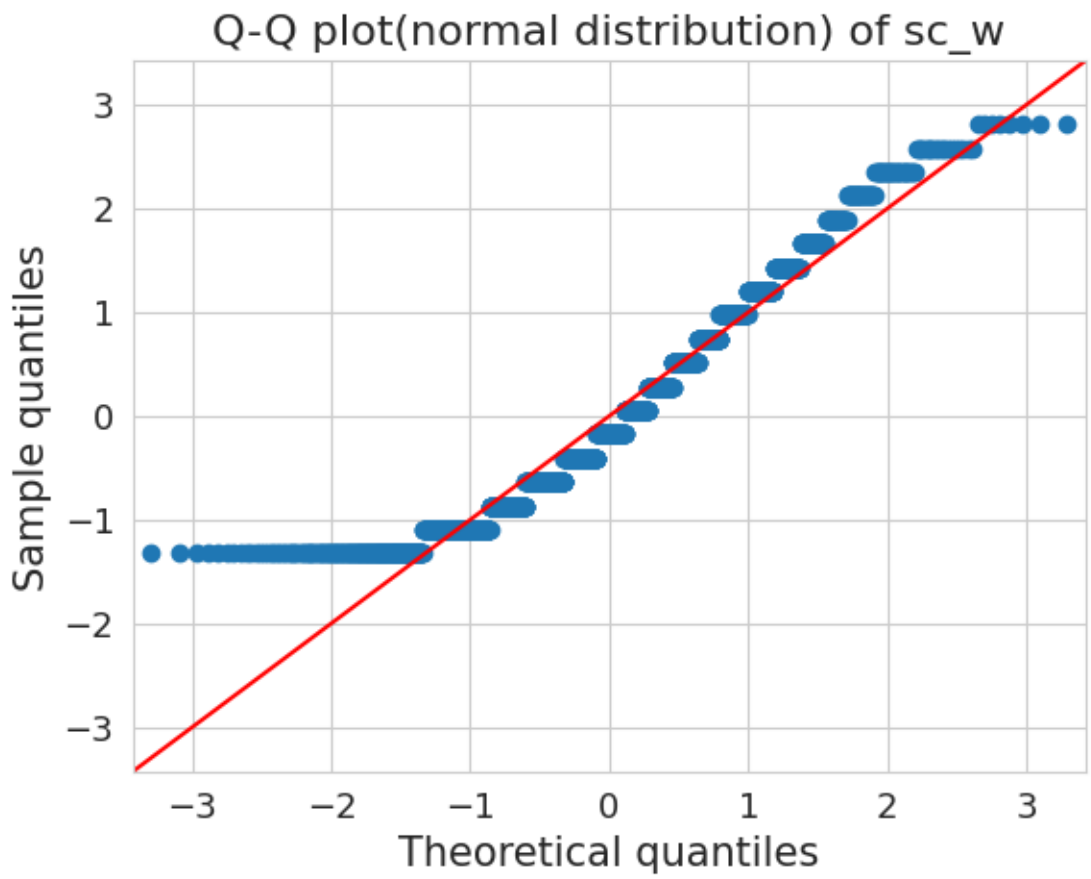

```
<Figure size 800x400 with 0 Axes>
```



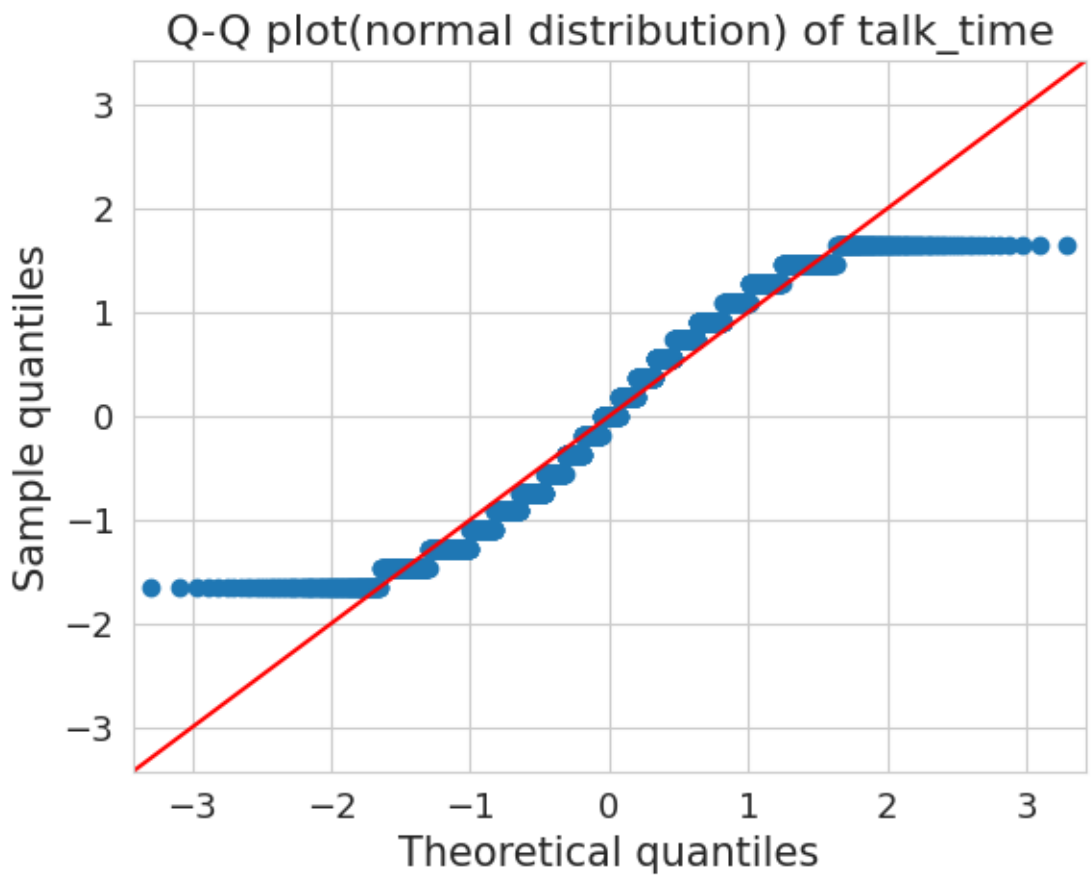

```
<Figure size 800x400 with 0 Axes>
```




```
<Figure size 800x400 with 0 Axes>
```




```
<Figure size 800x400 with 0 Axes>
```



[669]:

```
```

0.4.2 Train-Test Split

[670]:

```
X_train,X_test,y_train,y_test=train_test_split(dataset_predictors,dataset_target,test_size=0.
↪2)
```

0.4.3 Feature Scaling

[671]:

```
scaler=StandardScaler()
X_train[num_cols]=scaler.fit_transform(X_train[num_cols])
X_test[num_cols]=scaler.transform(X_test[num_cols])
X_scaled=pd.concat([X_train,X_test],axis=0)
y_scaled=pd.concat([y_train,y_test],axis=0)
```

[672]:

```
ld = LinearDiscriminantAnalysis()
ld.fit(X_train, y_train)
y_pred = ld.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

```
print("Accuracy:", accuracy)
```

```
Accuracy: 0.93
```

###Cross Validation

[673]:
```
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)
scores = []
for train_index, test_index in skf.split(X_scaled, y_scaled):
    x_train_fold, x_test_fold = X_scaled.to_numpy()[train_index], X_scaled.
 ↪to_numpy()[test_index]
    y_train_fold, y_test_fold = y_scaled.to_numpy()[train_index], y_scaled.
 ↪to_numpy()[test_index]
    ld.fit(x_train_fold, y_train_fold)
    scores.append(ld.score(x_test_fold, y_test_fold))
```

[674]:
```
scores
```

[674]:
```
[0.93, 0.935, 0.94, 0.95, 0.96, 0.945, 0.975, 0.94, 0.97, 0.96]
```

[675]:
```
np.mean(scores)
```

[675]:
```
0.9504999999999999
```

The best performing model gave an accuracy of 97%. The mean accuracy was 94%, which is very good performance.

1 Conclusion

The application of a Linear Discriminant Analysis model for two different datasets is carried out. It is observed that LDA has very specific requirements, which if not satisfied, may result in subpar performance as observed for the first dataset.

2 References

- Linear Discriminant Analysis (LDA) in Python
- Label Binarizer

3 LICENSE

Copyright <2022> 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.

Crash Course in Statistical Learning Quiz Questions

Due Apr 24 by 11:59pm **Points** 25 **Submitting** a file upload **Available** until Apr 25 at 11:59pm

This assignment was locked Apr 25 at 11:59pm.

You must have 10 quiz questions with answers for your section.

Crash Course in Statistical Learning Quiz Questions

Sai Kashyap Cheruku
NUID-002756849

1) **Which of the following statements about exploratory data analysis (EDA) is false?**

A) EDA can help identify outliers and anomalies in the dataset.
B) EDA can help determine the relationships between variables in the dataset.
C) EDA is typically performed after the feature selection process.
D) EDA can involve the use of statistical tests and hypothesis testing.

Ans) C) EDA is typically performed after the feature selection process.

2) **Which of the following statements is true about the standard error in machine learning?**

A) It measures the average deviation of the estimated regression line from the true regression line.
B) It measures the average deviation of the predicted values from the true values in the training data.
C) It measures the average deviation of the predicted values from the true values in the testing data.
D) It measures the variability of the sample mean from multiple random samples of the same size.

Ans) C) It measures the average deviation of the predicted values from the true values in the testing data.

3) **In regression analysis, the R2 metric is used to:**

A) Measure the proportion of the variance in the dependent variable that is explained by the independent variable
B) Measure the significance of the independent variable in predicting the dependent variable
C) Measure the correlation between the independent and dependent variables
D) Measure the error between the predicted and actual values of the dependent variable

Ans) A) Measure the proportion of the variance in the dependent variable that is explained by the independent variable.

4) **In a simple linear regression analysis with 30 observations, the estimated coefficient for the independent variable is 2.5 and the standard error of the coefficient is 0.8. Using a significance level of 0.05, what is the critical t-value for a two-tailed test?**

A) 1.96
B) 2.045
C) 2.042
D) 2.021

Ans) B) 2.045.

5) In a random forest model, what is the purpose of the "bagging" process?

A) To reduce the variance of the individual decision trees
B) To increase the bias of the individual decision trees
C) To increase the correlation between the individual decision trees
D) To increase the complexity of the individual decision trees

Ans) A) To reduce the variance of the individual decision trees.

6) What is the purpose of using multiple filters in a convolutional neural network (CNN)?

A) To reduce the computational cost of the network
B) To increase the size of the input image
C) To extract multiple features from the input image
D) To decrease the number of convolutional layers in the network

Ans) C) To extract multiple features from the input image.

7)Which of the following regularization techniques is specifically designed to address the problem of multicollinearity in linear regression models?

A) L1 regularization
B) L2 regularization
C) Elastic Net regularization
D) Ridge regression

Ans) D) Ridge regression.

8)What is the main purpose of the backpropagation algorithm in neural networks?

A) To compute the output of the neural network given an input
B) To update the weights of the neural network to minimize the error on a training set
C) To prevent overfitting of the neural network to the training data
D) To increase the size of the neural network by adding more layers

Ans)B) To update the weights of the neural network to minimize the error on a training set.

9)Which of the following is a common method to prevent overfitting in neural networks?

A) Increasing the learning rate
B) Decreasing the number of training epochs
C) Adding more hidden layers to the network
D) Using regularization techniques such as dropout or weight decay

Ans) D) Using regularization techniques such as dropout or weight decay.

10) Which of the following statements is true about soft margin SVMs?

A) Soft margin SVMs always have better generalization performance than hard margin SVMs.
B) Soft margin SVMs allow for misclassifications in the training data.
C) Soft margin SVMs are only applicable to linearly separable data.
D) Soft margin SVMs do not require any tuning of the regularization parameter.

Ans) B) Soft margin SVMs allow for misclassifications in the training data.

Use Dall-E to Create Artwork for Medium Articles and/or Educational Content

New Attempt

Due Apr 22 by 11:59pm **Points** 5 **Submitting** a file upload

Use Dall-E (or some other generative art AI) to Create Artwork for your Medium articles.

See the following article on how to do this https://medium.com/aiguys/how-to-use-dall-e-to-create-artwork-for-medium-articles-and-or-educational-content-50e2af9662ed

See **Using the OpenAI website for Dall-E** (https://openai.com/dall-e-2/)

Prompts

1) An Indian graduate student trying to climb a hill which is shaped like a normal distribution
2) A cartoon depicting a lion and deer, separated by barbed wire, rk laxman style
3) A mountain shaped like gaussian distribution
4) A sketch depicting Bayesian Decision Theory
5) A line of cars in an assembly line, with different colors

20 Exam Questions Part A

Due Apr 22 by 11:59pm **Points** 100 **Submitting** a website url or a file upload
Available until Apr 23 at 11:59pm

This assignment was locked Apr 23 at 11:59pm.

20 Exam Questions Part A

Create 20, 5-point moderately hard exam questions. You will be graded on whether your questions are related to the topics listed on the sample exam page, their correctness, and whether or not they are trivial. To generate using ChatGPT you might try the following:

GPT Generated Quiz Questions/Responses

Sai Kashyap Cheruku
NUID-002756849

1)
Prompt-> Generate a moderately hard multiple choice question and answer, on the topic of SVM kernels.
Question: How does the choice of kernel function affect the computational complexity of SVMs?
A) Choosing a linear kernel leads to a higher computational complexity compared to non-linear kernels.
B) Choosing a non-linear kernel leads to a higher computational complexity compared to linear kernels.
C) The choice of kernel function does not affect the computational complexity of SVMs.
D) The computational complexity of SVMs depends on the size of the dataset, and is not affected by the choice of kernel function.
Answer: B) Choosing a non-linear kernel leads to a higher computational complexity compared to linear kernels.
Explanation: The choice of kernel function affects the computational complexity of SVMs. Using a linear kernel results in a simpler optimization problem and a lower computational complexity compared to using a non-linear kernel. This is because the kernel trick used to project the data into a higher-dimensional space in order to make it linearly separable requires more computation for non-linear kernels. Therefore, choosing a non-linear kernel leads to a higher computational complexity compared to linear kernels. However, the overall computational complexity of SVMs also depends on the size of the dataset, as larger datasets require more computation to optimize the SVM.

2)
Prompt-> Generate a moderately hard multiple choice question, with answer and explanation on variable importance in multiple linear regression.
Question: When using the LASSO method for variable selection in multiple linear regression, how does the penalty parameter affect the importance of the predictor variables?
A) Increasing the penalty parameter leads to a decrease in the importance of all predictor variables.
B) Increasing the penalty parameter leads to an increase in the importance of all predictor variables.
C) Increasing the penalty parameter leads to a decrease in the importance of some predictor variables and an increase in the importance of others.
D) The penalty parameter does not affect the importance of the predictor variables in LASSO.
Answer: C) Increasing the penalty parameter leads to a decrease in the importance of some predictor variables and an increase in the importance of others.
Explanation: The LASSO (Least Absolute Shrinkage and Selection Operator) method is a technique used for variable selection in multiple linear regression. It works by adding a penalty term to the sum of the absolute values of the coefficients. The penalty parameter, denoted by lambda (λ), controls the strength of the penalty term. When λ is large, the penalty term is more dominant, and the LASSO method shrinks the coefficients towards zero, resulting in variable selection.
As the penalty parameter λ increases, some coefficients are shrunk more towards zero than others, resulting in the decrease in the importance of some predictor variables and an increase in the importance of others. The coefficients that are shrunk to exactly zero are excluded from the model, and the remaining coefficients are deemed the most important predictors. The LASSO method therefore provides a way to perform both variable selection and regularization in multiple linear regression.

3)

Prompt-> Generate a difficult multiple choice question, with answer and explanation on the log likelihood concept in logistic regression.
Sure, here's a challenging question on the log-likelihood concept in logistic regression:
Question: Which of the following statements is true about the log-likelihood function in logistic regression?
A) The log-likelihood function is maximized when the predicted probabilities of the logistic regression model match the observed binary outcomes.
B) The log-likelihood function is always negative and is minimized when the predicted probabilities of the logistic regression model match the observed binary outcomes.
C) The log-likelihood function is proportional to the sum of the squared errors between the predicted probabilities and the observed binary outcomes.
D) The log-likelihood function is only used to assess the model fit, and has no role in parameter estimation.

Answer: A) The log-likelihood function is maximized when the predicted probabilities of the logistic regression model match the observed binary outcomes.
Explanation: The log-likelihood function is a measure of the fit of the logistic regression model to the data. It is the logarithm of the likelihood function, which is the product of the predicted probabilities of the logistic regression model for each observation. The log-likelihood function is a concave function and is maximized when the predicted probabilities of the logistic regression model match the observed binary outcomes. In other words, the log-likelihood function is maximized when the model correctly predicts the binary outcomes for all observations in the dataset.

Option B is incorrect because the log-likelihood function is not always negative; it can be positive or negative depending on the model fit. Option C is incorrect because the log-likelihood function is not proportional to the sum of the squared errors; it is the logarithm of the likelihood function. Option D is incorrect because the log-likelihood function plays a central role in parameter estimation in logistic regression; the maximum likelihood estimation method is used to estimate the model parameters by maximizing the log-likelihood function.

4)
Prompt-> **Generate a moderately difficult multiple choice question, with answer and explanation on the topic of bayesian decision theory.**
Question: In Bayesian decision theory, what is the posterior distribution?

A) The probability distribution of the decision made by the decision maker.
B) The probability distribution of the loss function.
C) The probability distribution of the true state of nature, given the observed data.
D) The probability distribution of the prior probabilities.

Answer: C) The probability distribution of the true state of nature, given the observed data.

Explanation: In Bayesian decision theory, the posterior distribution is the probability distribution of the true state of nature, given the observed data. It is computed using Bayes' theorem, which states that the posterior distribution is proportional to the product of the prior distribution and the likelihood function.

The prior distribution represents the decision maker's initial belief about the distribution of the true state of nature before observing any data. The likelihood function represents the probability of observing the data given a particular value of the true state of nature. The posterior distribution combines the prior distribution and the likelihood function to update the decision maker's belief about the true state of nature after observing the data.

Option A is incorrect because the posterior distribution is not the probability distribution of the decision made by the decision maker; it is the probability distribution of the true state of nature. Option B is incorrect because the posterior distribution is not the probability distribution of the loss function; it is the probability distribution of the true state of nature. Option D is incorrect because the posterior distribution is not the probability distribution of the prior probabilities; it is the probability distribution of the true state of nature.

5)
Prompt->Generate a moderately hard question, with answer on the topic of normalization in machine learning.
Question: Which of the following normalization techniques is suitable for handling outliers in the data?

A) Min-Max scaling
B) Z-score normalization
C) Robust scaling
D) Log transformation

Answer: C) Robust scaling

Explanation: Robust scaling is a normalization technique that is suitable for handling outliers in the data. It is based on the median and interquartile range (IQR) instead of the mean and standard deviation, which are sensitive to outliers. In robust scaling, the data is first centered by subtracting the median, and then scaled by dividing by the IQR. This ensures that the scaling is not influenced by outliers and preserves the distribution of the data.

Option A is incorrect because Min-Max scaling is not suitable for handling outliers because it is sensitive to extreme values. Option B is incorrect because Z-score normalization is also sensitive to outliers because it is based on the mean and standard deviation. Option D is incorrect because log transformation is a data transformation technique that is not specifically designed for normalization or handling outliers.

6)
Prompt->Generate a moderately hard multiple choice question with answer on the concept of random forest regression.

Question: In random forest regression, what is the purpose of the out-of-bag (OOB) error estimate?

A) To estimate the generalization error of the model.
B) To measure the importance of each predictor variable in the model.
C) To optimize the number of trees in the forest.
D) To improve the accuracy of the model by reducing overfitting.

Answer: A) To estimate the generalization error of the model.

Explanation: The out-of-bag (OOB) error estimate is a method for estimating the generalization error of a random forest regression model. It is calculated by using only the observations that were not included in the bootstrap sample for each tree to predict the response variable. The OOB prediction error is then calculated as the difference between the predicted and actual response values. The average OOB prediction error across all trees is a measure of the model's generalization error.

Option B is incorrect because variable importance is typically measured using measures such as mean decrease impurity or mean decrease accuracy, which are not related to the OOB error estimate. Option C is incorrect because the number of trees in the forest is typically selected based on a validation set or cross-validation, not the OOB error estimate. Option D is incorrect because the OOB error estimate is a measure of the model's generalization error, not a method for reducing overfitting.

7)
Prompt-> Generate a multiple choice question with answers on the topic of feature selection methods.
Question: Which of the following feature selection methods is specifically designed for high-dimensional datasets?

A) Recursive feature elimination
B) Principal component analysis
C) Linear discriminant analysis
D) Stability selection

Answer: D) Stability selection

Explanation: Stability selection is a hybrid feature selection method that combines filter-based and wrapper-based approaches to handle high-dimensional datasets. It works by randomly subsampling the original dataset and applying a feature selection algorithm to each subsample to identify a set of stable features that are consistently selected across multiple subsamples. The stability of each feature is measured by a stability score that reflects the frequency of its selection across subsamples. The most stable features are then selected as the final set of features.

Option A is incorrect because recursive feature elimination is a wrapper-based feature selection method that may not perform well on high-dimensional datasets due to the large number of possible feature subsets. Option B is incorrect because principal component analysis (PCA) is a dimensionality reduction technique that may not preserve the interpretability of the original features. Option C is incorrect because linear discriminant analysis (LDA) is a supervised learning method that assumes a linear relationship between the features and the target variable, and may not work well on high-dimensional datasets with complex nonlinear relationships.

8)
Prompt-> Generate a difficult multiple choice question, with multiple answers on the topic of bias-variance trade-off.
Question: Which of the following statements accurately describe the bias-variance trade-off in machine learning?

A) Models with high bias tend to underfit the training data, while models with high variance tend to overfit the training data.
B) Increasing the complexity of a model generally leads to a decrease in bias and an increase in variance.
C) Regularization techniques can help control the trade-off between bias and variance.
D) The optimal balance between bias and variance depends on the specific problem and the amount of available training data.
Answer: A), B), C), and D)

Explanation: The bias-variance trade-off is a fundamental concept in machine learning that refers to the trade-off between the error due to bias and the error due to variance in a model. Models with high

bias tend to underfit the training data and have poor predictive performance on new data, while models with high variance tend to overfit the training data and have poor generalization performance on new data. Increasing the complexity of a model generally leads to a decrease in bias and an increase in variance, while decreasing the complexity of a model generally leads to an increase in bias and a decrease in variance. Regularization techniques can help control the trade-off between bias and variance by adding a penalty term to the objective function that discourages large weights and complex models. The optimal balance between bias and variance depends on the specific problem and the amount of available training data, as a model with more complexity may perform better if there is enough data to estimate the model parameters accurately.

9)
Prompt-> Generate a difficult multiple choice question, with multiple answers on the topic of ensemble learning.
Sure, here's a difficult multiple choice question with multiple answers on ensemble learning:

Question: Which of the following statements are true regarding ensemble learning?

A) Ensemble learning involves combining multiple models to improve the overall predictive performance.
B) Ensemble learning can only be used with decision trees.
C) Bagging and boosting are two popular ensemble learning methods.
D) Ensemble learning always produces better results than using a single model.

Answer: A) and C)

Explanation: Ensemble learning is a technique that involves combining multiple models to improve the overall predictive performance. By using a group of models instead of a single model, ensemble learning can reduce the variance of the predictions and improve the generalization ability of the model. Bagging and boosting are two popular ensemble learning methods. Bagging (bootstrap aggregating) involves training multiple models on different subsets of the training data and combining their predictions using a majority vote. Boosting involves iteratively training weak models on the misclassified samples from previous iterations and combining their predictions using a weighted sum. Option B is incorrect because ensemble learning can be used with any type of model, not just decision trees. Option D is incorrect because while ensemble learning can often improve the performance of a model, there are cases where a single model may perform better, particularly if the data is small or the models are not diverse enough.

10)
Prompt-> Generate a moderately hard multiple choice question with answers on the topic of hypothesis testing in machine learning

Question: What is a Type I error in hypothesis testing?
A) Rejecting the null hypothesis when it is true.
B) Failing to reject the null hypothesis when it is false.
C) Rejecting the alternative hypothesis when it is true.
D) Failing to reject the alternative hypothesis when it is false.

Answer: A)

Explanation: A Type I error in hypothesis testing occurs when the null hypothesis is rejected when it is actually true. In other words, a Type I error occurs when the significance level (alpha) is set too low, leading to the rejection of the null hypothesis when it should have been accepted. This type of error is

also known as a false positive. Option B describes a Type II error, where the null hypothesis is not rejected when it is actually false. Option C is incorrect because the alternative hypothesis is not rejected or accepted in hypothesis testing, only the null hypothesis. Option D is not a valid concept in hypothesis testing.

11)
Prompt->Generate a moderately difficult multiple choice question on the topic of gradient descent

Question: Which of the following statements about gradient descent is true?
A) Gradient descent always converges to the global minimum of the cost function.
B) Gradient descent can converge to a local minimum of the cost function depending on the starting point.
C) Gradient descent can only be used with convex cost functions.
D) Gradient descent updates all parameters simultaneously in each iteration.

Answer: B)

Explanation: Gradient descent is an iterative optimization algorithm used to minimize the cost function in machine learning. Option A is false because gradient descent can sometimes converge to a local minimum rather than the global minimum depending on the starting point and shape of the cost function. Option C is also false because gradient descent can be used with non-convex cost functions, although it may not always converge to a minimum. Option D is false because gradient descent updates the parameters one at a time in a specific order, such as using the Jacobian matrix or a random permutation.

12)
Prompt-> Generate a multiple choice question with answer on the topic of backpropogation in CNNs

Sure, here's a multiple choice question with answer on the topic of backpropagation in CNNs:

Question: What is the purpose of backpropagation in convolutional neural networks (CNNs)?

A) To calculate the activation values of each neuron in the network.
B) To adjust the weights and biases in the network based on the error between the predicted and actual output.
C) To convolve the input image with a set of learned filters to extract features.
D) To pool the output of the convolutional layers to reduce the size of the feature maps.

Answer: B)

Explanation: Backpropagation is an algorithm used to adjust the weights and biases in a neural network based on the error between the predicted and actual output. In CNNs, backpropagation is used to propagate the error through the network from the output layer to the input layer, and then adjust the weights and biases in the convolutional and fully connected layers. This process allows the network to learn from its mistakes and improve its predictions over time. Option A is incorrect because calculating the activation values of each neuron is done during the forward pass of the network. Option C is incorrect because convolution is performed during the forward pass to extract features, not during backpropagation. Option D is incorrect because pooling is also done during the forward pass to reduce the size of the feature maps.

13)

Prompt->Generate a moderately hard multiple choice question on the concept of clustering methods in machine learning.
Question: Which of the following clustering algorithms is most suitable for identifying clusters of varying shapes and sizes?

A) K-means clustering
B) Hierarchical clustering
C) DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
D) Mean Shift clustering

Answer: C) DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Explanation: DBSCAN is a density-based clustering algorithm that is effective at identifying clusters of varying shapes and sizes, as it does not require clusters to have a spherical shape or be of equal size. It works by defining clusters as dense regions of points that are separated by areas of lower density. This makes DBSCAN particularly useful in scenarios where clusters may be irregularly shaped or where there may be varying densities of data points within and between clusters. In contrast, K-means clustering and Mean Shift clustering are both centroid-based algorithms, which require clusters to have a spherical shape and be of equal size. Hierarchical clustering can be used to identify clusters of varying shapes and sizes, but it may not be as effective as DBSCAN in scenarios where clusters are irregularly shaped or where there is varying density of data points within and between clusters.

14)
Prompt-> Generate a very difficult multiple choice question ,with answers on the topic of model interpretability in machine learning.
Question: Which of the following model interpretability techniques is specifically designed to address the issue of sparsity and high-dimensional data?

A) Permutation feature importance
B) Partial dependence plots
C) Shapley values
D) Integrated gradients
E) Anchor explanations

Correct answer: E) Anchor explanations

Explanation: Permutation feature importance, partial dependence plots, Shapley values, and integrated gradients are all widely-used model interpretability techniques. However, none of them are specifically designed to address the issue of sparsity and high-dimensional data. Anchor explanations, on the other hand, is a technique that was specifically developed for sparse and high-dimensional data. It creates interpretable rules that apply to individual instances in the dataset, and can handle both categorical and continuous features.

15)
Prompt-> Generate a moderately difficult multiple choice question, with answers on the topic of optimizers in neural networks.
Question: Which of the following optimizers is specifically designed for use with recurrent neural networks (RNNs)?

A) Stochastic Gradient Descent (SGD)

B) Adagrad
C) Adam
D) RMSprop

Answer: D) RMSprop

Explanation: RMSprop is an optimizer that adapts the learning rate of each weight based on the magnitudes of recent gradients. It is particularly effective for RNNs, which are known to have issues with vanishing gradients. SGD, Adagrad, and Adam are all commonly used optimizers, but they are not specifically designed for use with RNNs.

16)
Prompt-> Generate a moderately hard multiple choice question, with answers on the topic of CNNs.
Question: What is the purpose of the activation function in a convolutional neural network (CNN)?

A) To help the network learn the optimal values for the filter weights
B) To introduce nonlinearity into the network and enable it to learn complex patterns
C) To prevent overfitting of the training data
D) To regulate the learning rate of the network during backpropagation

Answer: B) To introduce nonlinearity into the network and enable it to learn complex patterns

Explanation: The activation function is a key component of a CNN and is applied to the output of each convolutional or dense layer. Its purpose is to introduce nonlinearity into the network, allowing it to learn complex patterns in the data. Without an activation function, a CNN would be a series of linear transformations, which would limit its ability to learn complex features. While the filter weights are optimized during training, this is not the purpose of the activation function. While regularization techniques can be used to prevent overfitting, the activation function is not specifically designed for this purpose. Similarly, learning rate is controlled through the use of an optimizer, not the activation function.

17)
Prompt-> Generate a multiple choice question, with answer on the topic of activation functions in neural networks.

Question: Which of the following is a drawback of using the sigmoid activation function in neural networks?

A) It can cause the vanishing gradient problem
B) It can cause the exploding gradient problem
C) It is not differentiable
D) It is not bounded

Answer: A) It can cause the vanishing gradient problem. The sigmoid function has a derivative that is bounded between 0 and 0.25, which means that gradients can become very small as they propagate backwards through many layers. This can make it difficult for the network to learn, especially in deep architectures. The other options (B, C, and D) are not drawbacks of the sigmoid function. The exploding gradient problem can occur with other activation functions, such as the hyperbolic tangent (tanh), and the sigmoid function is in fact differentiable and bounded between 0 and 1.

18)

Prompt->Generate a difficult multiple choice question, with answer on the topic of dropout in neural network

Question: Which of the following techniques can be used to mitigate the problem of overfitting and improve the performance of a neural network suffering from dropout?

A) Increase the number of hidden layers in the network
B) Decrease the number of training epochs
C) Increase the dropout rate during training
D) Decrease the learning rate during training

Answer: C) Increase the dropout rate during training.

Explanation: Dropout is a regularization technique used to prevent overfitting in neural networks. During training, dropout randomly selects neurons in a layer and drops them out of the network with a certain probability. This forces the network to learn more robust features and prevents it from relying too heavily on any one feature. Increasing the dropout rate during training will increase the probability that neurons are dropped out, making the network more robust to overfitting. Increasing the number of hidden layers, decreasing the number of training epochs, or decreasing the learning rate may help prevent overfitting in some cases, but they are not directly related to dropout regularization.

19)
Prompt-> Generate a moderately difficult multiple choice question, with answer on the topic of r squared metric in machine learning.
Question: Which of the following statements regarding the R-squared metric in machine learning is true?

A) R-squared measures the accuracy of a classification model.
B) R-squared always takes a value between 0 and 1.
C) A high R-squared value indicates that the model is overfitting the data.
D) R-squared is unaffected by the number of independent variables used in the model.

Answer: B) R-squared always takes a value between 0 and 1.

Explanation: R-squared is a metric used to measure how well a regression model fits the data, and it always takes a value between 0 and 1, where 0 indicates that the model explains none of the variability in the dependent variable and 1 indicates that the model explains all of the variability in the dependent variable. Option A is incorrect because R-squared is used for regression models, not classification models. Option C is incorrect because a high R-squared value typically indicates that the model is a good fit for the data, although it is possible for a model to overfit the data and have a high R-squared value. Option D is incorrect because the number of independent variables used in the model can affect the value of R-squared, and it is important to use methods like adjusted R-squared to account for the number of variables in the model.

20)
Prompt-> Generate a moderately difficult multiple question on the topic of SVM.

Question: Which of the following statements regarding SVMs is true?

A) SVMs always converge to a global optimum.
B) SVMs with a linear kernel are unable to capture complex relationships in the data.

C) SVMs with a radial basis function (RBF) kernel are prone to overfitting when the gamma parameter is set too high.
D) The soft margin SVM allows for misclassifications of training data points, which can result in better generalization performance.

Answer: C) SVMs with a radial basis function (RBF) kernel are prone to overfitting when the gamma parameter is set too high.
Explanation: While SVMs are powerful machine learning models, there are several caveats to their use. Option A is incorrect because the optimization problem that SVMs solve is convex but not guaranteed to converge to a global optimum in all cases. Option B is incorrect because SVMs with a linear kernel can still capture complex relationships if the data is separable by a hyperplane. Option C is correct because the gamma parameter controls the shape of the RBF kernel, and if it is set too high, the model can fit the noise in the training data and result in poor generalization performance. Finally, option D is incorrect because the soft margin SVM allows for misclassifications of training data points, but this can lead to worse generalization performance if the model overfits to the training data.

Neural Network Type Classification | TMNIST

New Attempt

Due Apr 16 by 11:59pm **Points** 25 **Submitting** a file upload

Neural Network Type Classification | TMNIST

Find a "Hello World" Handwritten Digit Classification (MNIST) tutorial and create a similar tutorial for one of the TMNIST data sets.

The tutorial must be posted on Kaggle associated with the data set you use (see **https://www.kaggle.com/nikbearbrown/tmnist-alphabet-94-characters/code (https://www.kaggle.com/nikbearbrown/tmnist-alphabet-94-characters/code)**) Your notebook MUST not be identical to the other notebooks associated with the data.

Nice Notebook

https://www.kaggle.com/code/sheshngupta/tminst-character-recognition-94-4-accuracy (https://www.kaggle.com/code/sheshngupta/tminst-character-recognition-94-4-accuracy)

The TMNIST data sets

TMNIST Alphabet (94 characters)
Typography MNIST (TMNIST) with 94 characters and over 281,000 images

https://www.kaggle.com/nikbearbrown/tmnist-alphabet-94-characters/ (https://www.kaggle.com/nikbearbrown/tmnist-alphabet-94-characters/)

TMNIST-Glyphs (MNIST Style Glyph images)
Typography MNIST (TMNIST): Over 500000 images of 1812 glyphs in 2990 font styles

https://www.kaggle.com/nimishmagre/tmnist-glyphs-1812-characters (https://www.kaggle.com/nimishmagre/tmnist-glyphs-1812-characters)

TMNIST (Typeface MNIST)
A database of Typeface-based digits

https://www.kaggle.com/nimishmagre/tmnist-typeface-mnist (https://www.kaggle.com/nimishmagre/tmnist-typeface-mnist)

Nice model Notebook

https://www.kaggle.com/code/sheshngupta/tminst-character-recognition-94-4-accuracy (https://www.kaggle.com/code/sheshngupta/tminst-character-recognition-94-4-accuracy)

Grading Rubric

40 % (accuracy)

Minus 1 point for every 1% accuracy below 95% for ***TMNIST (Typeface MNIST)***

Minus 1 point for every 1% accuracy below 90% for ***TMNIST Alphabet (94 characters) or TMNIST-Glyphs (MNIST Style Glyph images)***

10 % (Uniqueness)

Up to 10 points may be deducted if your notebook looks too similar to the other notebooks posted (i.e. You will get 10 points if it looks like you wrote it and didn't copy it)

50 % (Explanation)

How well did you explain the math, code, and ideas behind your neural network?

neuralnetworkmnist

December 16, 2023

[52]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
↪outside of the current session
```

```
/kaggle/input/tmnist-alphabet-94-characters/94_character_TMNIST.csv
```

1 Abstract

The purpose of this notebook is to potray the procedures undertaken to build a neural-network model which is expected to classify grayscale images with high accuracy. The image dataset is taken from the Typography-MNIST collection.

The dataset contains 94 unique alphabetical characters in thousands of different font variations. Each sample contains:
* The font specification * A 28*28 grayscale pixel representation of the alphabet* The label of the sample(given instance of the alphabetical character)

Convolutional Neural Networks (CNNs) are a specific type of neural network that are particularly effective for image recognition and other computer vision applications. They are designed to automatically learn and extract features from images in a hierarchical manner.

We try to employ a CNN model to perform successful classification of the data samples.

[53]:
```
#Importing the necessary libraries
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
```

[54]:
```
# Reading Data
df = pd.read_csv('/kaggle/input/tmnist-alphabet-94-characters/
 ↪94_character_TMNIST.csv')
df.head(20)
```

[54]:

	names	labels	1	2	3	4	5	\
0	Salsa-Regular	6	0.0	0.0	0.0	0.0	0.0	
1	MouseMemoirs-Regular	D	0.0	0.0	0.0	0.0	0.0	
2	Creepster-Regular	f	0.0	0.0	0.0	0.0	0.0	
3	SeoulNamsan-Light	/	0.0	0.0	0.0	0.0	0.0	
4	HachiMaruPop-Regular	F	0.0	0.0	0.0	0.0	0.0	
5	Livvic-BoldItalic	x	0.0	0.0	0.0	0.0	0.0	
6	Sarabun-ExtraBold	J	0.0	0.0	0.0	0.0	0.0	
7	FiraSans-Light	8	0.0	0.0	0.0	0.0	0.0	
8	AlegreyaSansSC-BoldItalic	H	0.0	0.0	0.0	0.0	0.0	
9	NotoSerifSC-Light	k	0.0	0.0	0.0	0.0	0.0	
10	Taviraj-ExtraBold	@	0.0	0.0	0.0	0.0	0.0	
11	Philosopher-Bold	1	0.0	0.0	0.0	0.0	0.0	
12	Aleo-Regular	=	0.0	0.0	0.0	0.0	0.0	
13	Bellota-Italic	g	0.0	0.0	0.0	0.0	0.0	
14	SairaSemiCondensed-Medium	)	0.0	0.0	0.0	0.0	0.0	
15	AveriaSansLibre-Regular	2	0.0	0.0	0.0	0.0	0.0	
16	Aladin-Regular	d	0.0	0.0	0.0	0.0	0.0	
17	MontserratAlternates-ExtraLightItalic	^	0.0	0.0	0.0	0.0	0.0	
18	IstokWeb-Bold	3	0.0	0.0	0.0	0.0	0.0	
19	Simonetta-Black	@	0.0	0.0	0.0	0.0	0.0	

	6	7	8	…	775	776	777	778	779	780	781	782	783	784
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
1	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
2	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
3	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
4	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
5	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
6	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

```
7   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
8   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
9   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
10  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
11  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
12  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
13  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
14  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
15  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
16  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
17  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
18  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
19  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

[20 rows x 786 columns]
```

[55]:
```python
#Columns with non-zero null count
len([l for l in df.isnull().sum() if l!=0])
```

[55]: 0

[56]:
```python
df.describe()
```

[56]:
```
              1         2         3         4         5         6  
count  274093.0  274093.0  274093.0  274093.0  274093.0  274093.0  \
mean        0.0       0.0       0.0       0.0       0.0       0.0
std         0.0       0.0       0.0       0.0       0.0       0.0
min         0.0       0.0       0.0       0.0       0.0       0.0
25%         0.0       0.0       0.0       0.0       0.0       0.0
50%         0.0       0.0       0.0       0.0       0.0       0.0
75%         0.0       0.0       0.0       0.0       0.0       0.0
max         0.0       0.0       0.0       0.0       0.0       0.0

                   7              8              9             10  …
count  274093.000000  274093.000000  274093.000000  274093.000000  …  \
mean        0.000004       0.000011       0.001043       0.001518  …
std         0.001910       0.004271       0.475743       0.494410  …
min         0.000000       0.000000       0.000000       0.000000  …
25%         0.000000       0.000000       0.000000       0.000000  …
50%         0.000000       0.000000       0.000000       0.000000  …
75%         0.000000       0.000000       0.000000       0.000000  …
max         1.000000       2.000000     248.000000     245.000000  …

                 775            776            777            778
count  274093.000000  274093.000000  274093.000000  274093.000000  \
mean        0.002638       0.001562       0.001131       0.000894
std         0.320080       0.179093       0.150590       0.189578
```

```
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max       103.000000      59.000000      41.000000      88.000000

                779            780            781            782
count  274093.000000  274093.000000  274093.000000  274093.000000  \
mean        0.000489       0.000317       0.000204       0.000033
std         0.083345       0.067396       0.053345       0.012813
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max        21.000000      20.000000      19.000000       6.000000

                783       784
count  274093.000000  274093.0
mean        0.000033       0.0
std         0.012813       0.0
min         0.000000       0.0
25%         0.000000       0.0
50%         0.000000       0.0
75%         0.000000       0.0
max         6.000000       0.0

[8 rows x 784 columns]
```

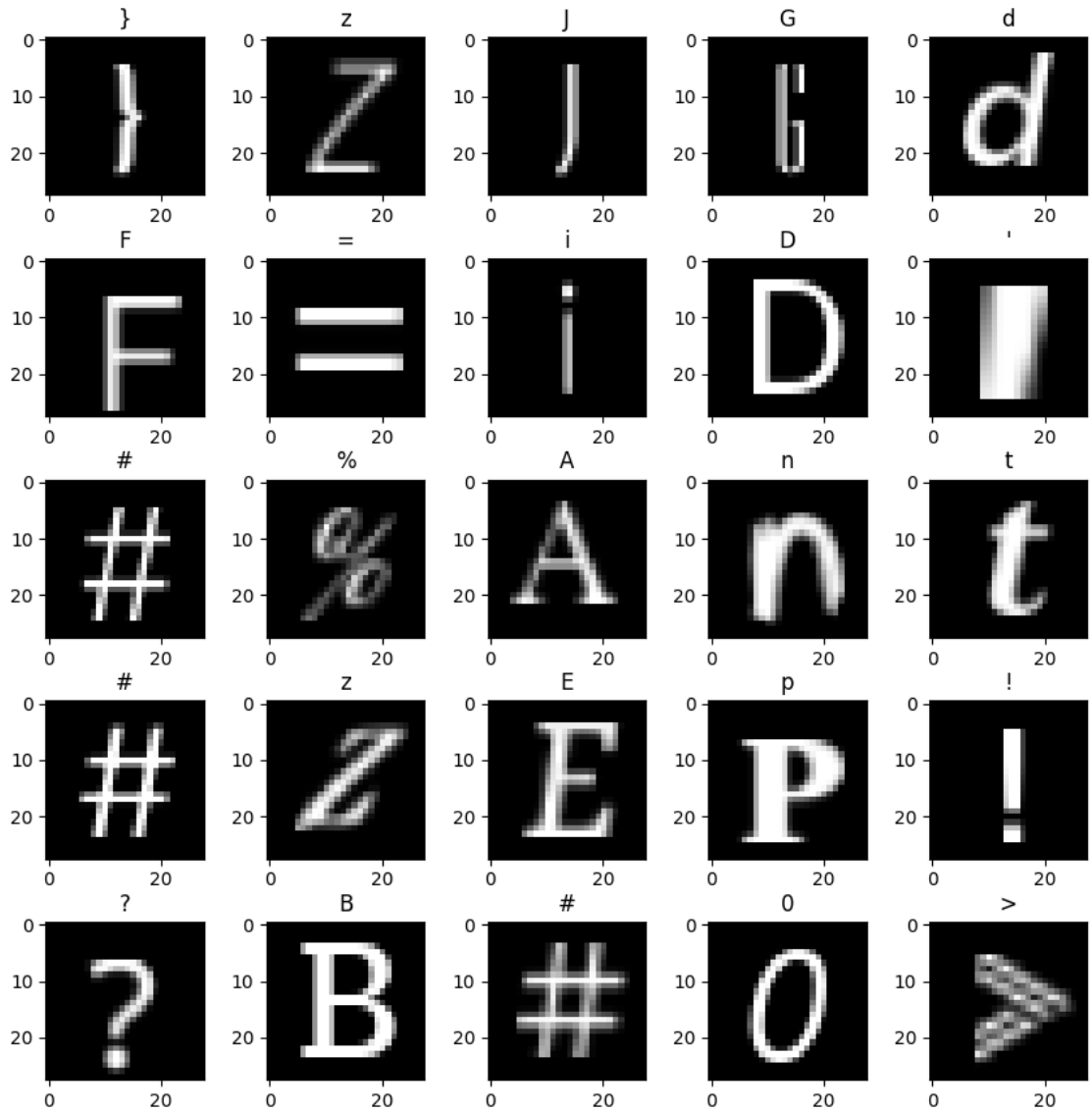
Scaling is needed for neural networks for several reasons:

- **Computational efficiency**: Neural networks can have a large number of parameters, and performing computations with large matrices can be computationally expensive. Scaling the input data to a smaller range can help reduce the computational burden.
- **Gradient Descent**: Gradient descent is a popular optimization algorithm used in training neural networks. However, it can be sensitive to the scale of the input features. Scaling the input features to a similar range can help ensure that the gradient descent algorithm converges more quickly and efficiently.
- **Regularization**: Scaling the input features can help with regularization by preventing some features from dominating the others. This can help prevent overfitting and improve the generalization performance of the model.

[57]:
```
#Number of unique instances of label class
labels=len(df['labels'].unique())
```

[58]:
```
#Plotting sample glyphs
def plot_im(idx):
  fig_=df.iloc[idx:idx+1,2:].to_numpy().reshape(28,28)
```

```
  plt.imshow(fig_,cmap='gray')
  plt.title(df.iloc[idx:idx+1,1:2].to_numpy()[0][0])
plt.figure(figsize=(10,10))
plt.subplots_adjust(left=0.1,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
indicies = np.random.randint(len(df), size=25)
for i in range(25):
        plt.subplot(5,5,i+1)
        idx =indicies[i]
        plot_im(idx)
plt.show()
```



[59]:
```
#Scaling the numerical data

scaler=MinMaxScaler()
fit=scaler.fit(df.iloc[:,2:])
scaled_pixels_df=pd.DataFrame(fit.transform(df.iloc[:,2:]),columns=range(1,785))
```

[60]:
```
# Created a scaled DataFrame and rearranging the columns
scaled_df=pd.concat([df.iloc[:,:2],scaled_pixels_df],axis=1)
cols=scaled_df.columns.to_list()
cols=[cols[0]]+cols[2:]+[cols[1]]
scaled_df=scaled_df[cols]
```

[61]:
```
#dropping the target column
df_names=df[['names']]
label_df=df[['labels']]
scaled_df=scaled_df.drop(['names','labels'],axis=1)
labels=df['labels'].unique().size
```

[62]:
```
# Splitting the data into train and test components
X_train,X_test,y_train,y_test=train_test_split(scaled_df,label_df,test_size=0.2)
X_train = np.reshape(X_train.values, (X_train.shape[0] ,28, 28,1))
X_test = np.reshape(X_test.values, (X_test.shape[0] ,28, 28,1))

# Encoding the target variable
y_train = OneHotEncoder().fit_transform(y_train)
y_test = OneHotEncoder().fit_transform(y_test )

y_train=y_train.toarray()
y_test=y_test.toarray()
```

2 Using a Convolutional Neural Network

Convolutional Neural Networks (CNNs) are designed to process input data such as pixels in an image. The process of a CNN involves several stages of convolutional layers, activation functions, pooling layers, and fully connected layers.

During training, the CNN adjusts the weights and biases of its neurons to minimize the error between the predicted output and the actual output. This process is known as backpropagation, and it allows the CNN to learn the features and patterns that are important for the given task.

Overall, the working of a CNN involves learning the important features and patterns of the input data through several layers of computations, allowing for accurate classification or prediction.

- Convolution: A convolution operation is the core building block of a CNN. It involves taking a small filter (also called a kernel or a weight matrix) and sliding it over the input image to compute a dot product between the filter and the corresponding portion of the input image. The result is a feature map that captures the presence of certain visual patterns or features in the image. The convolution operation can be expressed mathematically as:

  $$(f * w)(i, j) = \sum_m \sum_n f(m, n)w(i - m, j - n)$$

  Here, f is the input image, w is the filter, and i, j are the spatial coordinates of the output feature map.

- Pooling: After each convolutional layer, a pooling operation is often used to downsample the feature maps and reduce their dimensionality. The most common pooling operation is max pooling, which involves taking the maximum value within a local neighborhood of the feature map. The pooling operation can be expressed mathematically as:

  $$f_{pooled}(i, j) = max_{m,n} f(i + m, j + n)$$

Here, f is the input feature map, and f_{pooled} is the output pooled feature map.

- Activation functions: Activation functions are used to introduce non-linearity into the CNN and enable it to learn complex mappings between the input and output. The most common activation function used in CNNs is the Rectified Linear Unit (ReLU), which is defined as:

$$f(x) = max(0, x)$$

Here, x is the input to the activation function.

[63]:
```
from tensorflow.keras.models import Sequential,load_model
from tensorflow.keras.layers import Dense, Conv2D,␣
↪MaxPool2D,Flatten,Dropout,BatchNormalization
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import EarlyStopping
```

[64]:
```
model = Sequential ()
model.add(Conv2D(filters = 32 , kernel_size = (3,3),input_shape = (28, 28,1),␣
↪padding = 'same',))
model.add(BatchNormalization())
model.add(tf.keras.layers.Activation('relu'))
model.add(MaxPool2D(pool_size = (2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 64 , kernel_size = (3,3) ,padding = 'same'))
model.add(BatchNormalization())
model.add(tf.keras.layers.Activation('relu'))
model.add(MaxPool2D(pool_size = (2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 128 , kernel_size = (3,3) ,padding = 'same'))
model.add(BatchNormalization())
model.add(tf.keras.layers.Activation('relu'))
model.add(MaxPool2D(pool_size = (2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(filters = 256 , kernel_size = (3,3) ,padding = 'same'))
model.add(BatchNormalization())
model.add(tf.keras.layers.Activation('relu'))
model.add(MaxPool2D(pool_size = (2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(350))
model.add(BatchNormalization())
model.add(tf.keras.layers.Activation('relu'))
model.add(Dropout(0.25))
model.add(Dense(labels, activation = 'softmax'))
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_16 (Conv2D)          (None, 28, 28, 32)        320

 batch_normalization_20 (Bat  (None, 28, 28, 32)       128
 chNormalization)

 activation_20 (Activation)  (None, 28, 28, 32)        0

 max_pooling2d_16 (MaxPoolin  (None, 14, 14, 32)       0
 g2D)

 dropout_20 (Dropout)        (None, 14, 14, 32)        0

 conv2d_17 (Conv2D)          (None, 14, 14, 64)        18496

 batch_normalization_21 (Bat  (None, 14, 14, 64)       256
 chNormalization)

 activation_21 (Activation)  (None, 14, 14, 64)        0

 max_pooling2d_17 (MaxPoolin  (None, 7, 7, 64)         0
 g2D)

 dropout_21 (Dropout)        (None, 7, 7, 64)          0

 conv2d_18 (Conv2D)          (None, 7, 7, 128)         73856

 batch_normalization_22 (Bat  (None, 7, 7, 128)        512
 chNormalization)

 activation_22 (Activation)  (None, 7, 7, 128)         0

 max_pooling2d_18 (MaxPoolin  (None, 3, 3, 128)        0
 g2D)

 dropout_22 (Dropout)        (None, 3, 3, 128)         0

 conv2d_19 (Conv2D)          (None, 3, 3, 256)         295168

 batch_normalization_23 (Bat  (None, 3, 3, 256)        1024
 chNormalization)

 activation_23 (Activation)  (None, 3, 3, 256)         0

 max_pooling2d_19 (MaxPoolin  (None, 1, 1, 256)        0
```

```
 g2D)

 dropout_23 (Dropout)        (None, 1, 1, 256)         0

 flatten_4 (Flatten)         (None, 256)               0

 dense_8 (Dense)             (None, 350)               89950

 batch_normalization_24 (Bat  (None, 350)              1400
 chNormalization)

 activation_24 (Activation)  (None, 350)               0

 dropout_24 (Dropout)        (None, 350)               0

 dense_9 (Dense)             (None, 94)                32994

=================================================================
Total params: 514,104
Trainable params: 512,444
Non-trainable params: 1,660
_________________________________________________________________
```

2.1 Optimizer Method

2.1.1 RMSprop

Gradient descent works by iteratively adjusting the parameters in the direction of steepest descent of the loss function, inorder to improve the accuracy of a machine learning model. Root Mean Square Propagation is an optimization algorithm commonly used to train neural networks. RMSprop aims to solve the problem of the vanishing or exploding gradient that can occur in a deep neural network.

The formula for RMSprop can be represented as:
s = β *** s + (1 -** β**) * (grad** 2)
w = w - (α * grad) / (sqrt(s) + ϵ)**

where:

- **s** is the exponentially weighted moving average of the squared gradients
- β is the decay rate for the moving average (usually set to 0.9)
- **grad** is the gradient of the cost function with respect to the parameters
- **w** is the current value of the parameters
- α is the learning rate hyperparameter
- ϵ is a small constant added to the denominator to avoid division by zero.

[65]:
```
# Declaring and defining the hyperparameters
optimizer  = RMSprop(learning_rate=0.008)
Batch_ = 128
Epochs_ = 60
```

```
model.compile(loss = 'categorical_crossentropy', optimizer = optimizer, metrics␣
↪= ['accuracy'])
```

[ ]:
```
history= model.fit(X_train,y_train, validation_data =␣
↪(X_test,y_test),batch_size = Batch_ ,
                   epochs = Epochs_, verbose = 1)
```

```
Epoch 1/60
 866/1714 [==============>…] - ETA: 39s - loss: 0.7469 - accuracy:
0.7860
```

2.2 Epoch v/s Accuracy

The epoch vs accuracy plot shows the accuracy of the model on the training and validation datasets as the model is trained over a number of epochs.

By analyzing this plot, one can determine if the model is learning well or if it is overfitting or underfitting. If the accuracy on the training dataset is increasing over time, but the accuracy on the validation dataset is not, then the model is likely overfitting. Conversely, if the accuracy on both datasets is not improving, then the model may be underfitting.

[ ]:
```
#Epoch v/s Accuracy plot
plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.show()
```

[ ]:
```
# Metric specification of predictions
predictions = get_preds(X_test)
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print("test accuracy: " + str(test_acc))
print("test loss: "+str(test_loss))
```

3 License

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 MER-

CHANTABILITY, 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.

[ ]:

```
```

Combine Data Cleaning, Feature Selection, Modeling and Interpretability into one report

Due Apr 9 by 11:59pm **Points** 100 **Submitting** a website url or a file upload
Available until Apr 15 at 11:59pm

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

Combine Data Cleaning, Feature Selection, Modeling, and Interpretability into one report (Every assignment except TMINST) and make it a cohesive readable report.

The focus is readability and clarity.

The following must be answered:

1. What is the question?

2. What did you do?

3. How well did it work?

4. What did you learn?

Compilation (1)

December 16, 2023

1 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 and Leverage Analysis
- Fit Machine Learning models to the data and study their performances
- Examine Feature importance and Feature Selection for each of these models.
  Finally, we include the usage of Automated Machine Learning API's such as AutoML to gain a lucid understanding of the ML process and outcomes.

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. 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.

[211]:
```
!pip install shap
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: shap in /usr/local/lib/python3.9/dist-packages
(0.41.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-
packages (from shap) (1.2.2)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.9/dist-
packages (from shap) (23.0)
Requirement already satisfied: tqdm>4.25.0 in /usr/local/lib/python3.9/dist-
packages (from shap) (4.65.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages
(from shap) (1.4.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages
(from shap) (1.22.4)
Requirement already satisfied: slicer==0.0.7 in /usr/local/lib/python3.9/dist-
packages (from shap) (0.0.7)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages
(from shap) (1.10.1)
Requirement already satisfied: numba in /usr/local/lib/python3.9/dist-packages
(from shap) (0.56.4)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.9/dist-
packages (from shap) (2.2.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-
packages (from numba->shap) (67.6.1)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in
/usr/local/lib/python3.9/dist-packages (from numba->shap) (0.39.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-
packages (from pandas->shap) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.9/dist-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from scikit-learn->shap) (3.1.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-
packages (from scikit-learn->shap) (1.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-
packages (from python-dateutil>=2.8.1->pandas->shap) (1.16.0)
```

[212]:
```
from sklearn.preprocessing import MinMaxScaler
```

[213]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
import statsmodels.api as sm
```

```
from sklearn.model_selection import␣
↪train_test_split,cross_val_score,StratifiedKFold,RandomizedSearchCV
from sklearn import tree
import pylab as py

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import␣
↪classification_report,confusion_matrix,ConfusionMatrixDisplay,accuracy_score,precision_scor
from sklearn.tree import DecisionTreeClassifier
!pip install h2o
import h2o
import shap
from statsmodels.stats.outliers_influence import variance_inflation_factor
from h2o.estimators import H2ORandomForestEstimator
from h2o.automl import H2OAutoML
from sklearn.ensemble import RandomForestClassifier
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: h2o in /usr/local/lib/python3.9/dist-packages
(3.40.0.3)
Requirement already satisfied: future in /usr/local/lib/python3.9/dist-packages
(from h2o) (0.18.3)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-
packages (from h2o) (2.27.1)
Requirement already satisfied: tabulate in /usr/local/lib/python3.9/dist-
packages (from h2o) (0.8.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from requests->h2o) (2022.12.7)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-
packages (from requests->h2o) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from requests->h2o) (1.26.15)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from requests->h2o) (2.0.12)
```

[214]:
```
from sklearn.neighbors import LocalOutlierFactor
import scipy as sc
```

#Questions and Responses * What is the question?
I have tried to predict the presence of breast tumor from the readings of a patient's cell biopsy parameters. The prediction outcomes of diagnosis are expected to be "malignant" and "benign", encoded as "M" and "B" in the given dataset. The classification of a given feature set is arrived at by using machine learning techniques, which captures the trends in fitted data and assigns credible predictions for unseen observations. * What did you do?
After opting for the machine learning approach to make future predictions, I performed EDA on the data. Data interpretation and visualization, imputation,outlier analysis and feature selection operations were performed. The model fitting process was implemented as a succesor for EDA, the

models were interpreted and performances were analyzed.

[215]:
```
#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 Exploratory Data Analysis

[216]:
```
#Displaying the first 10 rows of the dataset
df.head(10)
```

[216]:
```
         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]
```

[217]:
```
#Description of some statistical features pertaining to the dataset
df.describe()
```

[217]:
```
                 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]
```

Observations:
* The ranges of the independent variable is varying significantly.(A scaling operation should be performed, depending on the chosen model to fit) * Some of the feature values are labelled as 0.(This may or may not be a regular occurance)

[218]:
```
# Dropping the 'id' column
df=df.drop('id',axis=1)
```

[219]:
```
#Checking for NaN values
df.isnull().sum()
```

[219]:
```
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
```

[220]:
```
def dummy(a):
  if(a=='M'):
    return 1
  else:
    return 0
df_initial=df
```

3.1 Observing Modalities

[222]:
```
f, ax = plt.subplots(nrows=30, ncols=1, figsize=(5, 100))
for a in range(30):
    if(df.columns[a]=='diagnosis'):
      continue
    sns.kdeplot(df[df.columns[a]],ax=ax[a])
```

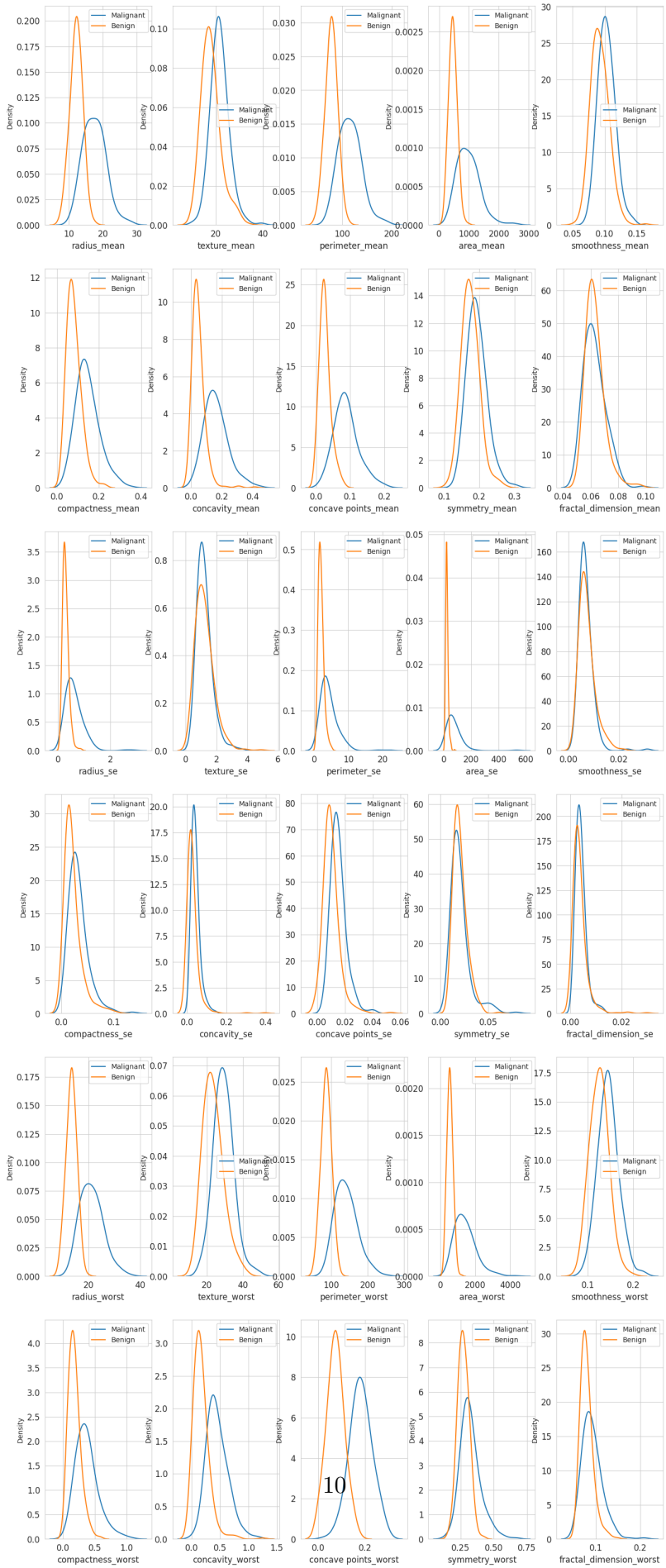
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
fractal_dimension_se
radius_worst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

It is inferred that the features follow unimodal and bimodal distributions.

3.2 Feature Distributions

[223]:
```
predictors=[f for f in df.columns if f not in ['diagnosis']]
i = 0
t0 = df[df['diagnosis'] == 'M']
t1 = df[df['diagnosis'] == 'B']

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 640x480 with 0 Axes>
```

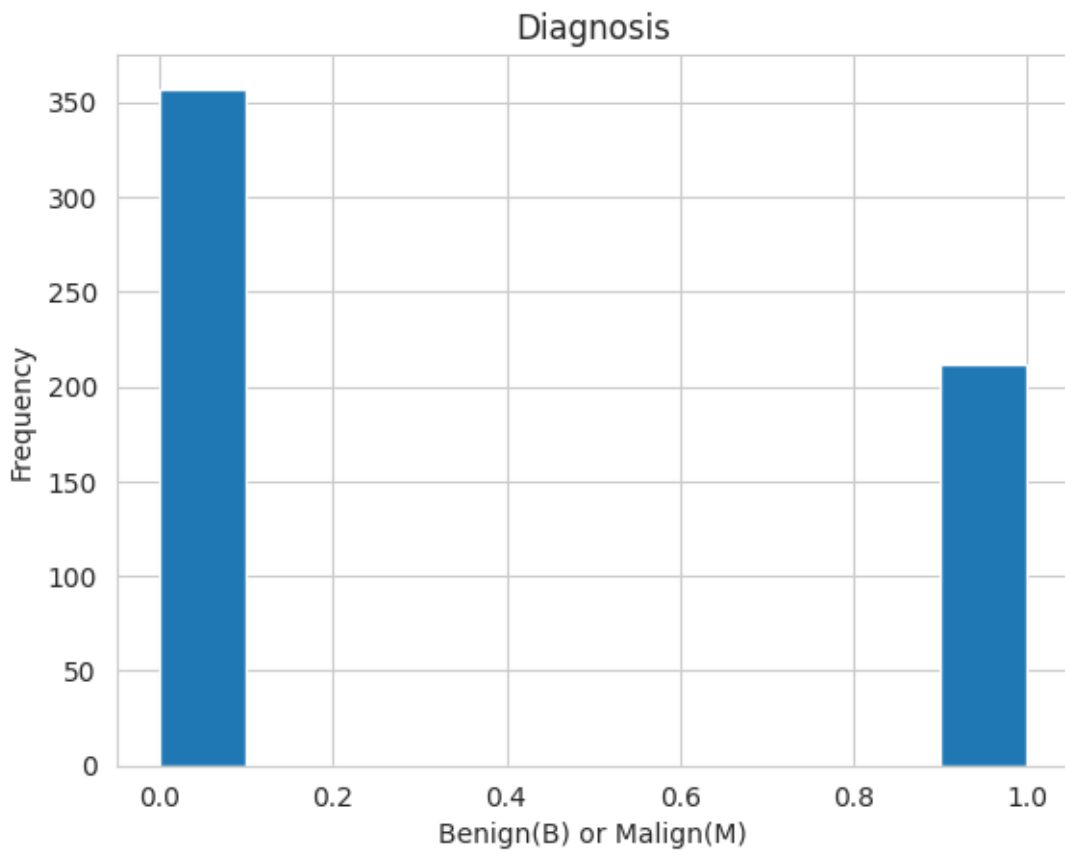

Malignant
Benign
Density
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
fractal_dimension_se
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 relationship between each of the predictor and target is not evident immediately, hence requiring further analysis.

[224]:
```
df_target=df['diagnosis'].apply(dummy)
df=df.drop(['diagnosis'],axis=1)
```

[225]:
```
#histogram plot for dependent feature
plt.hist(x=df_target)
plt.title("Diagnosis")
plt.xlabel("Benign(B) or Malign(M)")
plt.ylabel("Frequency")
```

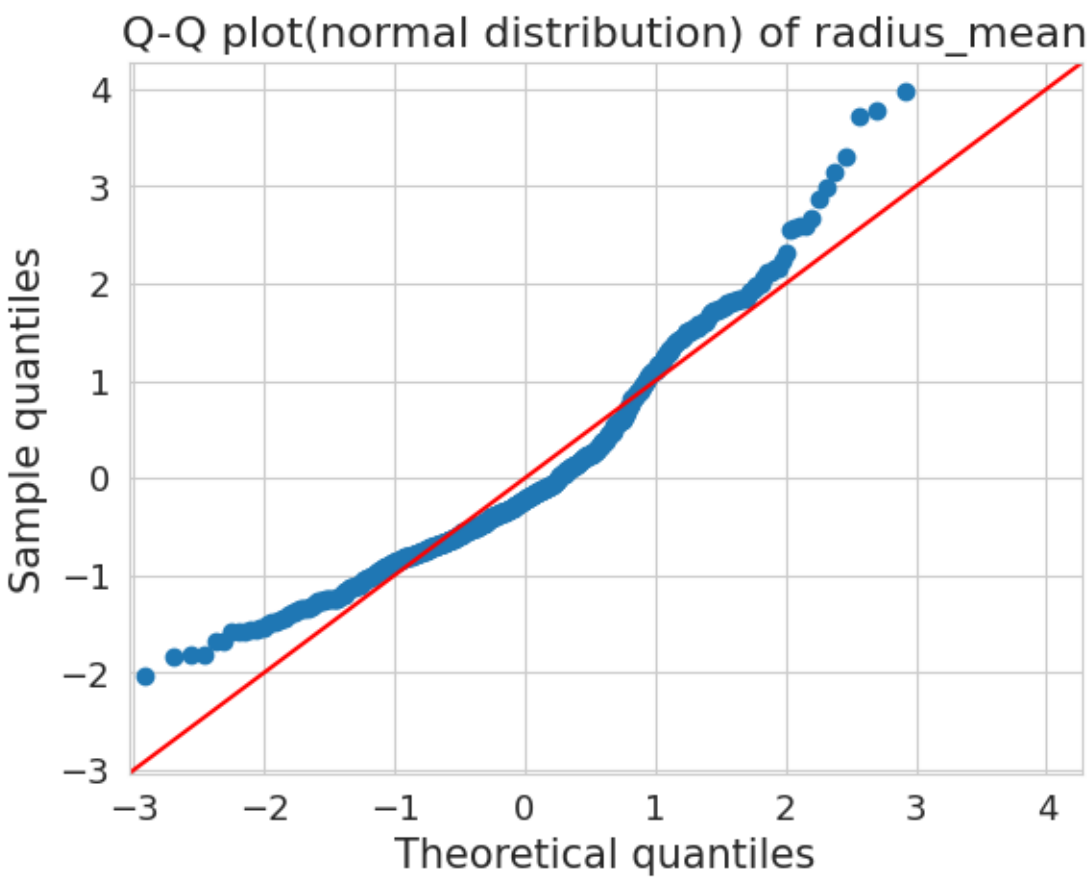
[225]: Text(0, 0.5, 'Frequency')



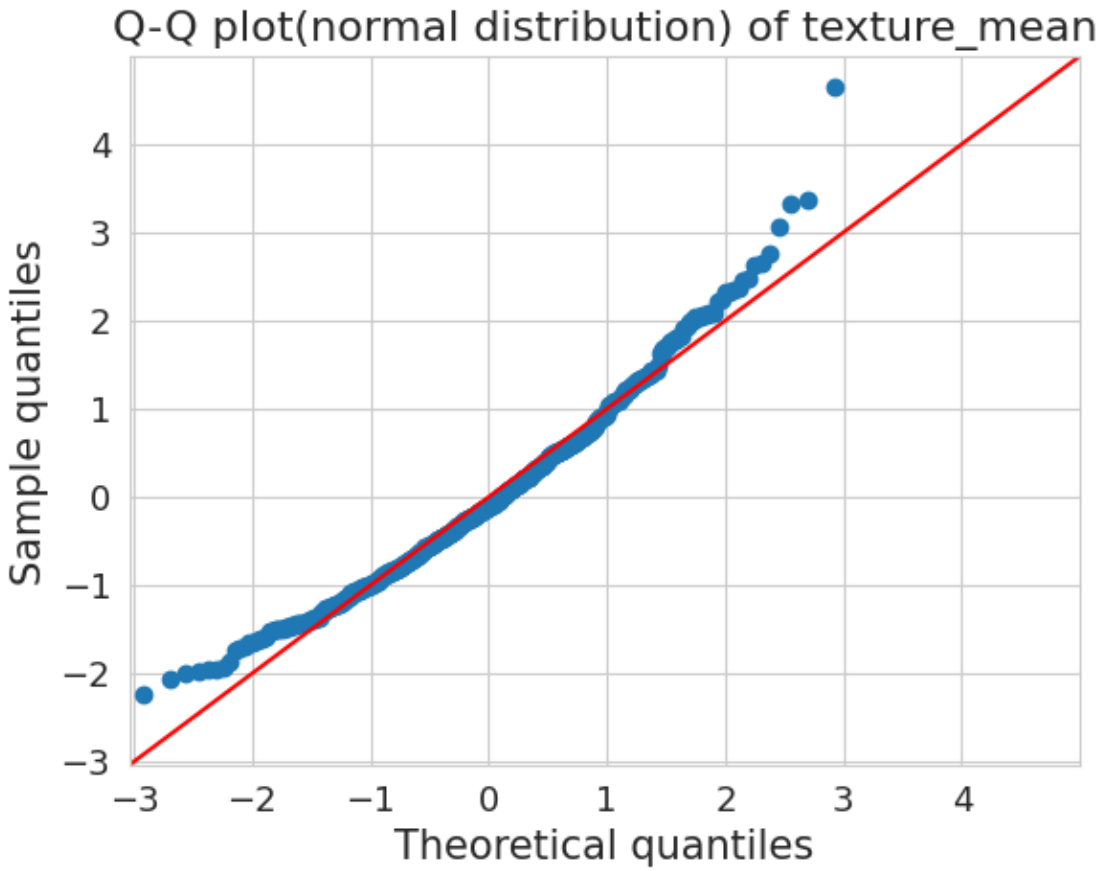
The bar graph shows a greater presence of "B" label in the dependent feature.

[226]:
```
#qqplot of independent features and various distributions
# Normal distribution
from statsmodels.graphics.gofplots import qqplot

for c in df.columns:
            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()
```

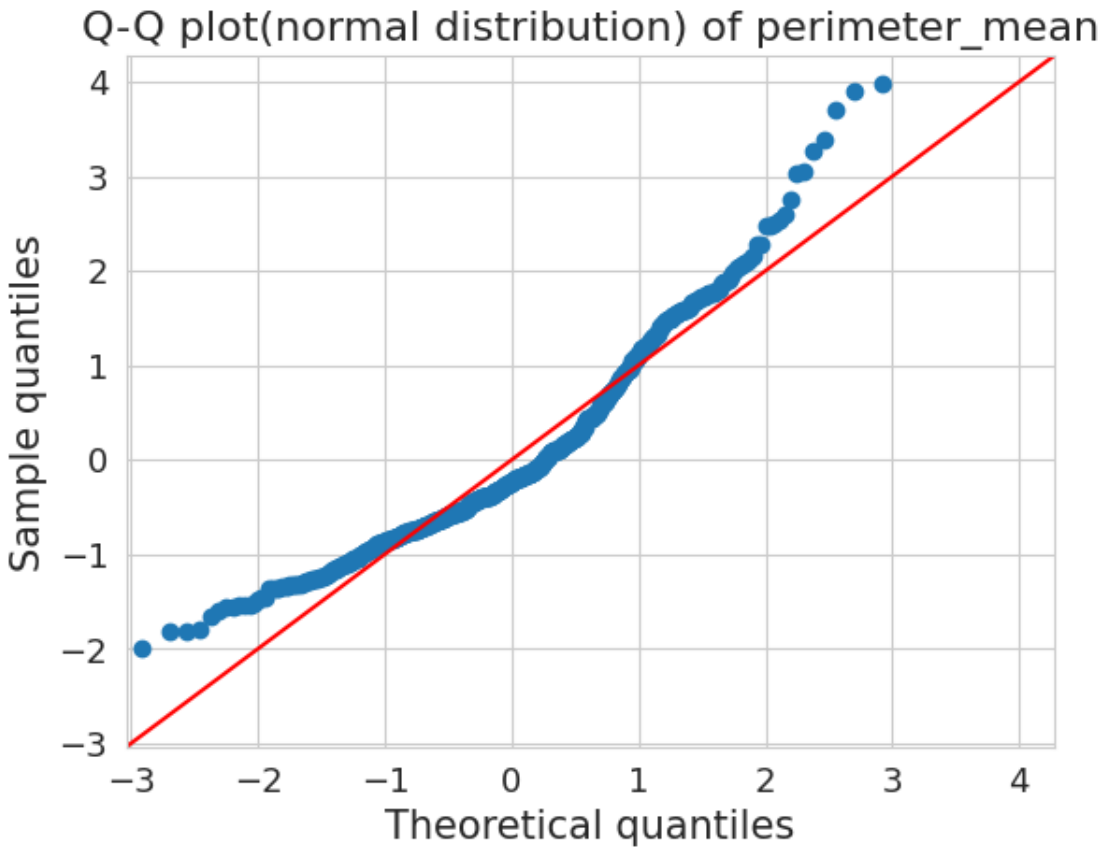
```
<Figure size 1000x400 with 0 Axes>
```



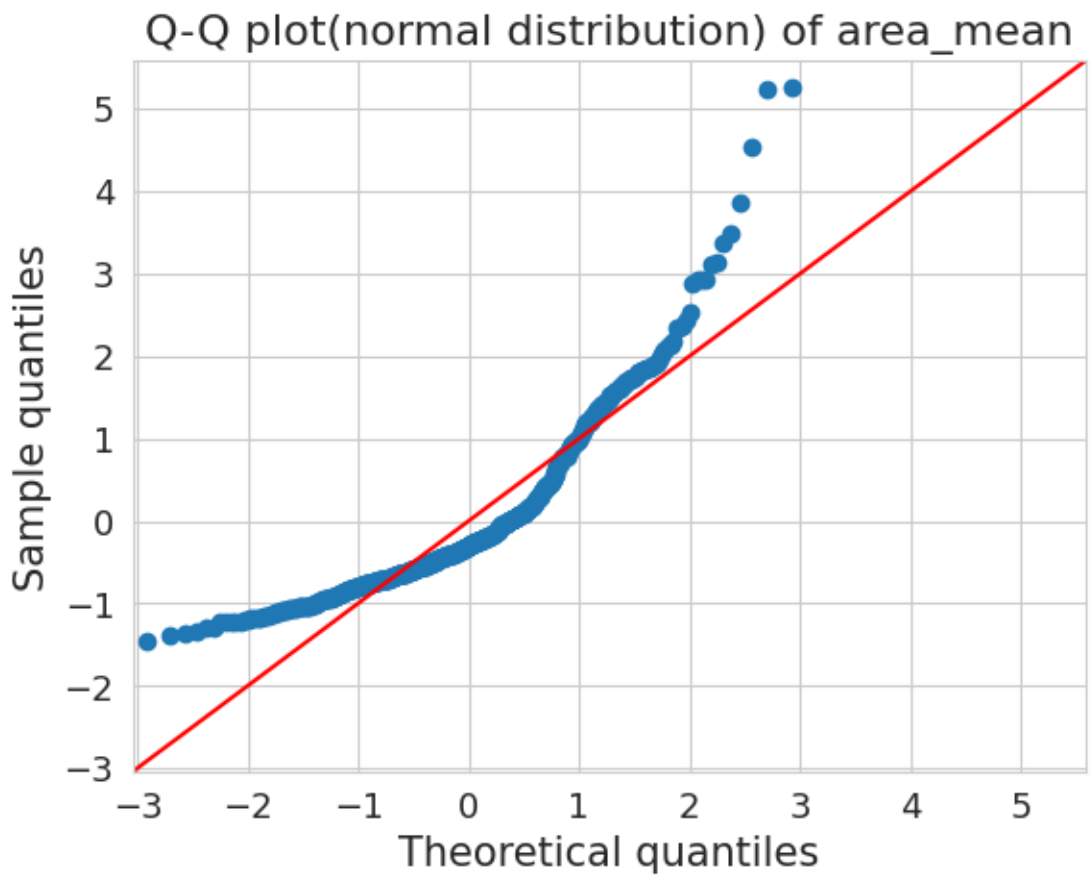

```
<Figure size 1000x400 with 0 Axes>
```



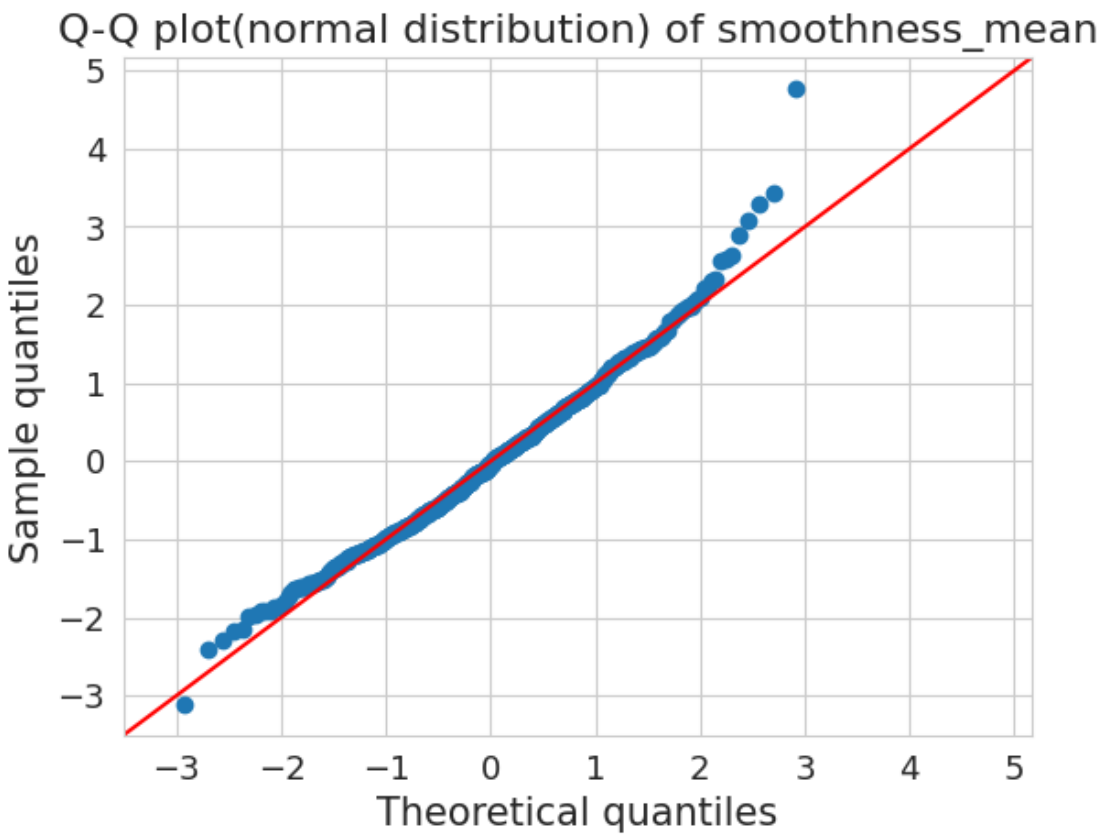

```
<Figure size 1000x400 with 0 Axes>
```



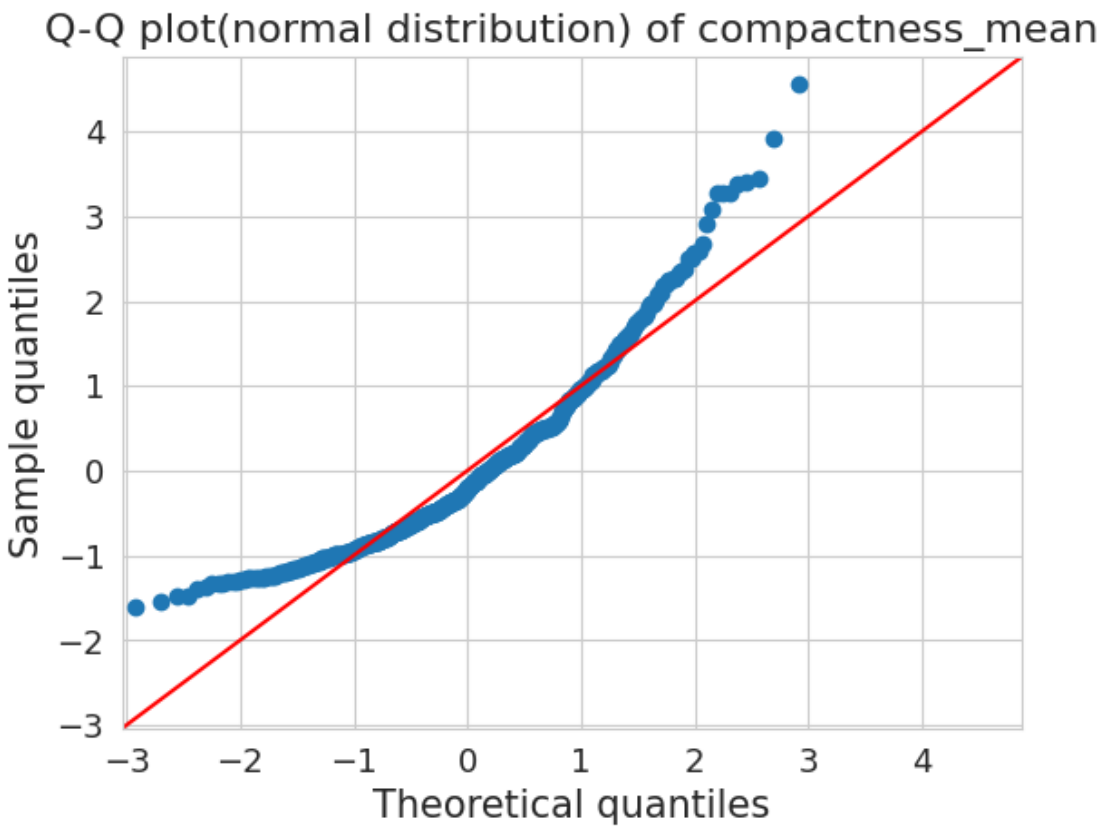

```
<Figure size 1000x400 with 0 Axes>
```



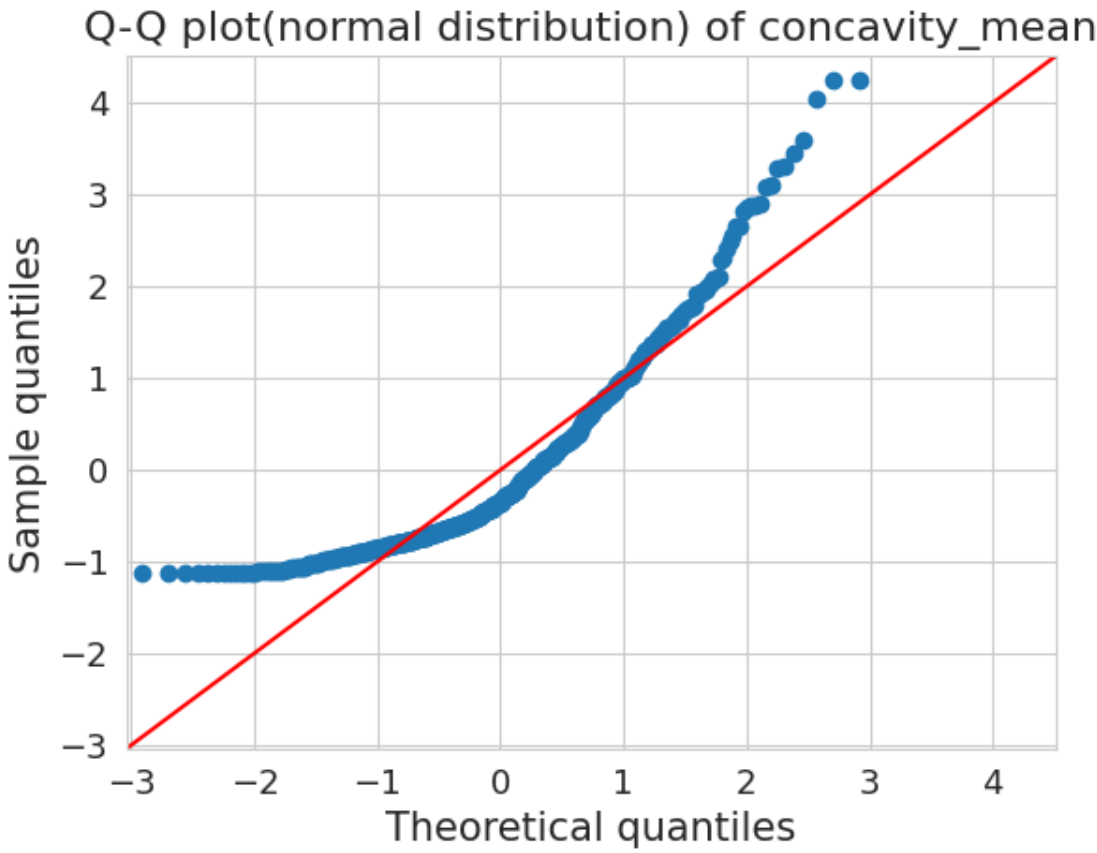

```
<Figure size 1000x400 with 0 Axes>
```



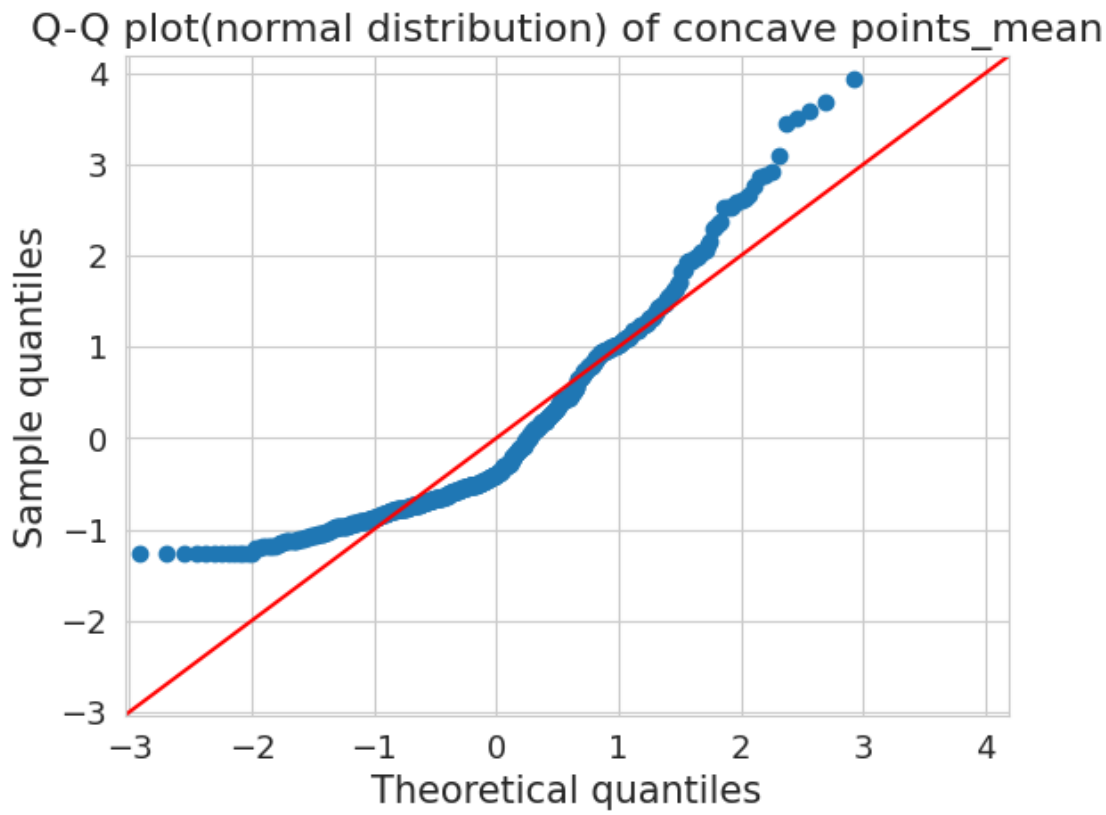

```
<Figure size 1000x400 with 0 Axes>
```



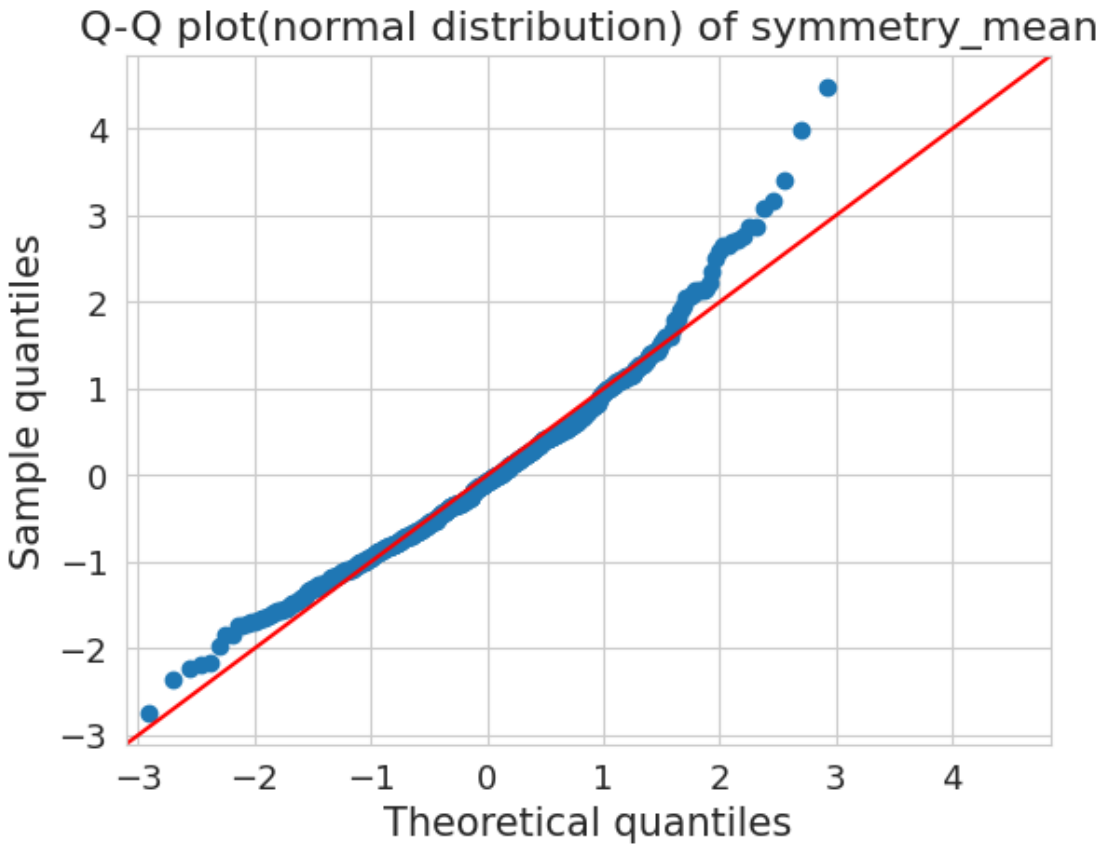

```
<Figure size 1000x400 with 0 Axes>
```



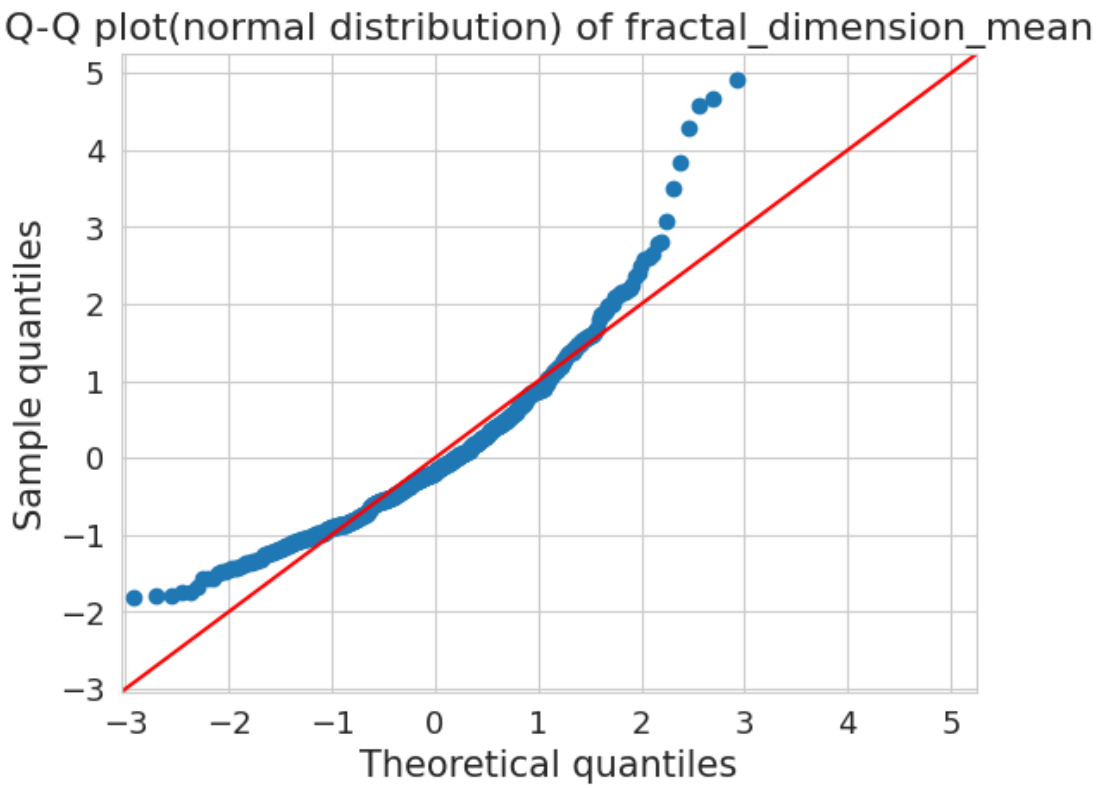

```
<Figure size 1000x400 with 0 Axes>
```



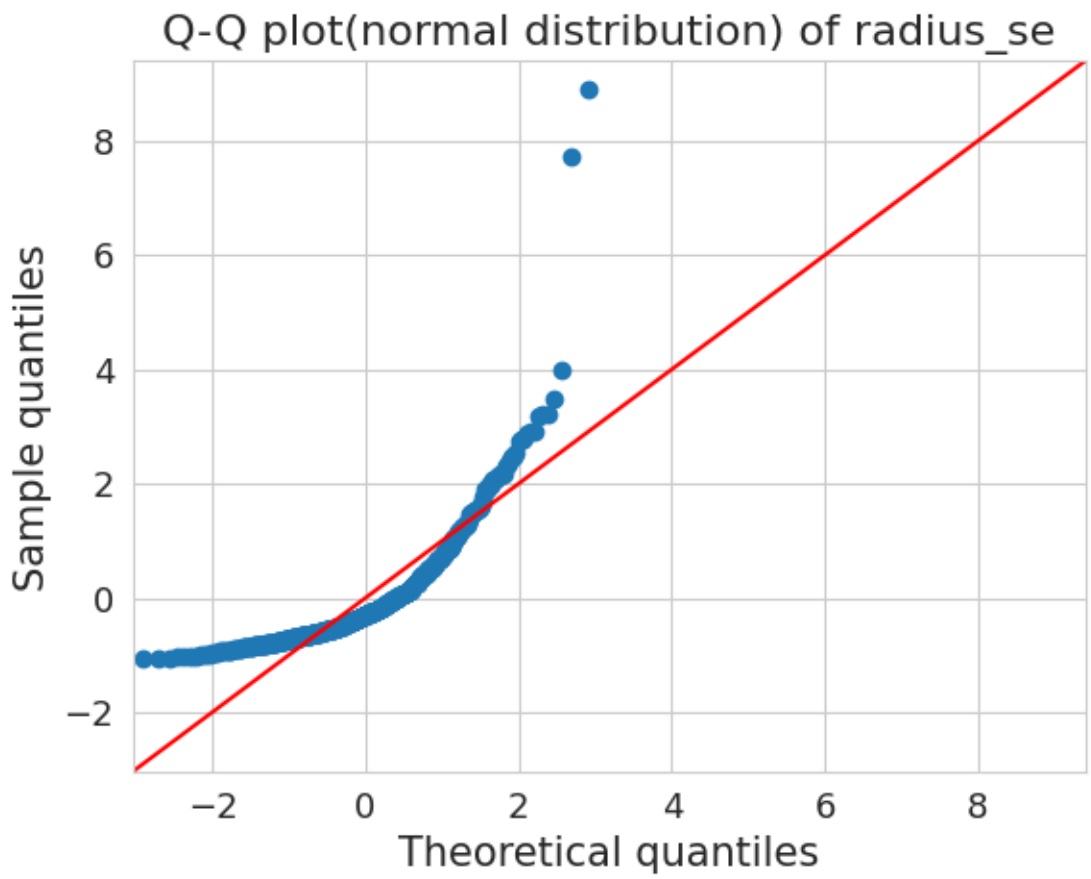

```
<Figure size 1000x400 with 0 Axes>
```



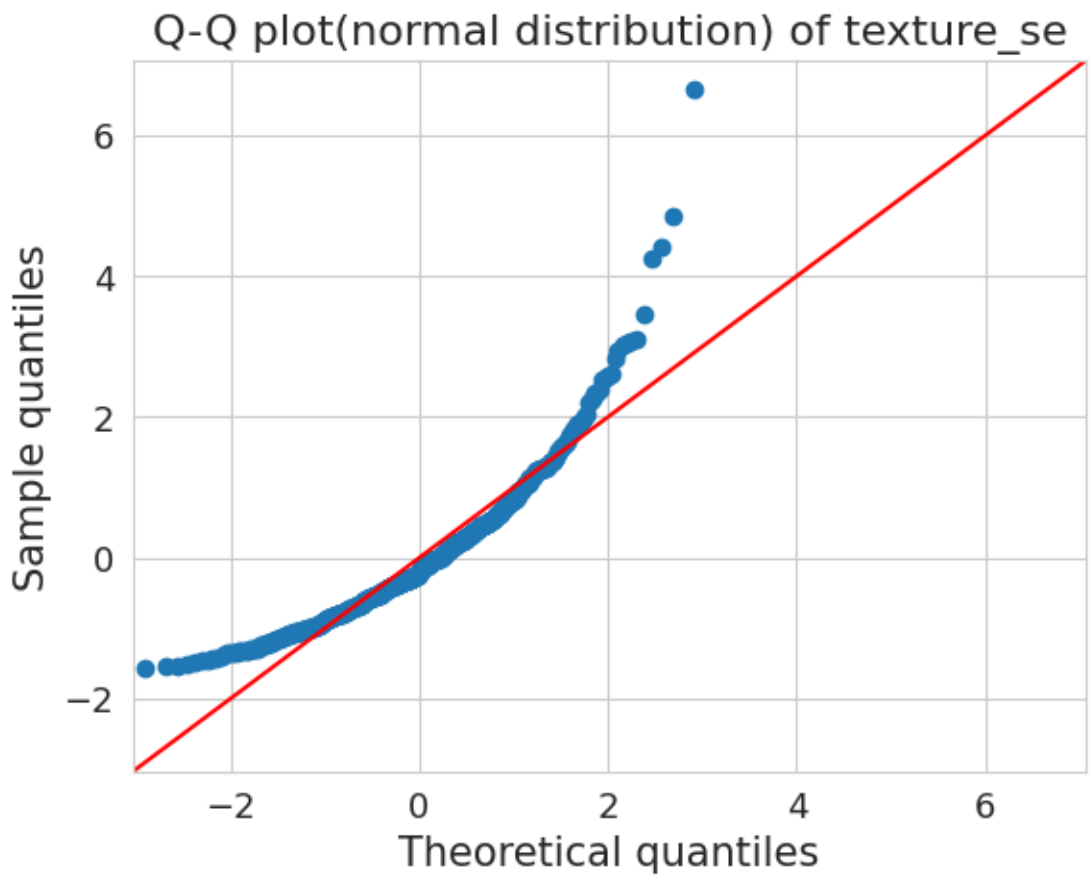

```
<Figure size 1000x400 with 0 Axes>
```



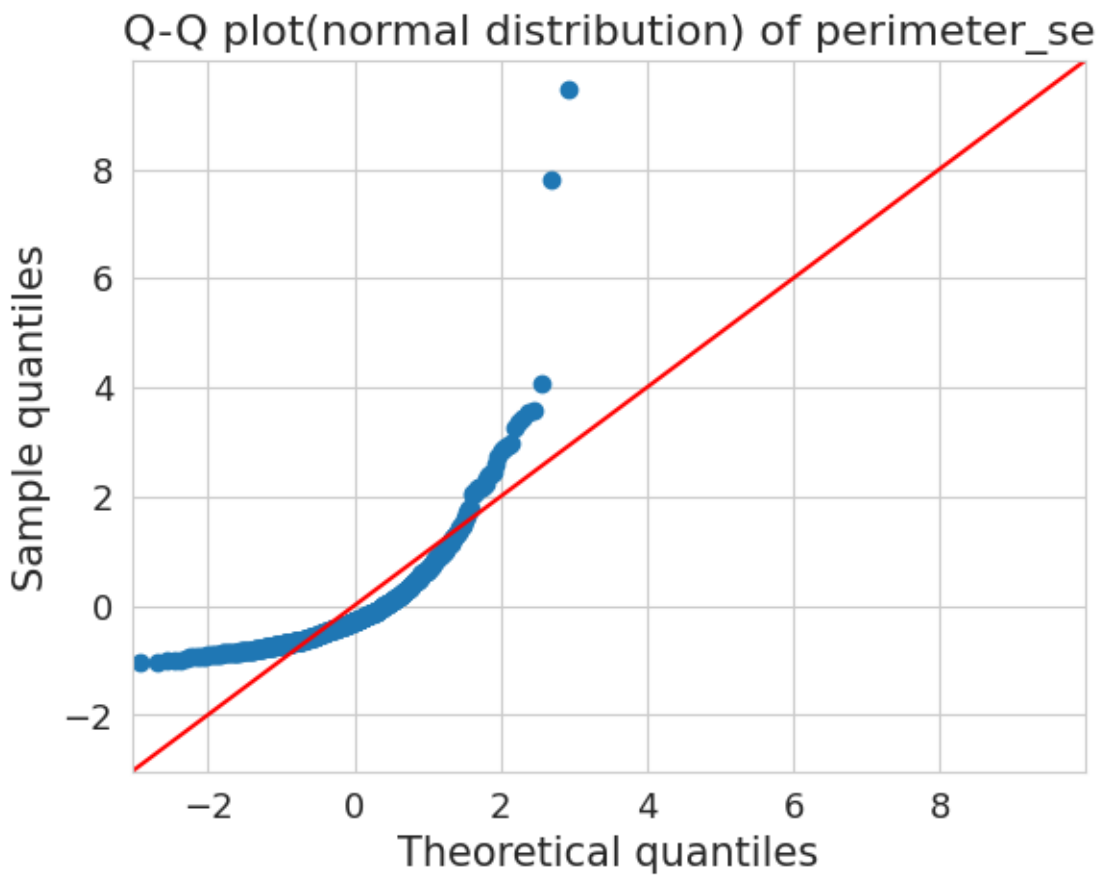

```
<Figure size 1000x400 with 0 Axes>
```



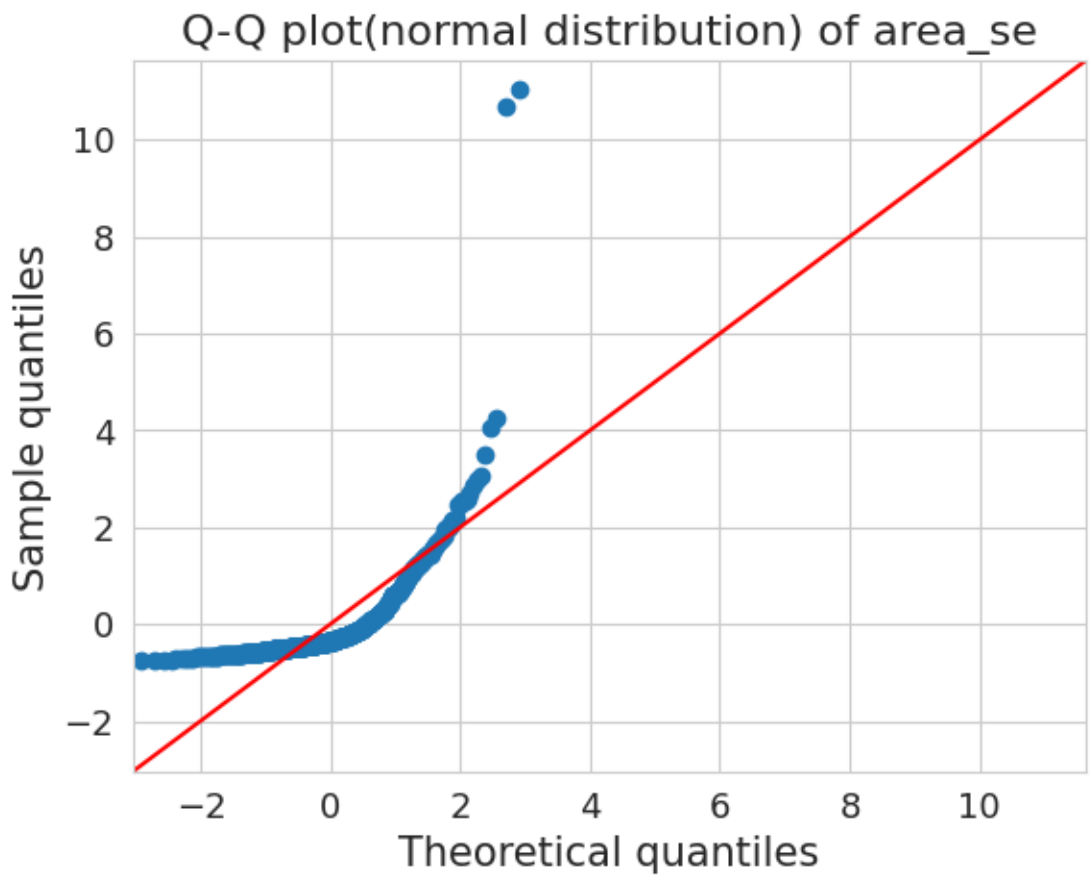

```
<Figure size 1000x400 with 0 Axes>
```



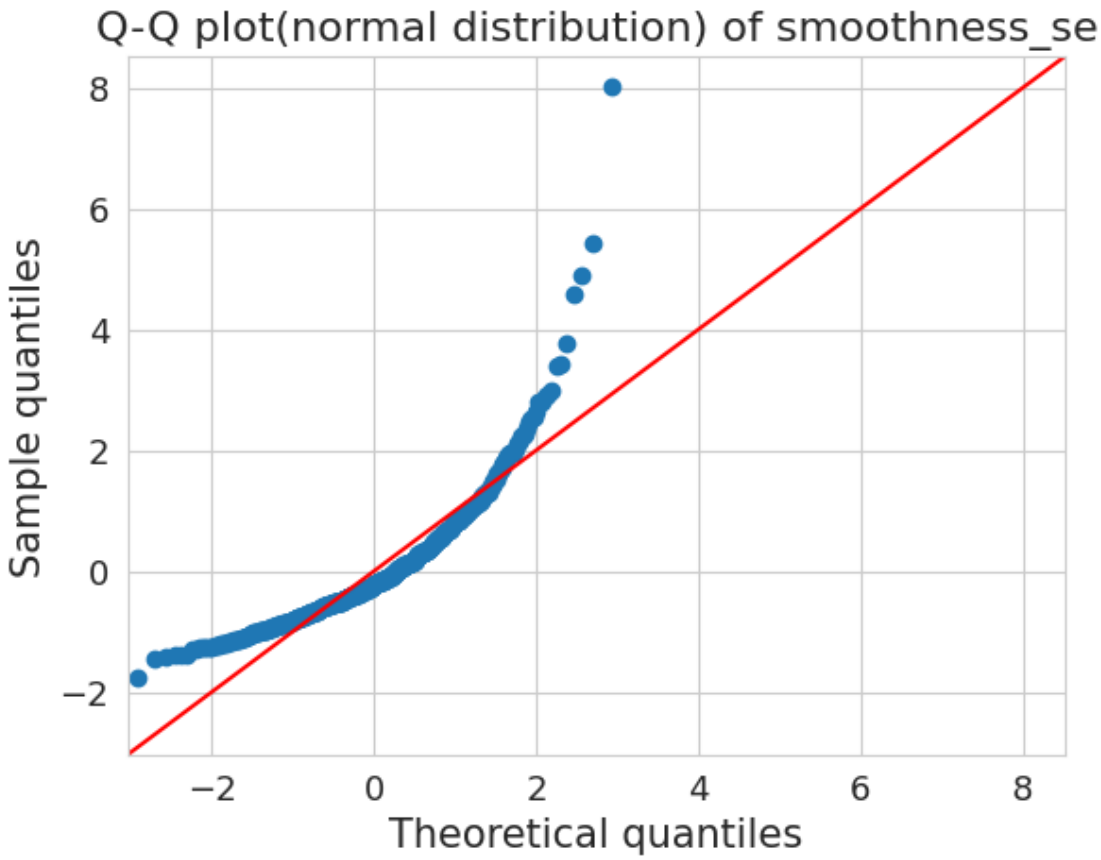

```
<Figure size 1000x400 with 0 Axes>
```



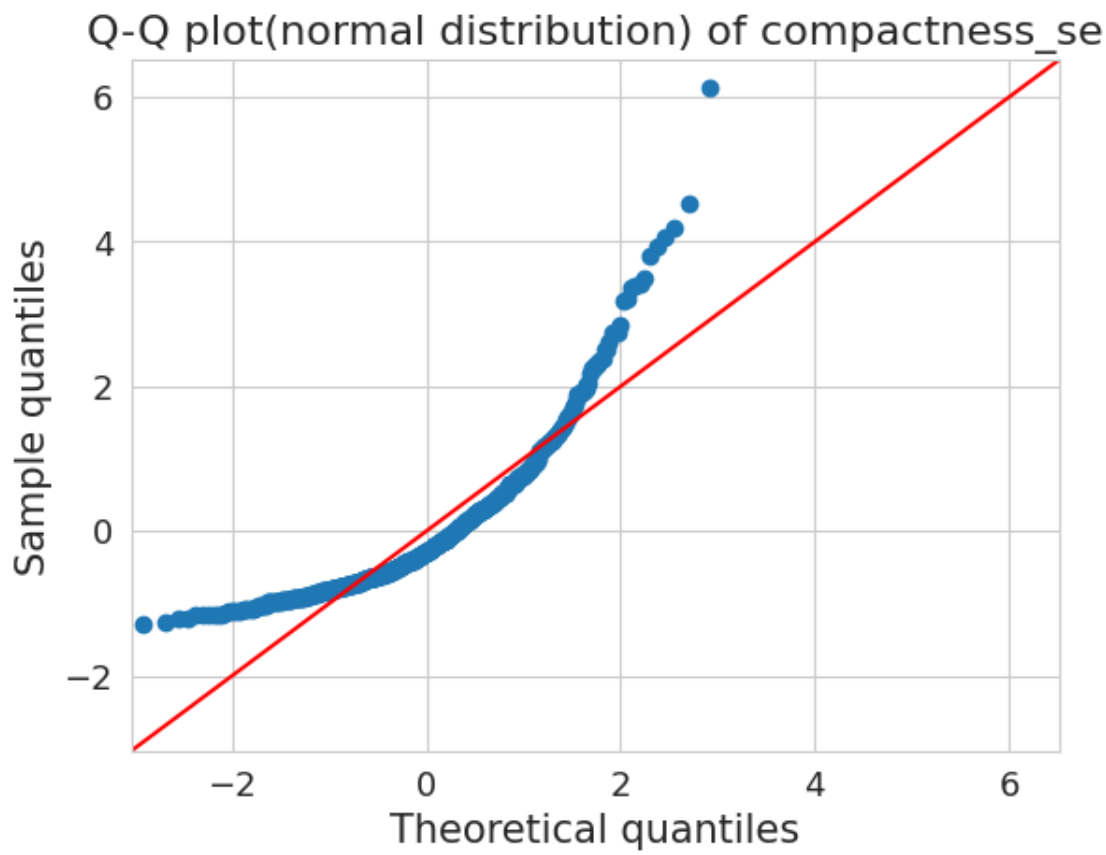

```
<Figure size 1000x400 with 0 Axes>
```



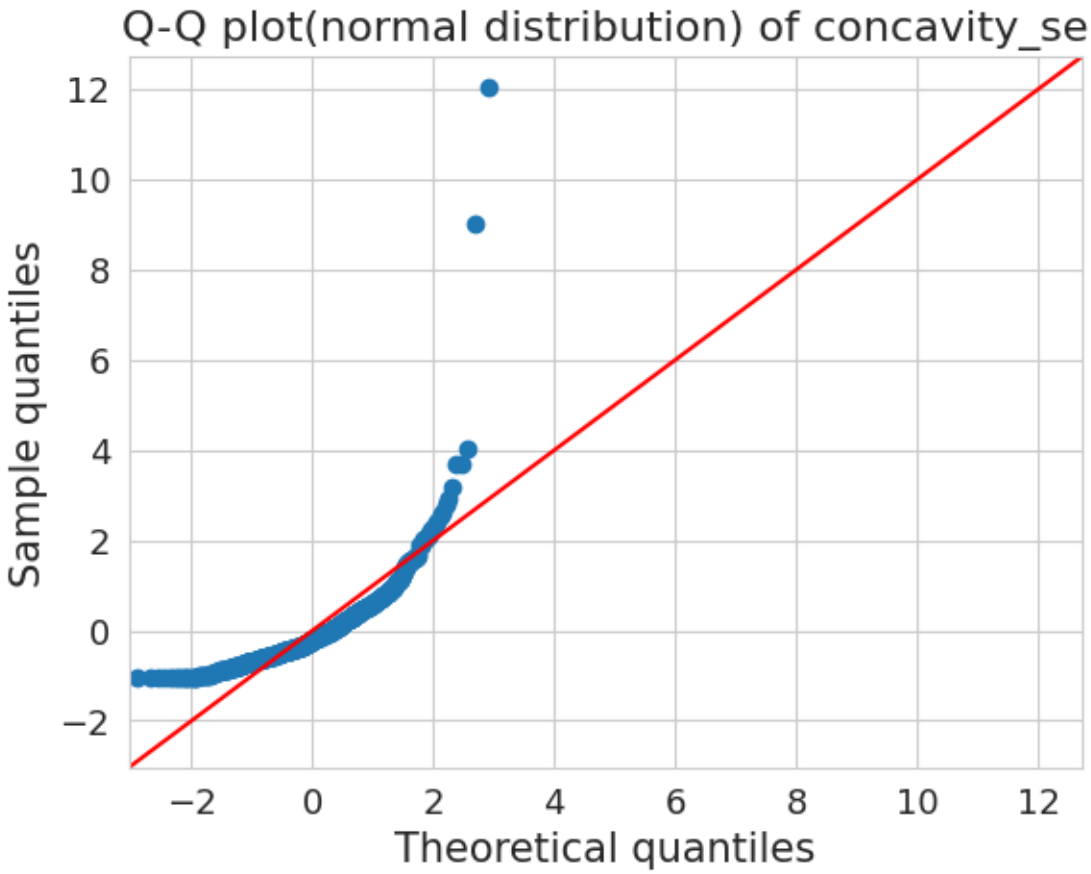

```
<Figure size 1000x400 with 0 Axes>
```



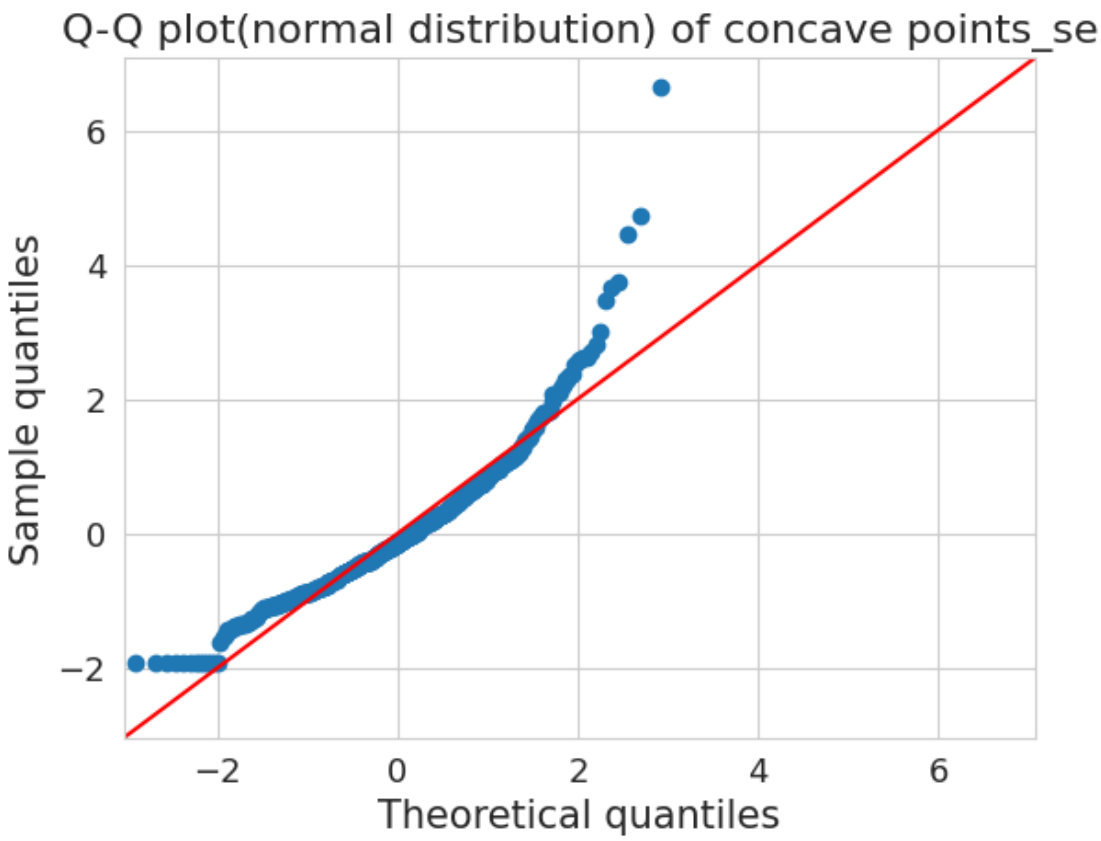

```
<Figure size 1000x400 with 0 Axes>
```



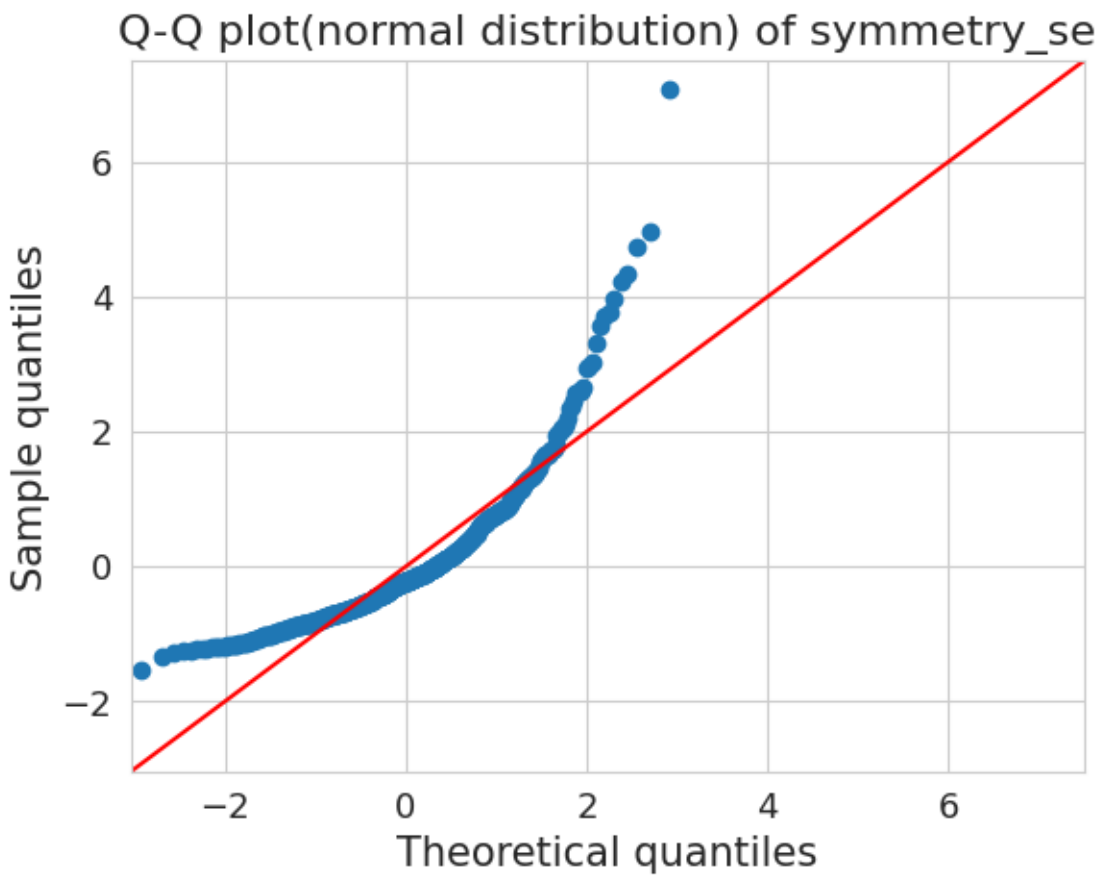

```
<Figure size 1000x400 with 0 Axes>
```



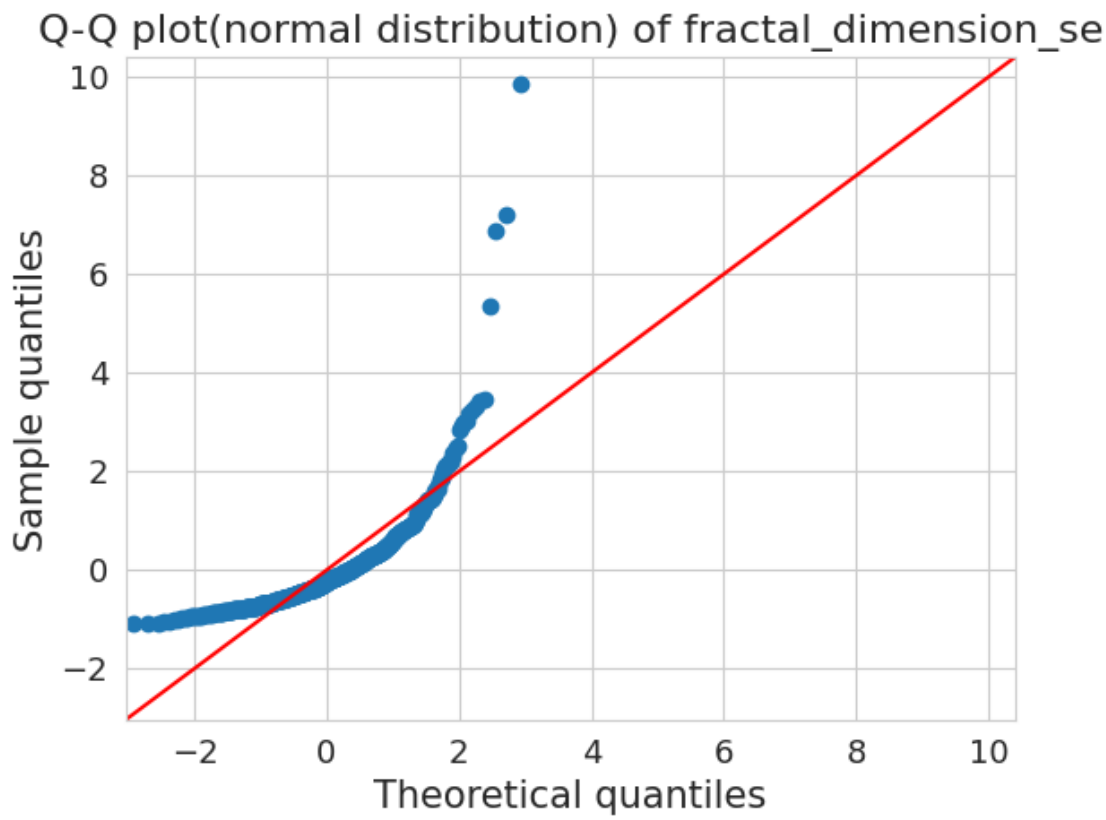

```
<Figure size 1000x400 with 0 Axes>
```



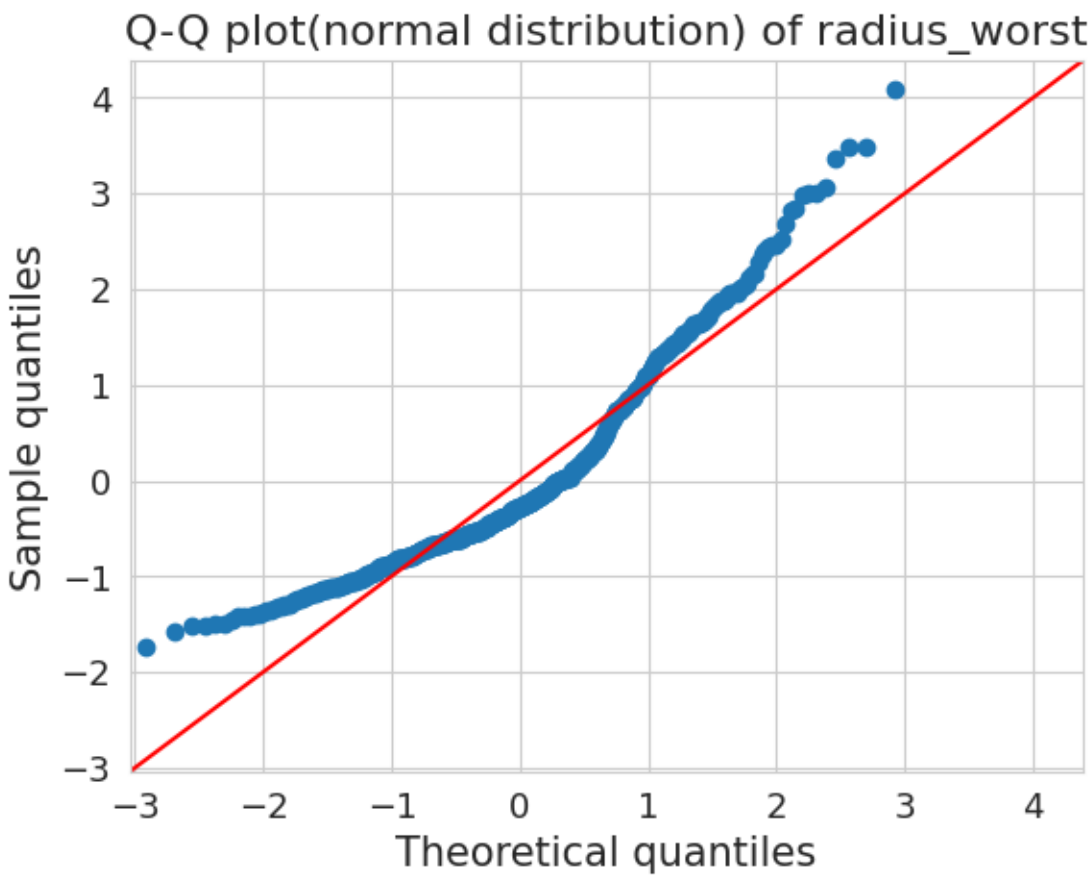

```
<Figure size 1000x400 with 0 Axes>
```



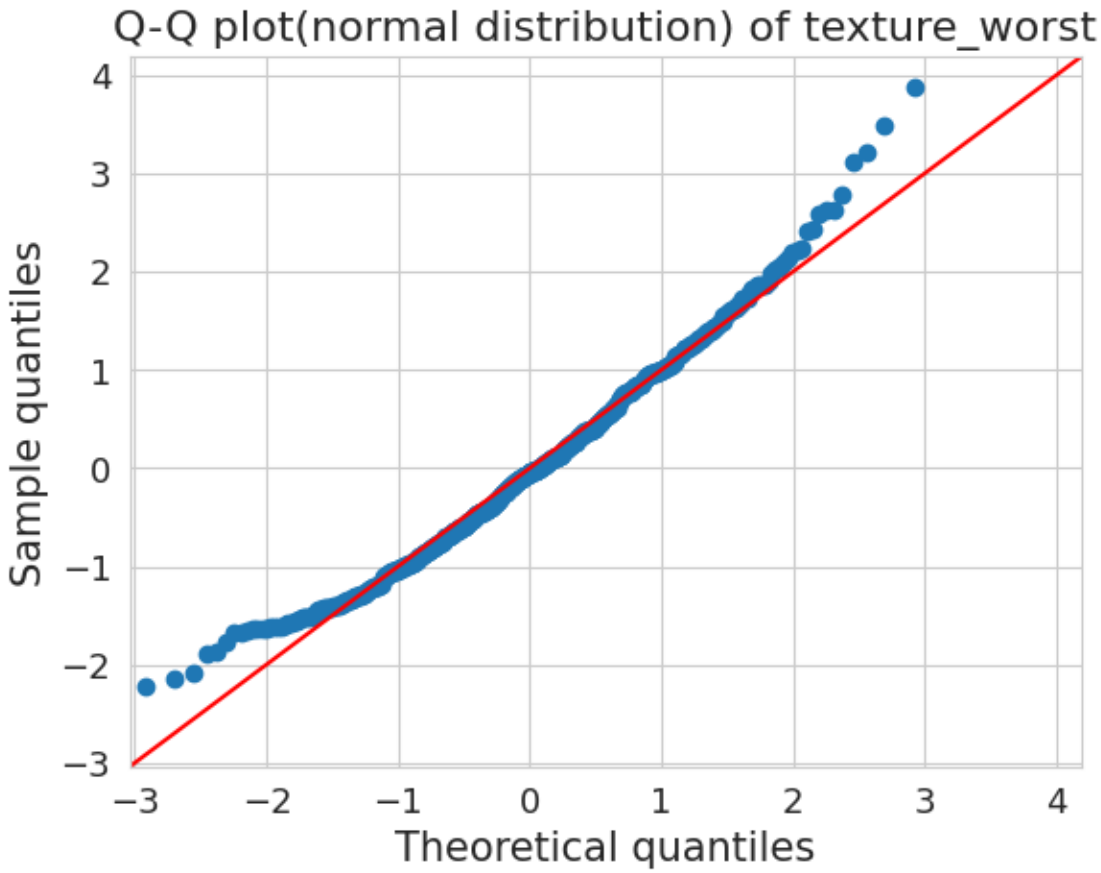

```
<Figure size 1000x400 with 0 Axes>
```



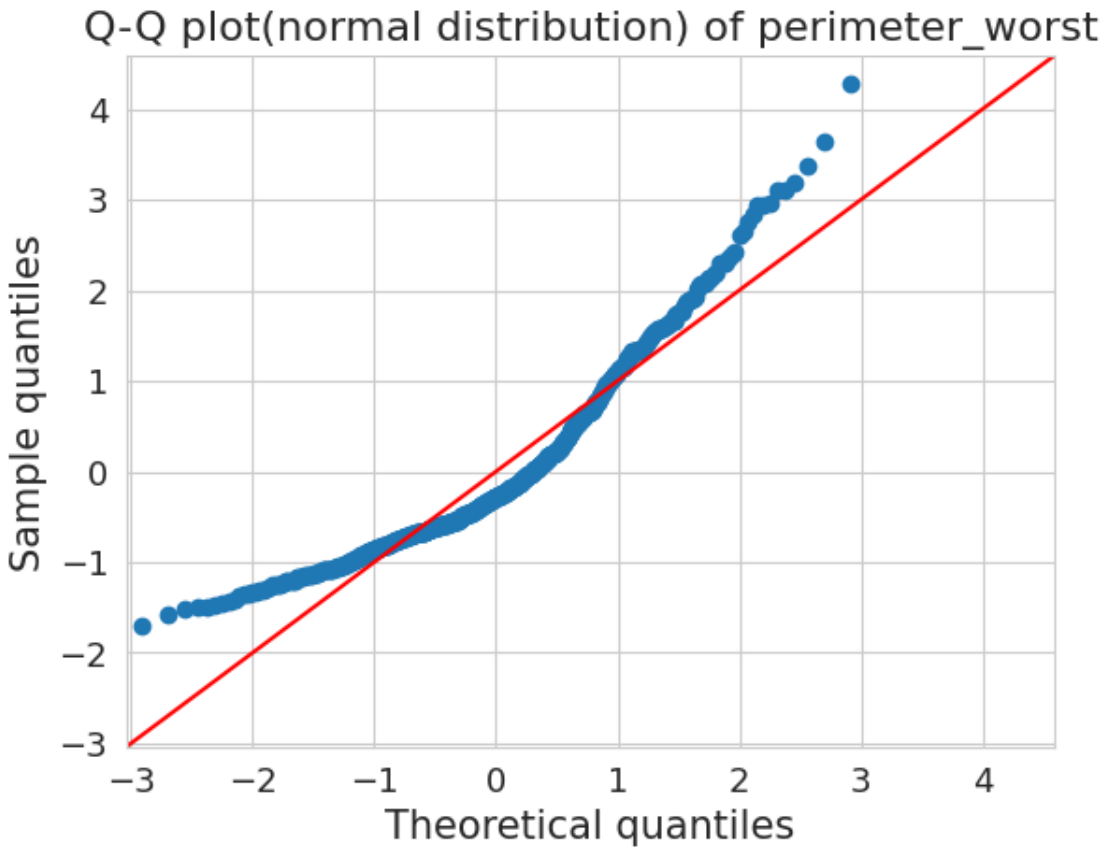

```
<Figure size 1000x400 with 0 Axes>
```



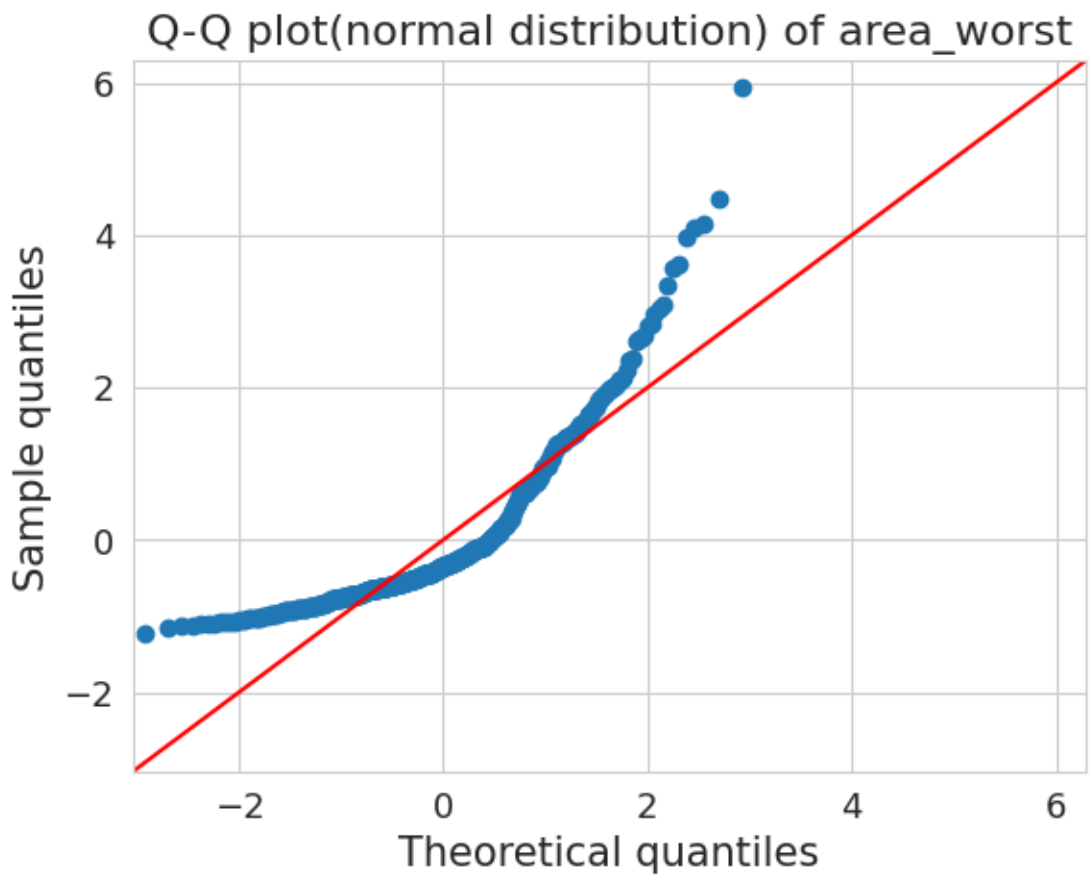

```
<Figure size 1000x400 with 0 Axes>
```



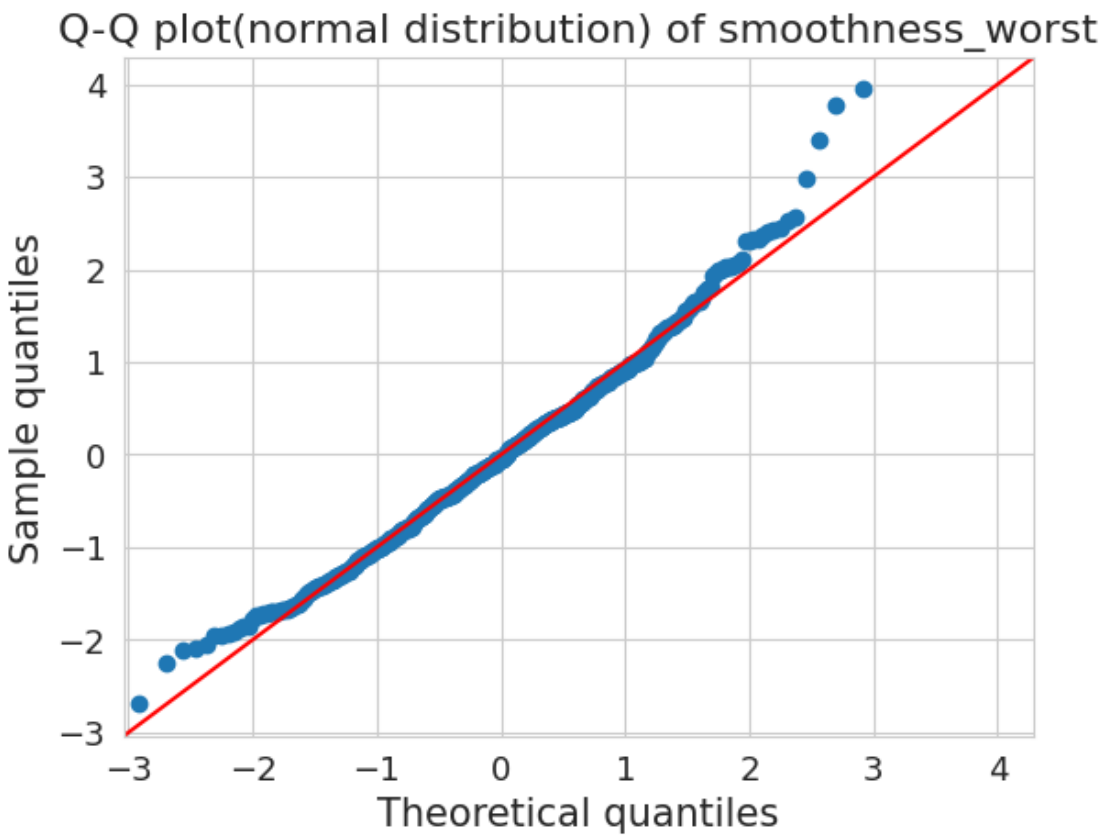

```
<Figure size 1000x400 with 0 Axes>
```



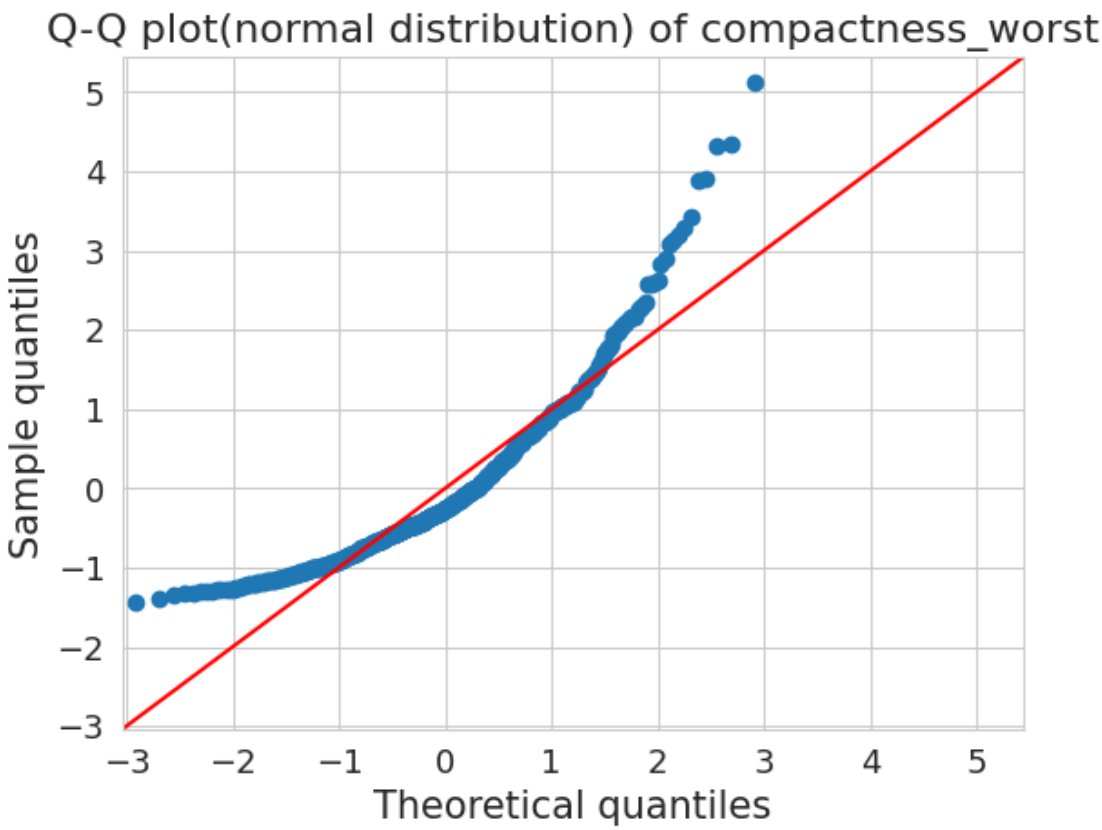

```
<Figure size 1000x400 with 0 Axes>
```



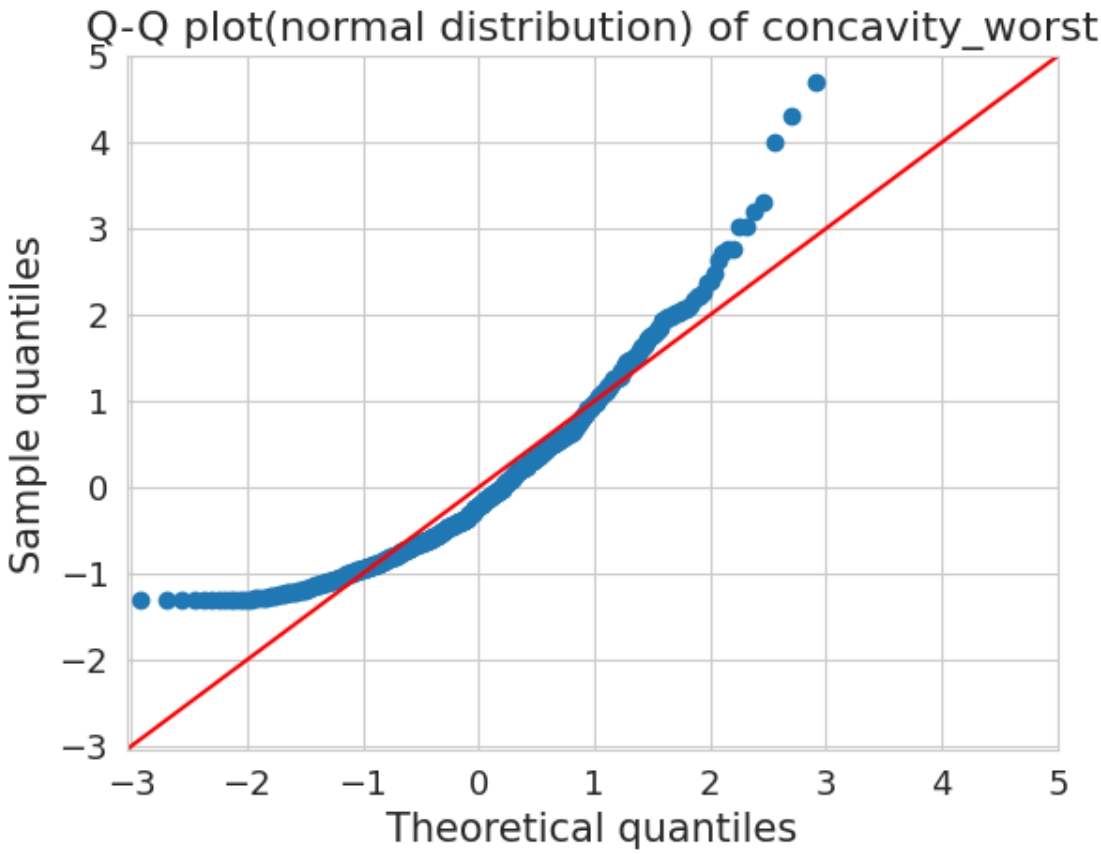

```
<Figure size 1000x400 with 0 Axes>
```



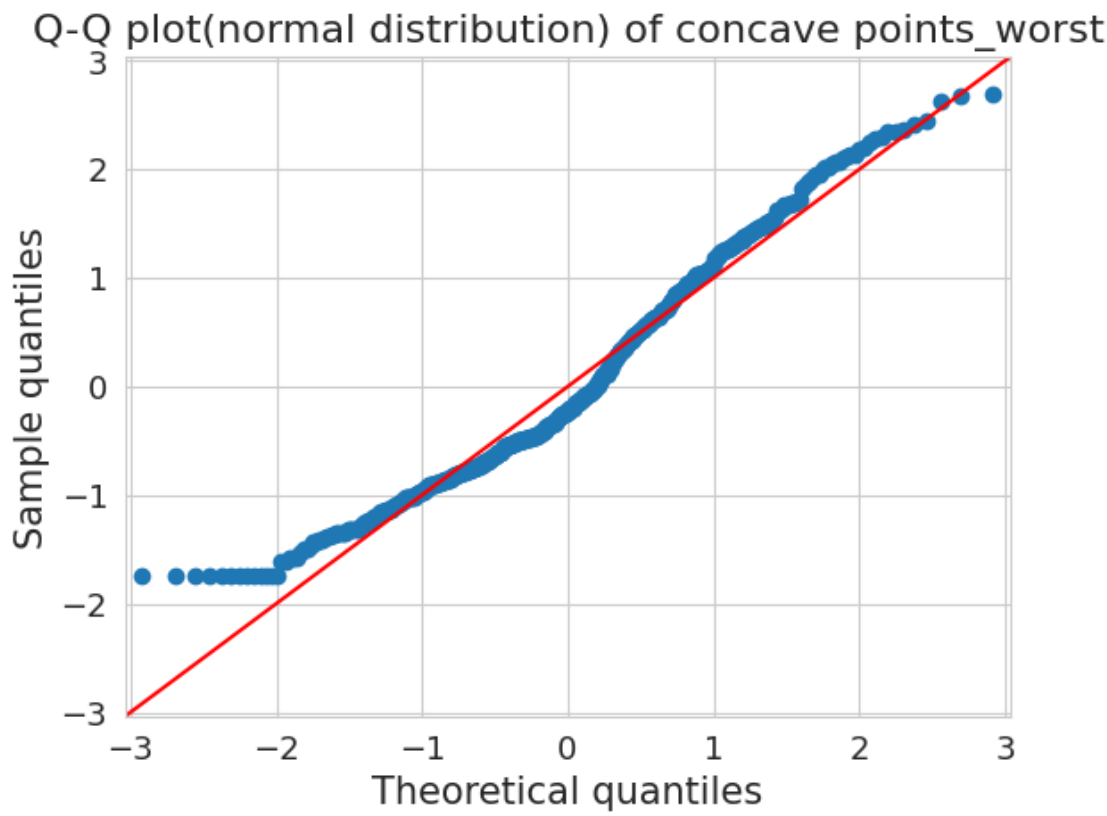

```
<Figure size 1000x400 with 0 Axes>
```



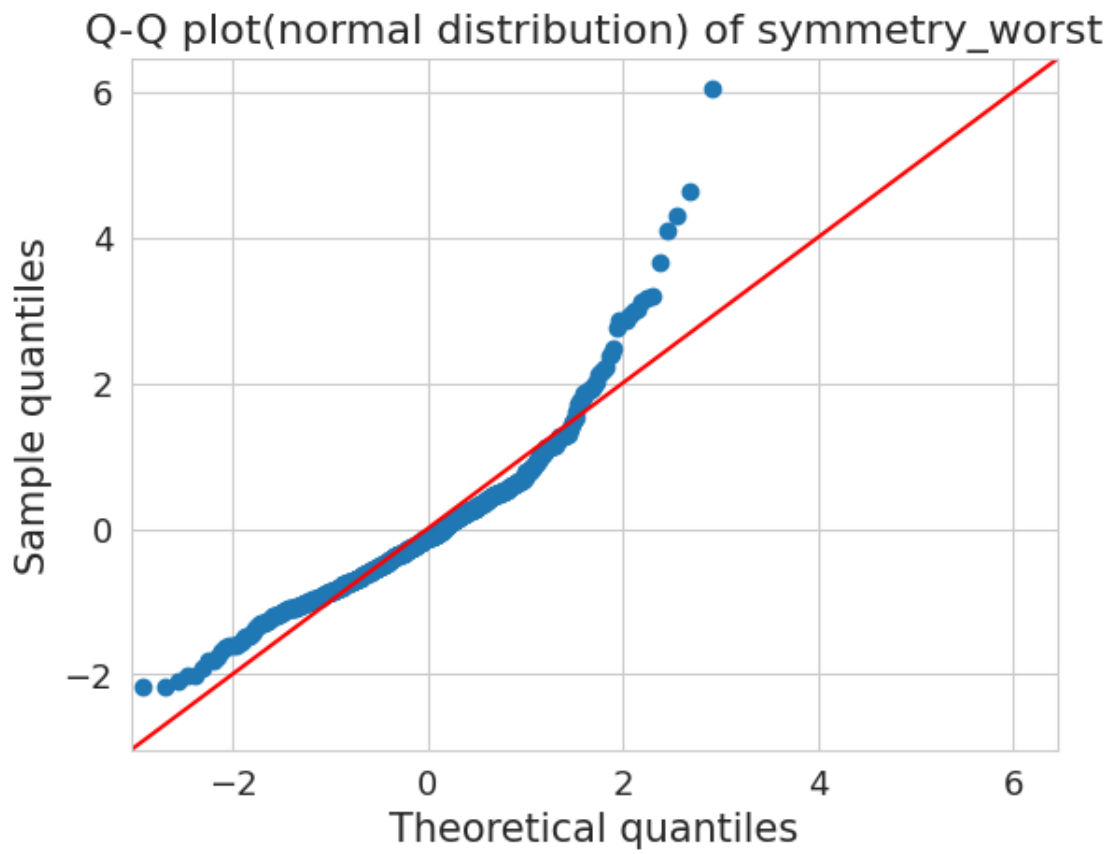

```
<Figure size 1000x400 with 0 Axes>
```



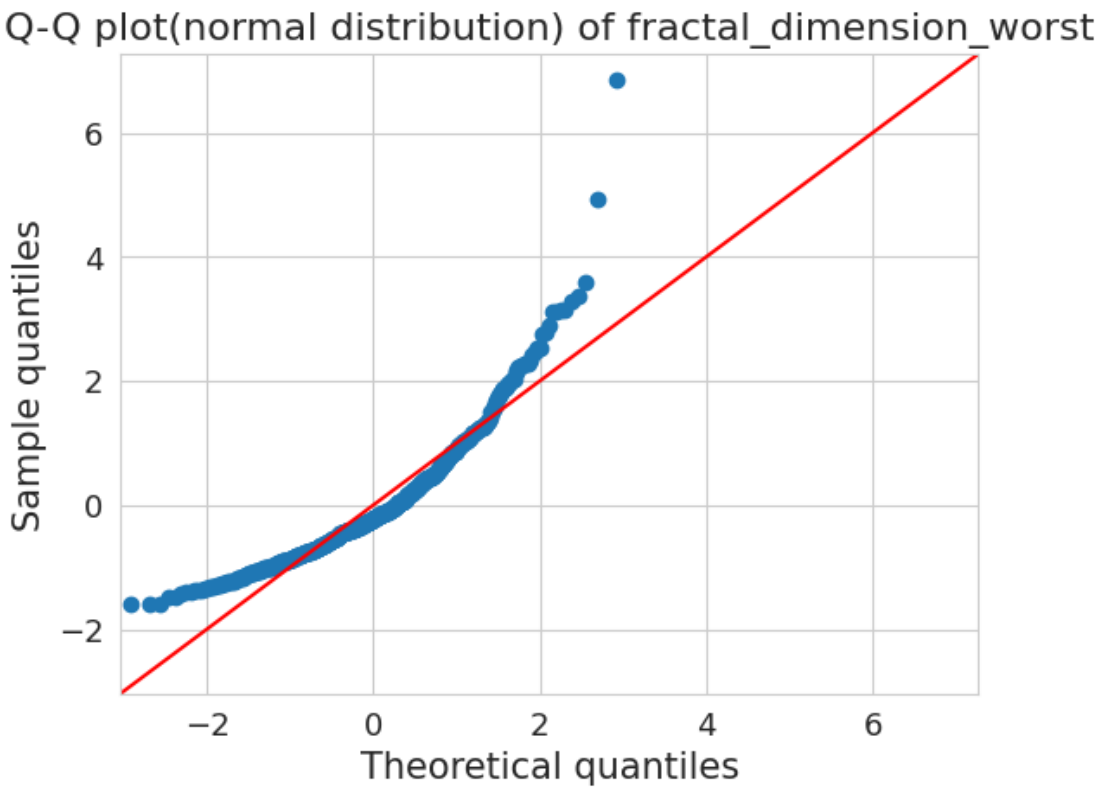

```
<Figure size 1000x400 with 0 Axes>
```




```
<Figure size 1000x400 with 0 Axes>
```

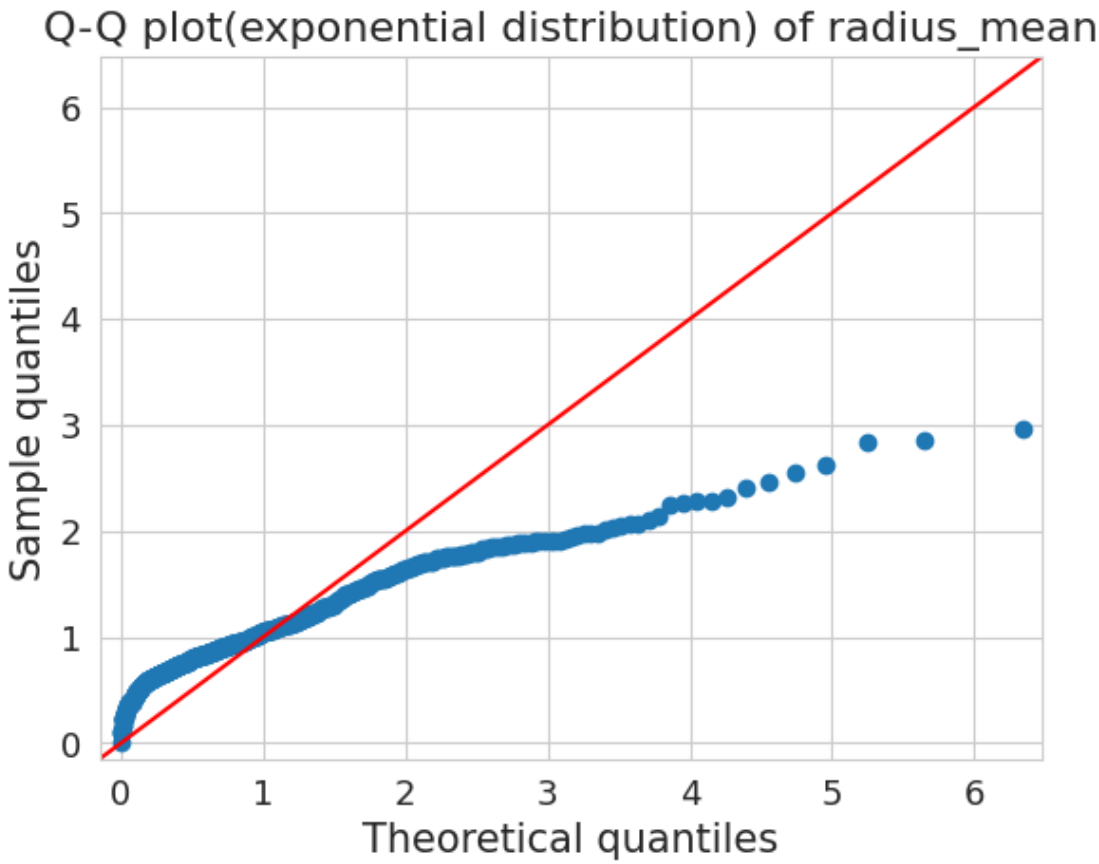



```
<Figure size 1000x400 with 0 Axes>
```

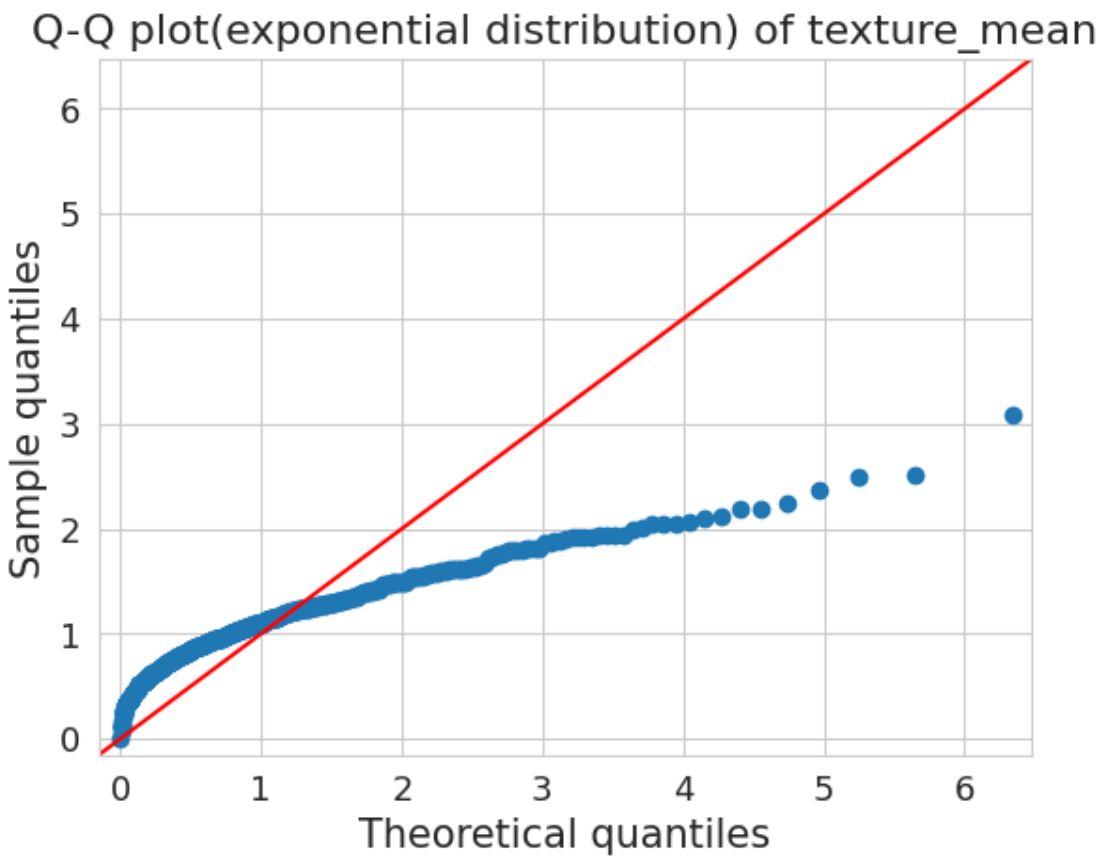


[227]:
```
#Exponential distribution
for c in df.columns:
        plt.figure(figsize=(8,5))
        fig=qqplot(df[c],dist=sc.stats.expon,line="45",fit='True')
        plt.xticks(fontsize=13)
        plt.yticks(fontsize=13)
        plt.xlabel("Theoretical quantiles",fontsize=15)
        plt.ylabel("Sample quantiles",fontsize=15)
        plt.title("Q-Q plot(exponential distribution) of {}".
 ↪format(c),fontsize=16)
        plt.grid(True)
        plt.show()
```

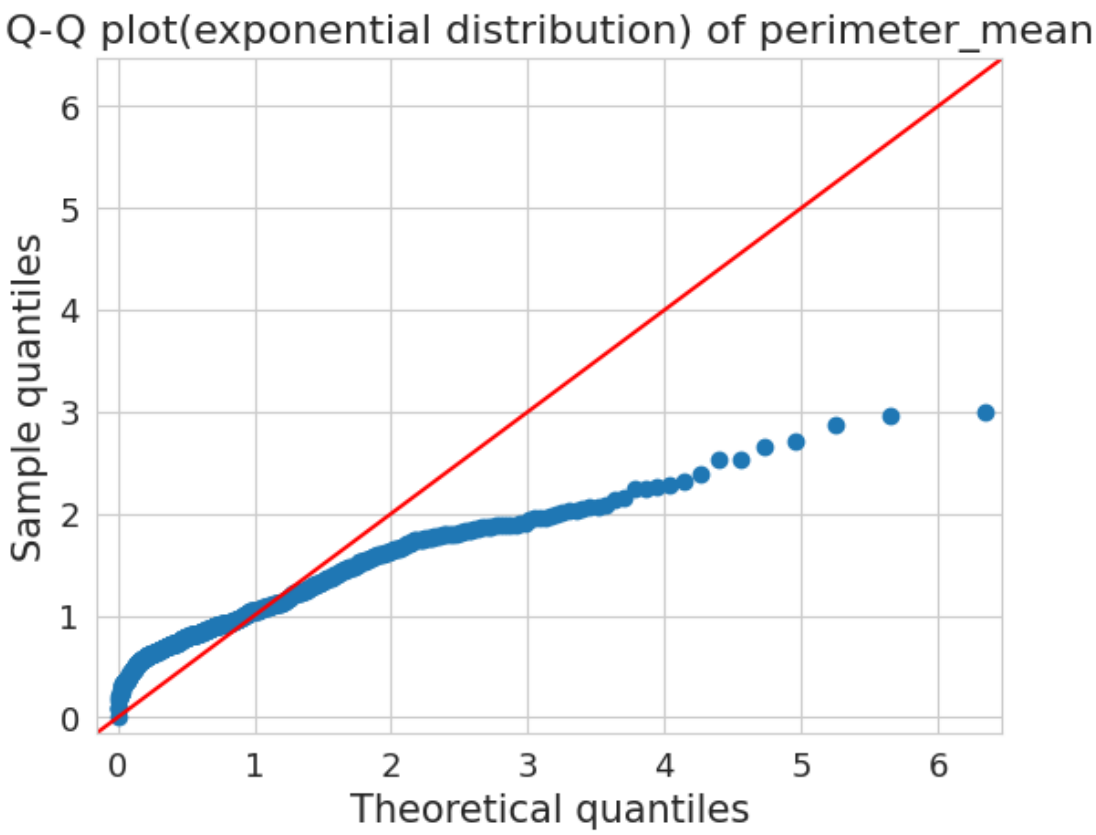
```
<Figure size 800x500 with 0 Axes>
```



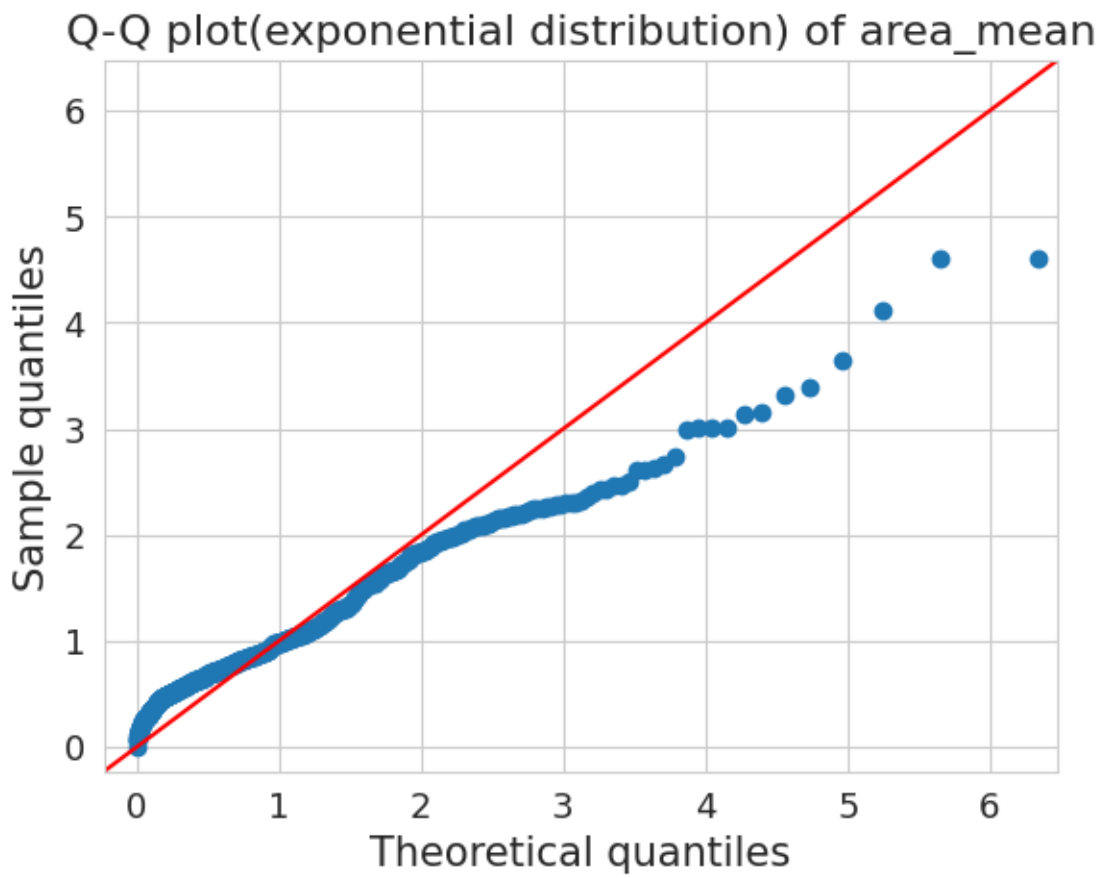

```
<Figure size 800x500 with 0 Axes>
```



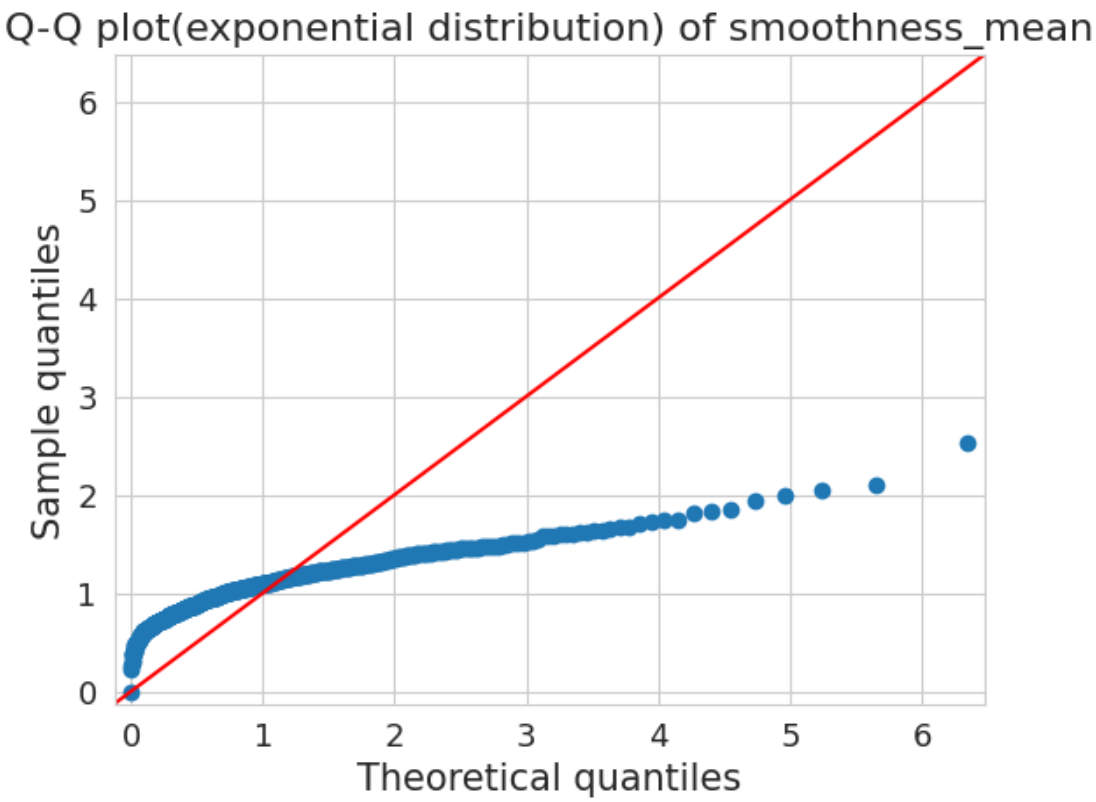

```
<Figure size 800x500 with 0 Axes>
```



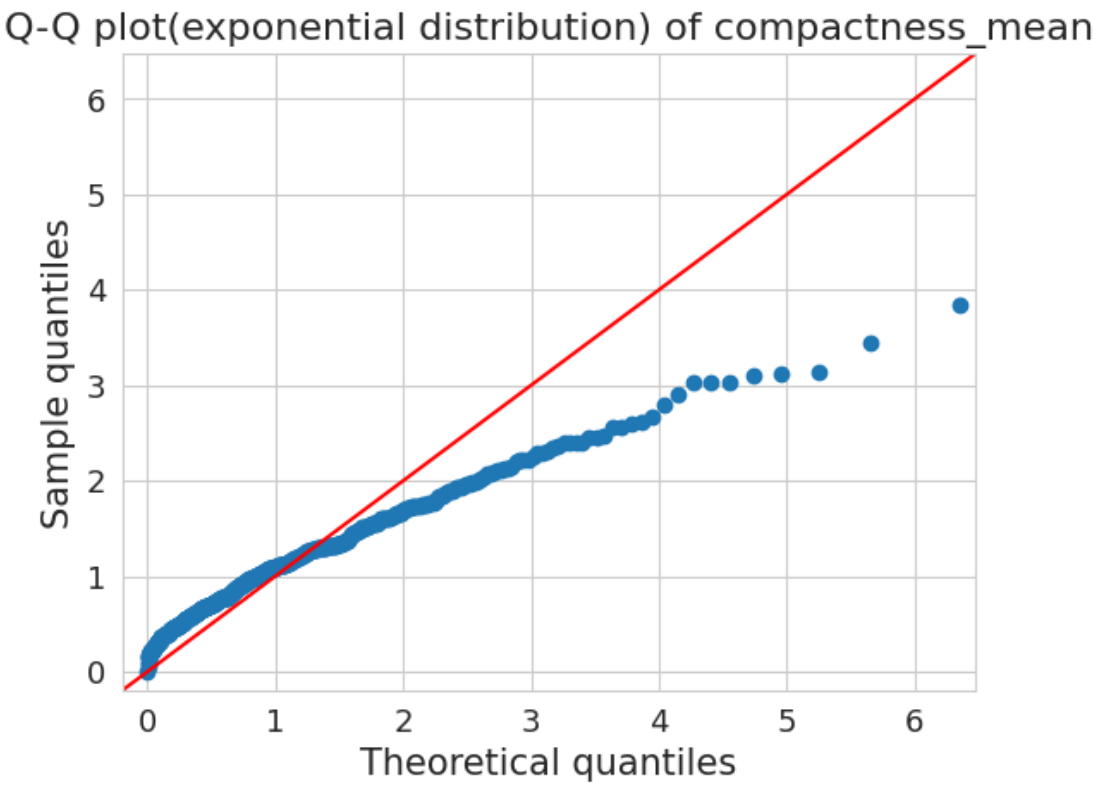

```
<Figure size 800x500 with 0 Axes>
```



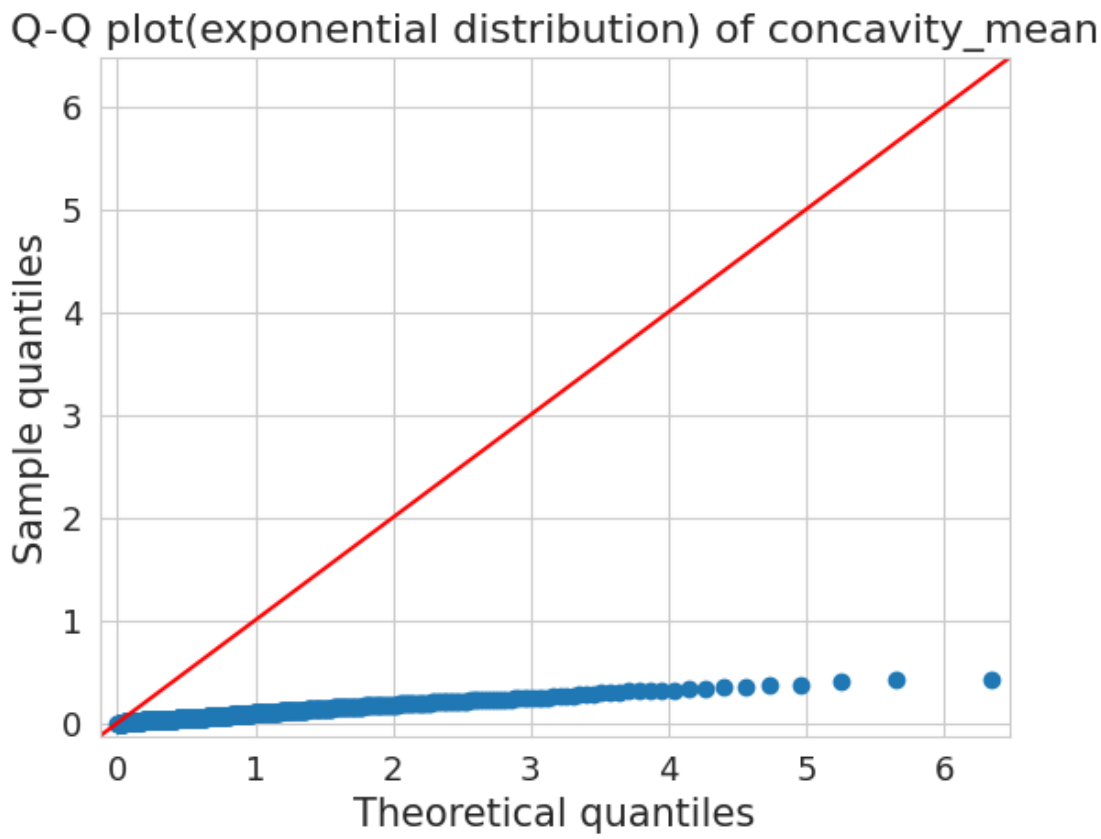

```
<Figure size 800x500 with 0 Axes>
```



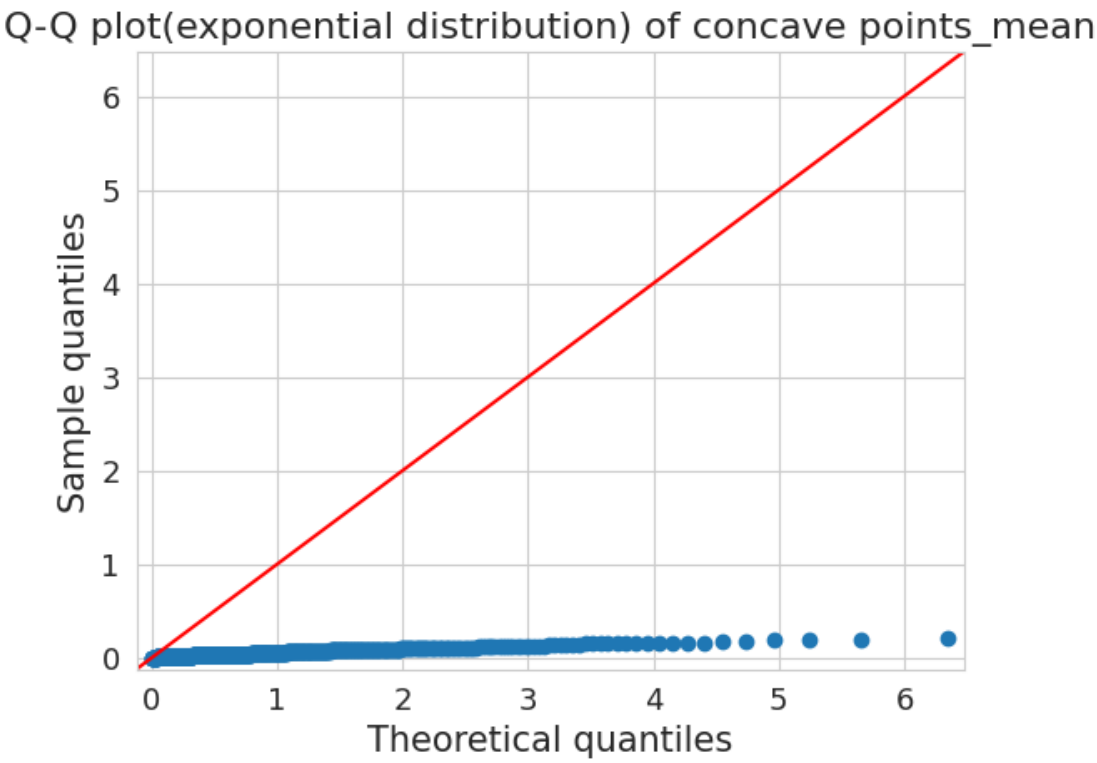

```
<Figure size 800x500 with 0 Axes>
```



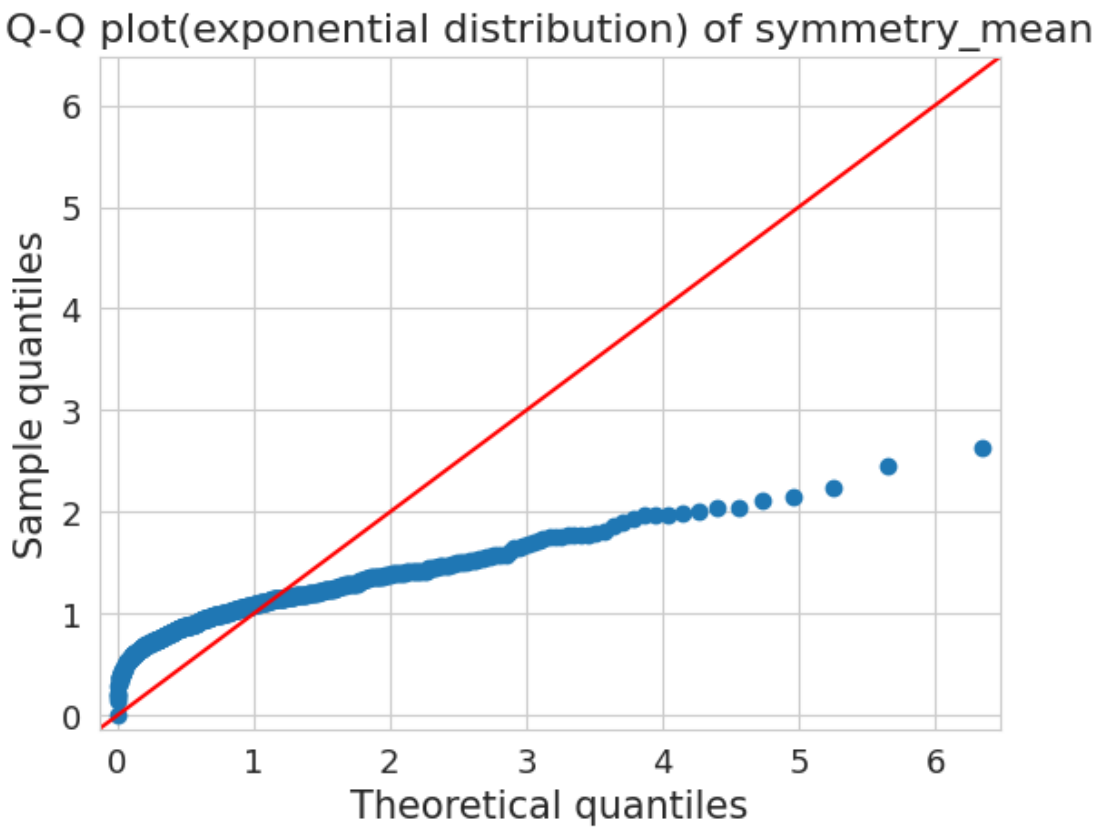

```
<Figure size 800x500 with 0 Axes>
```



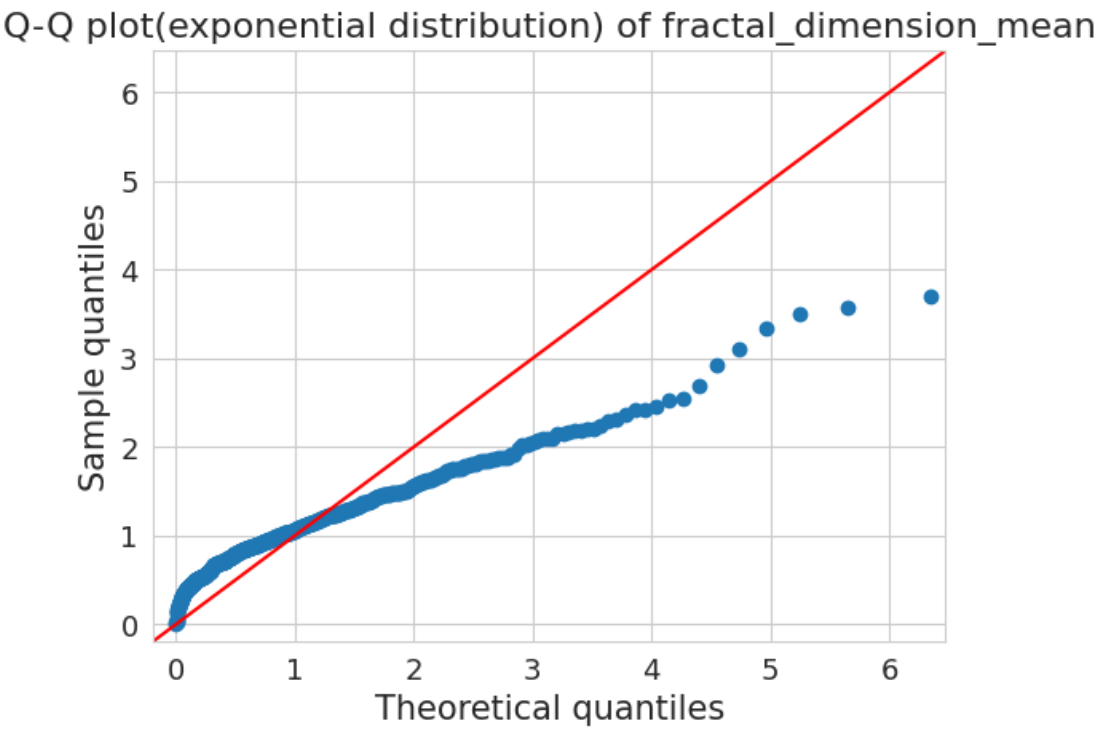

```
<Figure size 800x500 with 0 Axes>
```



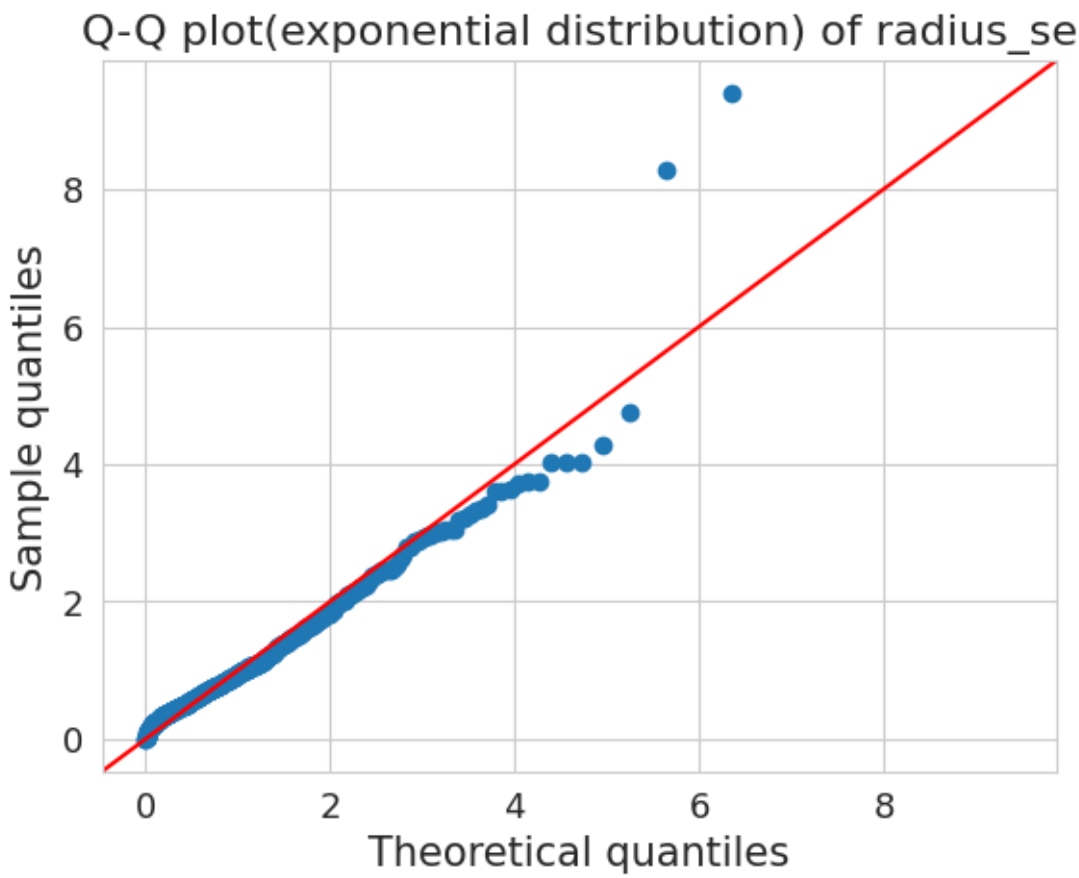

```
<Figure size 800x500 with 0 Axes>
```



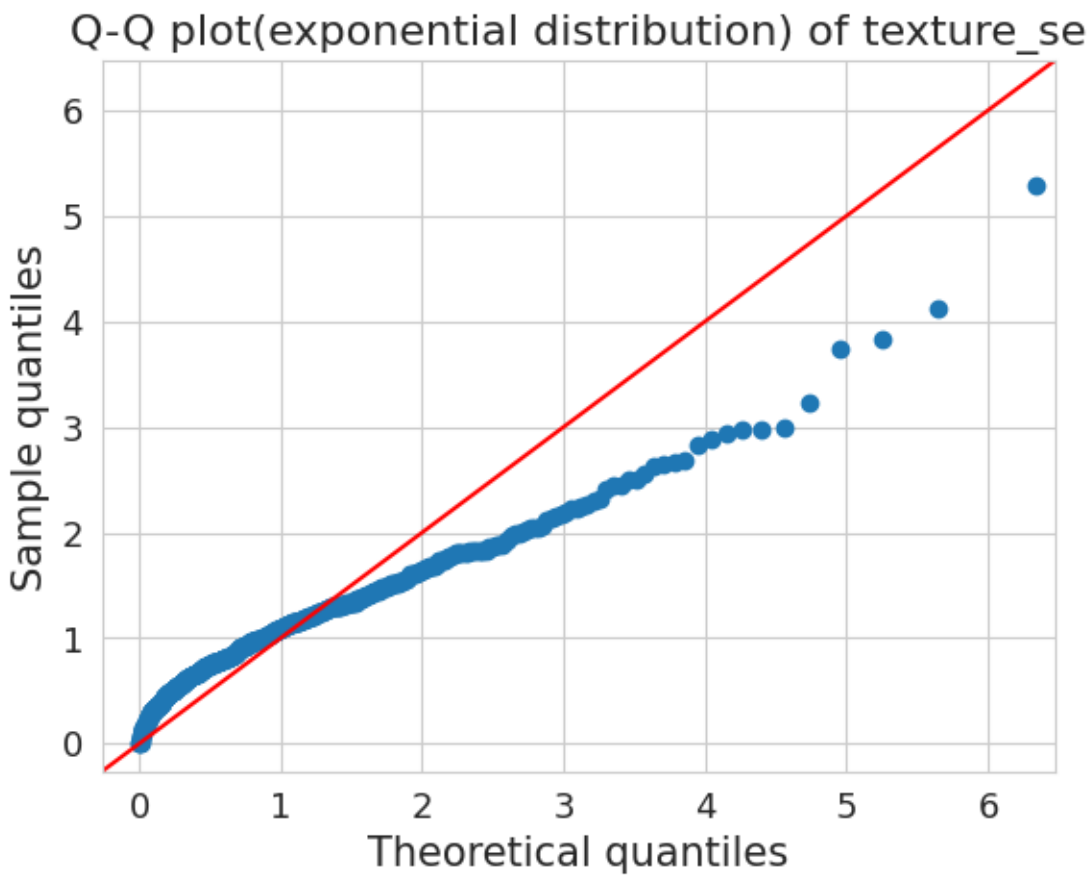

```
<Figure size 800x500 with 0 Axes>
```



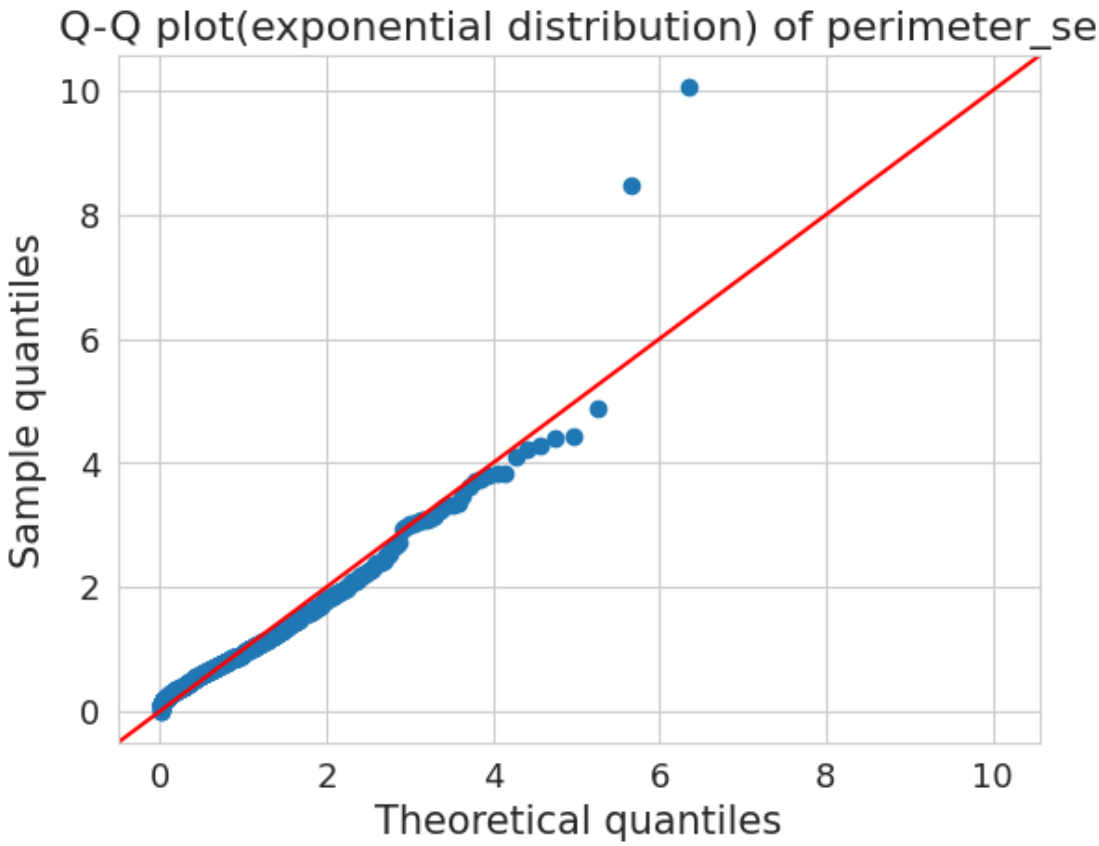

```
<Figure size 800x500 with 0 Axes>
```



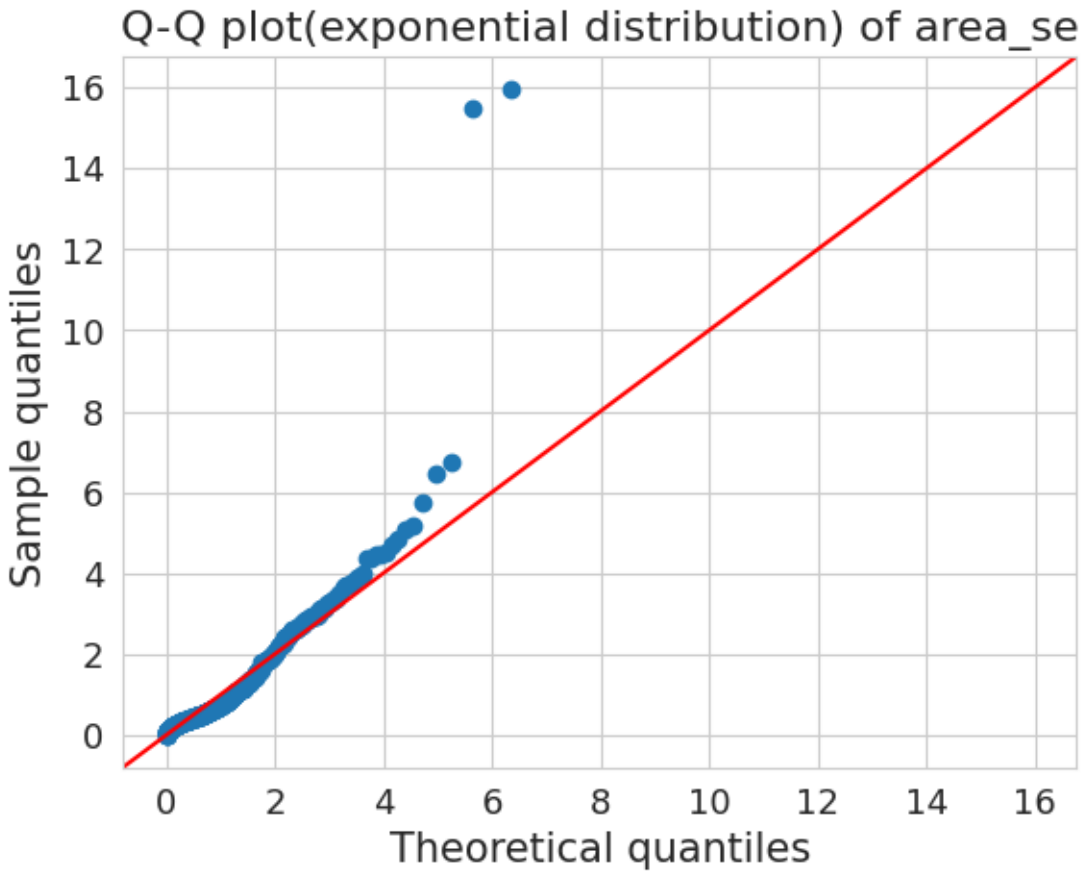

```
<Figure size 800x500 with 0 Axes>
```



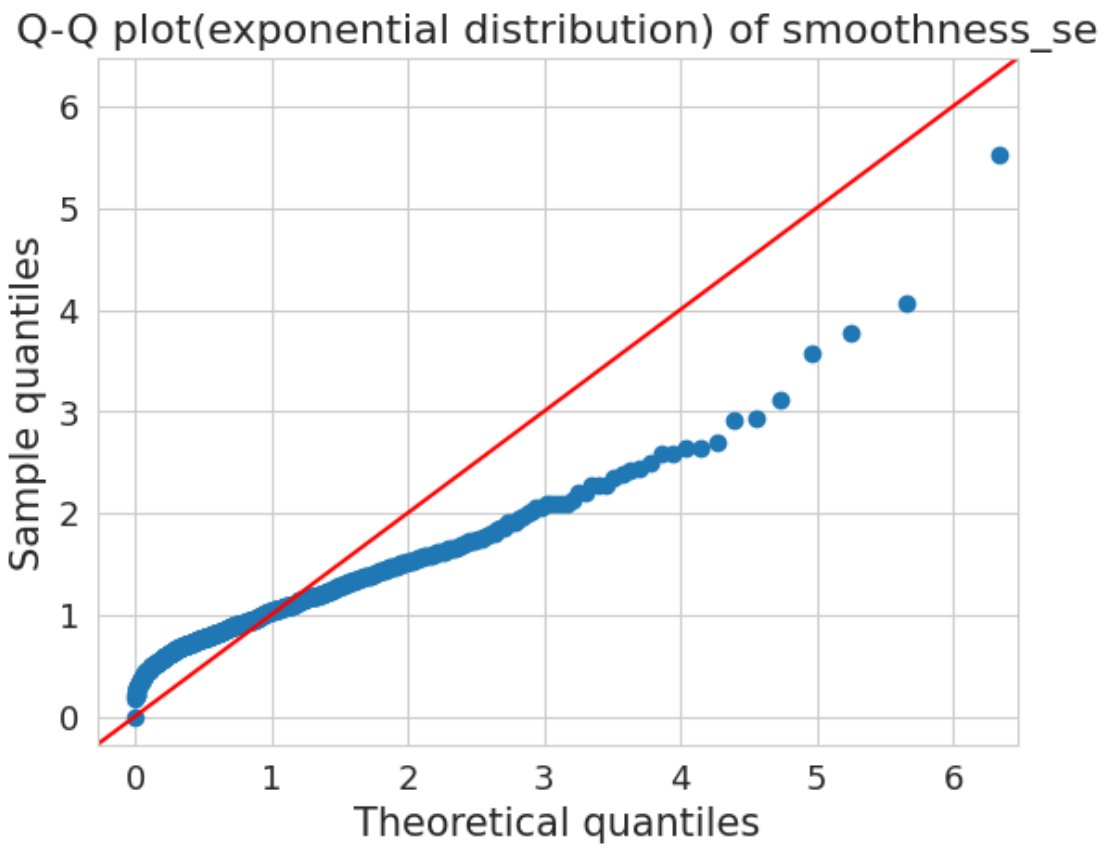

```
<Figure size 800x500 with 0 Axes>
```



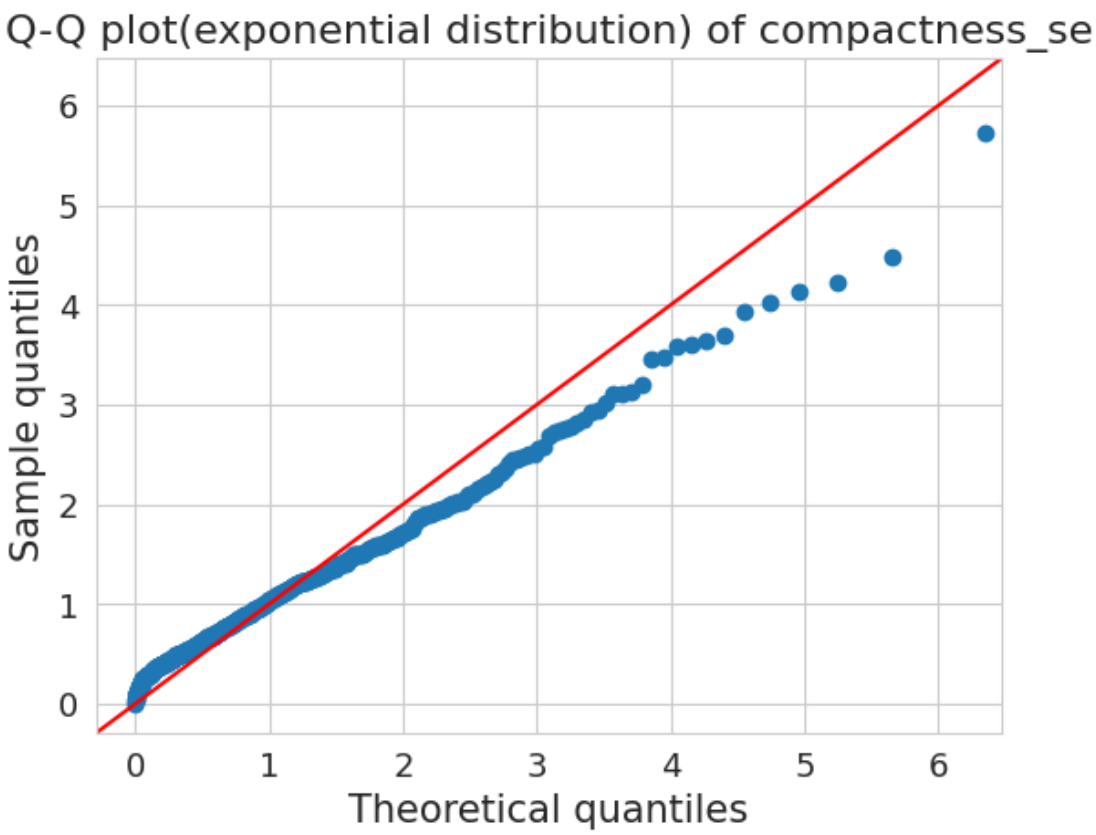

```
<Figure size 800x500 with 0 Axes>
```



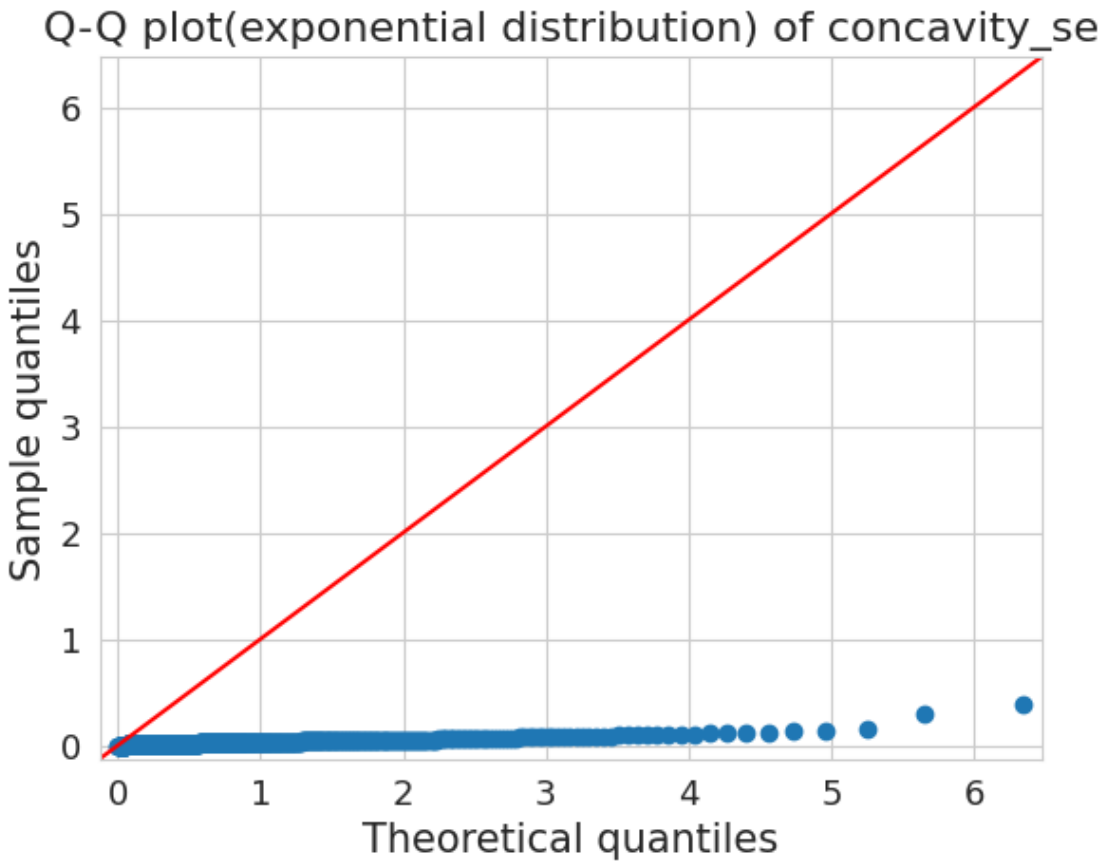

```
<Figure size 800x500 with 0 Axes>
```



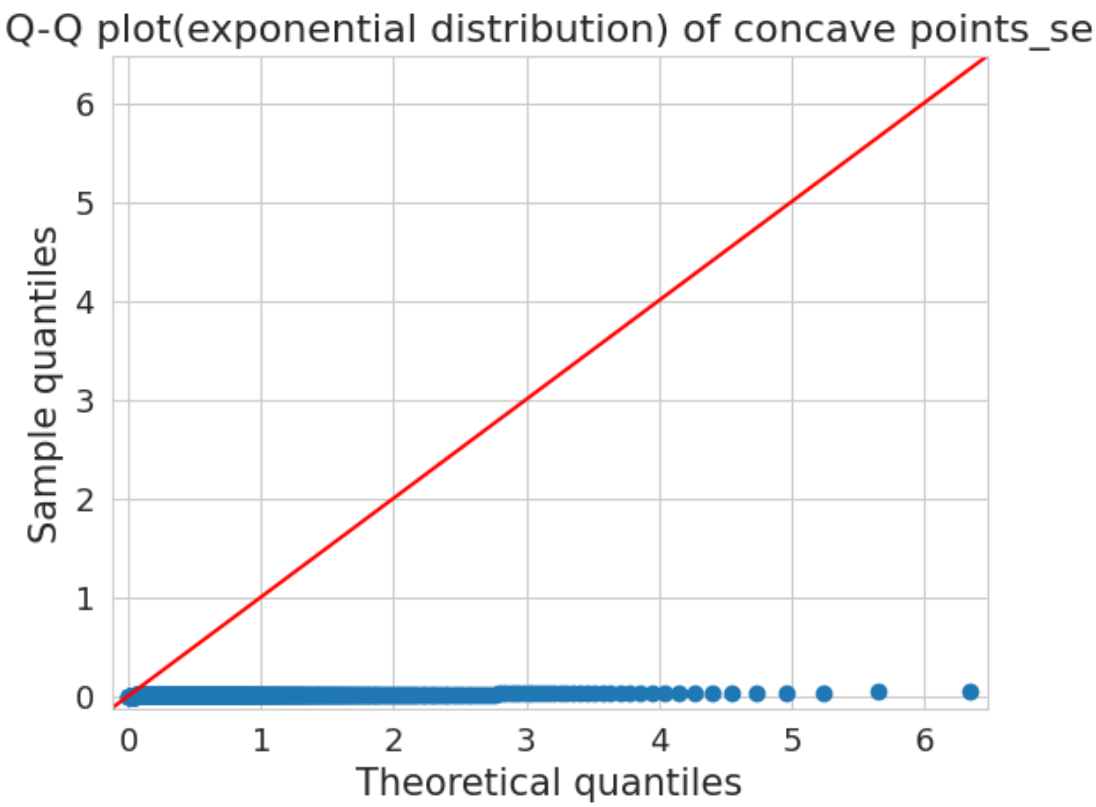

```
<Figure size 800x500 with 0 Axes>
```



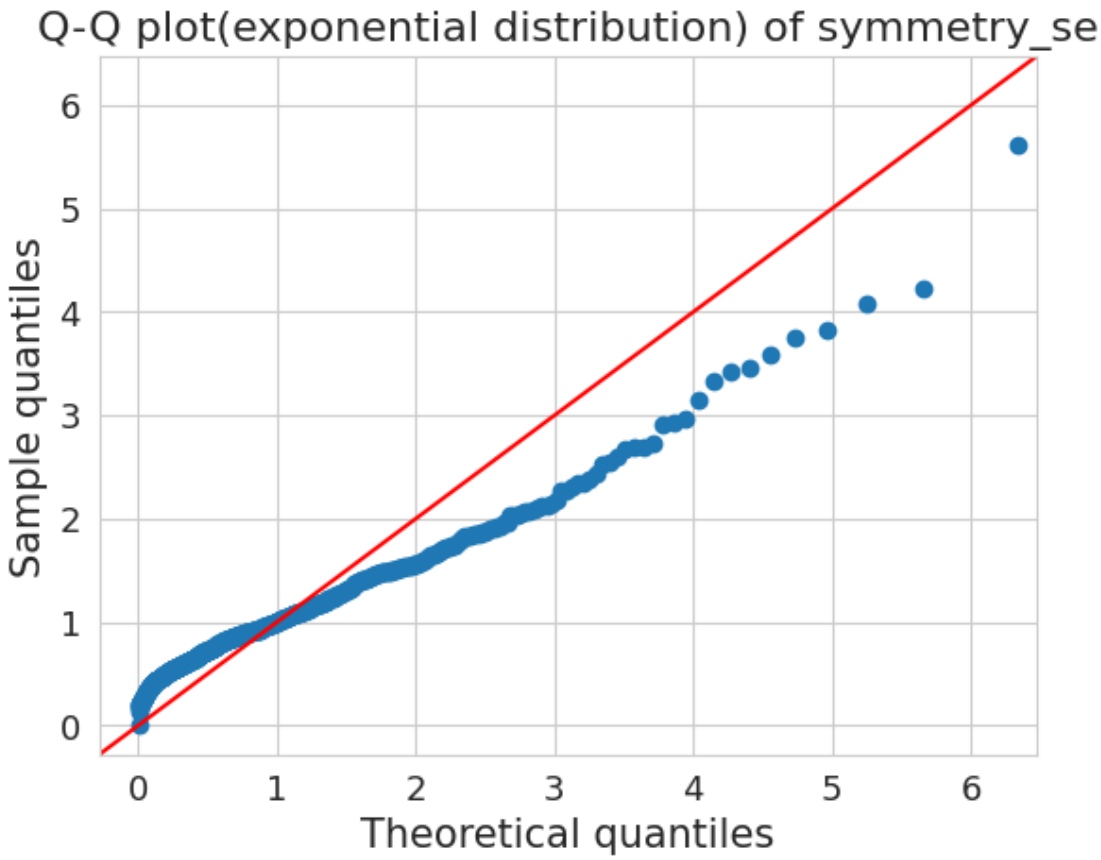

```
<Figure size 800x500 with 0 Axes>
```



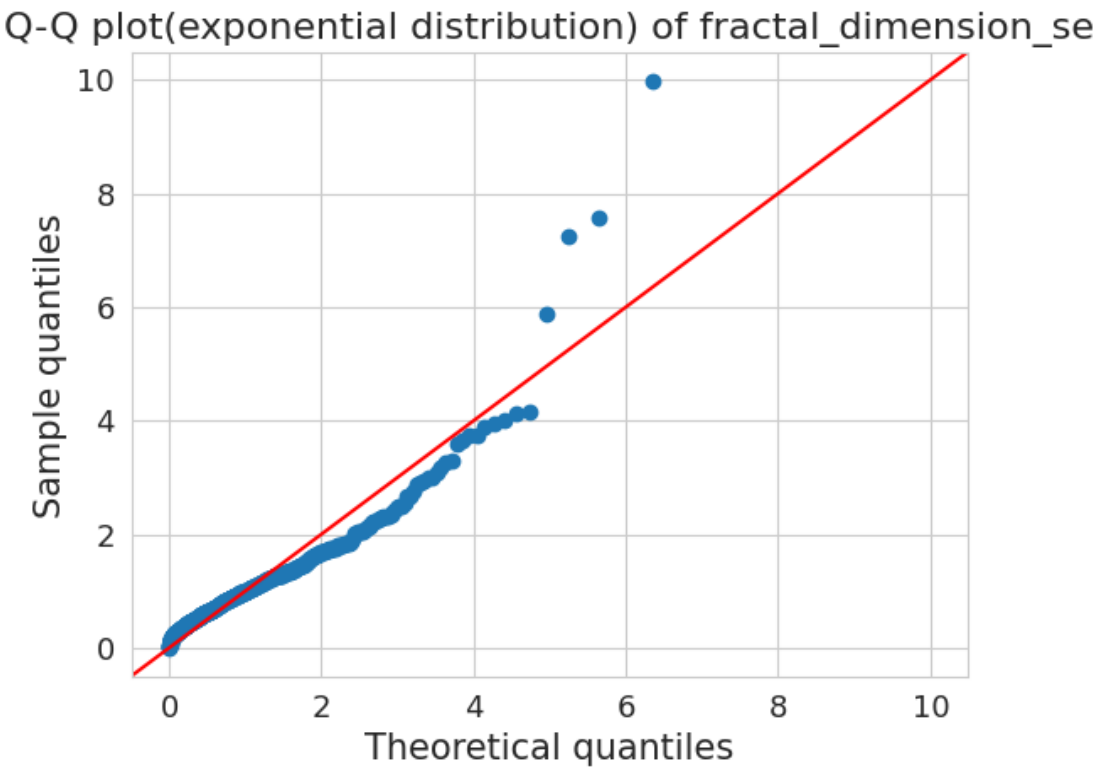

```
<Figure size 800x500 with 0 Axes>
```



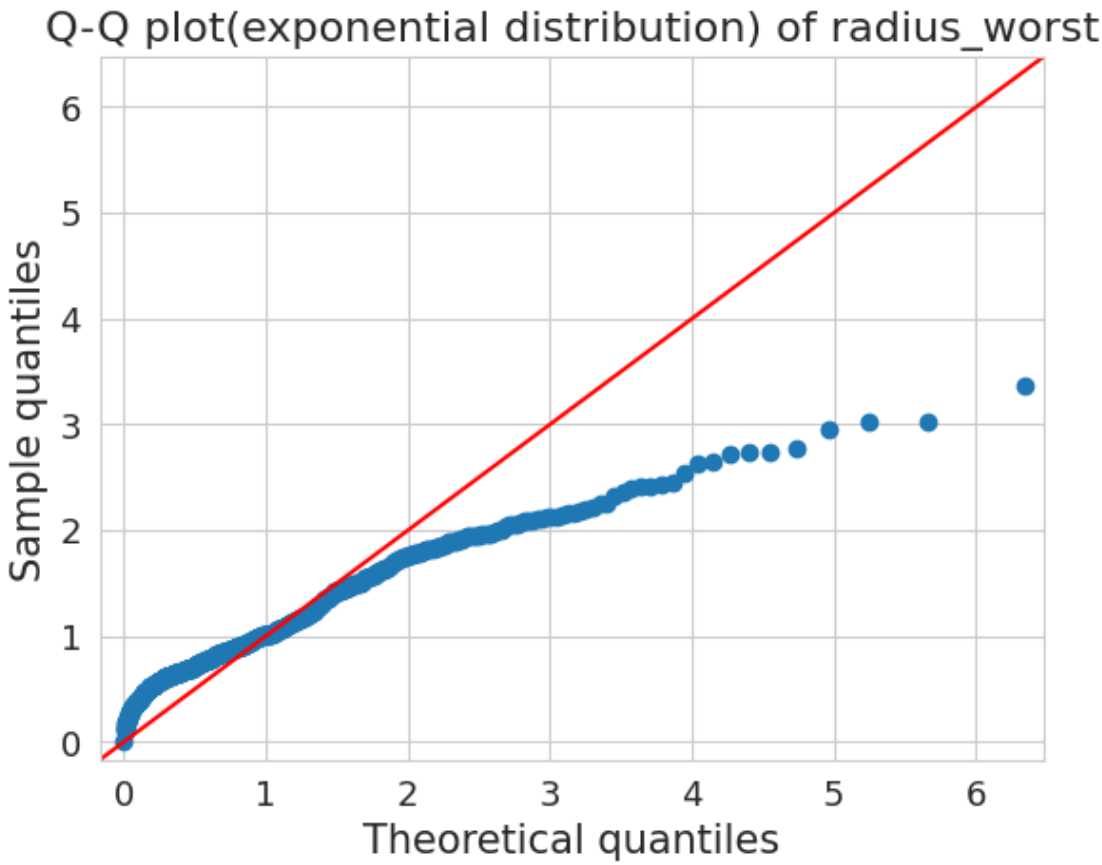

```
<Figure size 800x500 with 0 Axes>
```



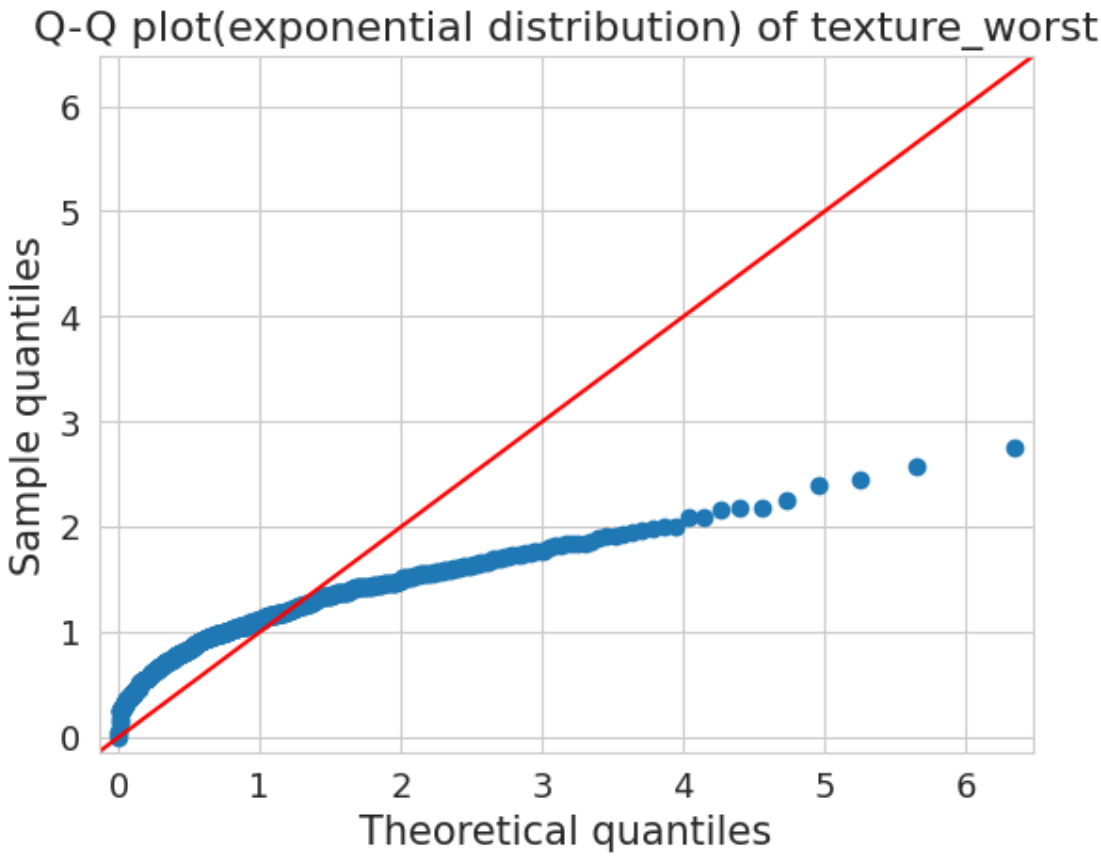

```
<Figure size 800x500 with 0 Axes>
```



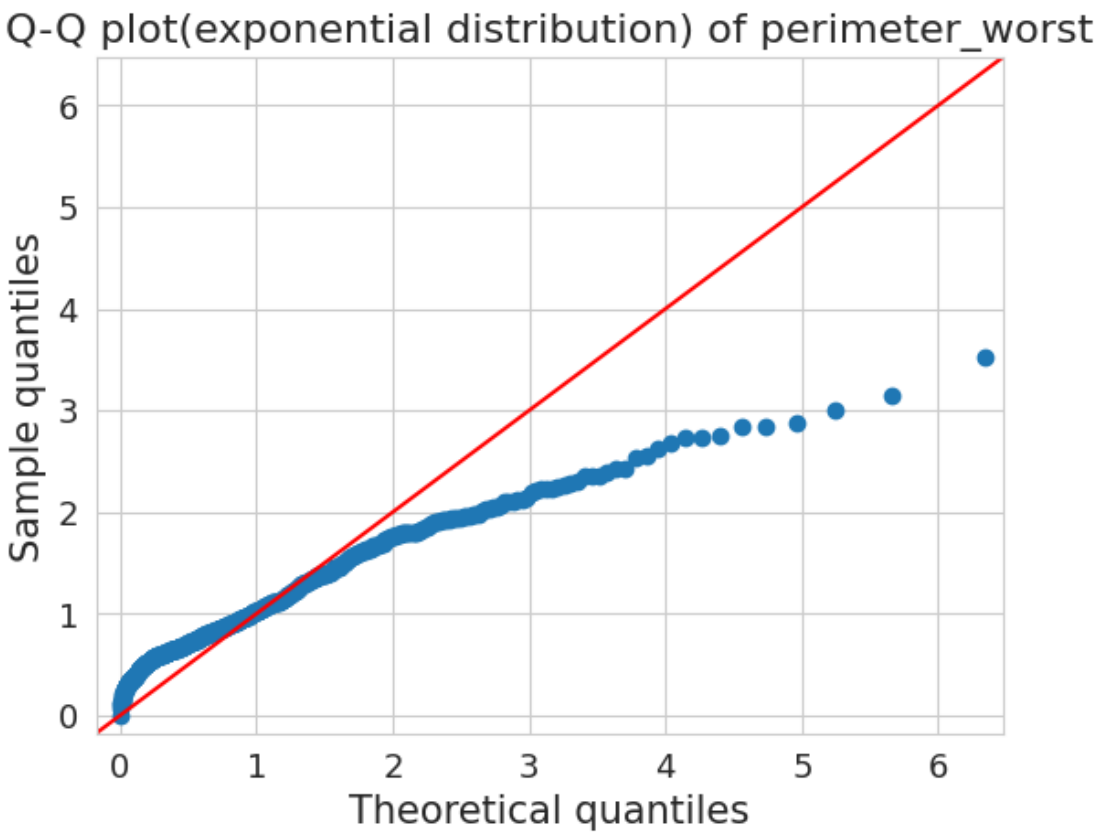

```
<Figure size 800x500 with 0 Axes>
```



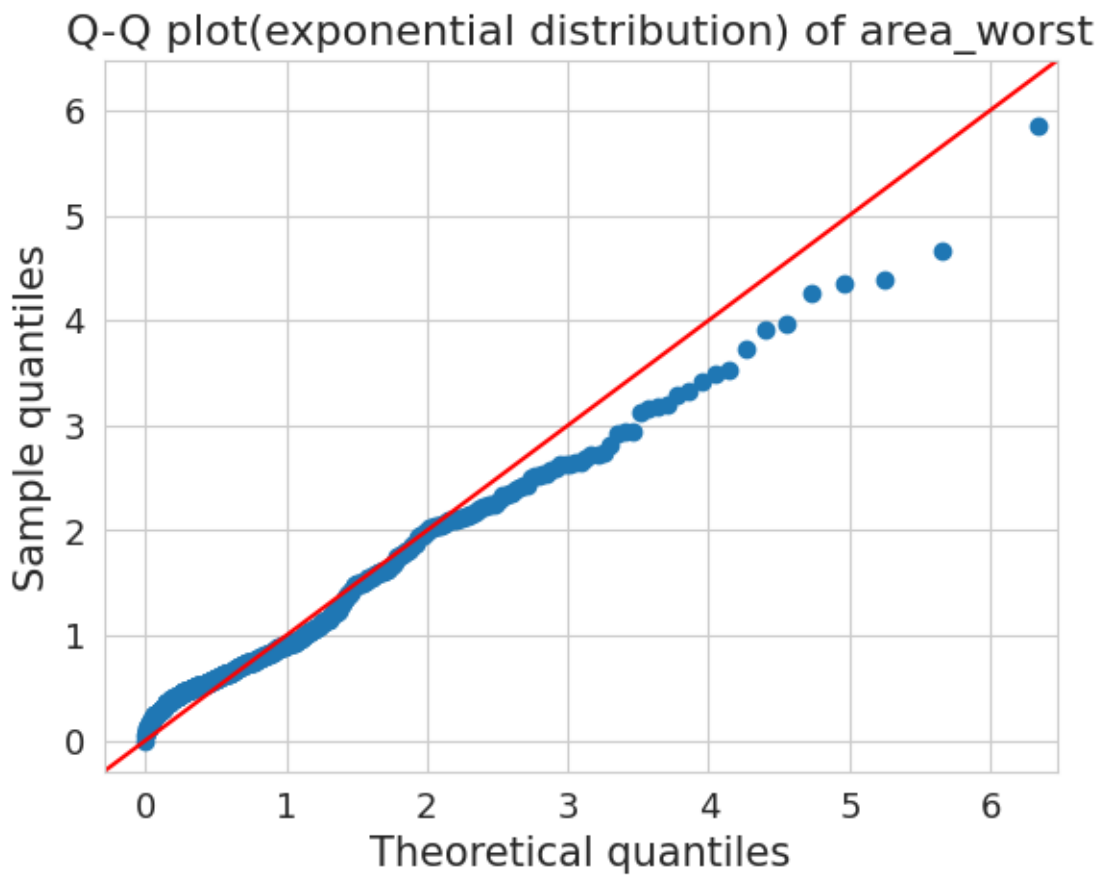

```
<Figure size 800x500 with 0 Axes>
```



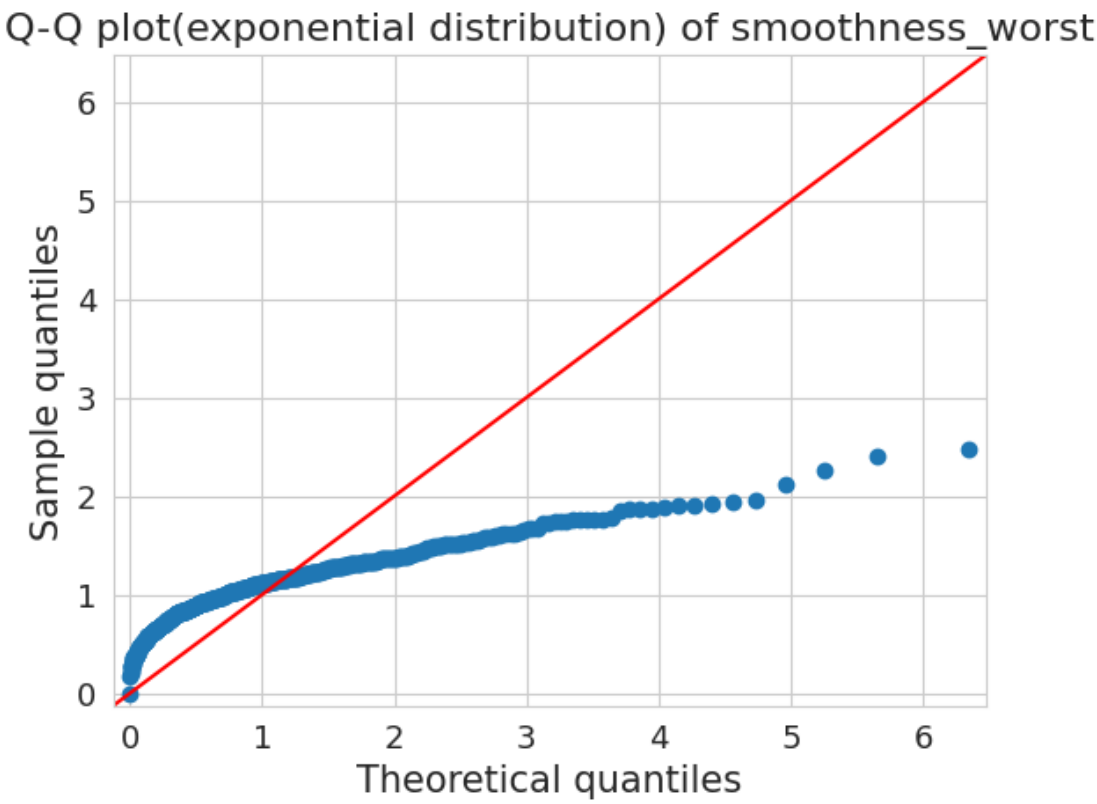

```
<Figure size 800x500 with 0 Axes>
```



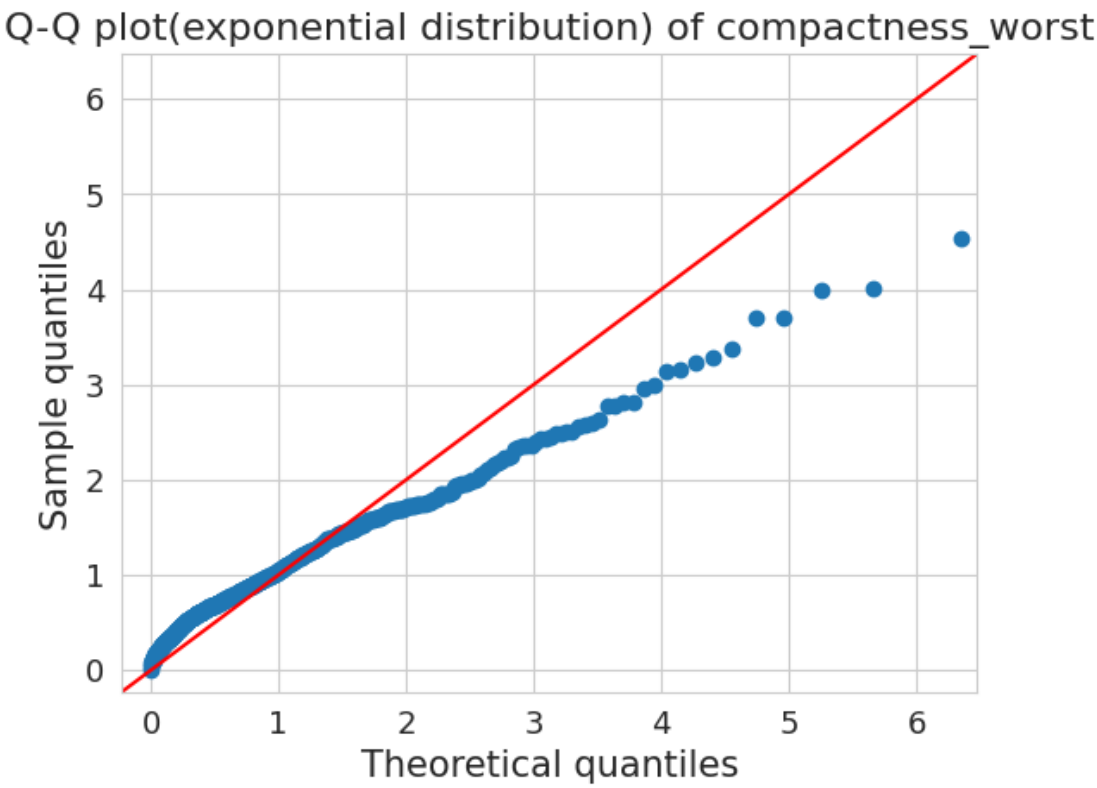

```
<Figure size 800x500 with 0 Axes>
```



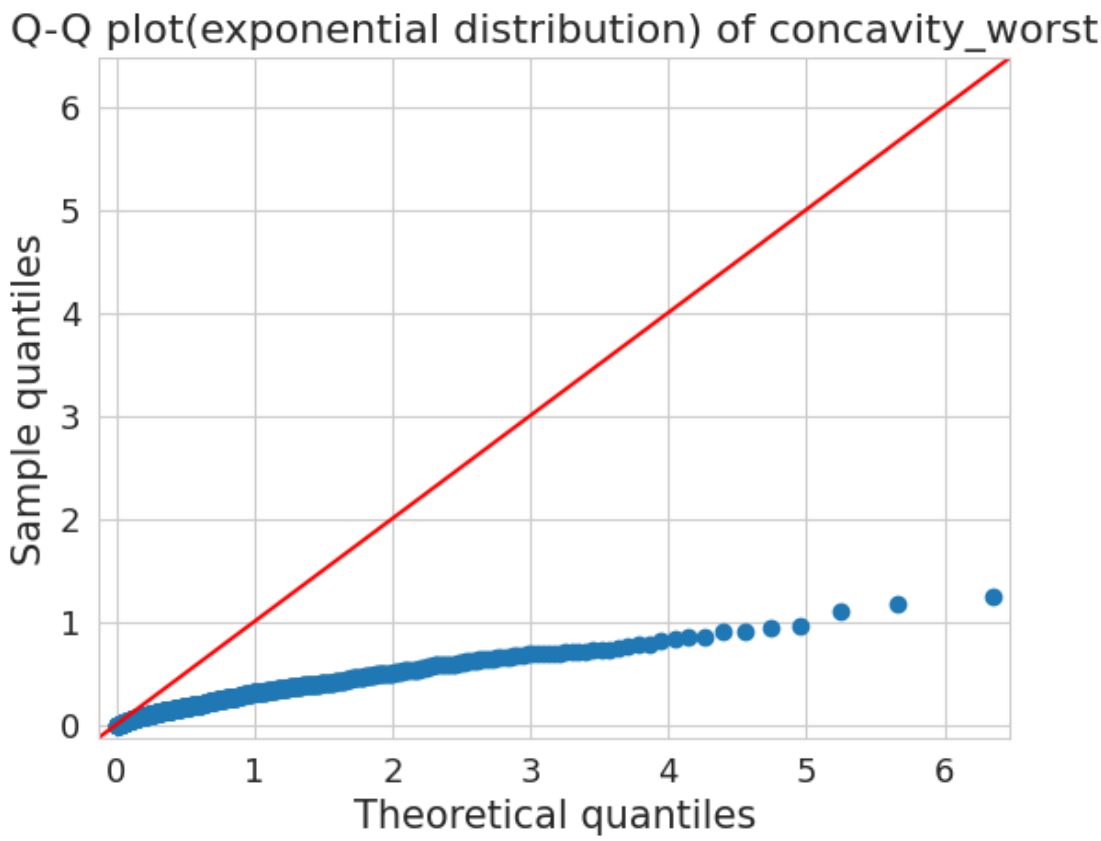

```
<Figure size 800x500 with 0 Axes>
```



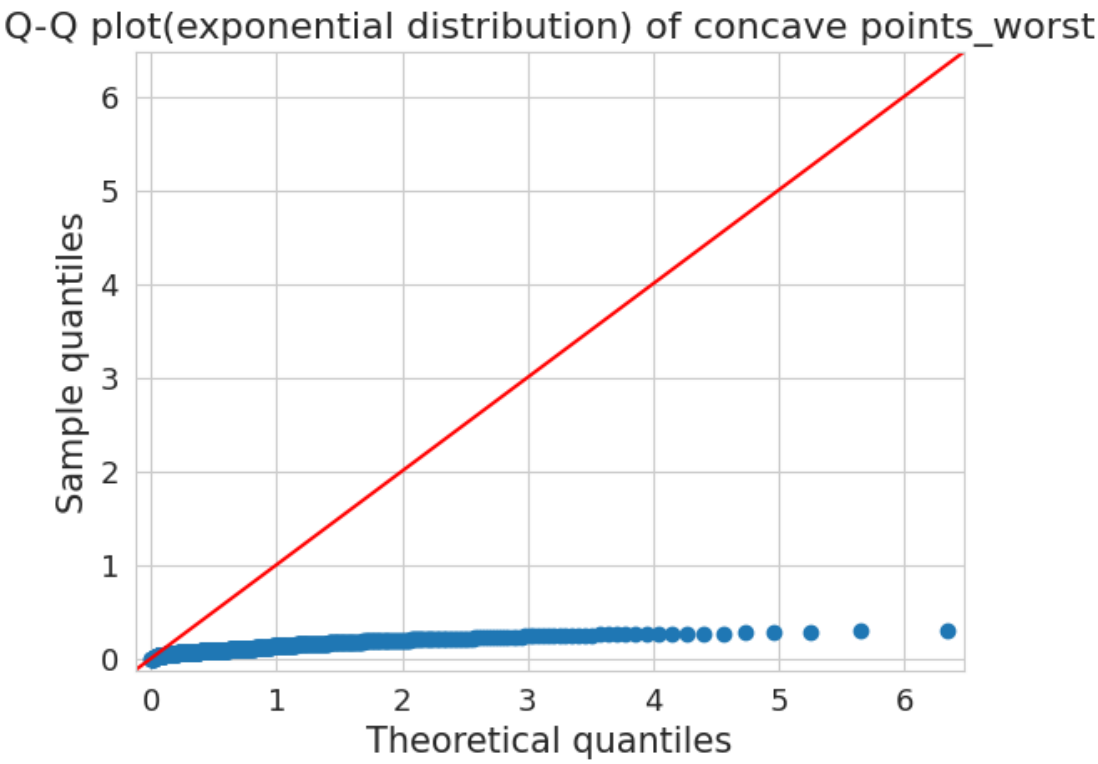

```
<Figure size 800x500 with 0 Axes>
```



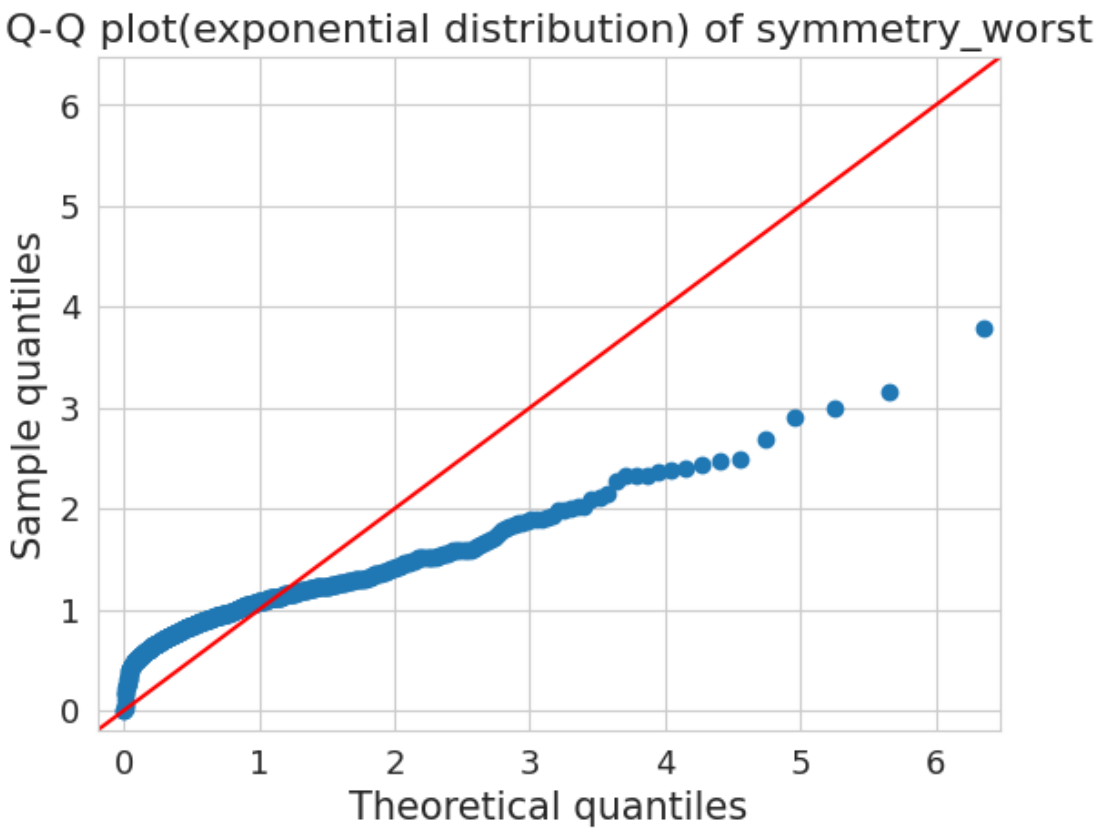

```
<Figure size 800x500 with 0 Axes>
```



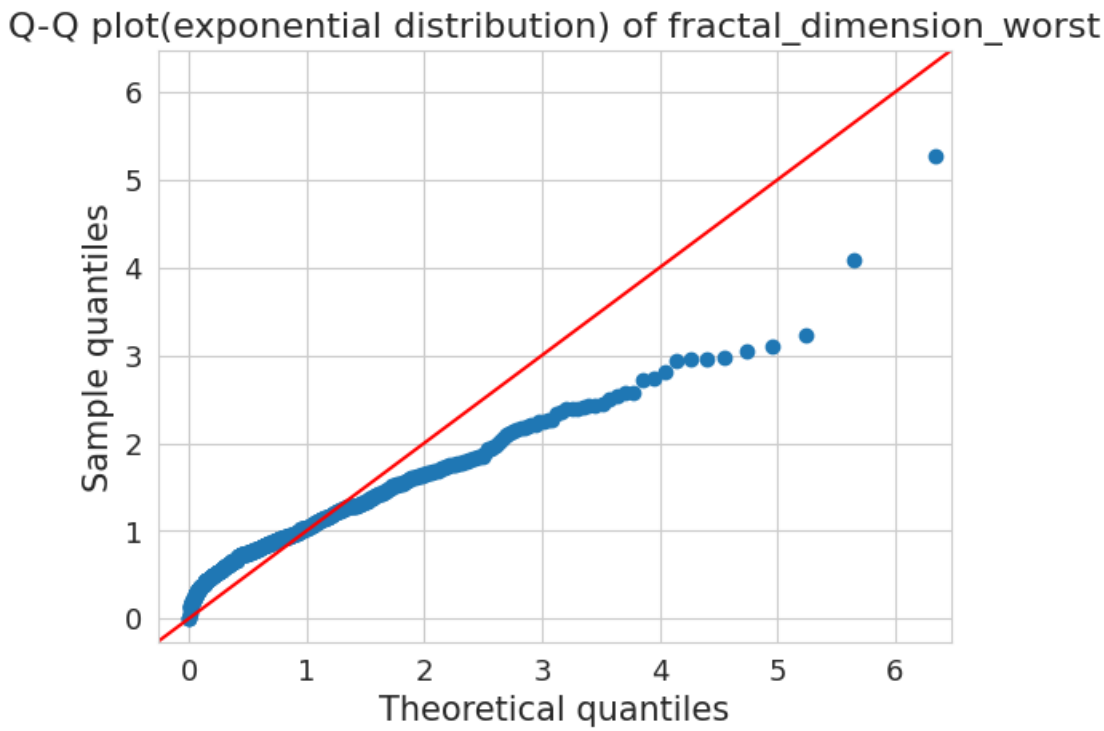

```
<Figure size 800x500 with 0 Axes>
```




```
<Figure size 800x500 with 0 Axes>
```

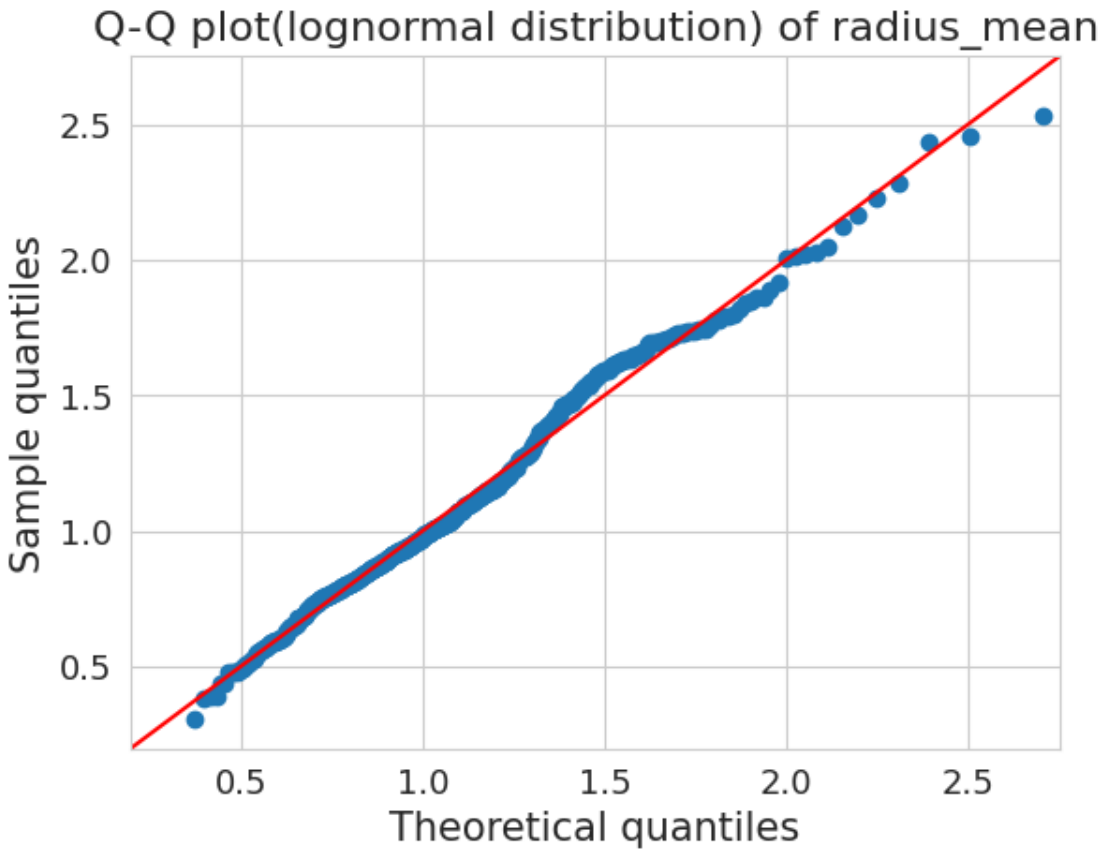



```
<Figure size 800x500 with 0 Axes>
```

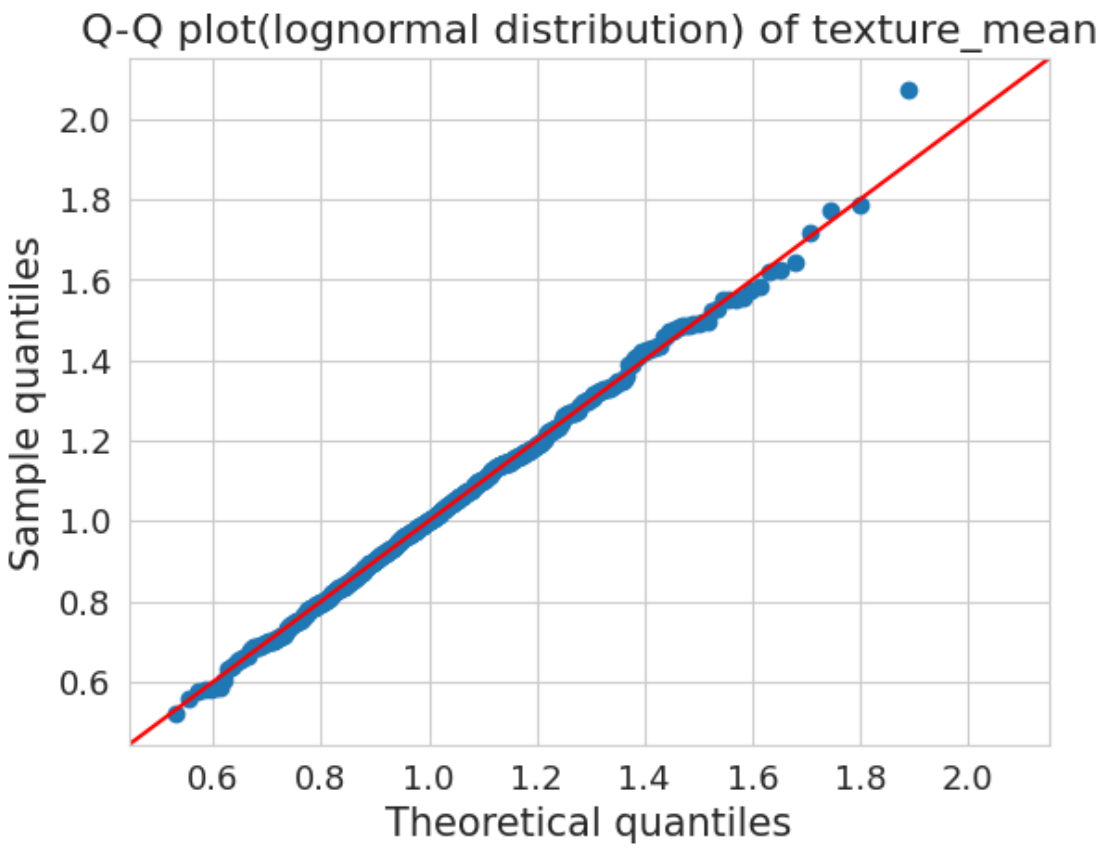


[228]:
```
#lognormal distribution
for c in df.columns:
        plt.figure(figsize=(8,5))
        fig=qqplot(df[c],dist=sc.stats.lognorm,line="45",fit='True')
        plt.xticks(fontsize=13)
        plt.yticks(fontsize=13)
        plt.xlabel("Theoretical quantiles",fontsize=15)
        plt.ylabel("Sample quantiles",fontsize=15)
        plt.title("Q-Q plot(lognormal distribution) of {}".
 ↪format(c),fontsize=16)
        plt.grid(True)
        plt.show()
```

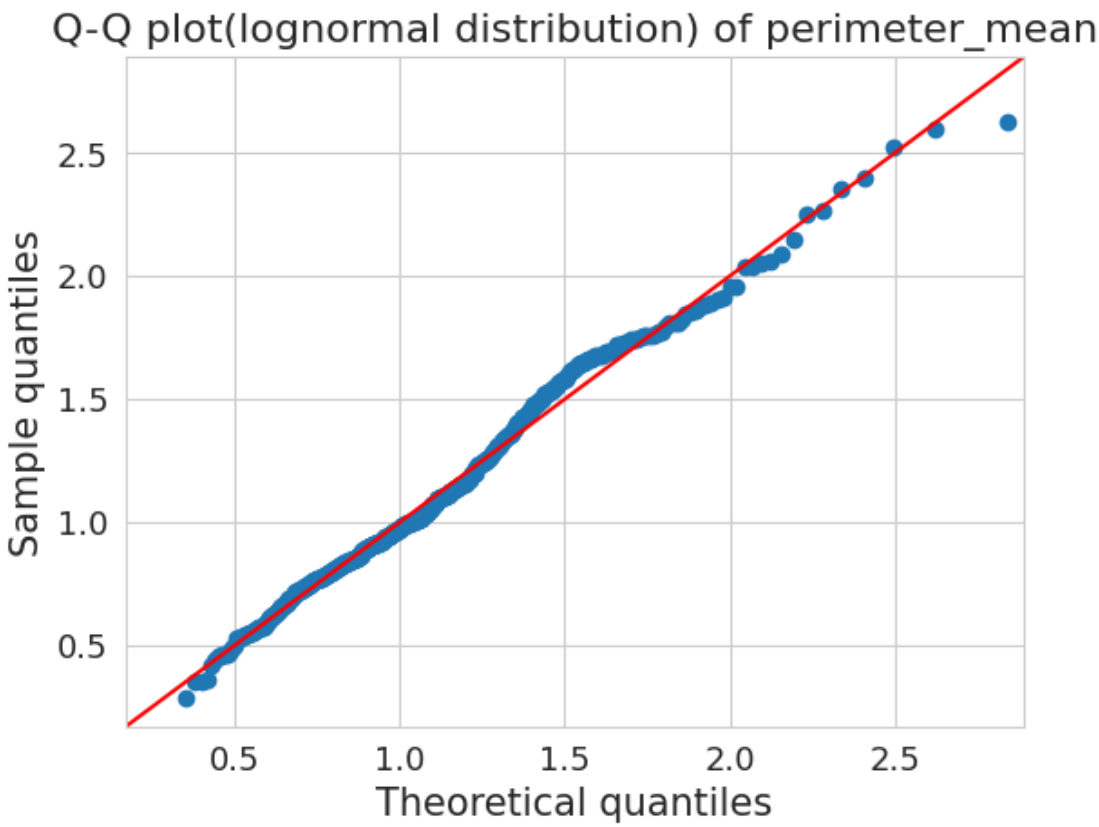
```
<Figure size 800x500 with 0 Axes>
```



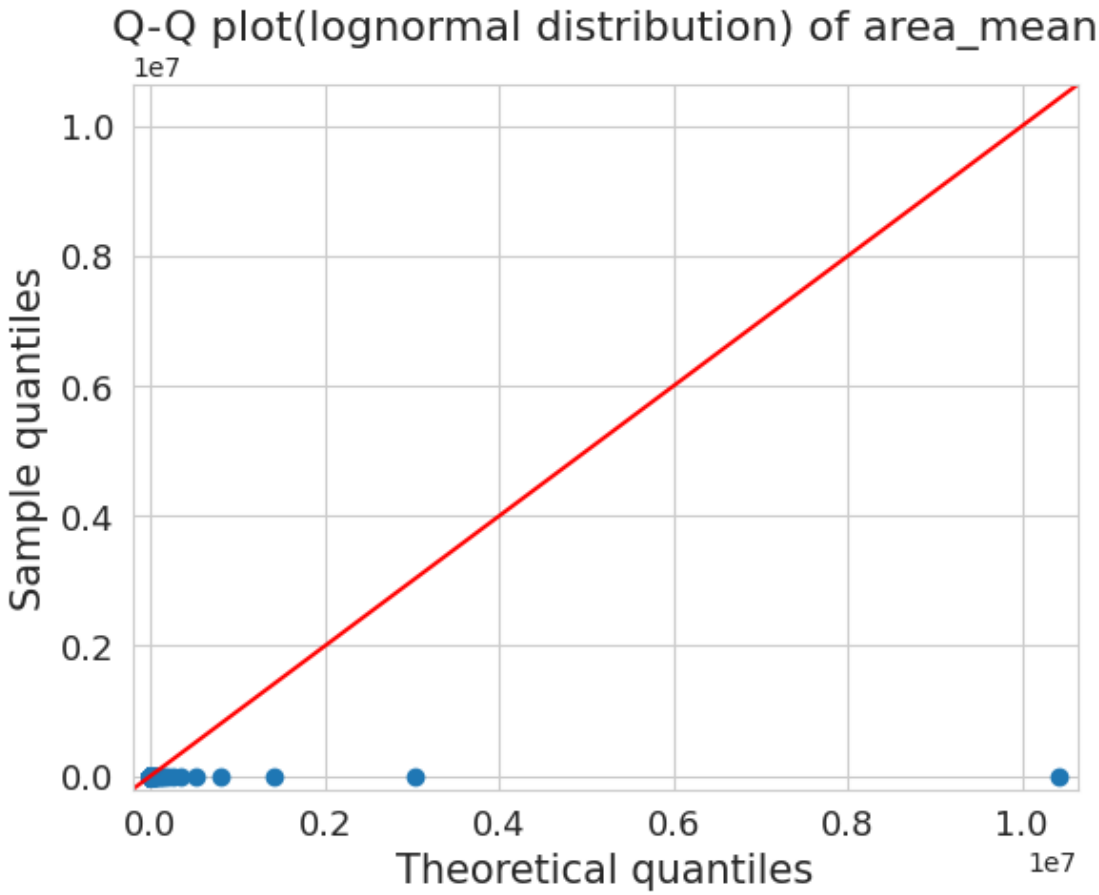

```
<Figure size 800x500 with 0 Axes>
```



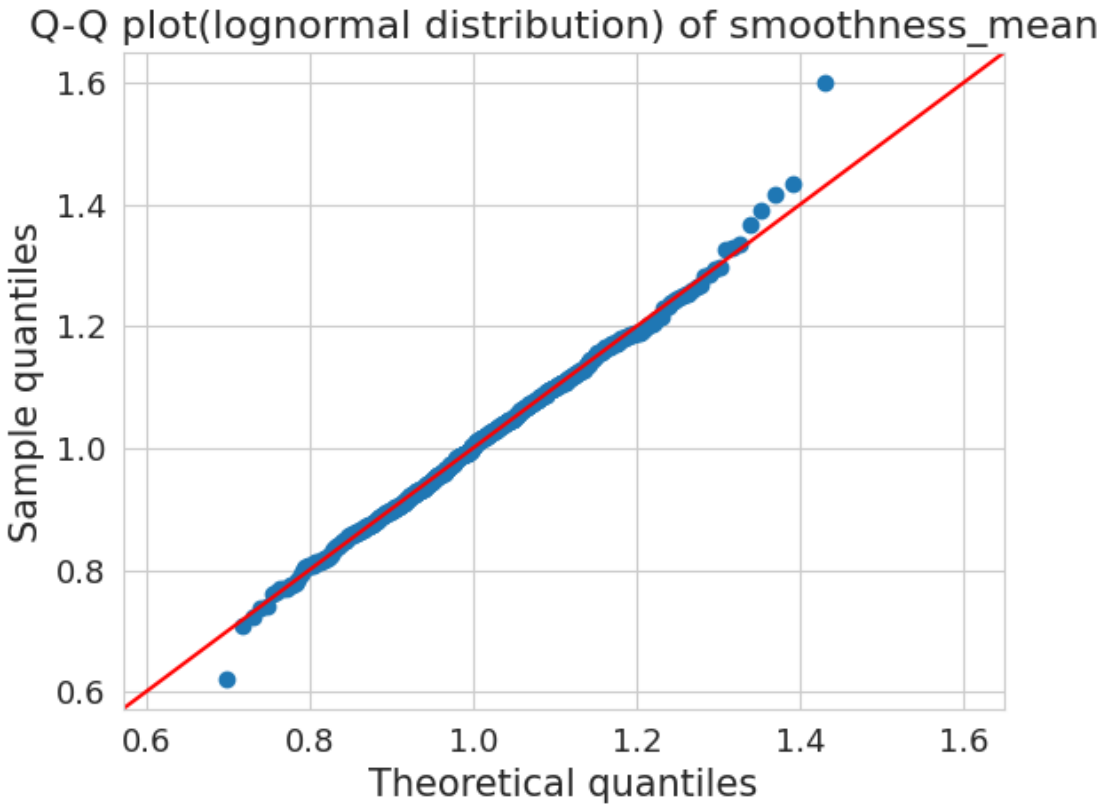

```
<Figure size 800x500 with 0 Axes>
```



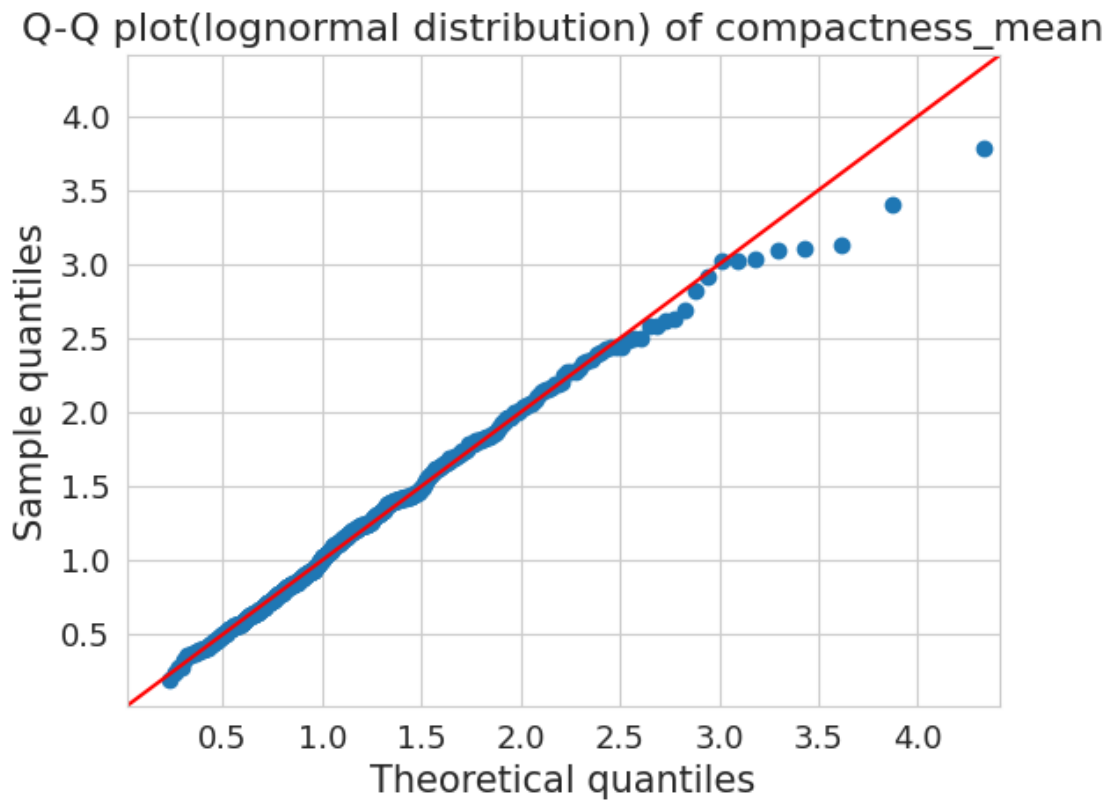

```
<Figure size 800x500 with 0 Axes>
```



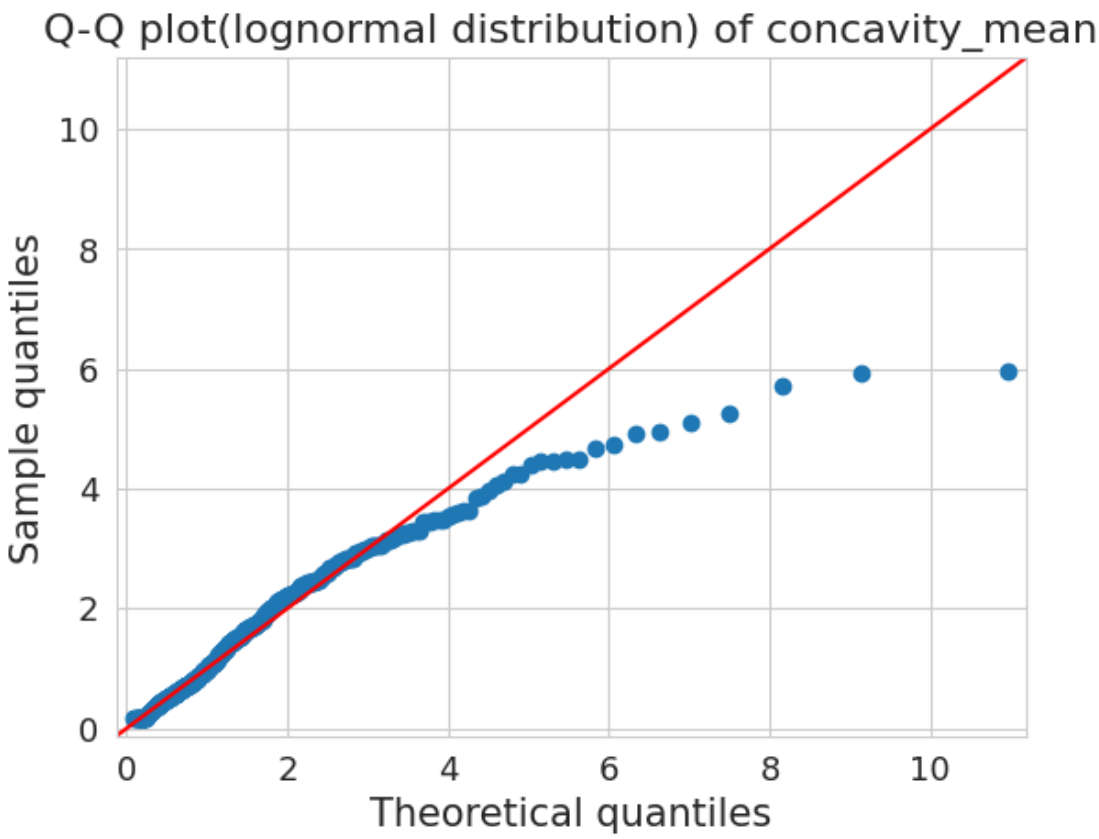

```
<Figure size 800x500 with 0 Axes>
```



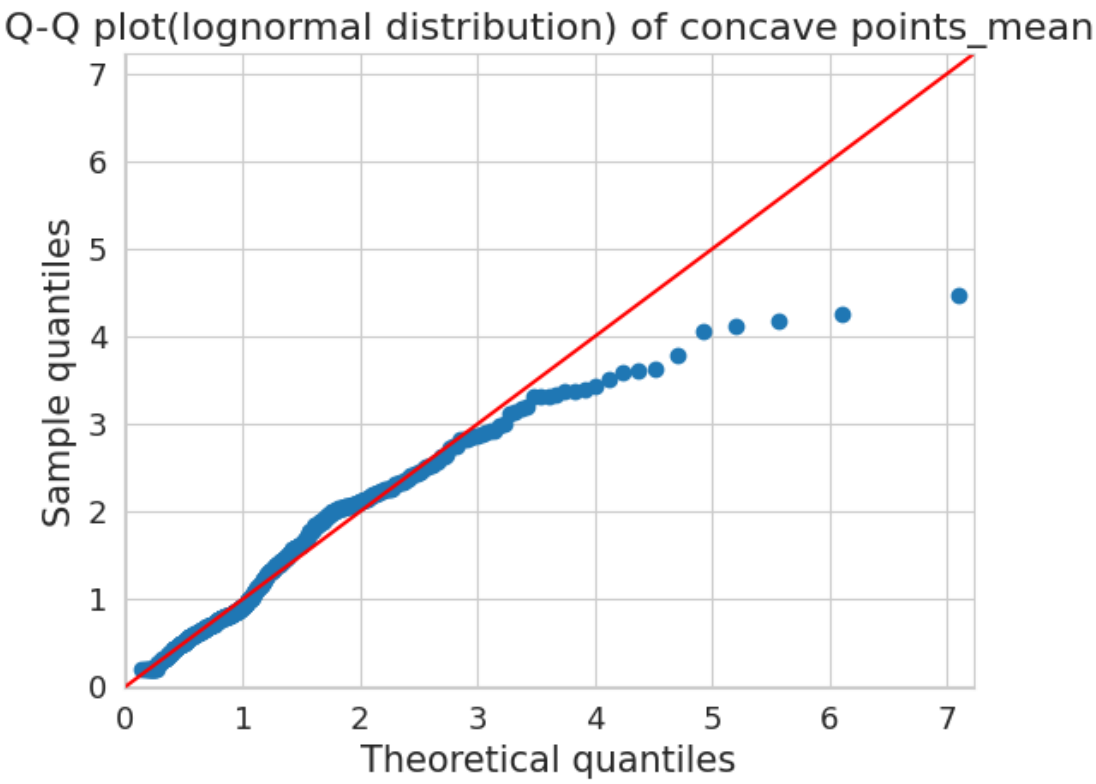

```
<Figure size 800x500 with 0 Axes>
```



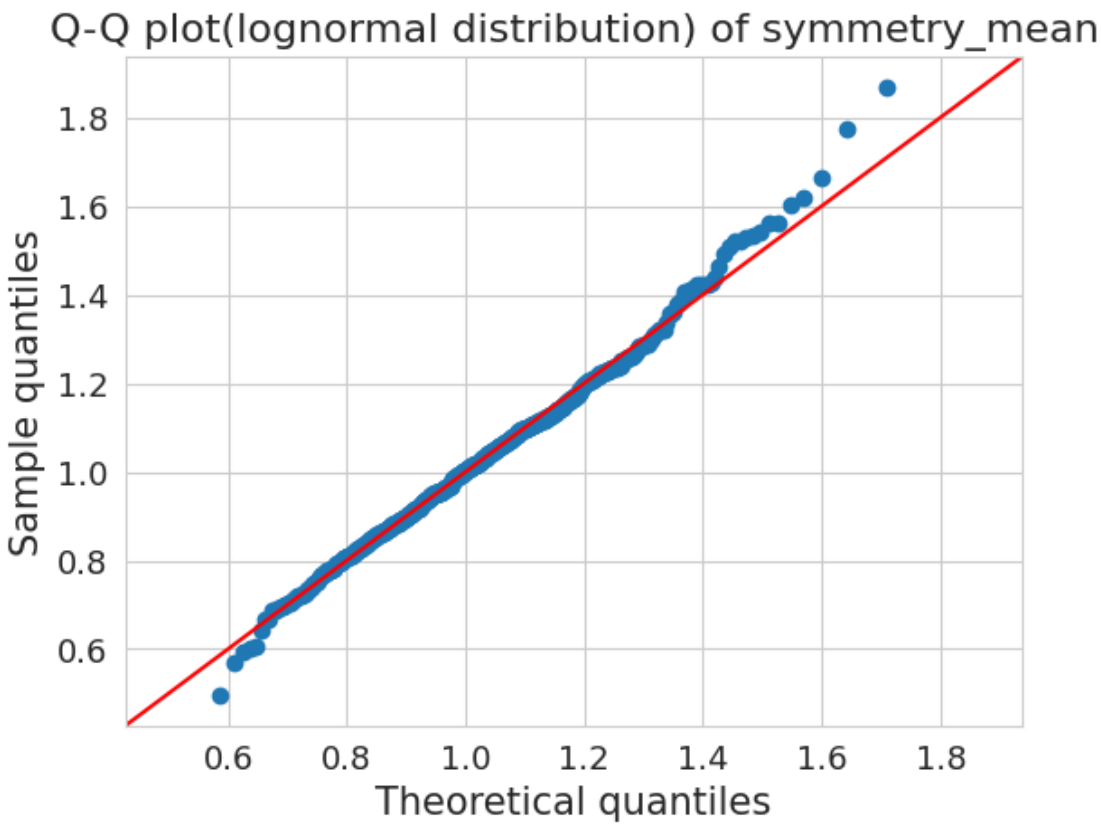

```
<Figure size 800x500 with 0 Axes>
```



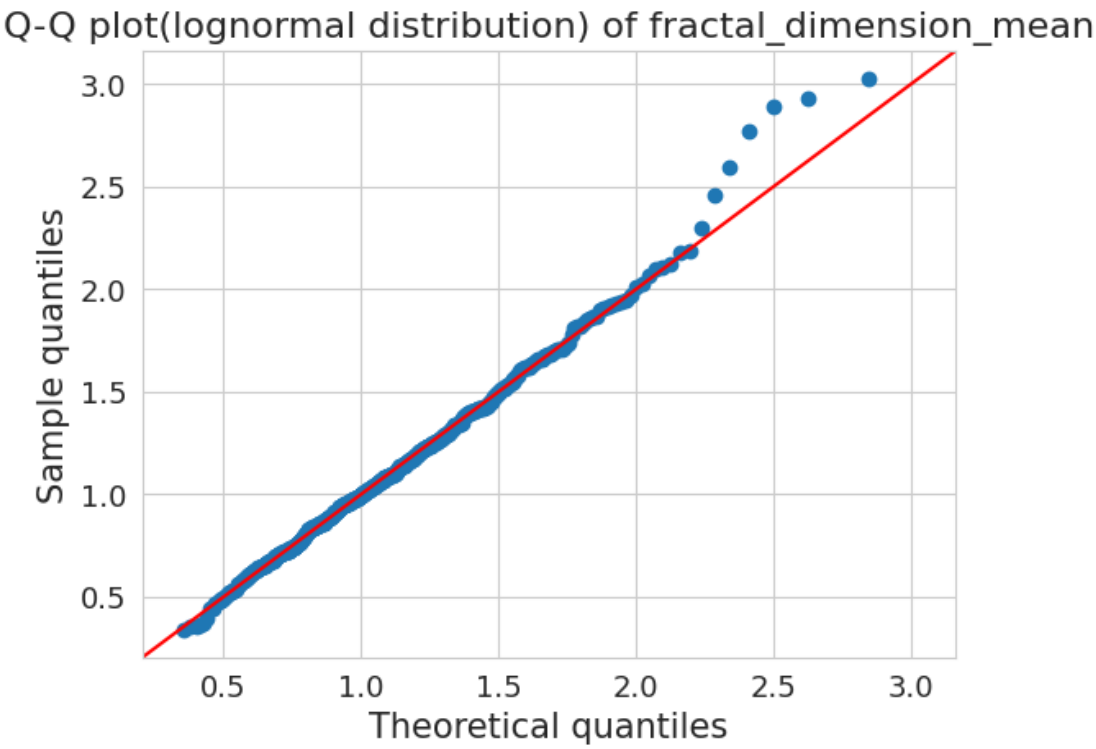

```
<Figure size 800x500 with 0 Axes>
```



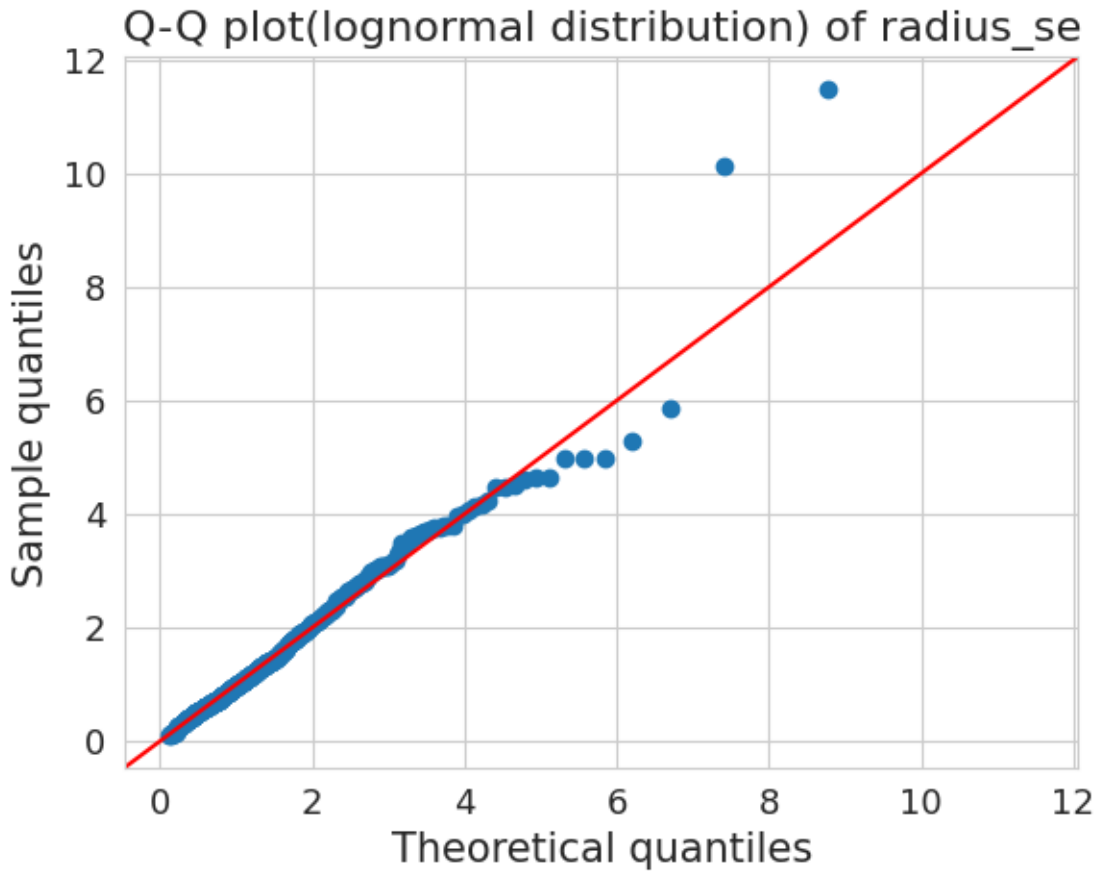

```
<Figure size 800x500 with 0 Axes>
```



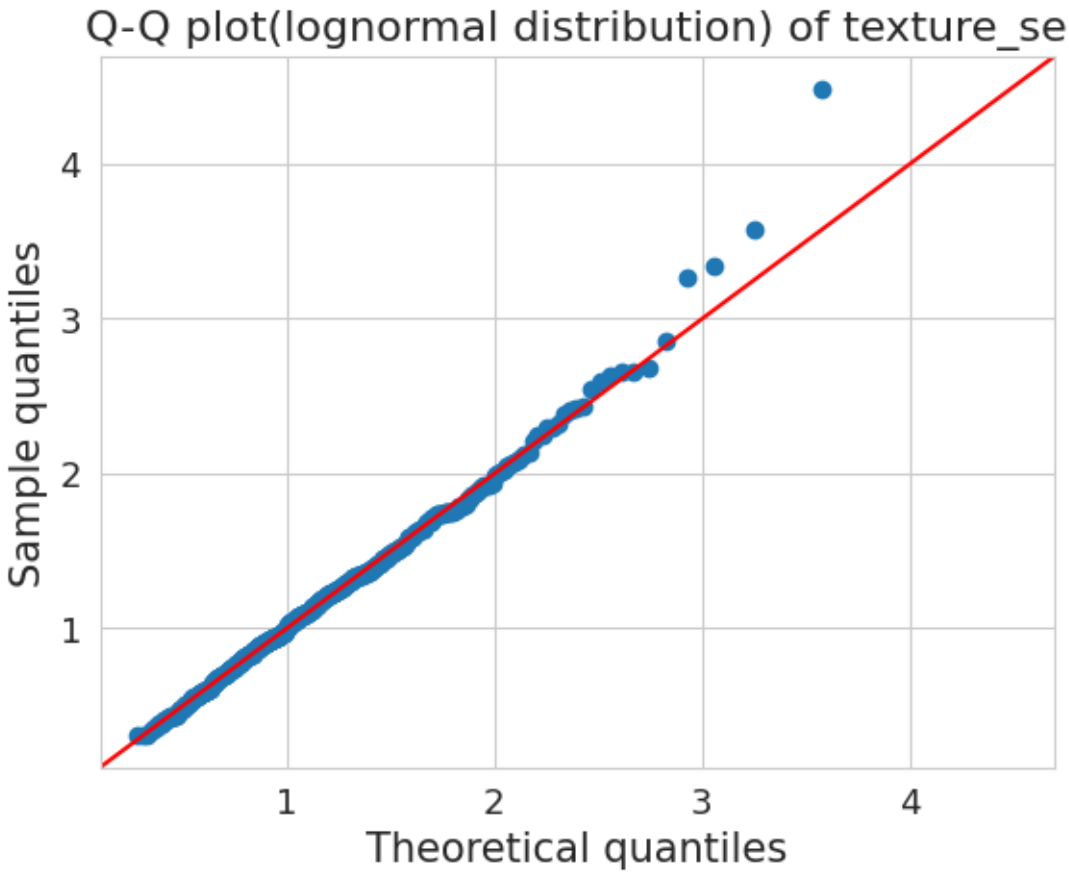

```
<Figure size 800x500 with 0 Axes>
```



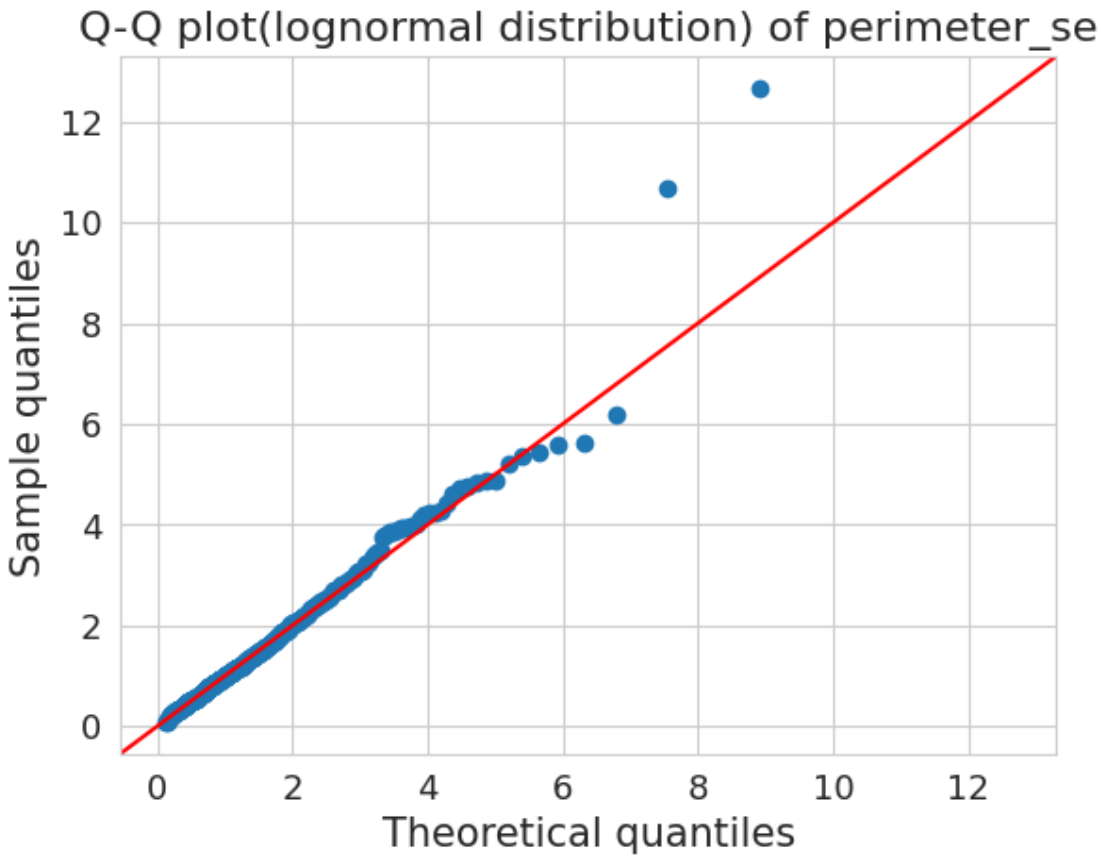

```
<Figure size 800x500 with 0 Axes>
```



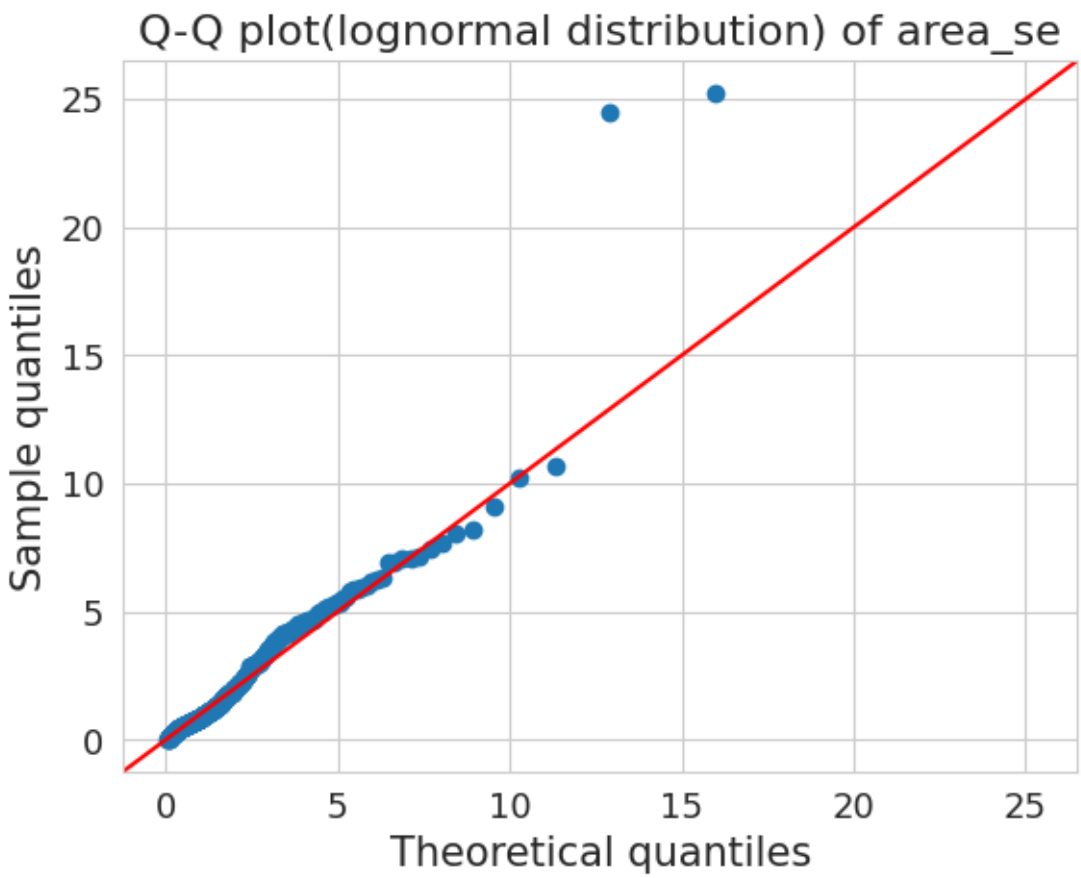

```
<Figure size 800x500 with 0 Axes>
```



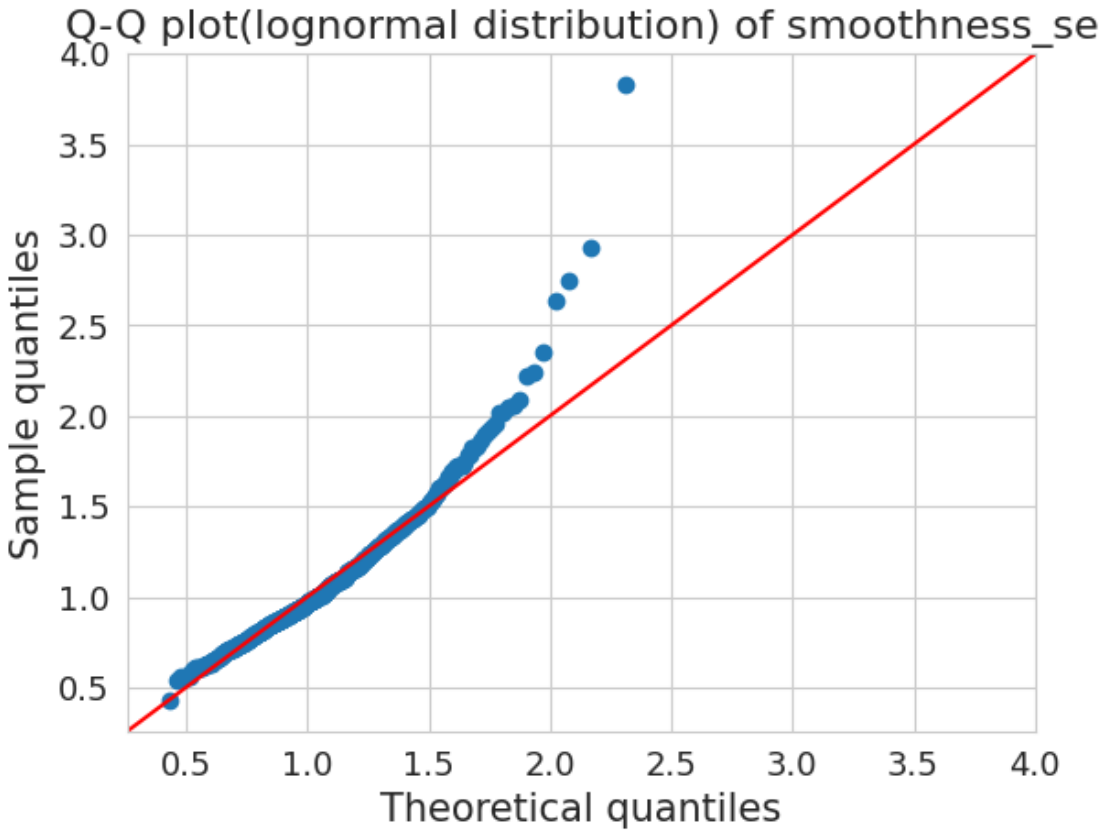

```
<Figure size 800x500 with 0 Axes>
```



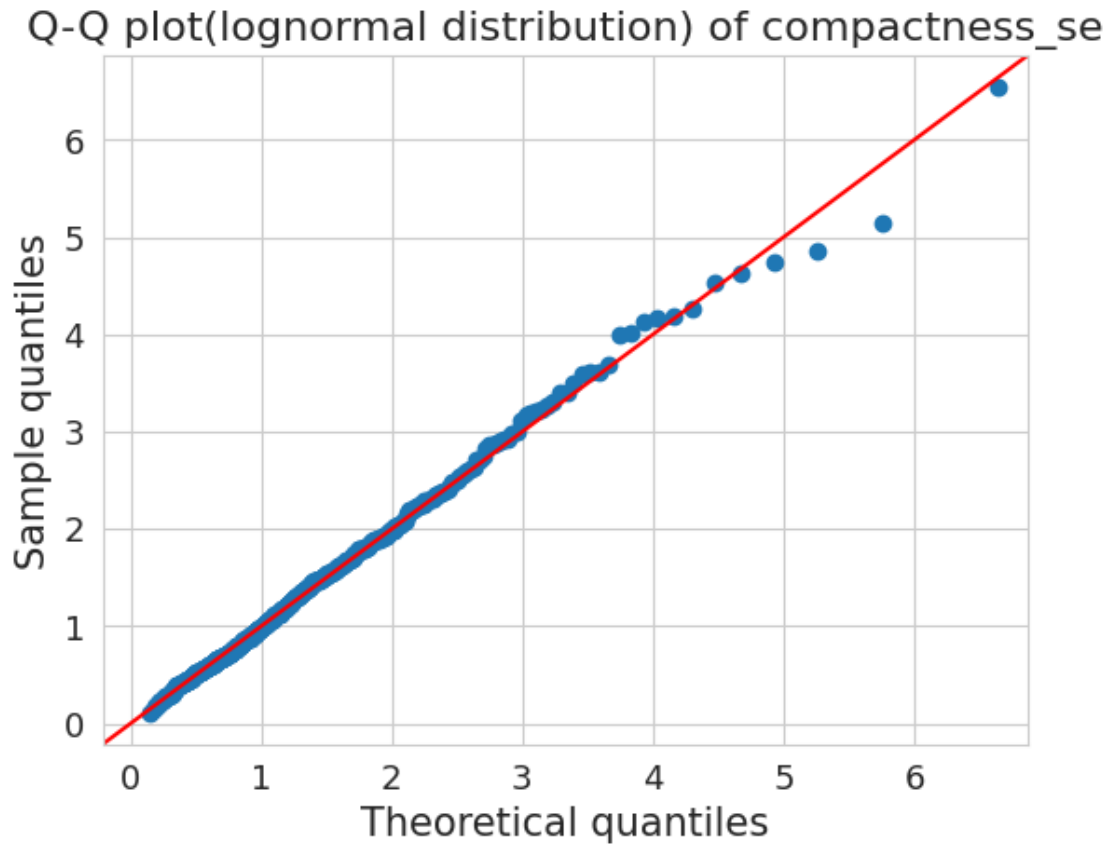

```
<Figure size 800x500 with 0 Axes>
```



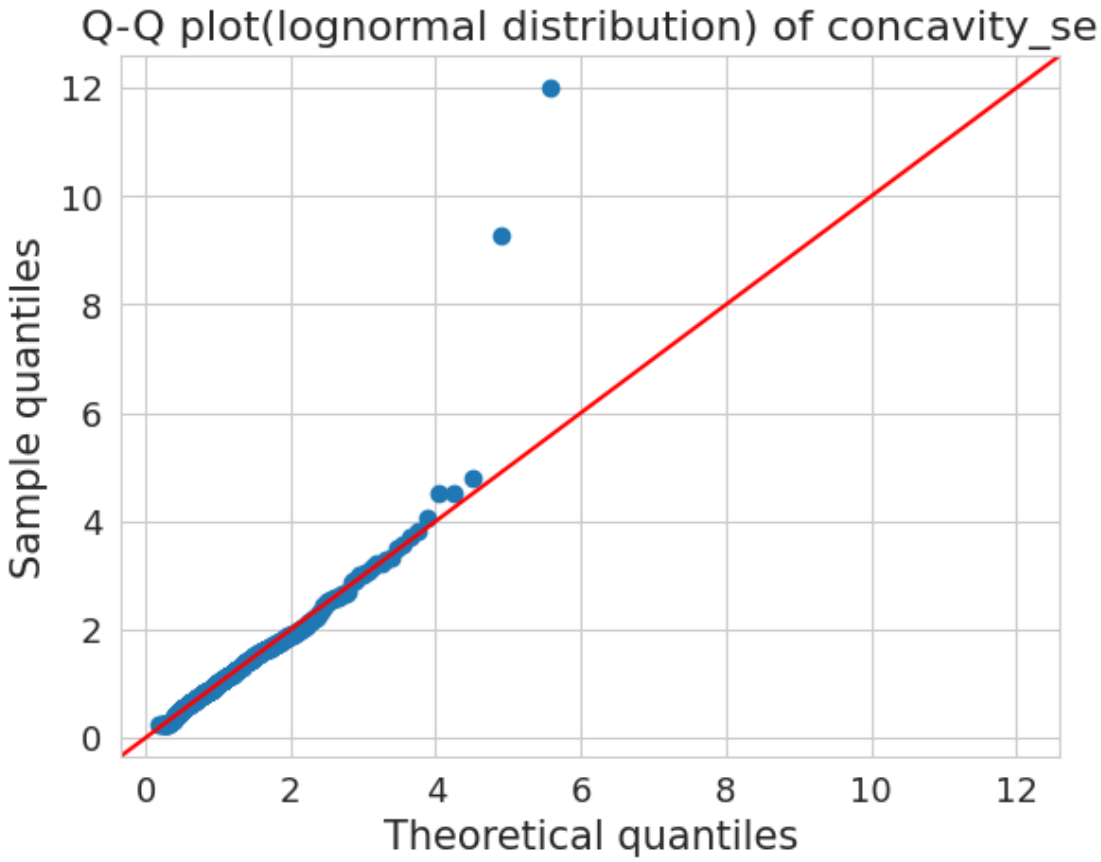

```
<Figure size 800x500 with 0 Axes>
```



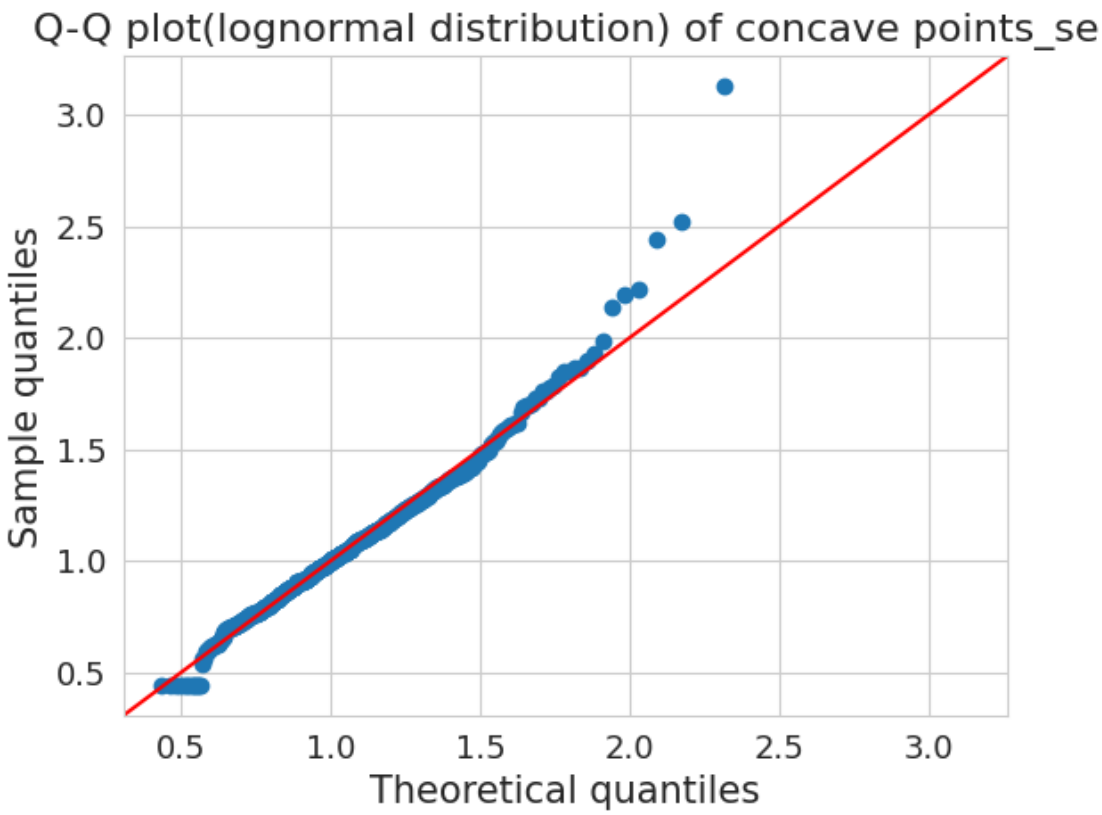

```
<Figure size 800x500 with 0 Axes>
```



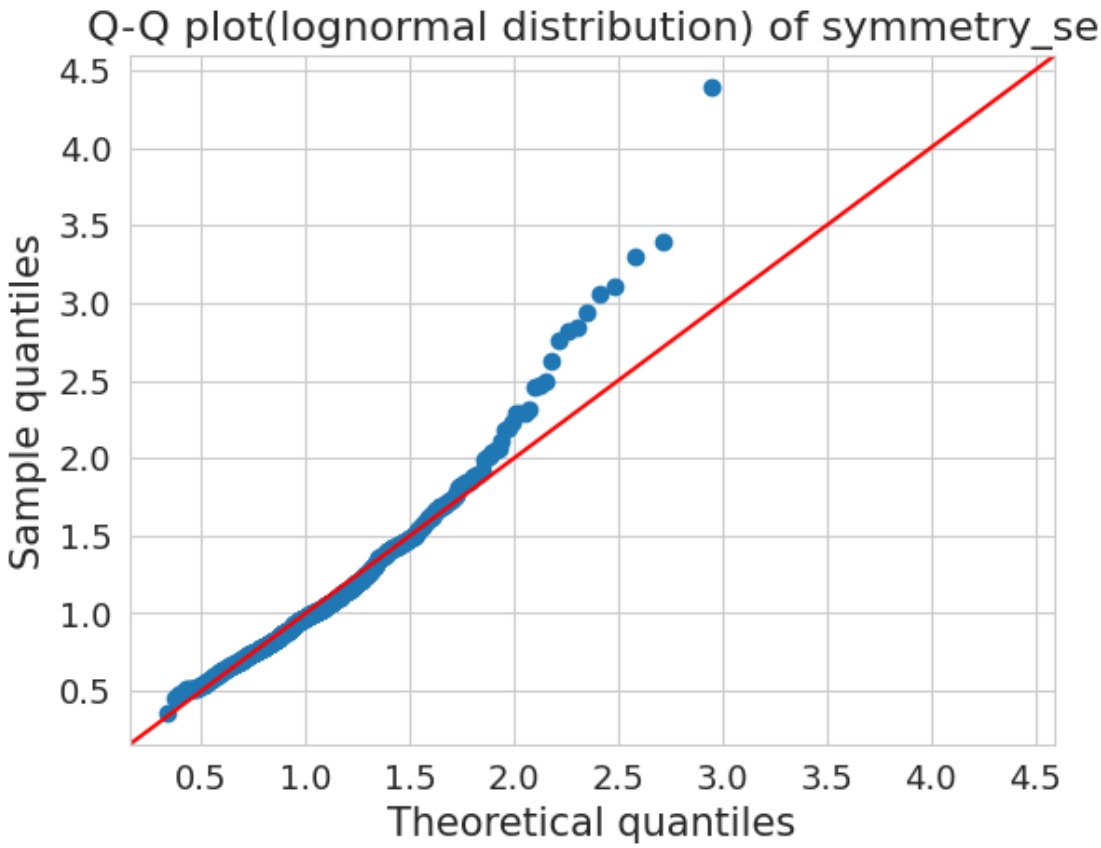

```
<Figure size 800x500 with 0 Axes>
```



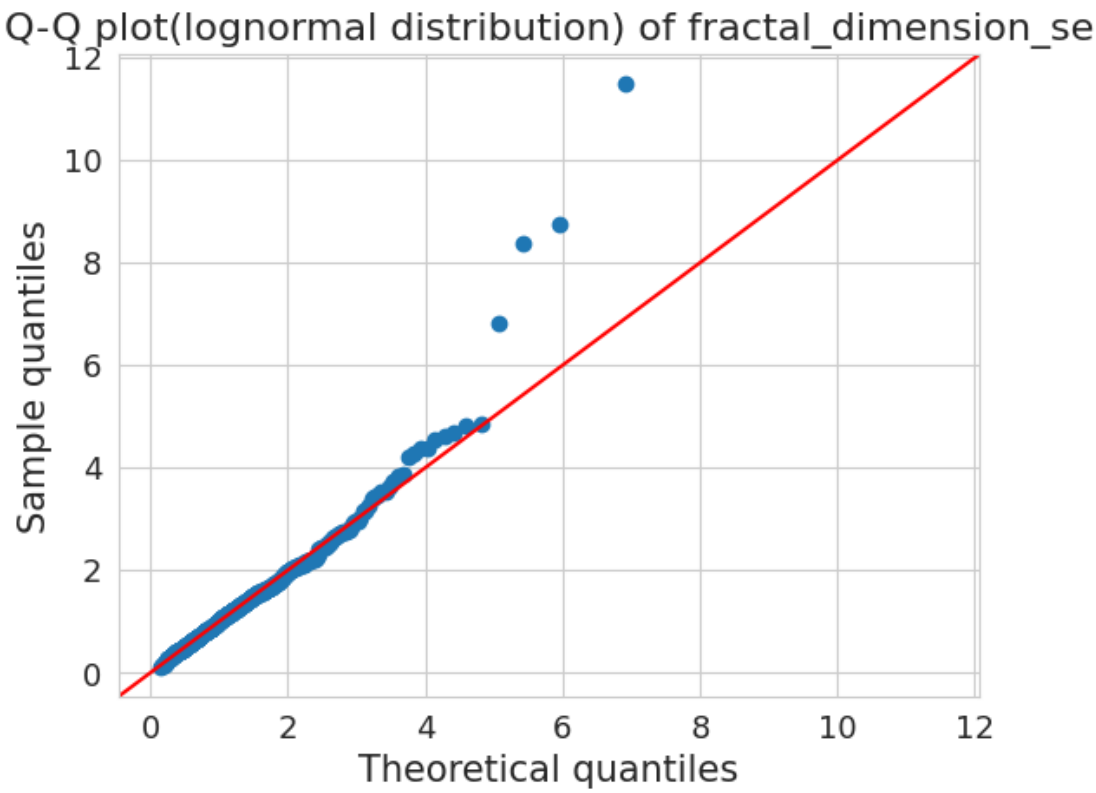

```
<Figure size 800x500 with 0 Axes>
```



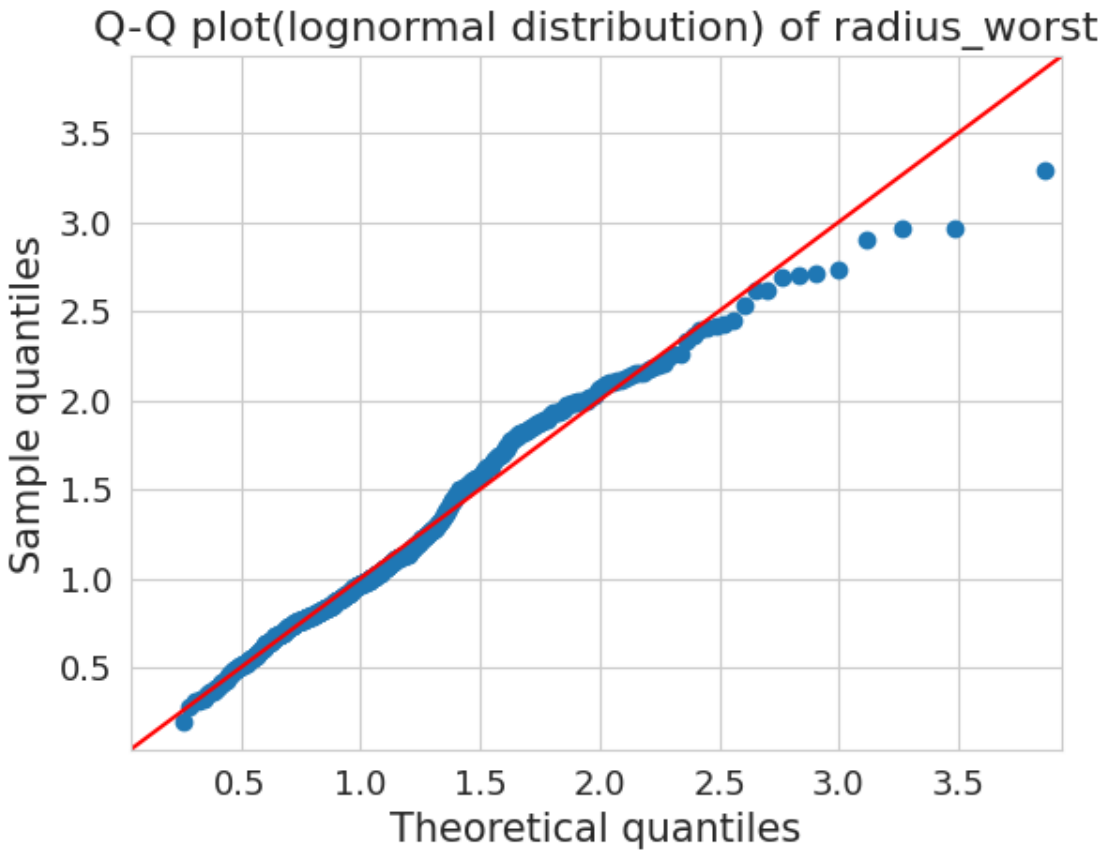

```
<Figure size 800x500 with 0 Axes>
```



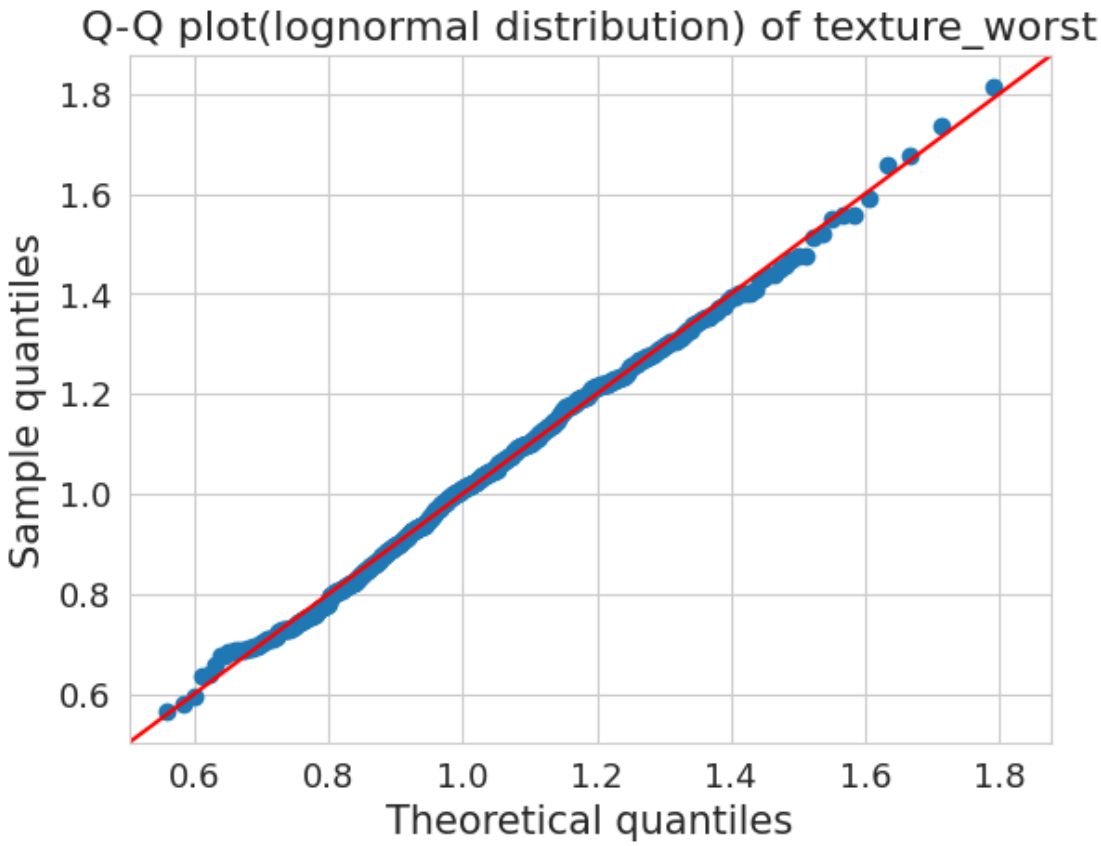

```
<Figure size 800x500 with 0 Axes>
```



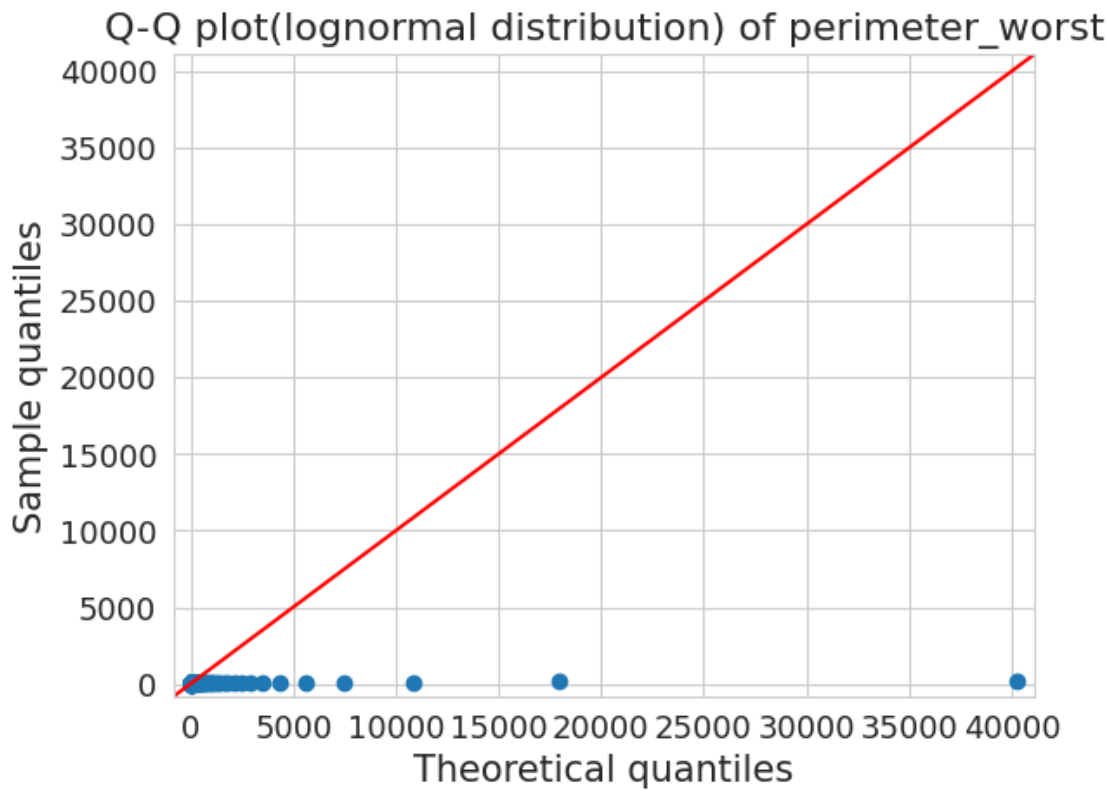

```
<Figure size 800x500 with 0 Axes>
```



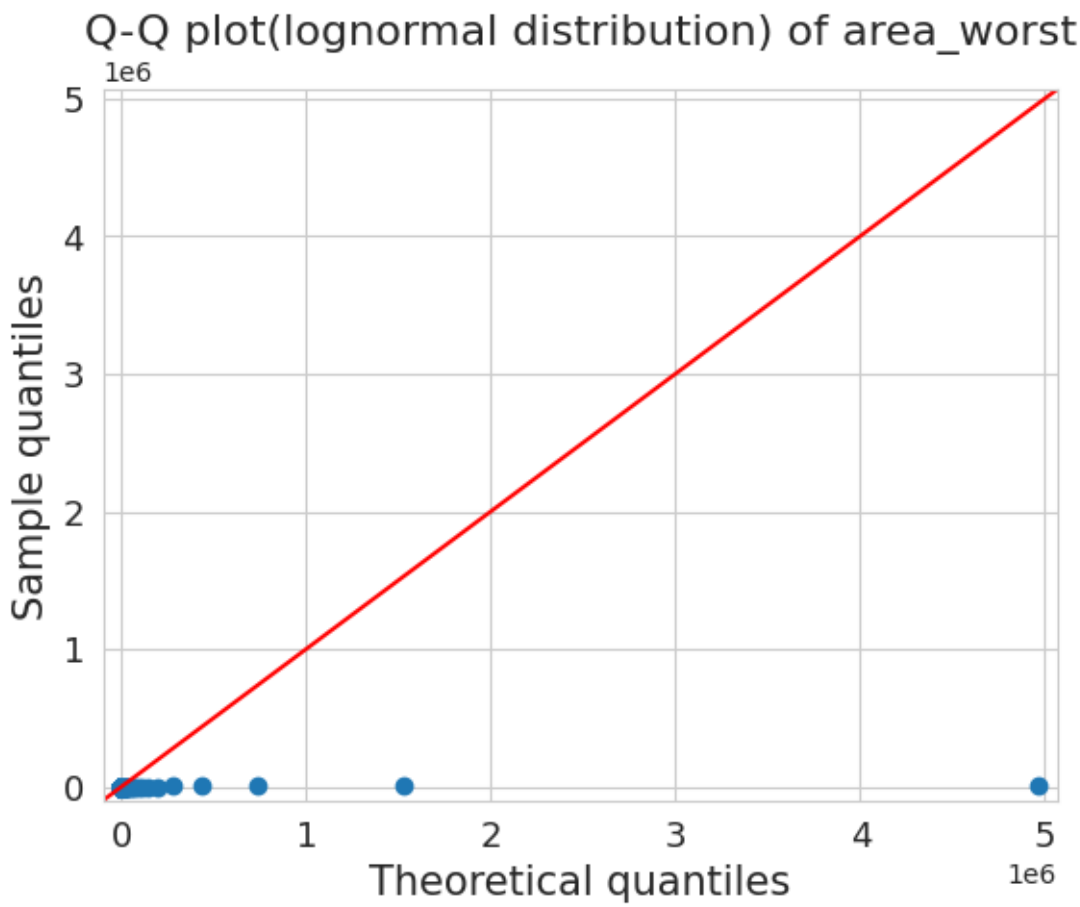

```
<Figure size 800x500 with 0 Axes>
```



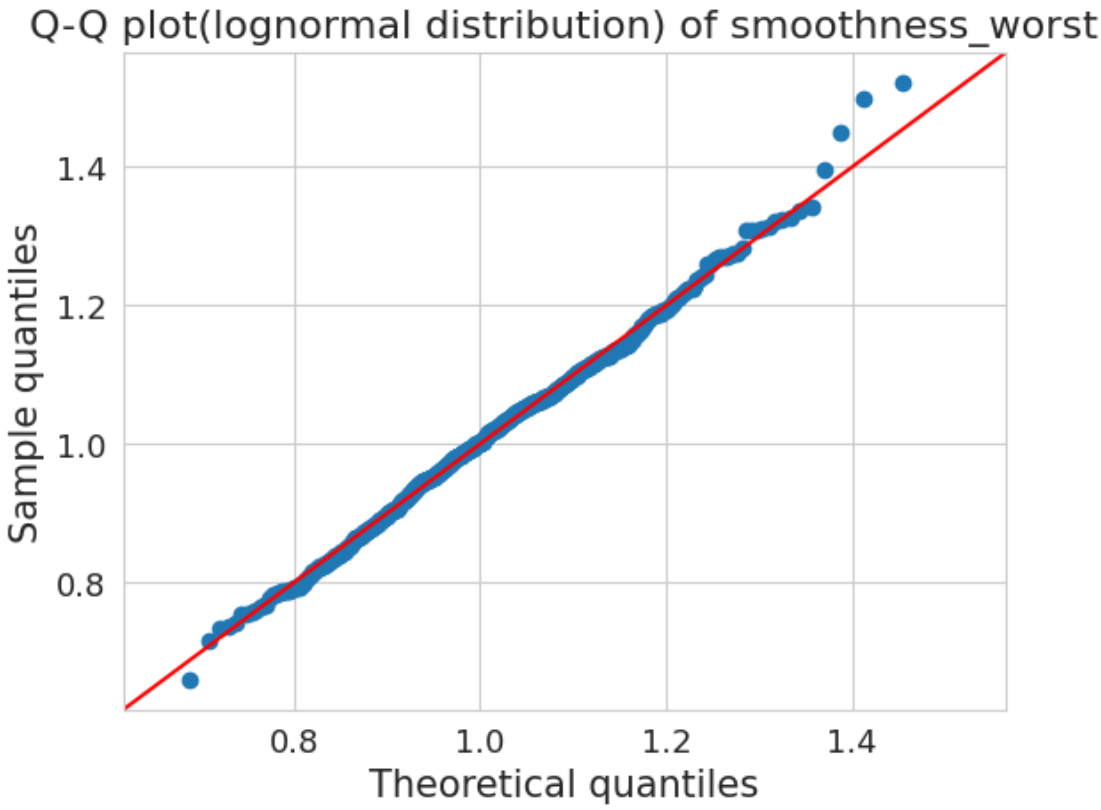

```
<Figure size 800x500 with 0 Axes>
```



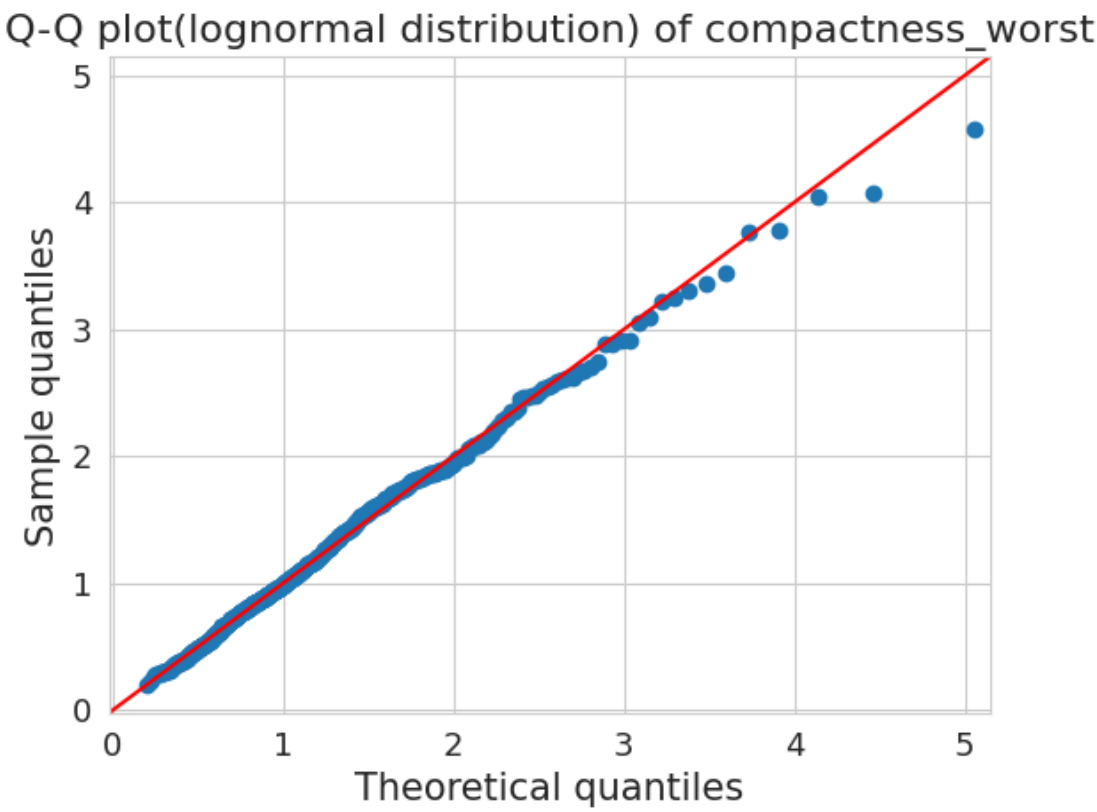

```
<Figure size 800x500 with 0 Axes>
```



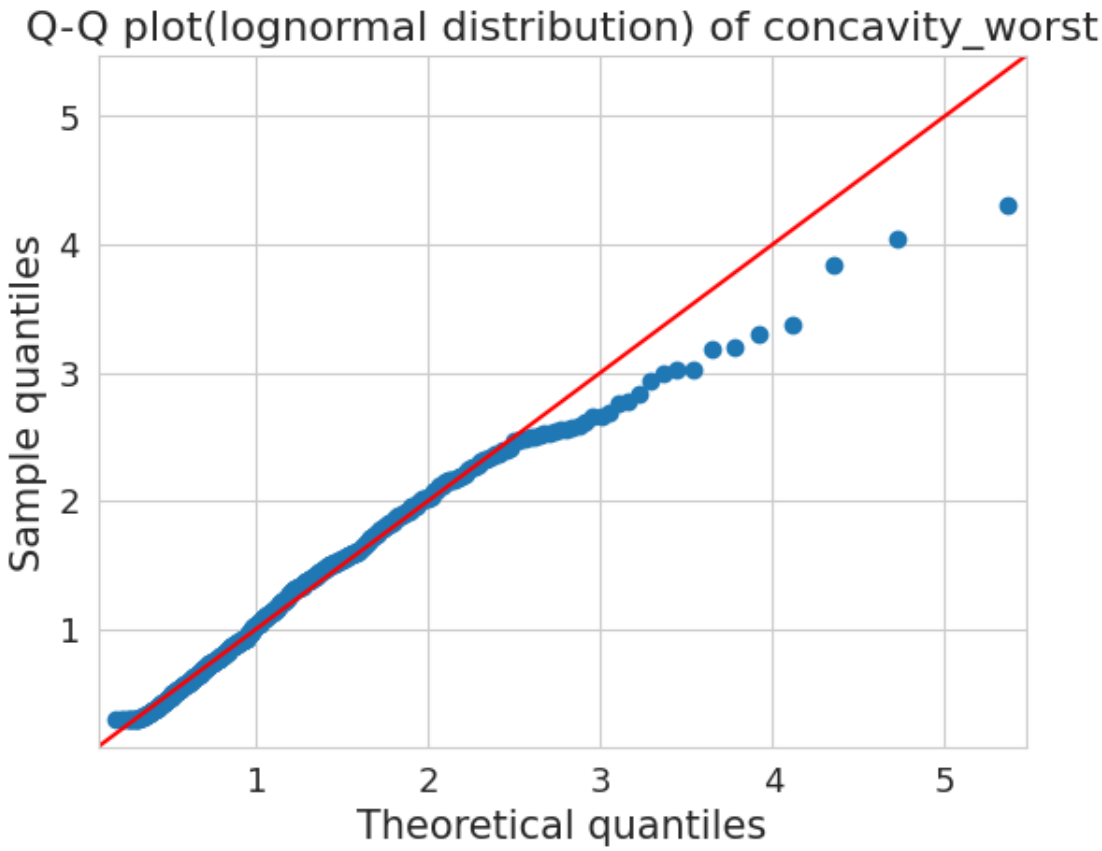

```
<Figure size 800x500 with 0 Axes>
```



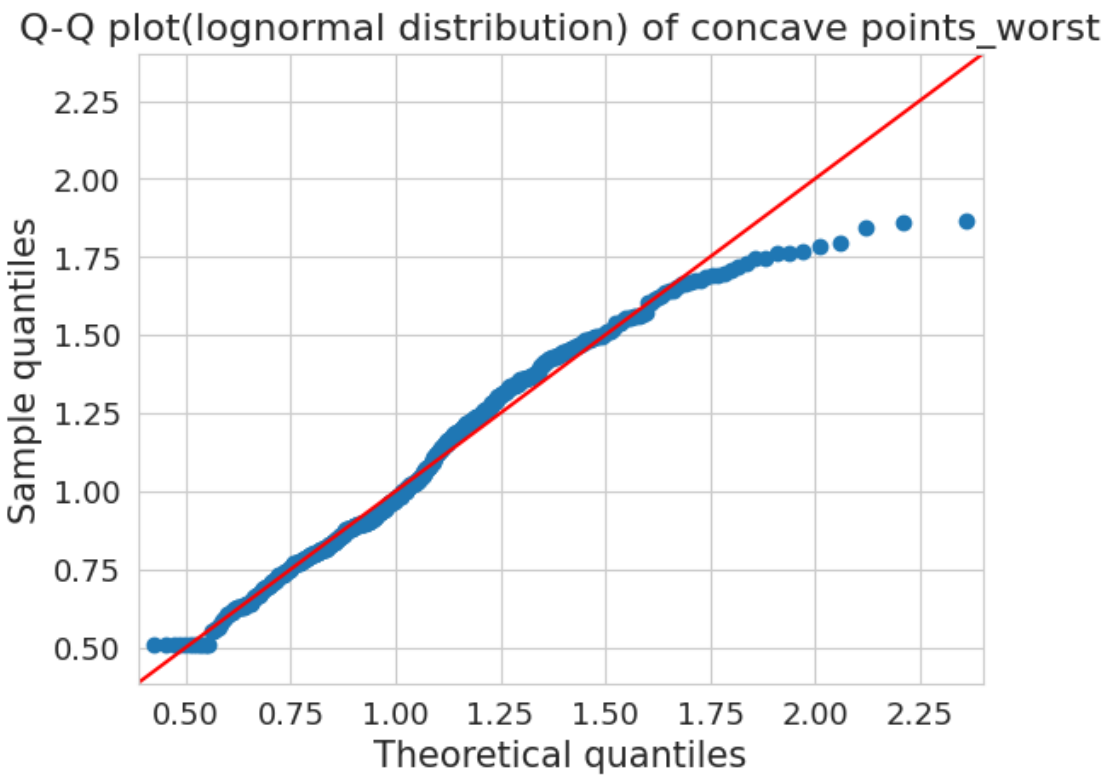

```
<Figure size 800x500 with 0 Axes>
```



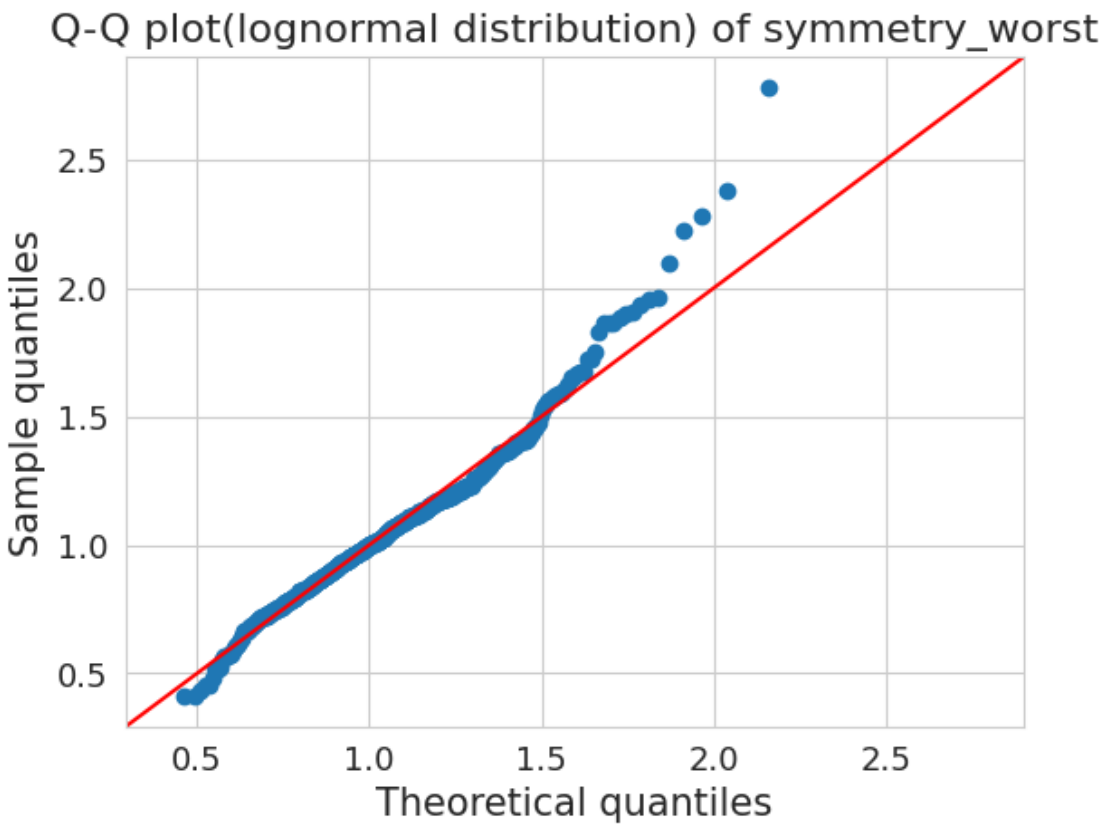

```
<Figure size 800x500 with 0 Axes>
```



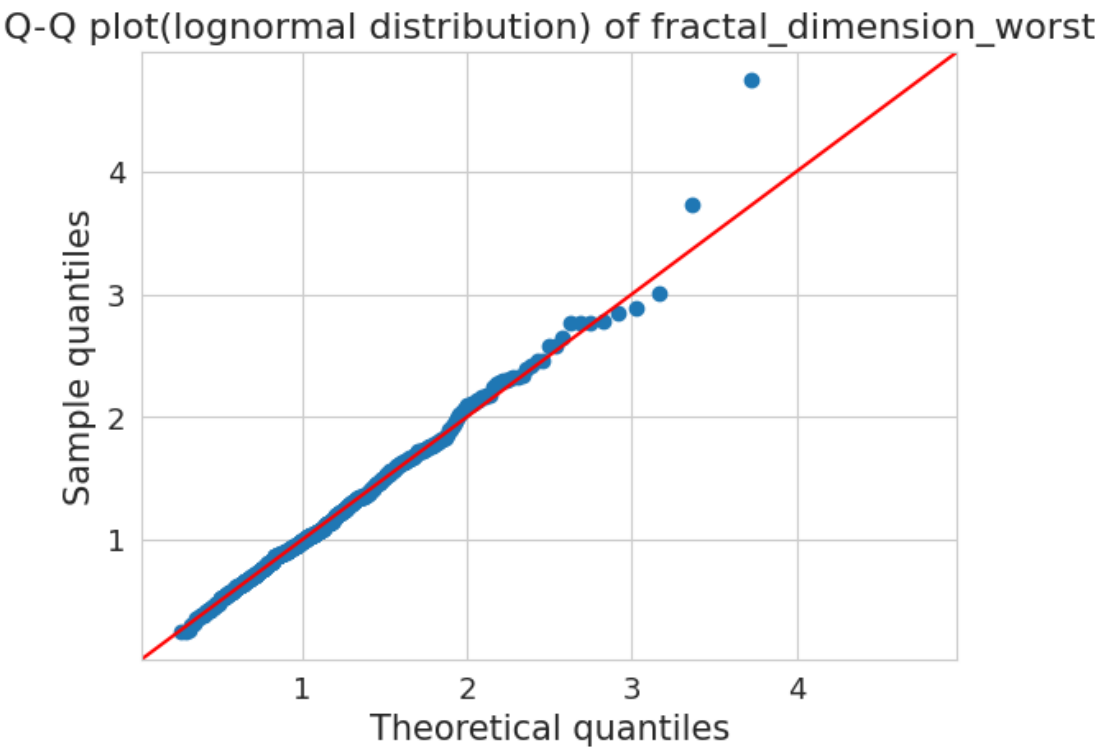

```
<Figure size 800x500 with 0 Axes>
```




```
<Figure size 800x500 with 0 Axes>
```

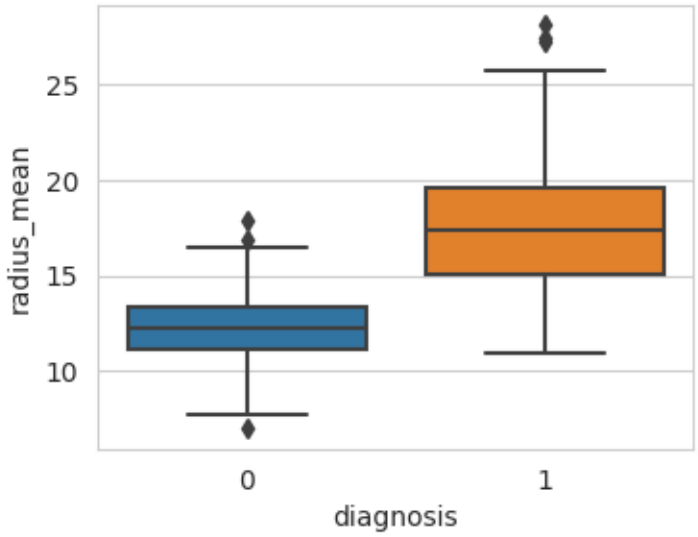


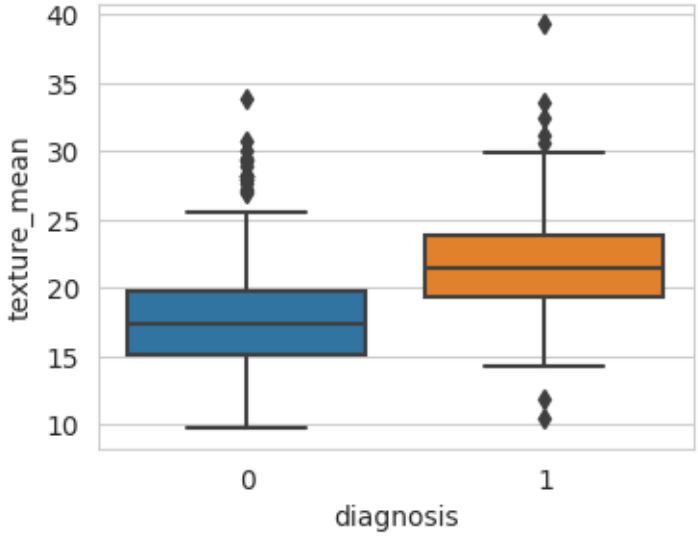

```
<Figure size 800x500 with 0 Axes>
```

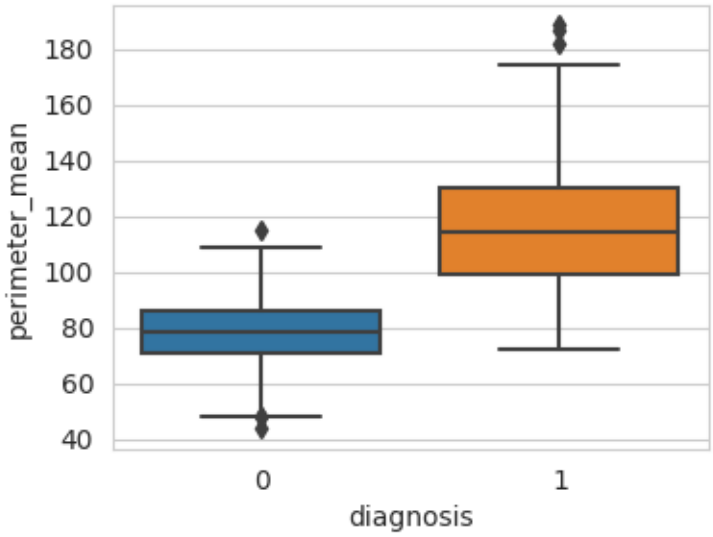


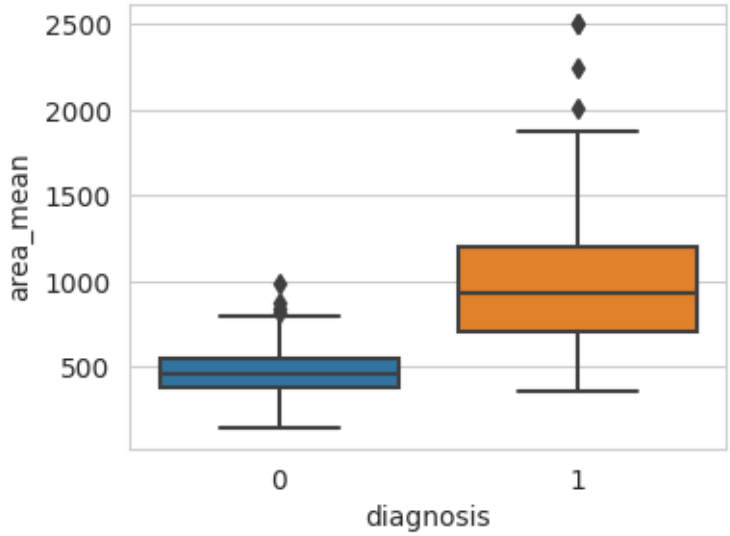
The qqplot is constructed for the independent features, on different distributions.
It is observed that majority of the features follow a *log normal distribution*

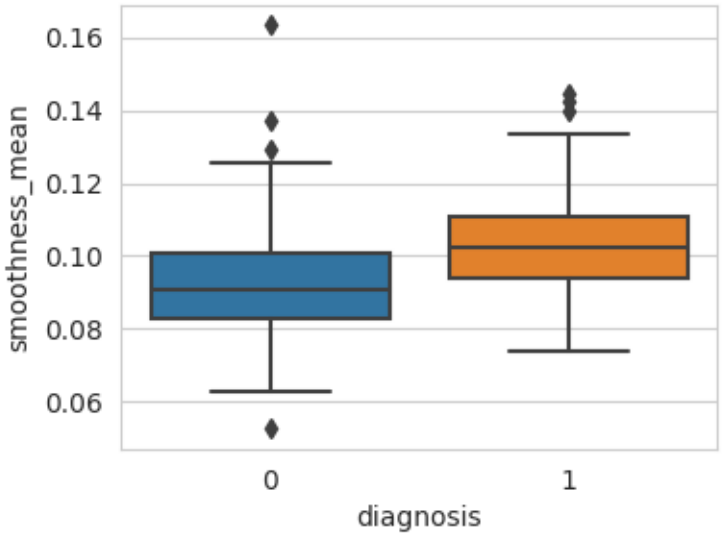
[229]:
```
# Boxplot to show the distribution of dependent variable for different␣
↪independent variables
for a in df.columns:
  plt.figure(figsize=(4,3))
  fig = sns.boxplot(x=df_target, y=df[a])
```

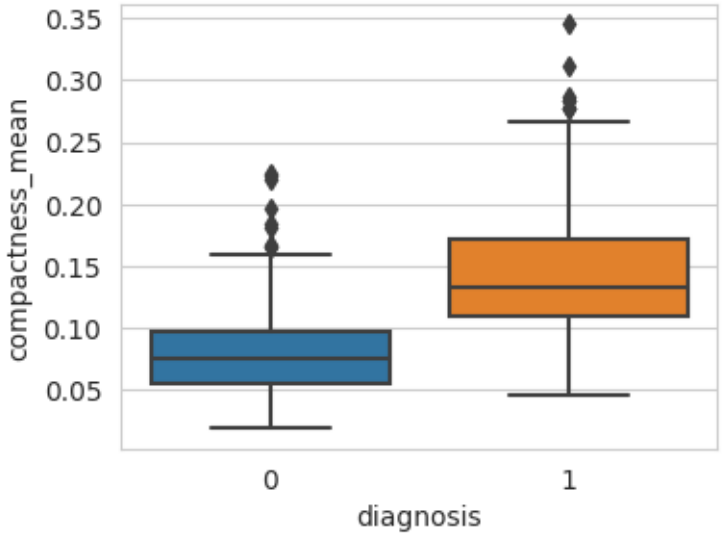

25
20
15
10
radius_mean
0
1
diagnosis

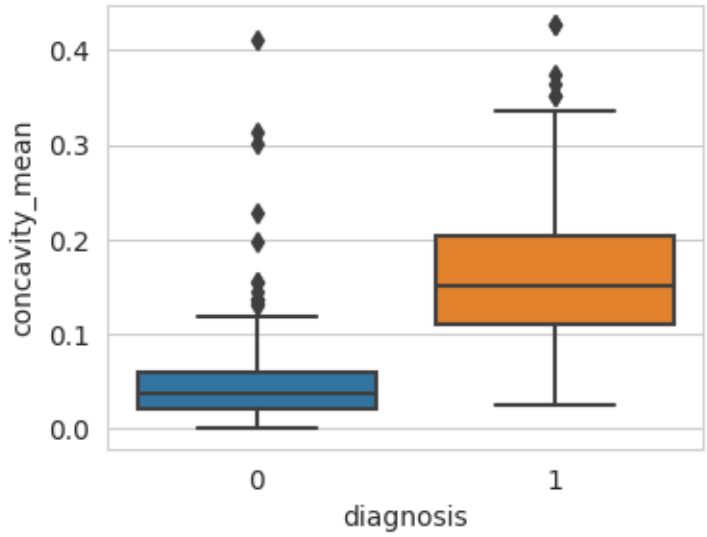

40
35
30
25
20
15
10
texture_mean
0
1
diagnosis

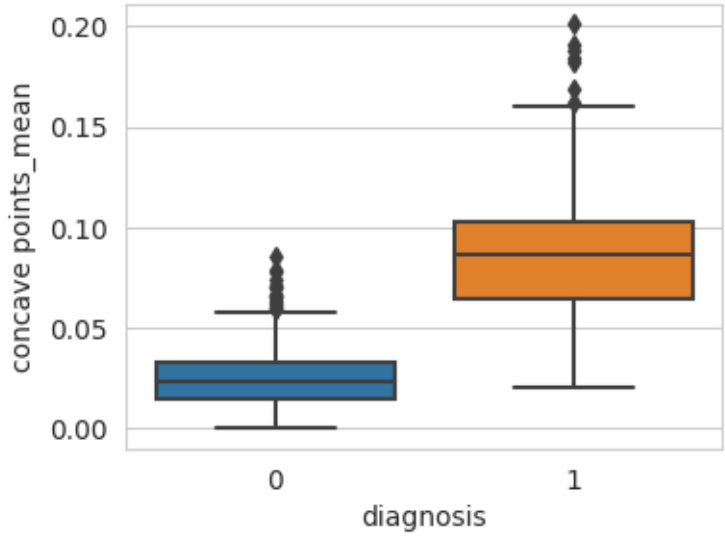

perimeter_mean
180
160
140
120
100
80
60
40
0
1
diagnosis

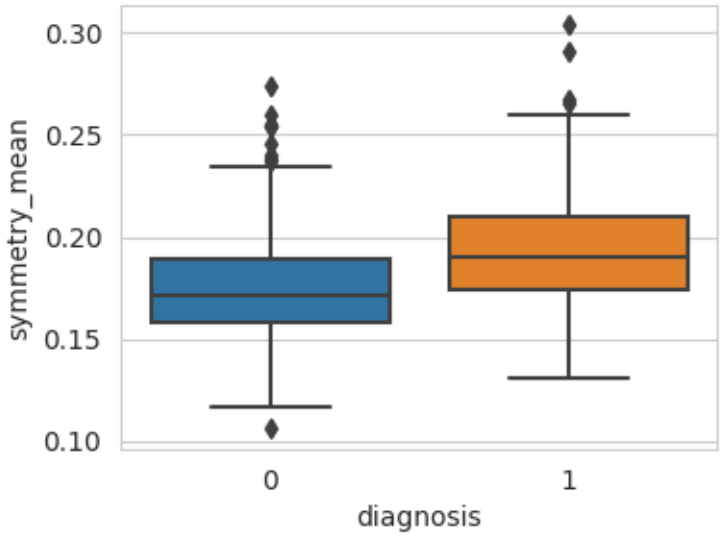

area_mean
2500
2000
1500
1000
500
0
1
diagnosis

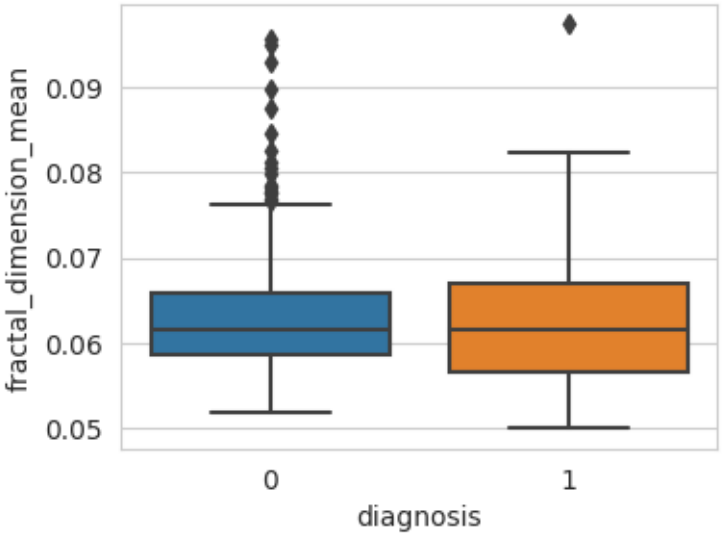

0.16
0.14
0.12
0.10
0.08
0.06
smoothness_mean
0
1
diagnosis

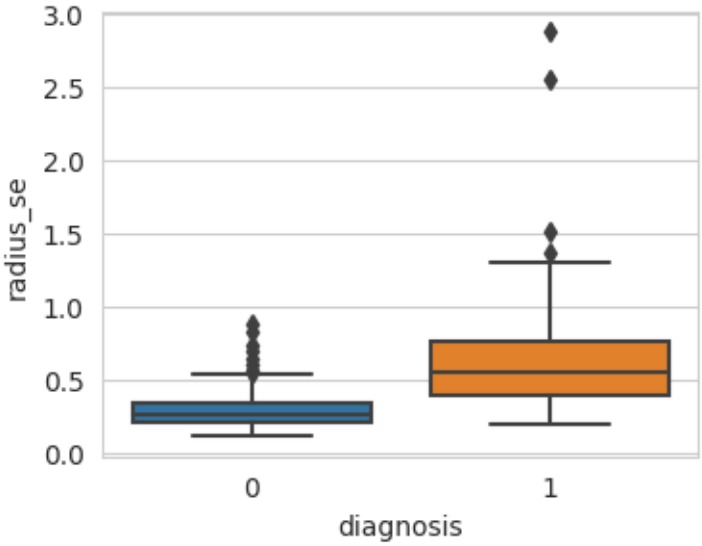

0.35
0.30
0.25
0.20
0.15
0.10
0.05
compactness_mean
0
1
diagnosis

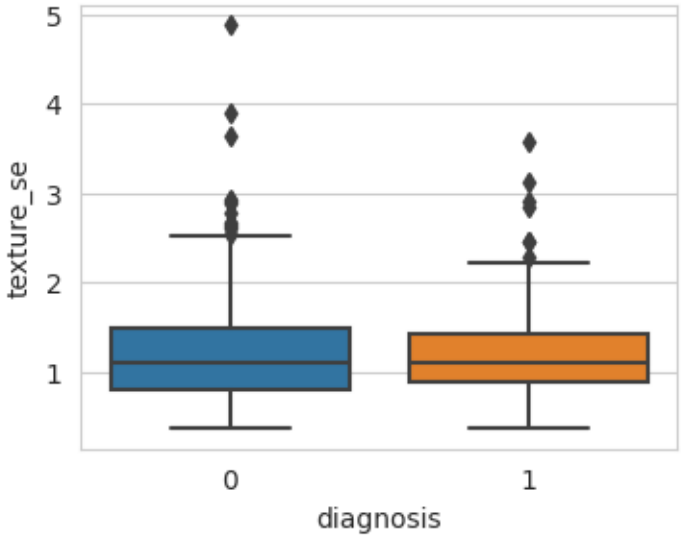

0.4
0.3
0.2
0.1
0.0
concavity_mean
0
1
diagnosis

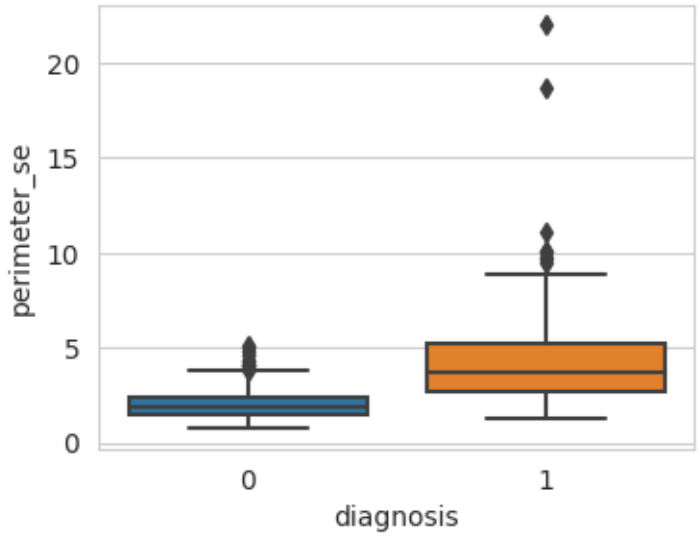

0.20
0.15
0.10
0.05
0.00
concave points_mean
0
1
diagnosis

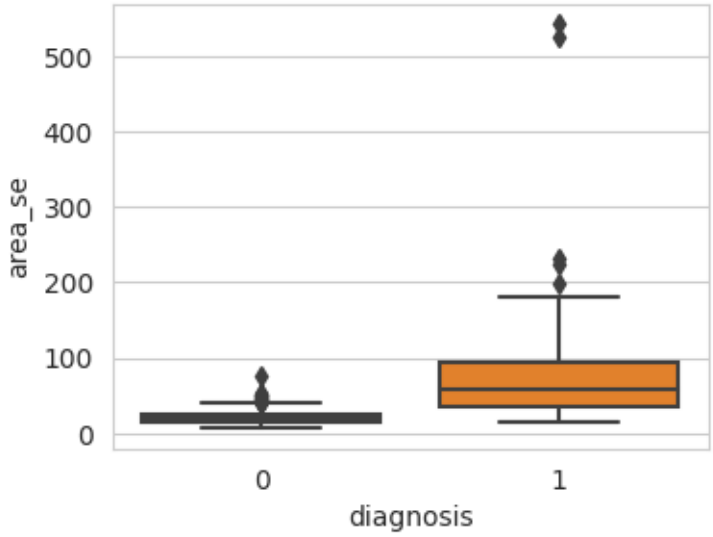

0.30
0.25
0.20
0.15
0.10
symmetry_mean
0
1
diagnosis

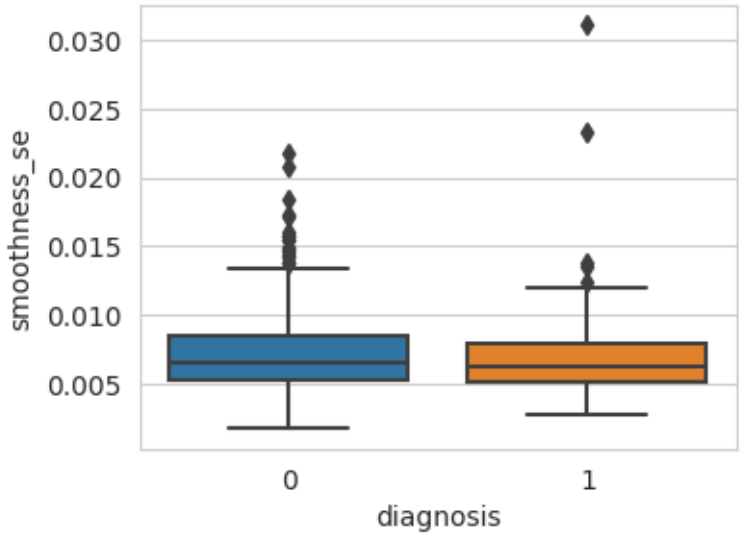

0.09
0.08
0.07
0.06
0.05
fractal_dimension_mean
0
1
diagnosis

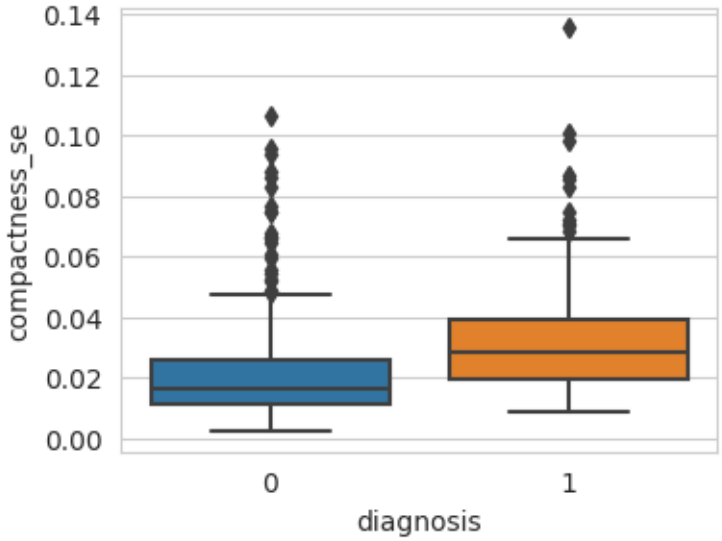

3.0
2.5
2.0
1.5
1.0
0.5
0.0
radius_se
0
1
diagnosis

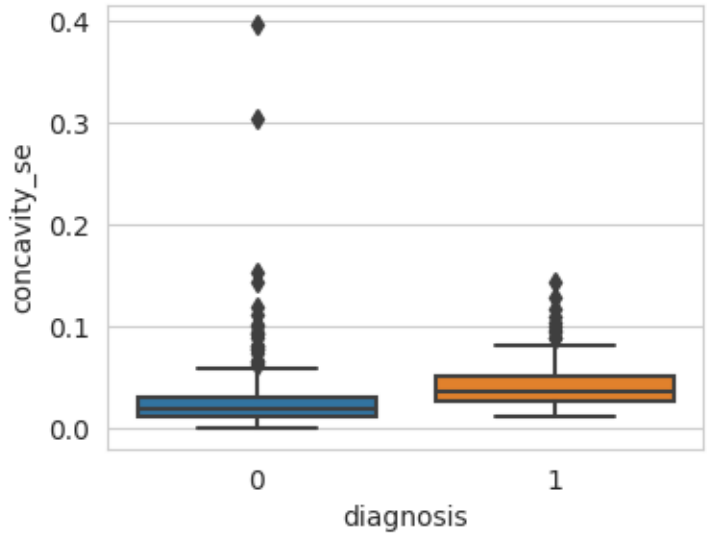

5
4
3
2
1
texture_se
0
1
diagnosis

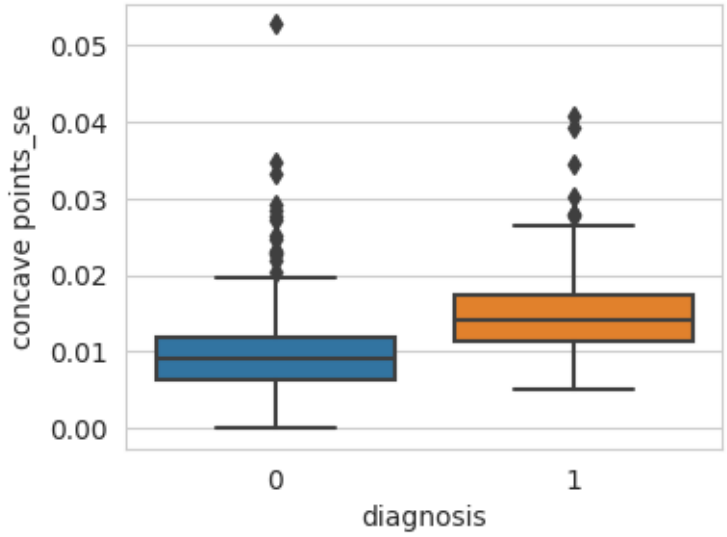

perimeter_se
20
15
10
5
0
0
1
diagnosis

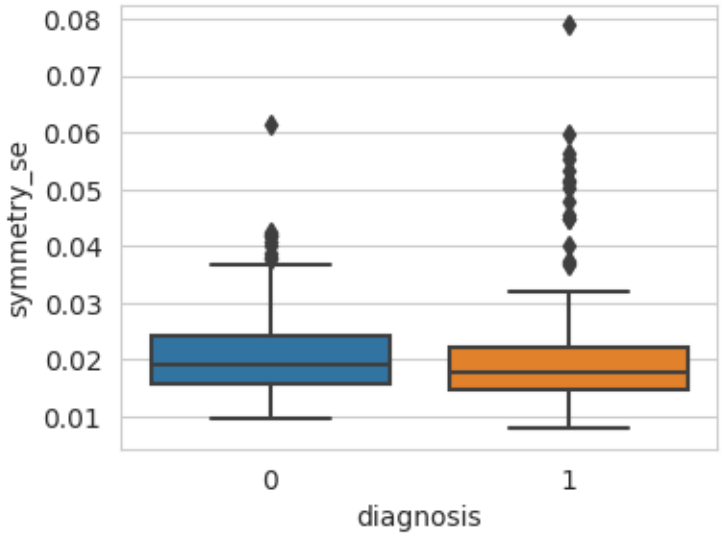

area_se
500
400
300
200
100
0
0
1
diagnosis

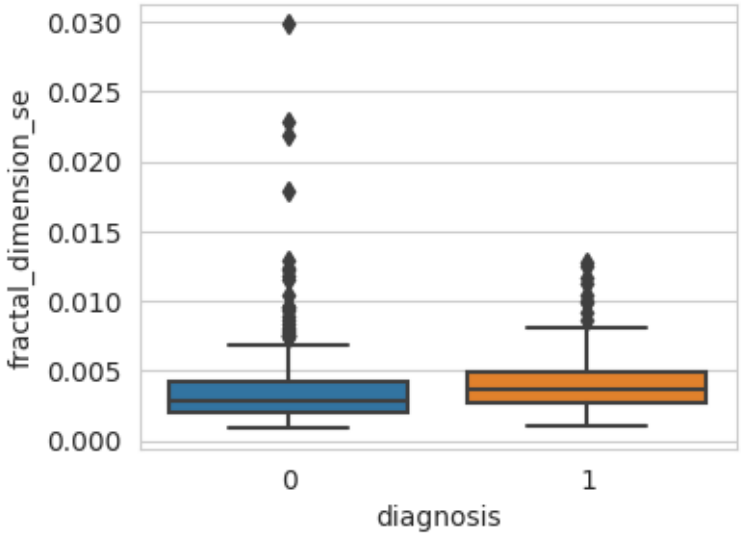

0.030
0.025
0.020
0.015
0.010
0.005
smoothness_se
0
1
diagnosis

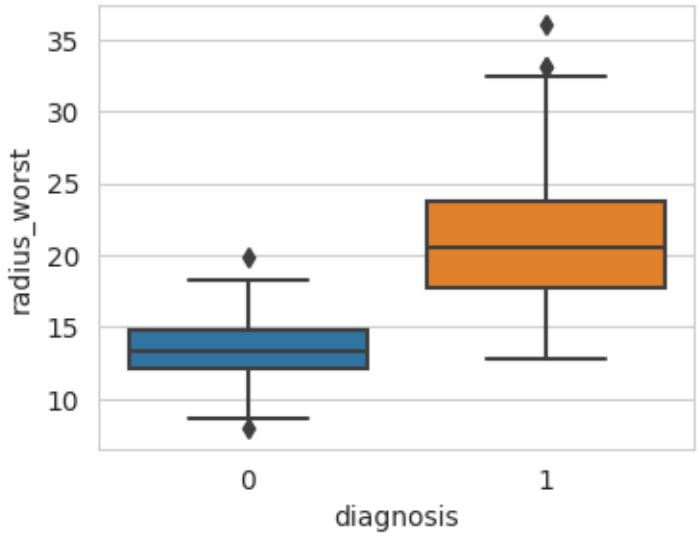

0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
compactness_se
0
1
diagnosis

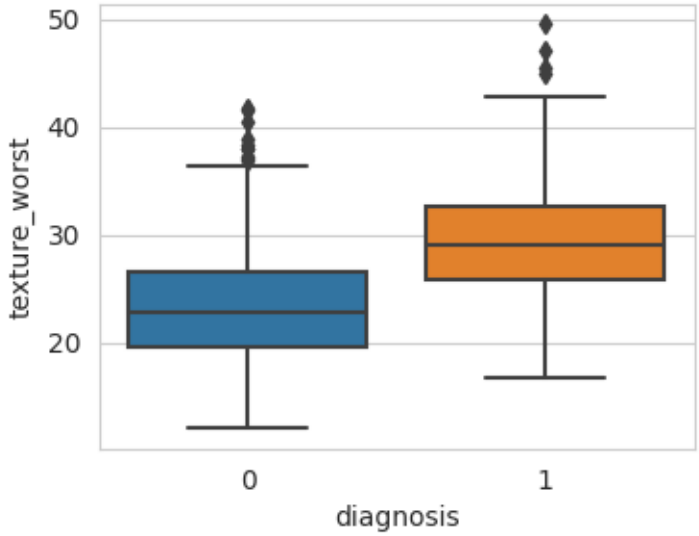

0.4
0.3
0.2
0.1
0.0
concavity_se
0
1
diagnosis

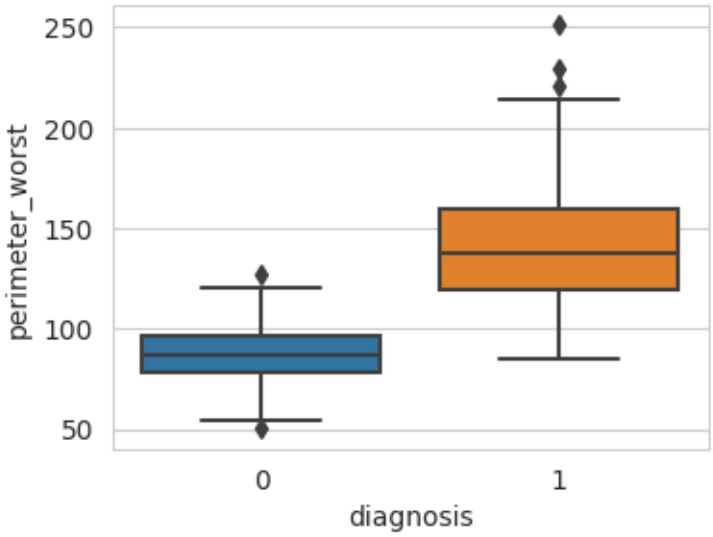

0.05
0.04
0.03
0.02
0.01
0.00
concave points_se
0
1
diagnosis

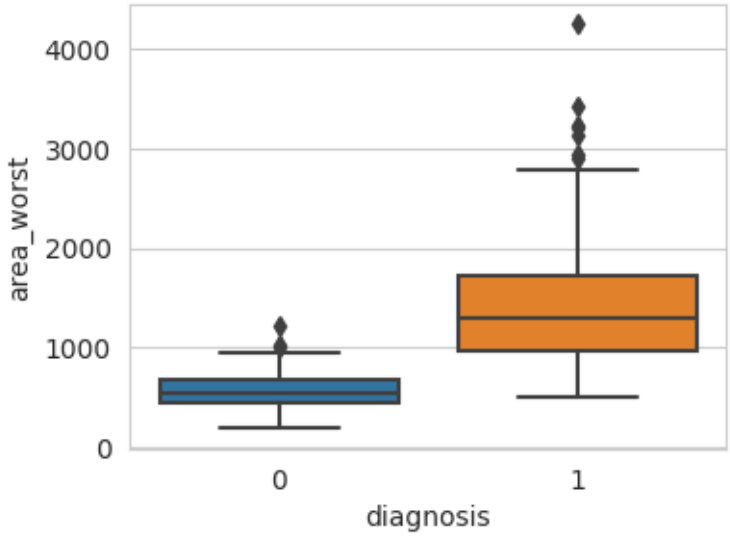

0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
symmetry_se
0
1
diagnosis

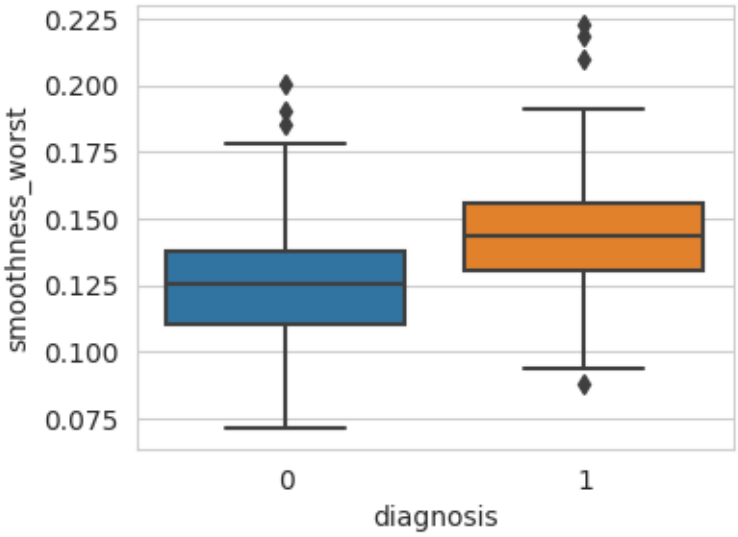

0.030
0.025
0.020
0.015
0.010
0.005
0.000
fractal_dimension_se
0
1
diagnosis

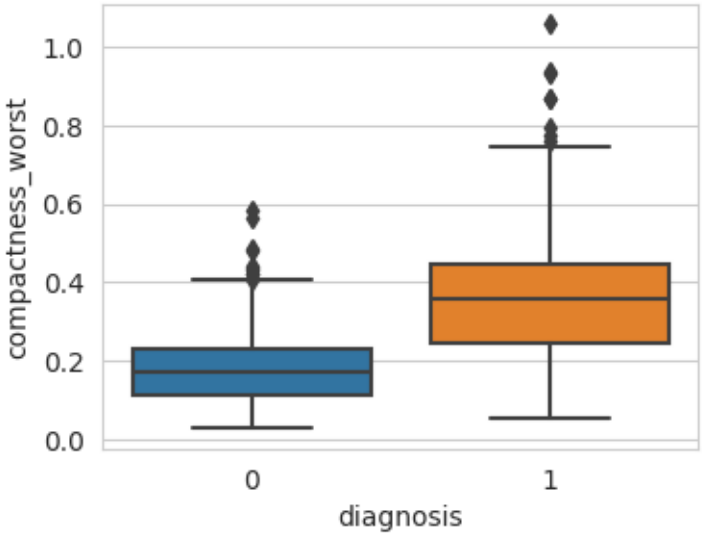

35
30
25
20
15
10
radius_worst
0
1
diagnosis

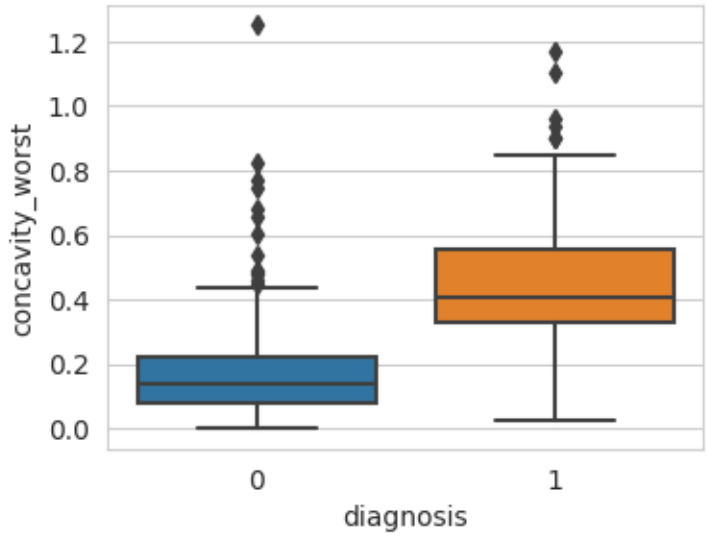

50
40
30
20
texture_worst
0
1
diagnosis

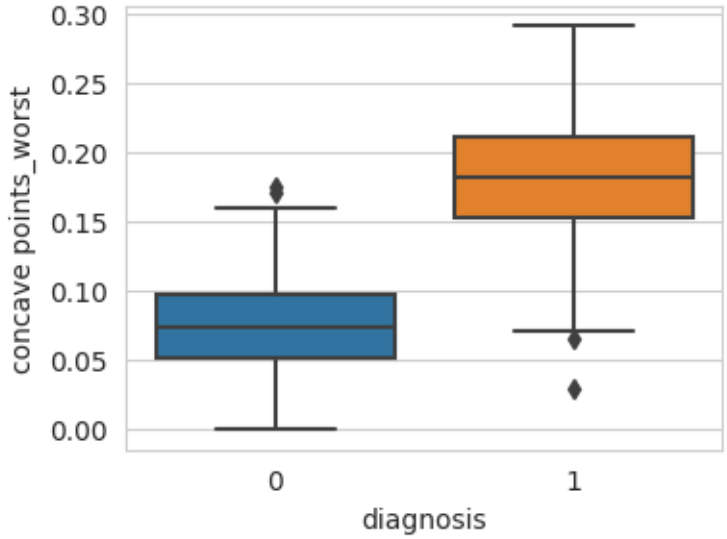

250
200
150
100
50
perimeter_worst
0
1
diagnosis

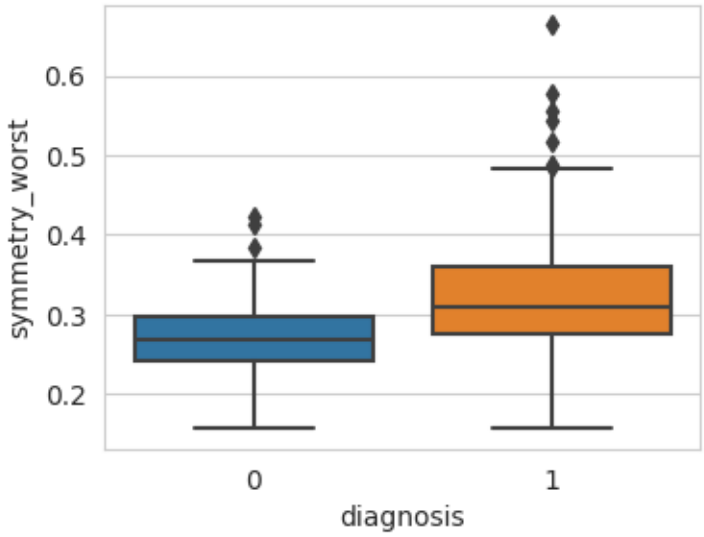

4000
3000
2000
1000
0
area_worst
0
1
diagnosis

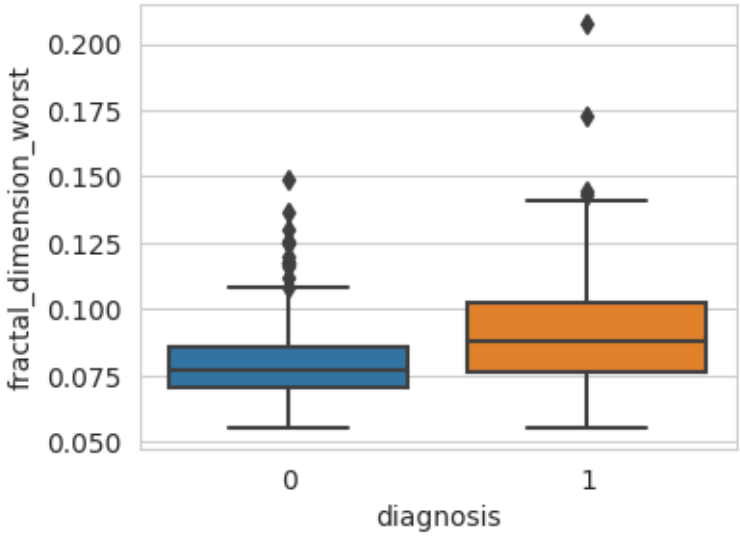

0.225
0.200
0.175
0.150
0.125
0.100
0.075
smoothness_worst
0
1
diagnosis


1.0
0.8
0.6
0.4
0.2
0.0
compactness_worst
0
1
diagnosis


concavity_worst
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
1
diagnosis


concave points_worst
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
1
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.

3.3 Anomaly Detection

[245]:
```
def cf_display(test,pred):
  cf =␣
 ↪ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(test,pred),display_labels=['Benign
  cf.plot()
  plt.show()

def metric_scores(test,pred):
  accuracy = accuracy_score(test,pred)
  precision=precision_score(test,pred)
  recall=recall_score(test,pred)
  print("Accuracy:", accuracy)
  print("Precision: ",precision)
  print("Recall ",recall)
def scaled_boxplot(df):
  scaler=MinMaxScaler()
  df_scaled=pd.DataFrame(scaler.fit_transform(df),columns=df.columns)
  plt.rcParams.update({'figure.max_open_warning': 0})
  plt.figure(figsize=(10,100))
  cols=list(df_scaled.columns)
  for a in range(1,30):
    plt.subplot(30,1,a)
    sns.boxplot(df_scaled[cols[a]])
    plt.title("Distribution of "+cols[a])
def sample_feature_importance(shap_vals,idx):
        return shap.plots.waterfall(shap_vals[idx],max_display=10)
def find_best_model_from_grid(grid, parameter):
    model_list = []
    for item in grid:
        if parameter == "r2":
            if not (item.r2() == "NaN"):
                model_list.append(item.r2())
            else:
                model_list.append(0.0)
        elif parameter == "auc":
            if not (item.auc() == "NaN"):
                model_list.append(item.auc())
            else:
                model_list.append(0.0)
    max_index = model_list.index(max(model_list))
    best_model = grid[max_index]
    print("Model ID with best R2: " + best_model.model_id)
    if parameter == "r2":
        print("Best R2: " + str(best_model.r2()))
    elif parameter == "auc":
        print("Best AUC: " + str(best_model.auc()))
```

```
    return best_model
def pdp(feature,model,train,index,shap_values):
  shap.partial_dependence_plot(
            feature,
            model.predict,
            train,
            ice=False,
            model_expected_value=True,
            feature_expected_value=True,
            shap_values=shap_values[index:index+1,:])
```

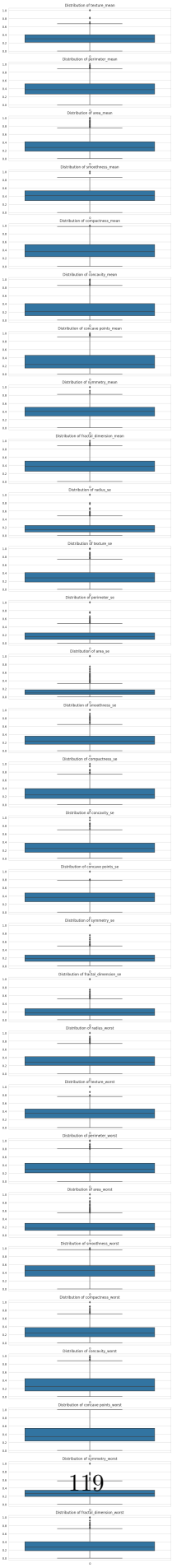
3.4 IsolationForest

An method called "Isolation Forest" is employed to find outliers. The method builds a tree-based structure and measures an observations's level of abnormality by counting how many splits are necessary to isolate it.

[231]:
```
from sklearn.ensemble import IsolationForest
rs=np.random.RandomState(100)
clf = IsolationForest(max_samples=100,random_state=rs, contamination=.15)
clf.fit(df)
anomalies=clf.predict(df)
mask=anomalies!=-1
df_array=df.to_numpy()[mask,:]
target_array=df_target.to_numpy()[mask]
df=pd.DataFrame(df_array,columns=df.columns)
df_target=pd.DataFrame(target_array,columns=['diagnosis'])
```

```
X does not have valid feature names, but IsolationForest was fitted with feature
names
```

[232]:
```
#feature distributions after performing outlier detection
scaled_boxplot(df)
```


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

3.5 Collinearity

3.5.1 Variance Inflation Factor

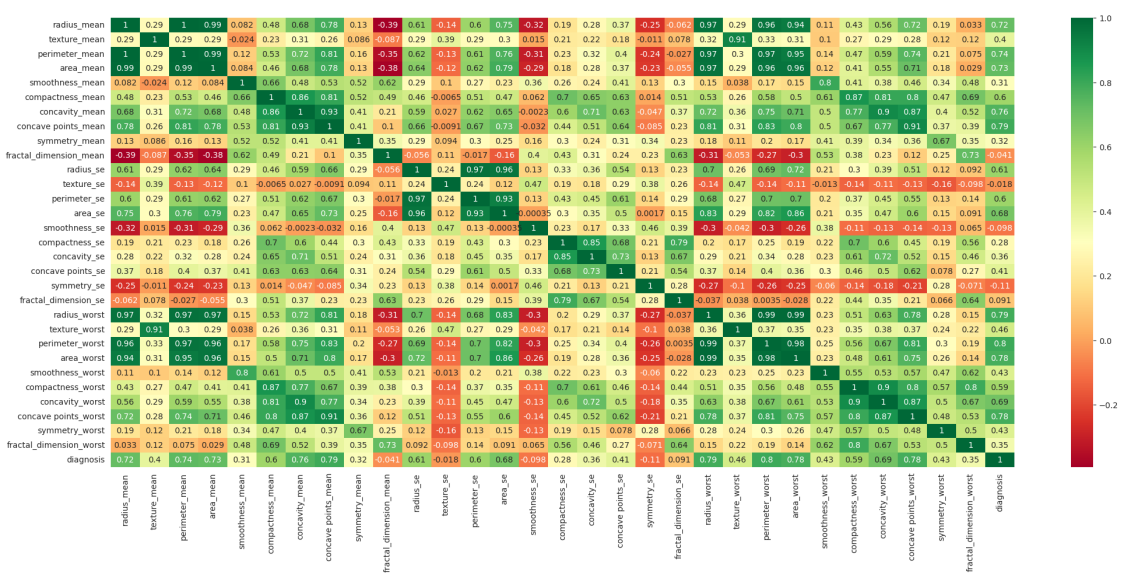
[233]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_data = pd.DataFrame()
dataset=pd.concat([df,df_target],axis=1)
vif_data["feature"] = dataset.columns
vif_data["VIF"] = [variance_inflation_factor(dataset.values, i)
                          for i in range(len(dataset.columns))]
mc_features=[]
for a in zip(vif_data['feature'],vif_data['VIF']):
  if a[1]>10:
    mc_features.append(a[0])
```

The VIF values show a great number of features are exhibiting multicollinearity.
The features with high VIF values may be ignored while performing regression analysis, such as in logistic regression.

##Correlation The relationship between features and the target is crucial because it reveals which features are most useful for forecasting the target variable. This is because a feature with strong correlation to the target variable would hold crucial information that can be used to precisely predict the target variable.

[234]:
```
corr_mat=dataset.corr()
plt.figure(figsize=(25,10))
sns.heatmap(corr_mat, annot=True, cmap='RdYlGn')
```

[234]: <Axes: >



A majority of features exhibit considerable correlation with the target.
The situation of high multicollinearity and considerable correlation demands further investigation to arrive at variable significances.

3.6 Feature Significance

Feature significance is arrived at by using feature selection methods. The interpretation of regression coefficients to arrive at feature importance is avoided due to high collinearity of features.

[235]:
```
from sklearn.feature_selection import SelectKBest,chi2
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
scaler=MinMaxScaler()
df_scaled=scaler.fit_transform(df)
selector=SelectKBest(score_func=chi2,k=25)
selected=selector.fit_transform(df_scaled,df_target)
support=selector.get_support()
selected_feat=list(df.loc[:,support].columns)
print(selected_feat)
```

```
['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
'smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean',
'symmetry_mean', 'radius_se', 'perimeter_se', 'area_se', 'compactness_se',
'concavity_se', 'concave points_se', 'radius_worst', 'texture_worst',
'perimeter_worst', 'area_worst', 'smoothness_worst', 'compactness_worst',
'concavity_worst', 'concave points_worst', 'symmetry_worst',
'fractal_dimension_worst']
```

The most important features, computed by using the KBest feature selection method for a random forest model can be observed above.

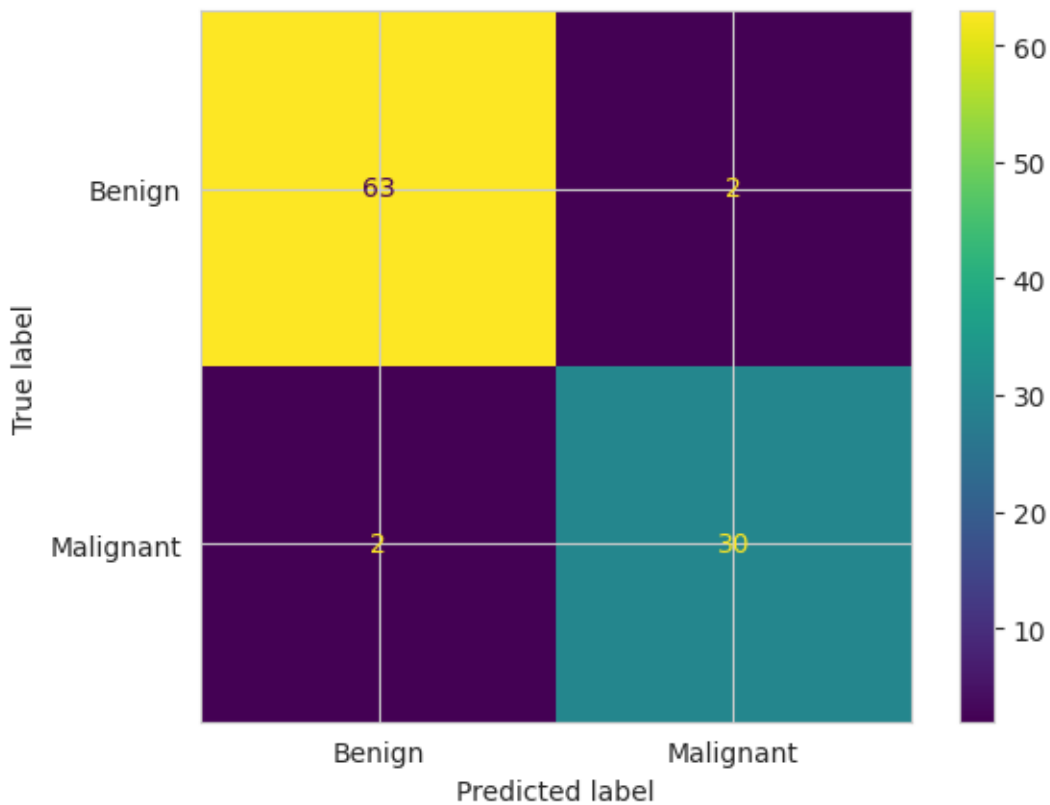
KBest feature selection is a machine learning technique that is used to pick the most crucial features from a dataset. Using a specified statistical metric, such as mutual information, the ANOVA F-test, or the chi-squared test, the top K features with the highest scores are chosen. The user chooses the K value, which can either be assigned to a specific integer or selected through cross-validation. By deleting unnecessary or redundant features, this method can be used to decrease the dimensionality of a dataset and enhance model performance.

[236]:
```
df_best=df[selected_feat]
X_train,X_test,y_train,y_test=train_test_split(df_best,df_target,test_size=0.
 ↪2,random_state=100)
train_df_best=pd.concat([X_train,y_train],axis=1)
test_df_best=pd.concat([X_test,y_test],axis=1)

X_train.reset_index(inplace=True,drop=True)
y_train.reset_index(inplace=True,drop=True)
```

```
X_test.reset_index(inplace=True,drop=True)
y_test.reset_index(inplace=True,drop=True)
```

[237]:
```
# testing the performance of the KBest features fit to the Random Forest model
rf_fit=rf.fit(X_train,y_train)
y_pred=rf_fit.predict(X_test)
pred_df_best=pd.DataFrame(y_pred,columns=['diagnosis'])
metric_scores(y_test,y_pred)
cf_display(y_test,y_pred)
```

```
A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().

Accuracy: 0.9587628865979382
Precision:  0.9375
Recall  0.9375
```



4 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.

[238]:
```
#initializing the h2o cluster
h2o.init()
```

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

-------------------------- ␣
↪-------------------------------------------------------------------------------------------
H2O_cluster_uptime:         2 hours 19 mins
H2O_cluster_timezone:       Etc/UTC
H2O_data_parsing_timezone:  UTC
H2O_cluster_version:        3.40.0.3
H2O_cluster_version_age:    5 days
H2O_cluster_name:           H2O_from_python_unknownUser_paagfe
H2O_cluster_total_nodes:    1
H2O_cluster_free_memory:    2.952 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,␣
↪"colab_language_server": "/usr/colab/bin/language_service"}
H2O_internal_security:      False
Python_version:             3.9.16 final
-------------------------- ␣
↪-------------------------------------------------------------------------------------------
```

[239]:
```
# Train-test split of the h2o frame
df=h2o.H2OFrame(df_initial)
df['diagnosis']=df['diagnosis'].asfactor()
for a in df.columns:
  df[a]=df[a].asnumeric()

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

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

[240]:
```
#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)

train['diagnosis']=train['diagnosis'].asfactor()
aml.train(training_frame=train,y='diagnosis')
```

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

[240]:
```
Model Details
=============
H2OStackedEnsembleEstimator : Stacked Ensemble
Model Key: StackedEnsemble_BestOfFamily_1_AutoML_4_20230410_25250


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: 0.005662005783955975
RMSE: 0.07524630079914876
LogLoss: 0.028432286041025674
AUC: 0.999661447312738
AUCPR: 0.9994926260108352
Gini: 0.999322894625476
Null degrees of freedom: 447
Residual degrees of freedom: 443
Null deviance: 594.7659284347461
Residual deviance: 25.475328292759002
AIC: 35.475328292759

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.4629786759770367
       0    1    Error    Rate
-----  ---  ---  -------  -----------
0      278  0    0        (0.0/278.0)
1      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: 0.023825674194027462
RMSE: 0.15435567431755615
LogLoss: 0.08775914847879626
AUC: 0.9940753279729159
AUCPR: 0.9924507004700177
Gini: 0.9881506559458317
Null degrees of freedom: 447
Residual degrees of freedom: 443
Null deviance: 595.8508793671206
```

```
Residual deviance: 78.63219703700145
AIC: 88.63219703700145

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.49034265320686343
       0    1    Error    Rate
-----  ---  ---  -------  ------------
0      272  6    0.0216   (6.0/278.0)
1      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.
```

[241]:
```
#leaderboard visualization
leaders=aml.leaderboard
leaders.head(rows=leaders.nrows)
aml.leader.model_performance(test_data=test)
```

[241]:
```
ModelMetricsBinomialGLM: stackedensemble
** Reported on test data. **

MSE: 0.012585102897409118
RMSE: 0.11218334500900354
LogLoss: 0.0523940263058841
AUC: 0.9972875226039783
AUCPR: 0.9957565798651044
Gini: 0.9945750452079567
Null degrees of freedom: 120
Residual degrees of freedom: 116
Null deviance: 156.78875756908388
Residual deviance: 12.679354366023954
AIC: 22.679354366023954

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.7282519933220777
       0    1    Error    Rate
-----  ---  ---  -------  -----------
0      79   0    0        (0.0/79.0)
1      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  40
max f2                       0.728252     0.980861  40
max f0point5                 0.728252     0.995146  40
max accuracy                 0.728252     0.991736  40
max precision                1            1         0
max recall                   0.0755553    1         50
max specificity              1            1         0
max absolute_mcc             0.728252     0.981829  40
max min_per_class_accuracy   0.728252     0.97619   40
max mean_per_class_accuracy  0.728252     0.988095  40
```

```
max tns                      1            79        0
max fns                      1            41        0
max fps                      1.84394e-05  79        120
max tps                      0.0755553    42        50
max tnr                      1            1         0
max fnr                      1            0.97619   0
max fpr                      1.84394e-05  1         120
max tpr                      0.0755553    1         50

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
```

[242]:
```
explain_model = aml.explain(frame = test, figsize = (8,6))
```

5 Leaderboard

Leaderboard shows models with their metrics. When provided with H2OAutoML object, the leaderboard shows 5-fold cross-validated metrics by default (depending on the H2OAutoML settings), otherwise it shows metrics computed on the frame. At most 20 models are shown by default.

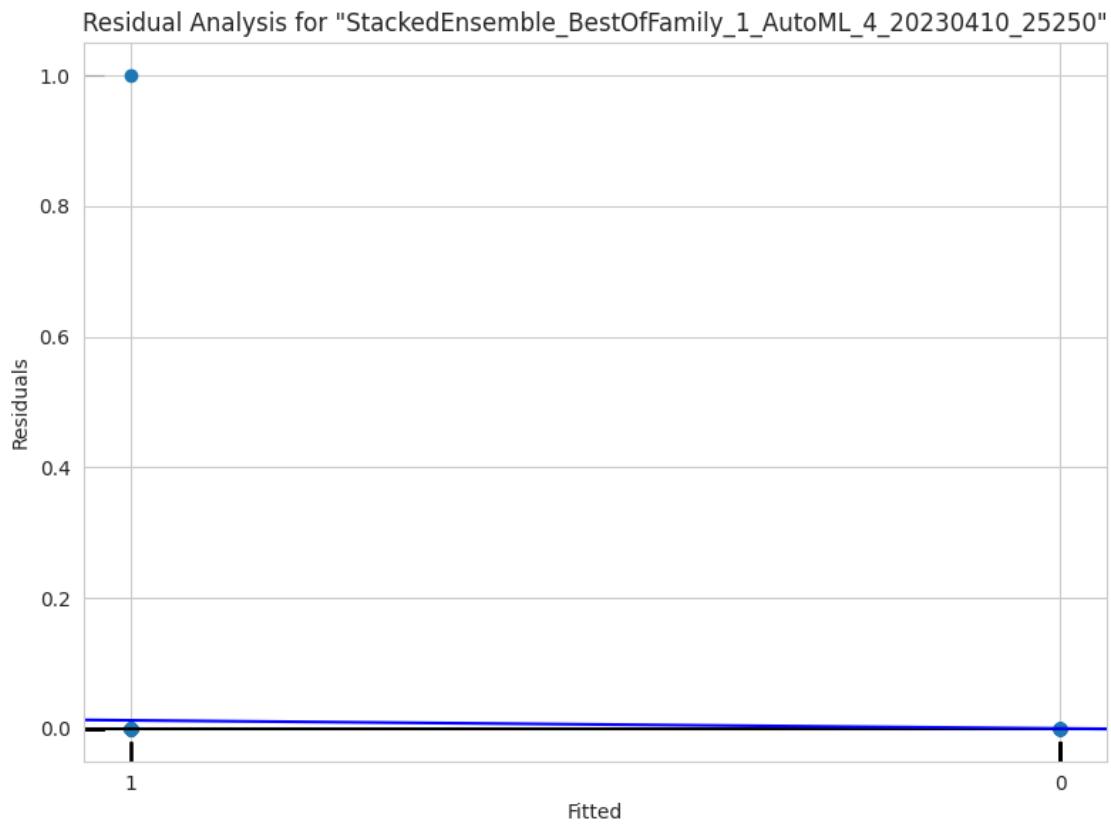
```
model_id                                                    auc    logloss    ␣
↪aucpr    mean_per_class_error      rmse        mse    training_time_ms   ␣
↪predict_time_per_row_ms  algo
-----------------------------------------------------  --------  --------- ␣
↪--------  ----------------------  --------  ---------  ------------------ ␣
↪-------------------------  ---------------
GBM_grid_1_AutoML_4_20230410_25250_model_1              0.99789   0.0674769  0.
↪996142               0.0182339  0.140165  0.0196462                 787       ␣
↪           0.070006  GBM
GLM_1_AutoML_4_20230410_25250                           0.99789   0.054696   0.
↪996563               0.0119048  0.118975  0.0141551                 100       ␣
↪           0.069305  GLM
GBM_2_AutoML_4_20230410_25250                           0.99789   0.0568212  0.
↪996419               0.0182339  0.122617  0.015035                  826       ␣
↪           0.079692  GBM
```

```
GBM_3_AutoML_4_20230410_25250                                0.997288  0.0611831  0.
↪995402                0.0182339  0.131906  0.0173991                 947        ␣
↪            0.072644  GBM
StackedEnsemble_BestOfFamily_1_AutoML_4_20230410_25250  0.997288  0.052394   0.
↪995757                0.0119048  0.112183  0.0125851                6197        ␣
↪            0.116889  StackedEnsemble
StackedEnsemble_AllModels_1_AutoML_4_20230410_25250     0.996986  0.0513361  0.
↪995377                0.0119048  0.110828  0.0122828                7677        ␣
↪            0.192709  StackedEnsemble
GBM_4_AutoML_4_20230410_25250                                0.996383  0.0655409  0.
↪994048                0.024563   0.137206  0.0188254                1512        ␣
↪            0.072902  GBM
XRT_1_AutoML_4_20230410_25250                                0.996082  0.09039    0.
↪9939                  0.0238095  0.151474  0.0229444                 397        ␣
↪            0.052822  DRF
XGBoost_grid_1_AutoML_4_20230410_25250_model_2               0.995781  0.0807877  0.
↪993147                0.0301386  0.149039  0.0222128                 865        ␣
↪            0.054877  XGBoost
GBM_5_AutoML_4_20230410_25250                                0.995781  0.0864384  0.
↪992903                0.0301386  0.157857  0.0249187                 671        ␣
↪            0.071564  GBM
XGBoost_grid_1_AutoML_4_20230410_25250_model_7               0.995781  0.0967837  0.
↪993028                0.0301386  0.164384  0.0270221                 389        ␣
↪            0.052375  XGBoost
XGBoost_1_AutoML_4_20230410_25250                            0.99563   0.127219   0.
↪992295                0.0301386  0.174149  0.0303279                 108        ␣
↪            0.056129  XGBoost
XGBoost_grid_1_AutoML_4_20230410_25250_model_1               0.995479  0.0938203  0.
↪992469                0.0357143  0.16202   0.0262505                 892        ␣
↪            0.054993  XGBoost
XGBoost_grid_1_AutoML_4_20230410_25250_model_8               0.995479  0.0701317  0.
↪993679                0.0119048  0.127952  0.0163718                 301        ␣
↪            0.060597  XGBoost
GBM_grid_1_AutoML_4_20230410_25250_model_2                   0.995479  0.0805979  0.
↪992544                0.0301386  0.15531   0.0241213                 416        ␣
↪            0.064445  GBM
GBM_grid_1_AutoML_4_20230410_25250_model_4                   0.994274  0.065884   0.
↪992522                0.0119048  0.123964  0.0153671                 484        ␣
↪            0.075405  GBM
GBM_grid_1_AutoML_4_20230410_25250_model_5                   0.993972  0.0748131  0.
↪991949                0.0182339  0.137978  0.0190378                1011        ␣
↪            0.066921  GBM
GBM_grid_1_AutoML_4_20230410_25250_model_3                   0.993972  0.0773751  0.
↪991372                0.024563   0.14692   0.0215855                 366        ␣
↪            0.071966  GBM
```

```
XGBoost_2_AutoML_4_20230410_25250                          0.993671  0.101257   0.
↪989216                0.0372212  0.175897  0.0309398                  146
↪           0.062216  XGBoost
DRF_1_AutoML_4_20230410_25250                              0.99352   0.102582   0.
↪99076                 0.024563   0.162931  0.0265465                  141
↪           0.0561    DRF
[20 rows x 10 columns]
```

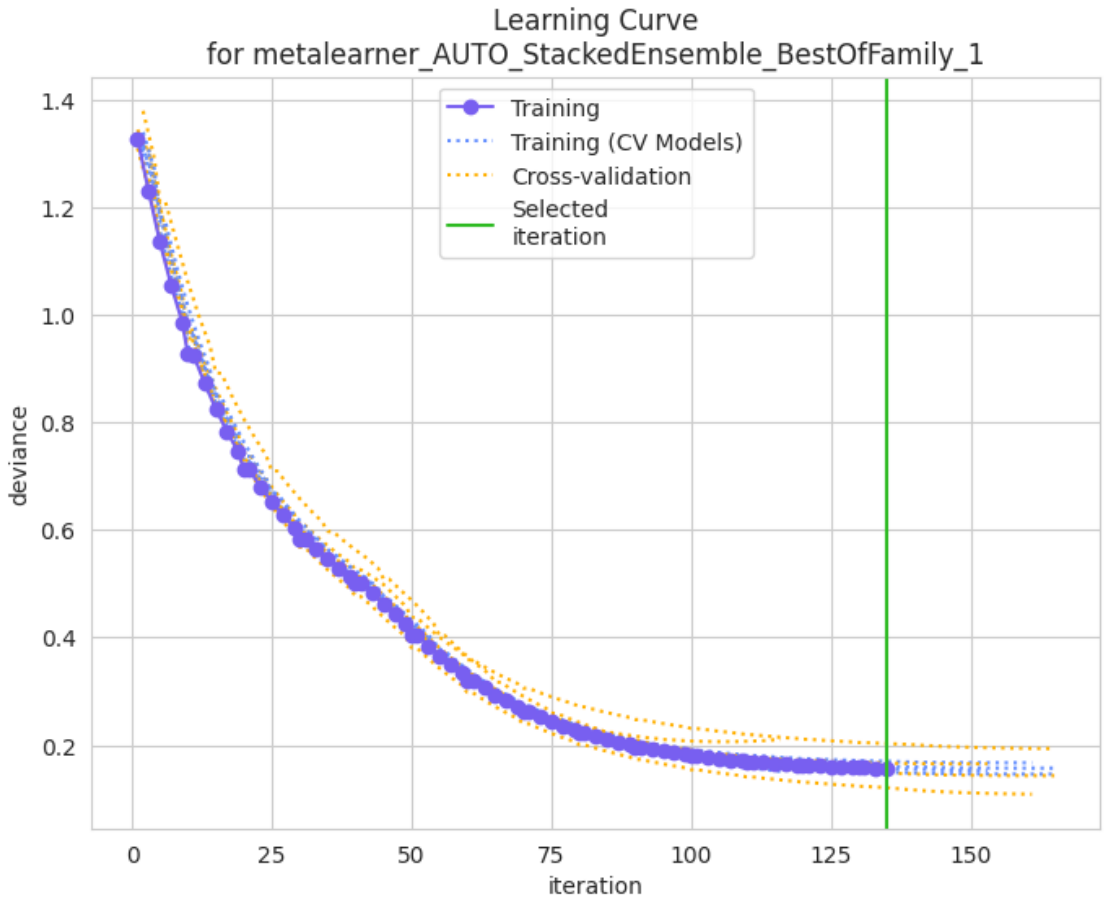
6 Residual Analysis

Residual Analysis plots the fitted values vs residuals on a test dataset. Ideally, residuals should be randomly distributed. Patterns in this plot can indicate potential problems with the model selection, e.g., using simpler model than necessary, not accounting for heteroscedasticity, autocorrelation, etc. Note that if you see "striped" lines of residuals, that is an artifact of having an integer valued (vs a real valued) response variable.



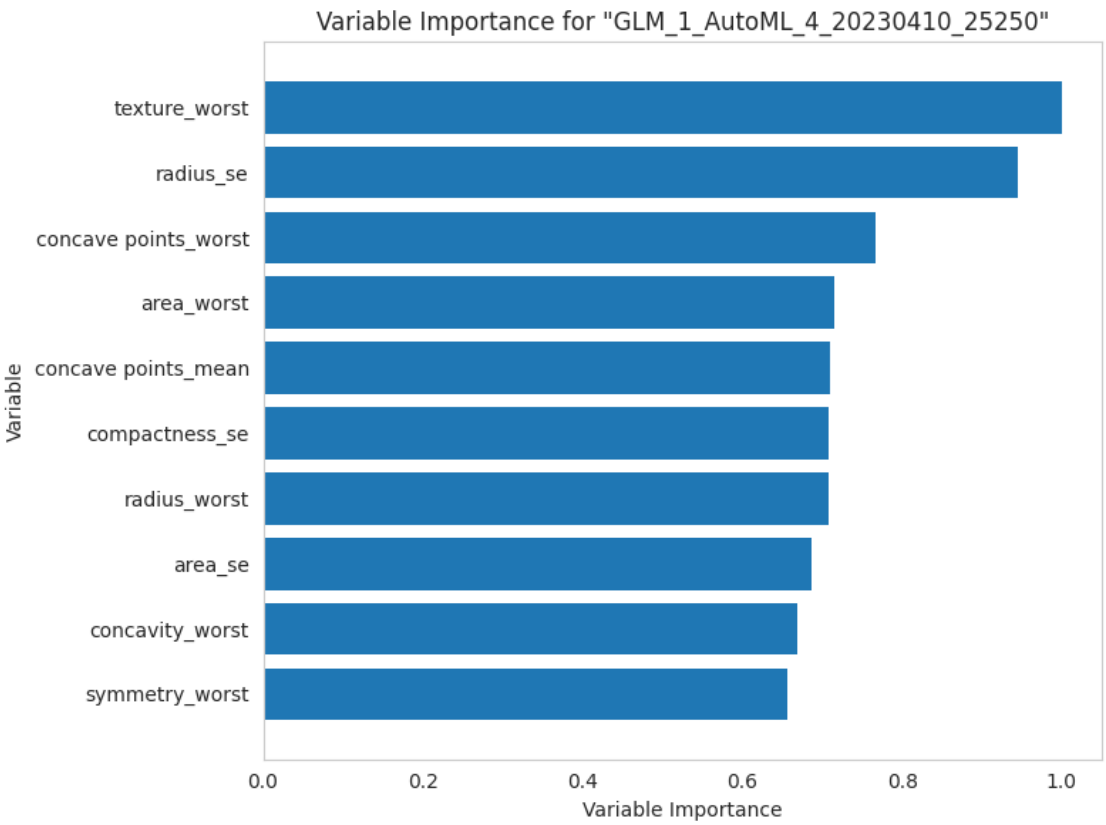
7 Learning Curve Plot

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.



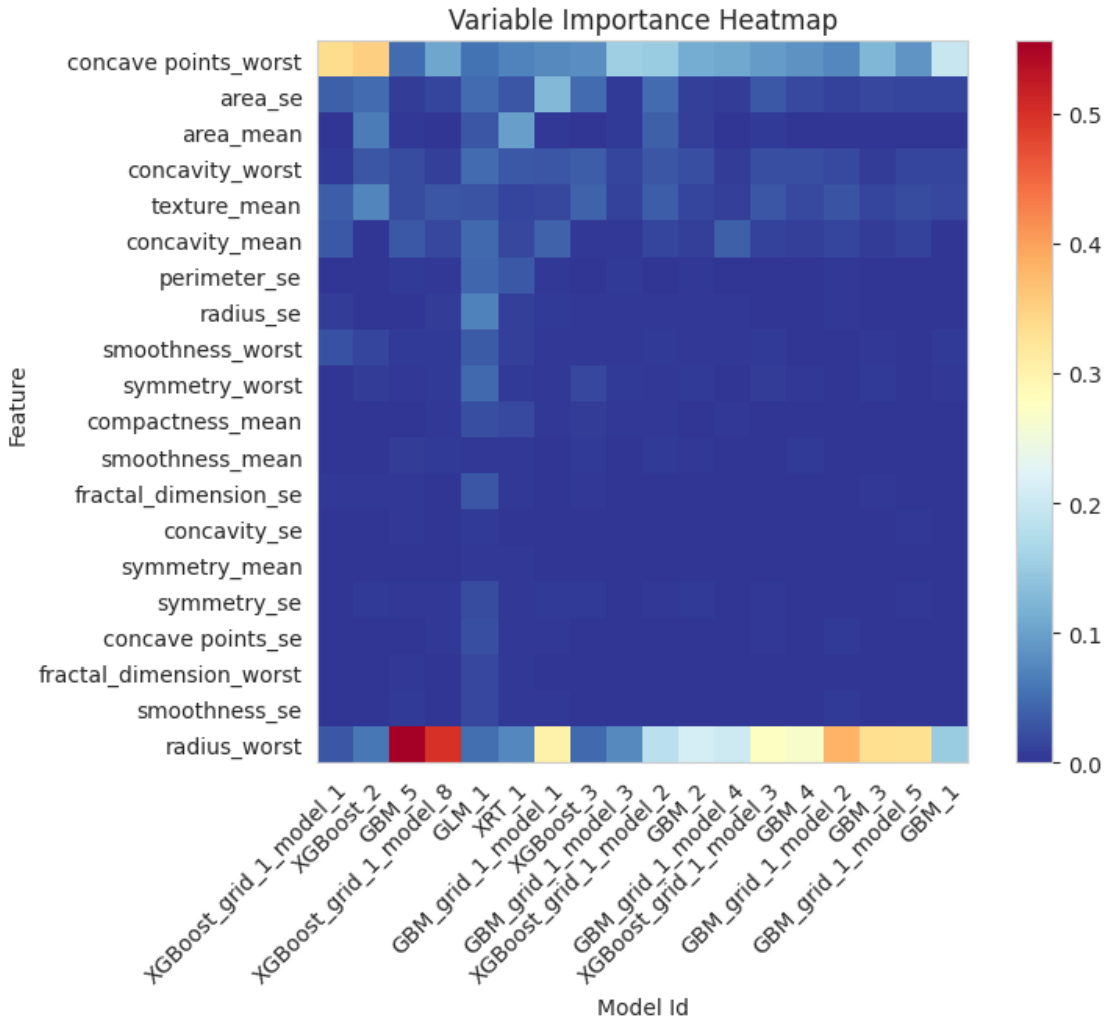
8 Variable Importance

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



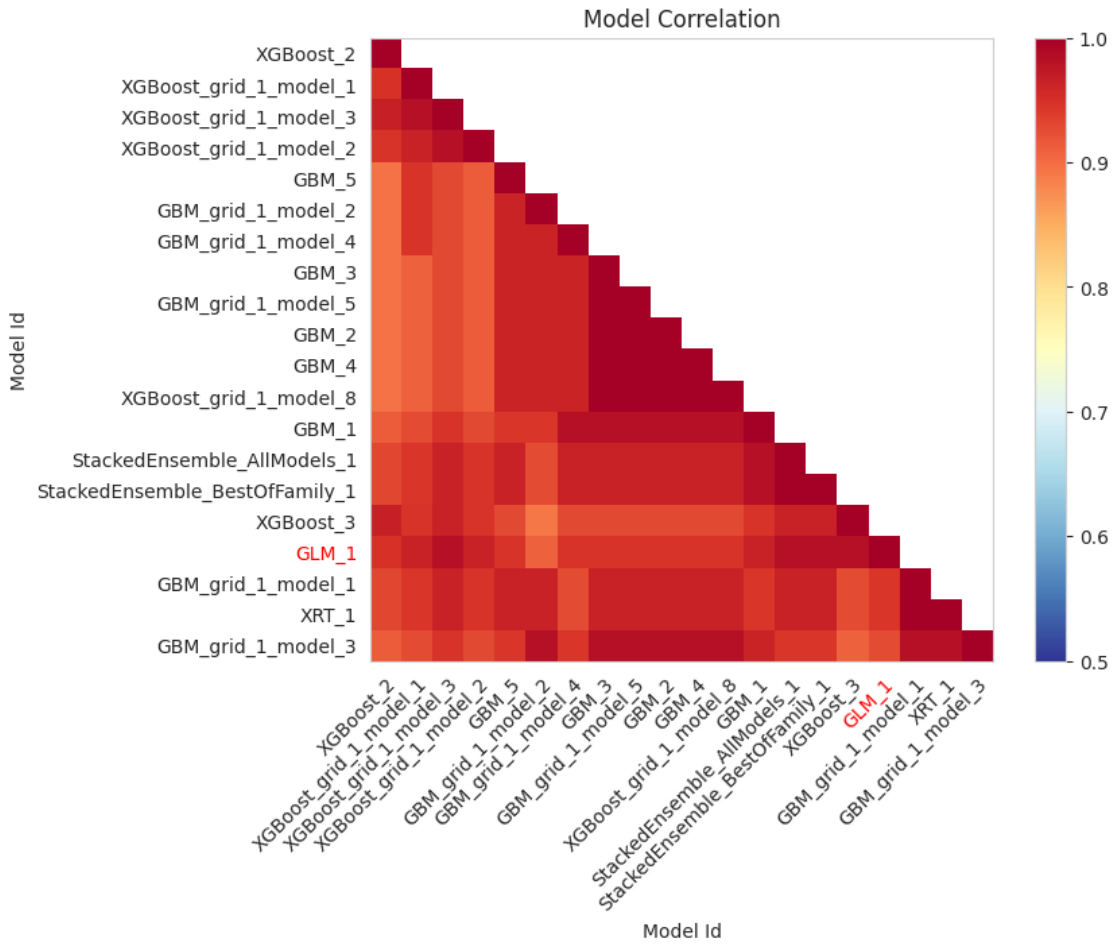
9 Variable Importance Heatmap

Variable importance heatmap shows variable importance across multiple models. Some models in H2O return variable importance for one-hot (binary indicator) encoded versions of categorical columns (e.g. Deep Learning, XGBoost). In order for the variable importance of categorical columns to be compared across all model types we compute a summarization of the the variable importance across all one-hot encoded features and return a single variable importance for the original categorical feature. By default, the models and variables are ordered by their similarity.



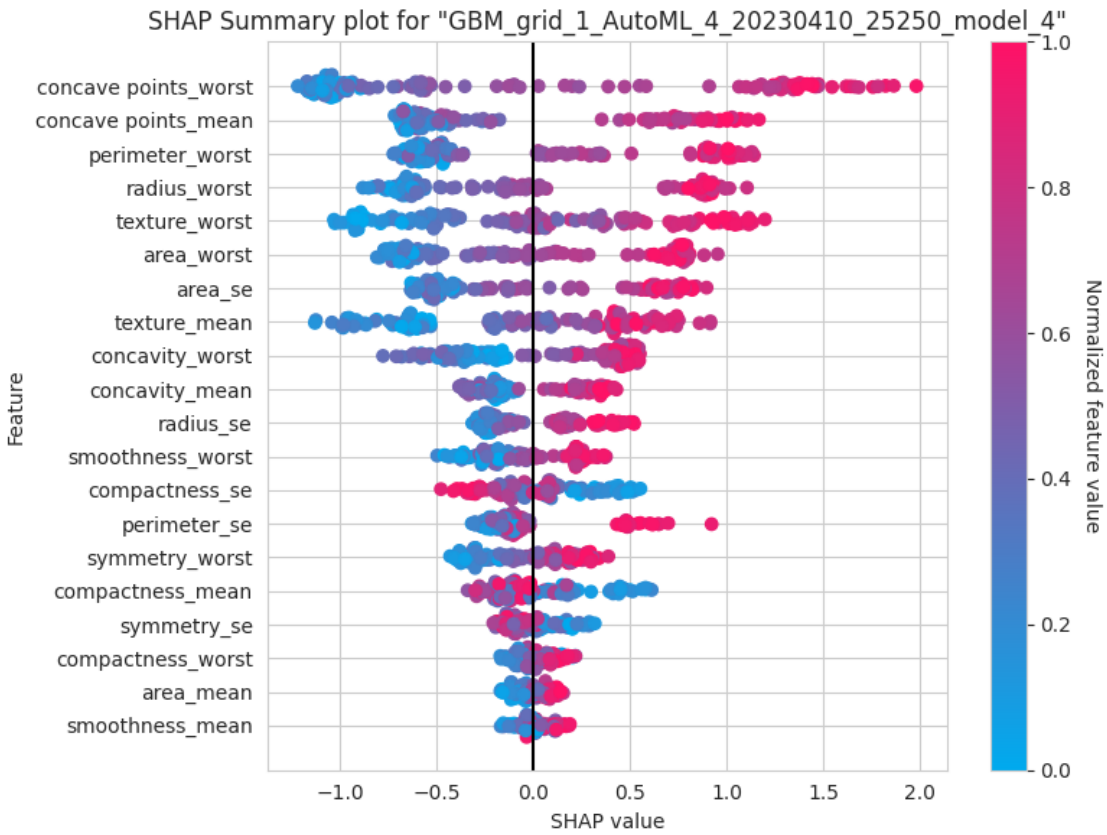
10 Model Correlation

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.



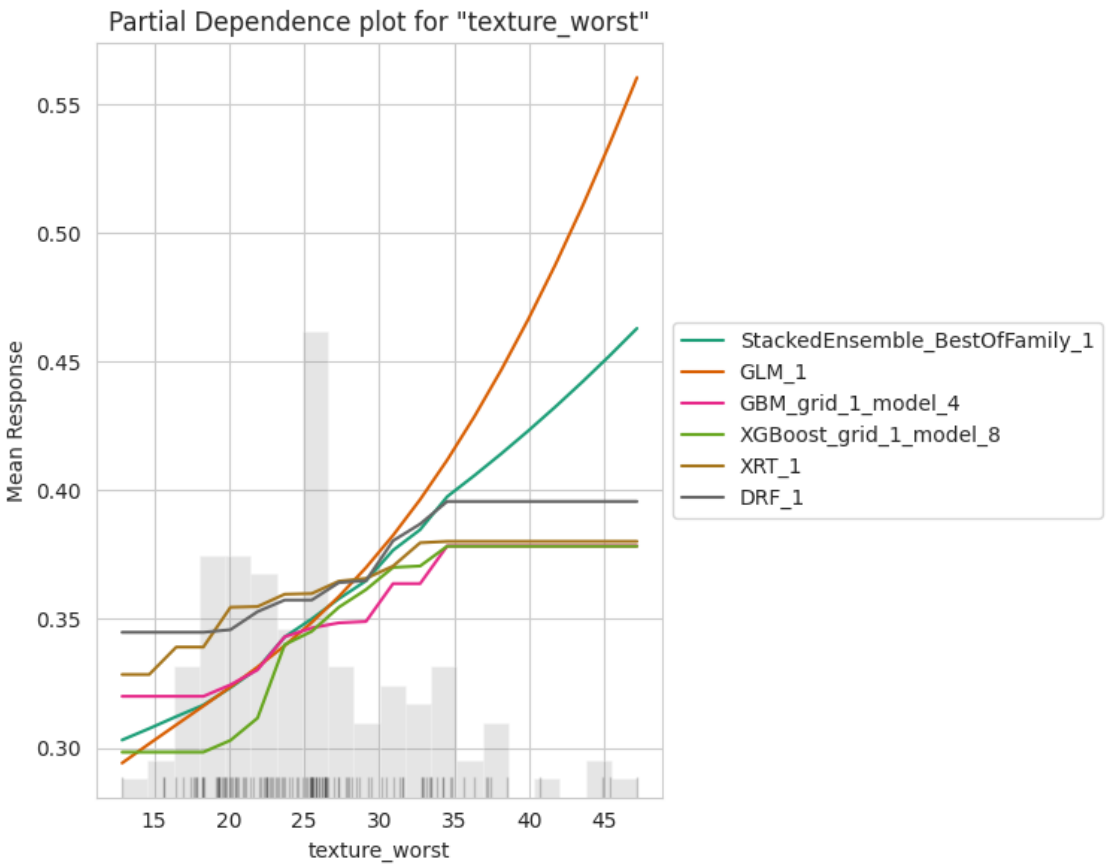
11 SHAP Summary

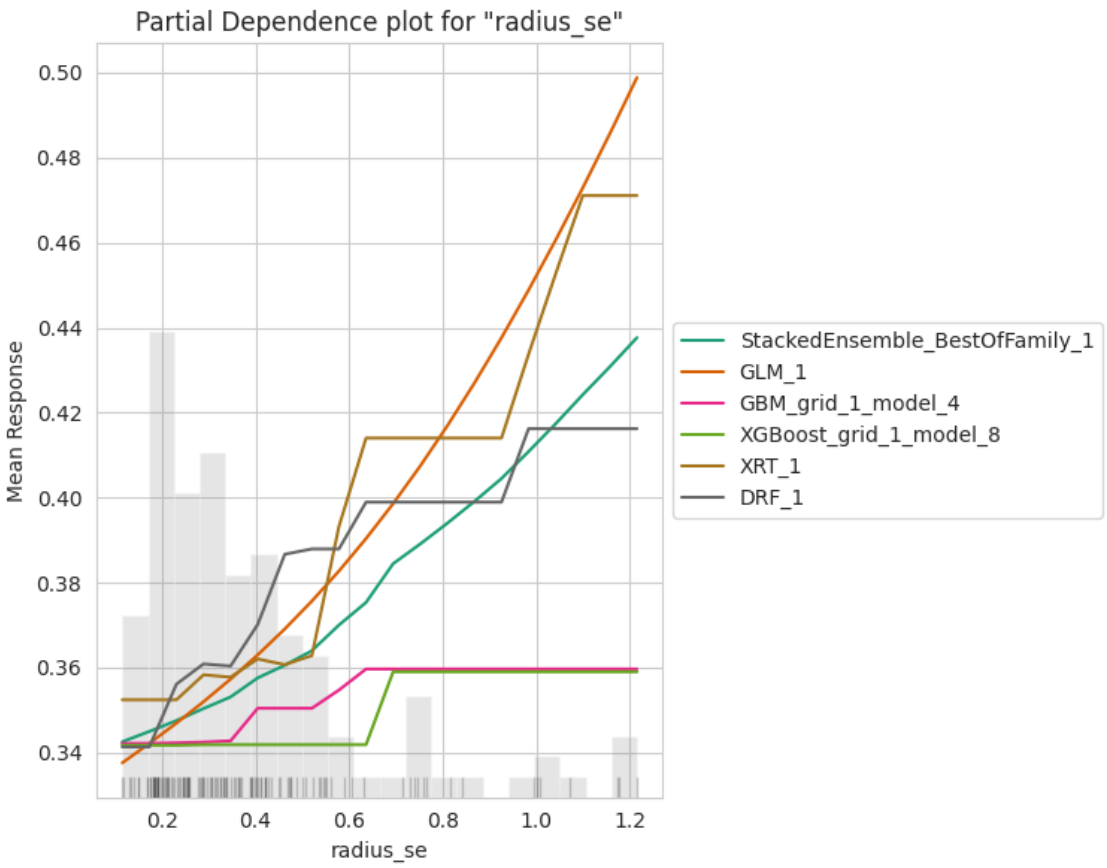
SHAP summary plot shows the contribution of the features for each instance (row of data). The sum of the feature contributions and the bias term is equal to the raw prediction of the model, i.e., prediction before applying inverse link function.

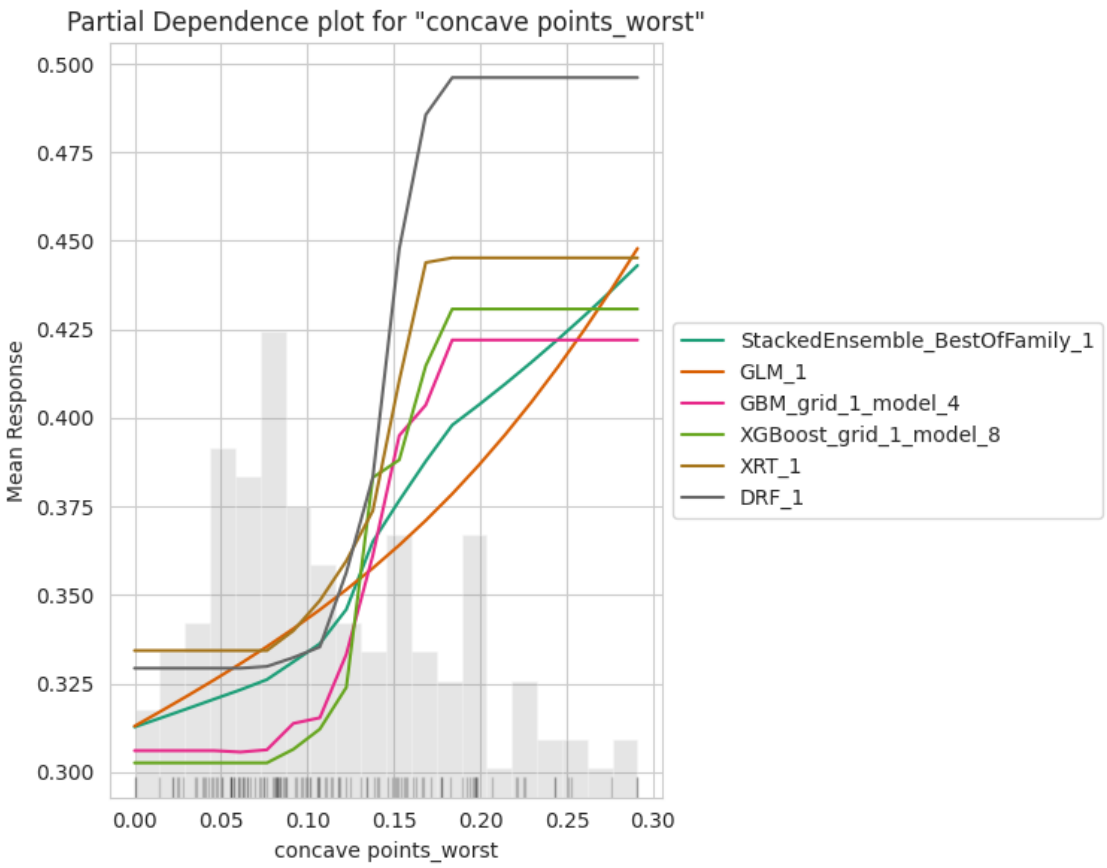


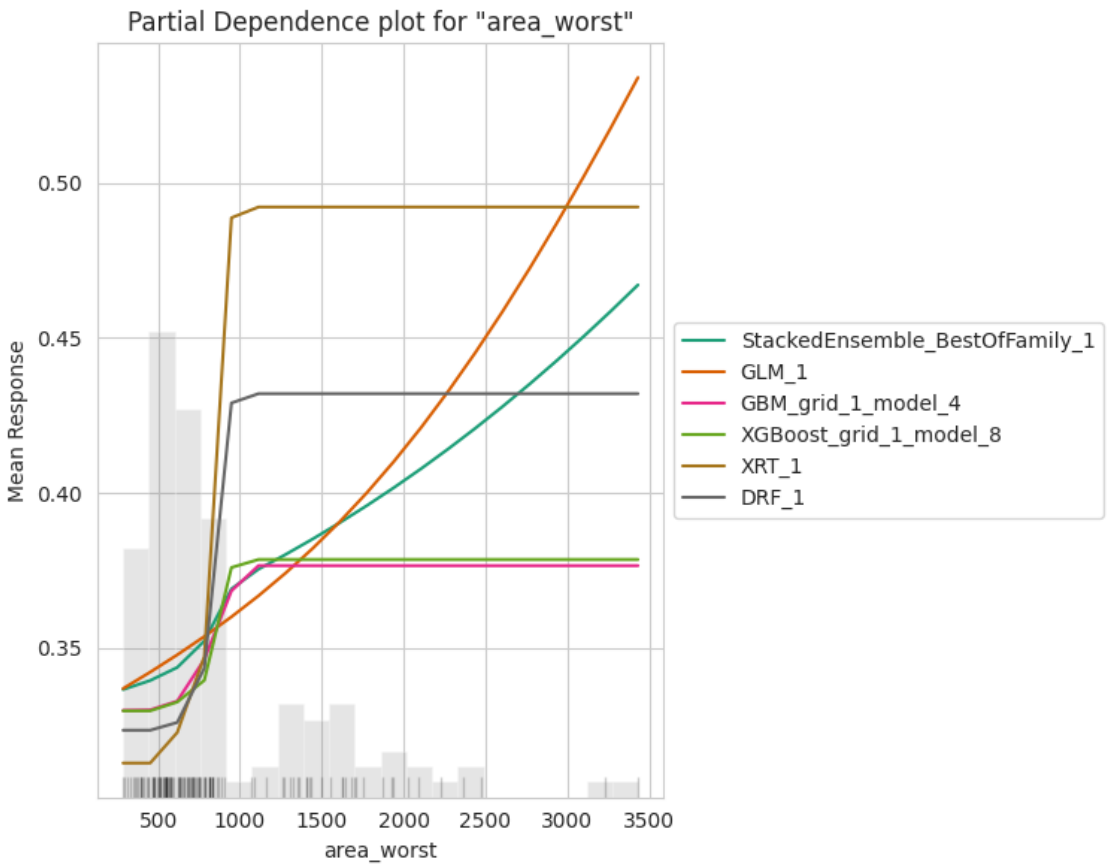
12 Partial Dependence Plots

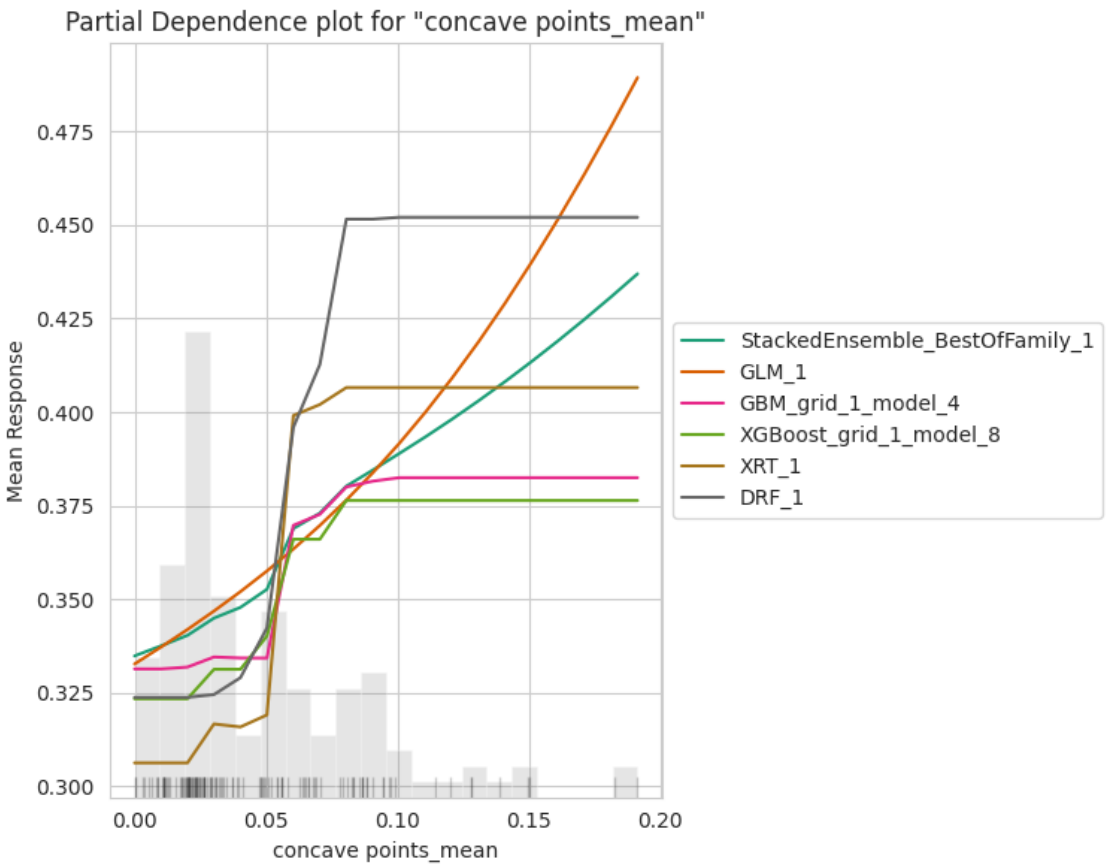
Partial dependence plot (PDP) gives a graphical depiction of the marginal effect of a variable on the response. The effect of a variable is measured in change in the mean response. PDP assumes independence between the feature for which is the PDP computed and the rest.


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

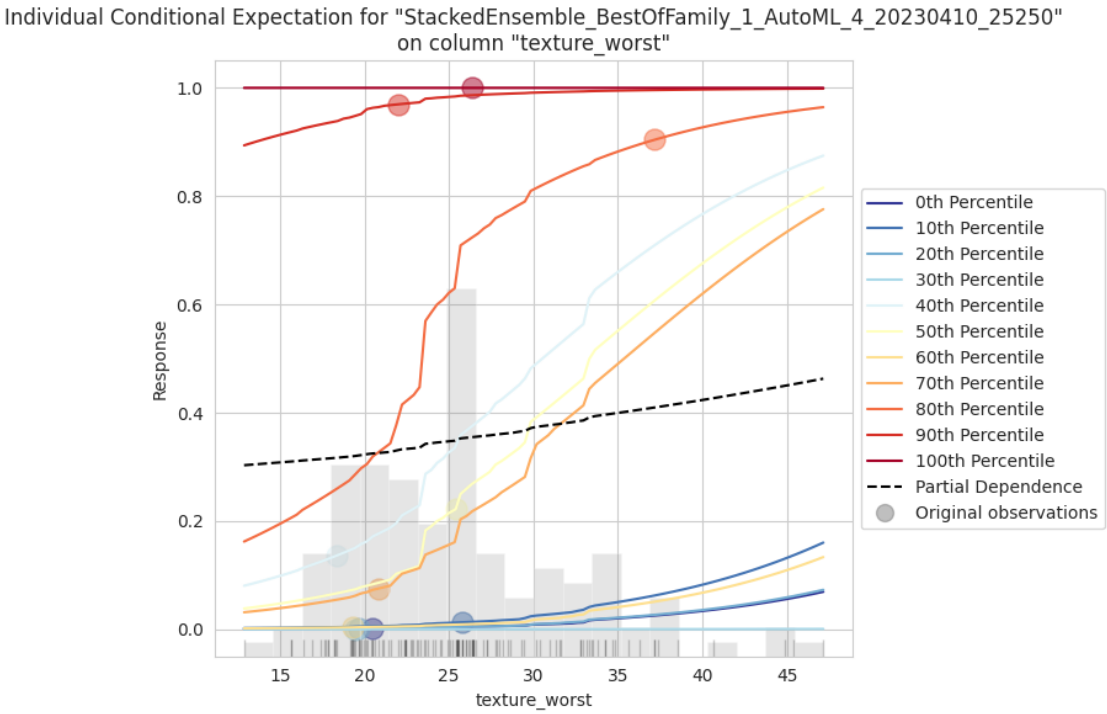

Partial Dependence plot for "concave points_worst"
Mean Response
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0.00
0.05
0.10
0.15
0.20
0.25
0.30
concave points_worst
StackedEnsemble_BestOfFamily_1
GLM_1
GBM_grid_1_model_4
XGBoost_grid_1_model_8
XRT_1
DRF_1

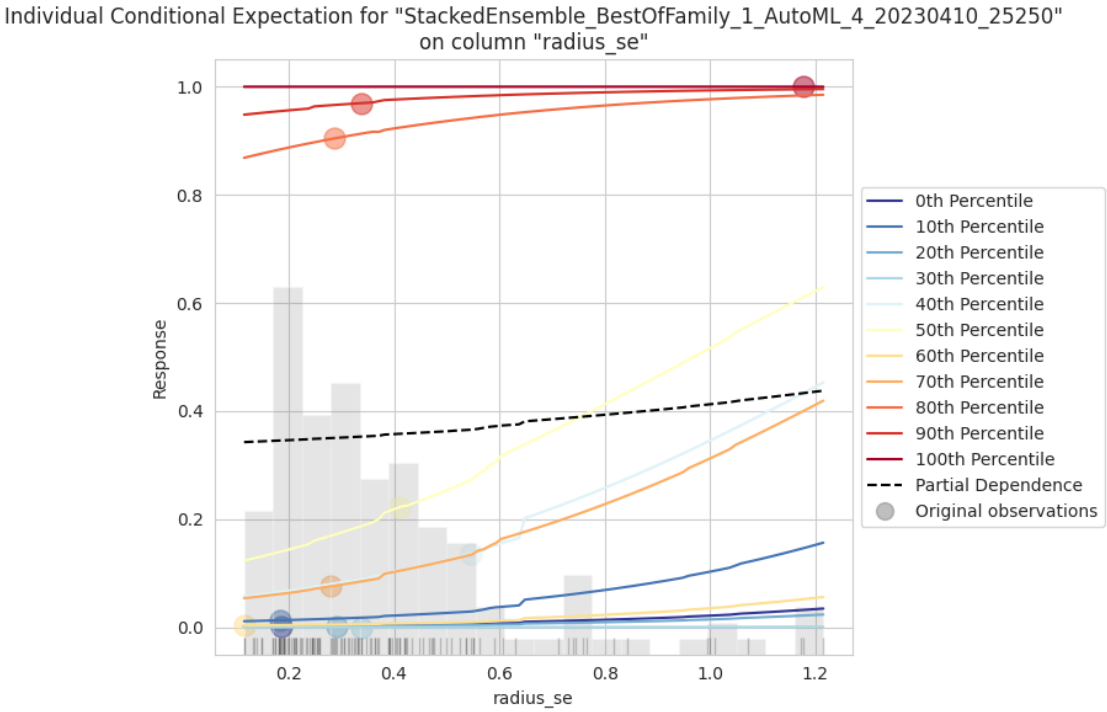

Partial Dependence plot for "area_worst"
Mean Response
0.50
0.45
0.40
0.35
500
1000
1500
2000
2500
3000
3500
area_worst
StackedEnsemble_BestOfFamily_1
GLM_1
GBM_grid_1_model_4
XGBoost_grid_1_model_8
XRT_1
DRF_1

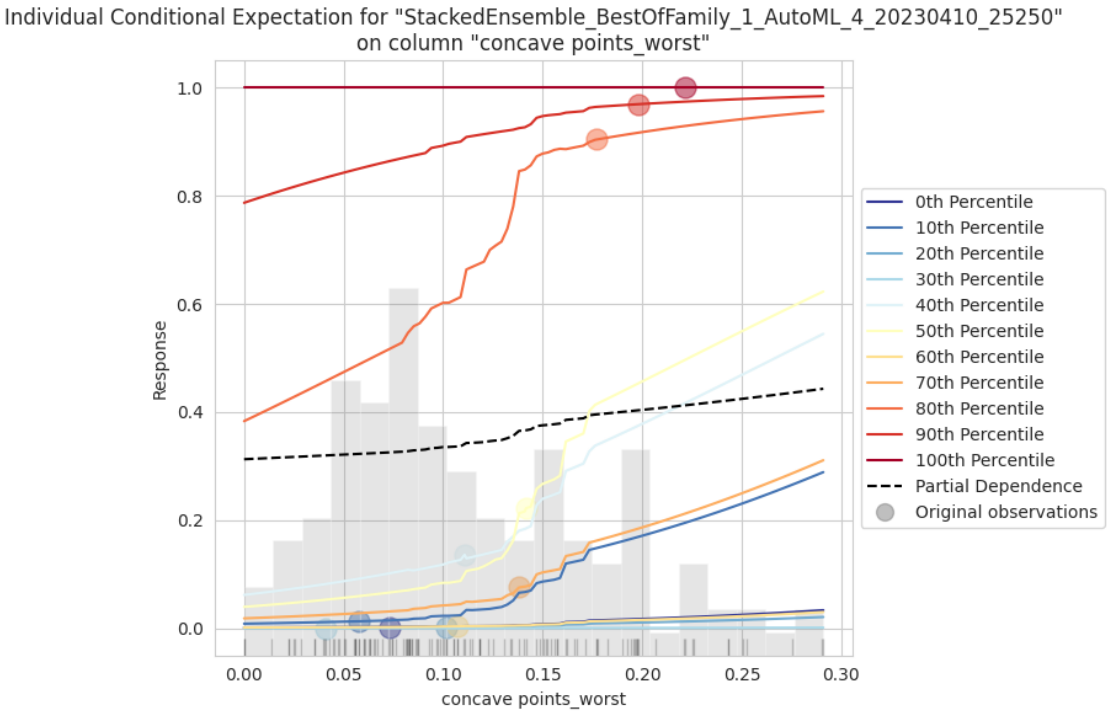


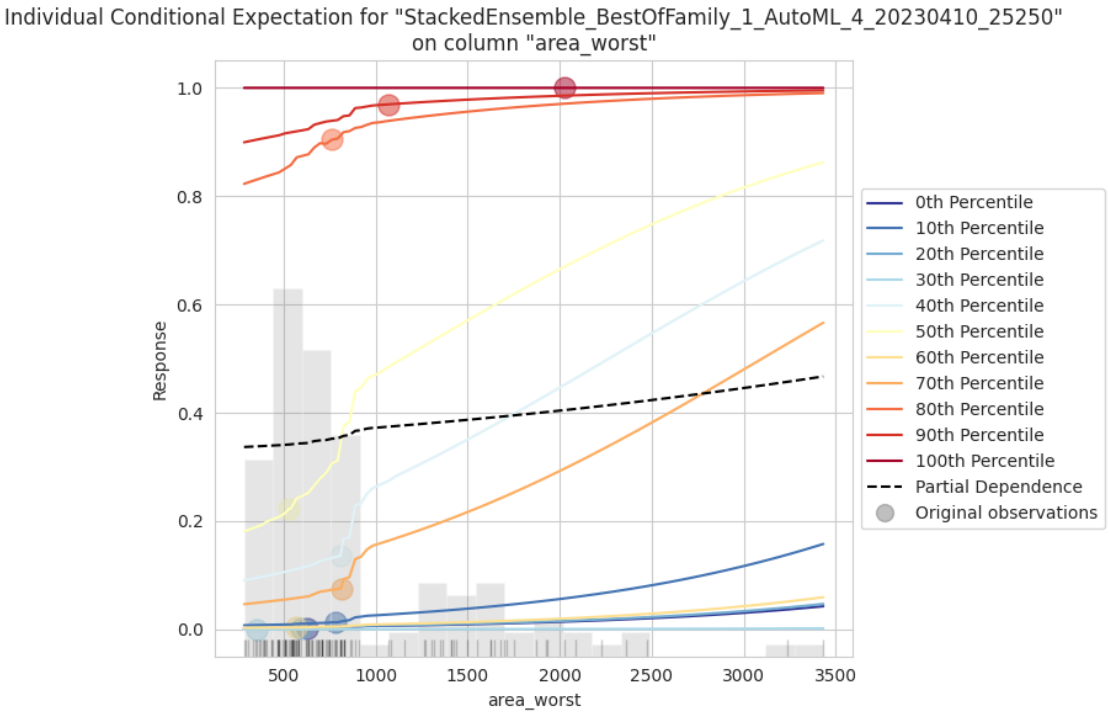
13 Individual Conditional Expectation

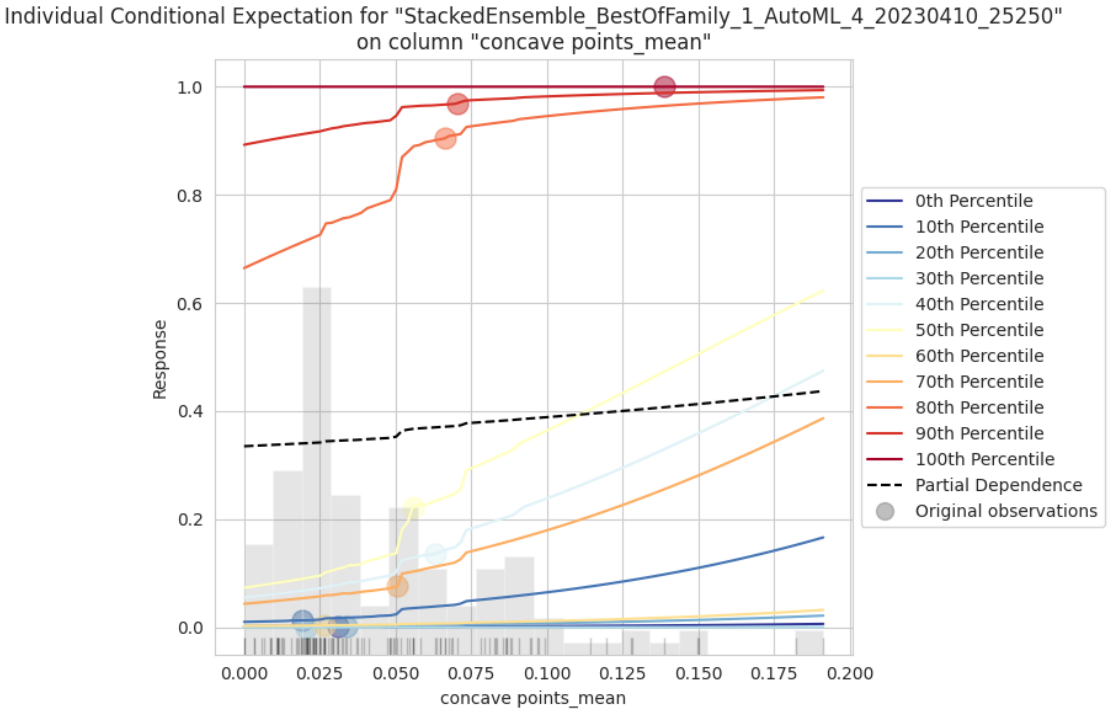
An Individual Conditional Expectation (ICE) plot gives a graphical depiction of the marginal effect of a variable on the response. ICE plots are similar to partial dependence plots (PDP); PDP shows the average effect of a feature while ICE plot shows the effect for a single instance. This function will plot the effect for each decile. In contrast to the PDP, ICE plots can provide more insight, especially when there is stronger feature interaction.


Individual Conditional Expectation for "StackedEnsemble_BestOfFamily_1_AutoML_4_20230410_25250"
on column "texture_worst"
Response
texture_worst
0th Percentile
10th Percentile
20th Percentile
30th Percentile
40th Percentile
50th Percentile
60th Percentile
70th Percentile
80th Percentile
90th Percentile
100th Percentile
Partial Dependence
Original observations


Individual Conditional Expectation for "StackedEnsemble_BestOfFamily_1_AutoML_4_20230410_25250"
on column "radius_se"
Response
radius_se
0th Percentile
10th Percentile
20th Percentile
30th Percentile
40th Percentile
50th Percentile
60th Percentile
70th Percentile
80th Percentile
90th Percentile
100th Percentile
Partial Dependence
Original observations


Individual Conditional Expectation for "StackedEnsemble_BestOfFamily_1_AutoML_4_20230410_25250"
on column "concave points_worst"
Response
concave points_worst
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0th Percentile
10th Percentile
20th Percentile
30th Percentile
40th Percentile
50th Percentile
60th Percentile
70th Percentile
80th Percentile
90th Percentile
100th Percentile
Partial Dependence
Original observations


Individual Conditional Expectation for "StackedEnsemble_BestOfFamily_1_AutoML_4_20230410_25250"
on column "area_worst"
Response
area_worst
0th Percentile
10th Percentile
20th Percentile
30th Percentile
40th Percentile
50th Percentile
60th Percentile
70th Percentile
80th Percentile
90th Percentile
100th Percentile
Partial Dependence
Original observations


Individual Conditional Expectation for "StackedEnsemble_BestOfFamily_1_AutoML_4_20230410_25250"
on column "concave points_mean"
Response
concave points_mean
0th Percentile
10th Percentile
20th Percentile
30th Percentile
40th Percentile
50th Percentile
60th Percentile
70th Percentile
80th Percentile
90th Percentile
100th Percentile
Partial Dependence
Original observations

13.1 Hyperparameter Tuning

[243]:
```
from h2o.grid.grid_search import H2OGridSearch
```

[254]:
```
hyper_params = {
    'ntrees': [50, 100, 200],
    'max_depth': [3, 5, 7]
}
model=aml.leader
grid = H2OGridSearch(model=model, hyper_params=hyper_params)
grid.train(x=train.names[:-1], y=train.names[-1], training_frame=train,␣
 ↪validation_frame=test)
hyperparams=grid.get_grid()[0]
```

```
---------------------------------------------------------------------------
H2OResponseError                          Traceback (most recent call last)
<ipython-input-254-34b83a2c9484> in <cell line: 46>()
     44
     45 # train the grid search
---> 46 grid.train(x=train.names[:-1], y=train.names[-1], training_frame=train,␣
 ↪validation_frame=test)

/usr/local/lib/python3.9/dist-packages/h2o/grid/grid_search.py in train(self, x␣
 ↪y, training_frame, offset_column, fold_column, weights_column,␣
 ↪validation_frame, **params)
    353         x = list(xset)
    354         parms["x"] = x
--> 355         self.build_model(parms)
    356         return self
    357

/usr/local/lib/python3.9/dist-packages/h2o/grid/grid_search.py in␣
 ↪build_model(self, algo_params)
    390         else:
    391             self.model._estimator_type = "unsupervised"
--> 392         self._model_build(x, y, training_frame, validation_frame,␣
 ↪algo_params)
    393
    394     def _model_build(self, x, y, tframe, vframe, kwargs):

/usr/local/lib/python3.9/dist-packages/h2o/grid/grid_search.py in␣
 ↪_model_build(self, x, y, tframe, vframe, kwargs)
    408         if self.grid_id is not None: kwargs["grid_id"] = self.grid_id
```

```
    409         rest_ver = kwargs.pop("_rest_version") if "_rest_version" in␣
 ↪kwargs else None
--> 410         self._run_grid_job(kwargs, rest_ver=rest_ver)
    411
    412     def _run_grid_job(self, params, end_point="", rest_ver=None):

/usr/local/lib/python3.9/dist-packages/h2o/grid/grid_search.py in␣
 ↪_run_grid_job(self, params, end_point, rest_ver)
    412     def _run_grid_job(self, params, end_point="", rest_ver=None):
    413         algo = self.model.algo
--> 414         grid = H2OJob(h2o.api("POST /99/Grid/%s%s" % (algo, end_point),␣
 ↪data=params), job_type=(algo + " Grid Build"))
    415         if self._future:
    416             self._job = grid

/usr/local/lib/python3.9/dist-packages/h2o/h2o.py in api(endpoint, data, json,␣
 ↪filename, save_to)
    122     # type checks are performed in H2OConnection class
    123     _check_connection()
--> 124     return h2oconn.request(endpoint, data=data, json=json,␣
 ↪filename=filename, save_to=save_to)
    125
    126

/usr/local/lib/python3.9/dist-packages/h2o/backend/connection.py in␣
 ↪request(self, endpoint, data, json, filename, save_to)
    497                     save_to = save_to(resp)
    498                 self._log_end_transaction(start_time, resp)
--> 499                 return self._process_response(resp, save_to)
    500
    501             except (requests.exceptions.ConnectionError, requests.
 ↪exceptions.HTTPError) as e:

/usr/local/lib/python3.9/dist-packages/h2o/backend/connection.py in␣
 ↪_process_response(response, save_to)
    851         if status_code in {400, 404, 412} and isinstance(data,␣
 ↪H2OErrorV3):
    852             data.show_stacktrace = False
--> 853             raise H2OResponseError(data)
    854
    855         # Server errors (notably 500 = "Server Error")

H2OResponseError: Server error water.exceptions.H2OIllegalArgumentException:
  Error: Illegal argument: max_depth of function: grid: Unknown hyper parameter␣
 ↪for grid search!
  Request: POST /99/Grid/stackedensemble
```

```
  data: {'parallelism': '1', 'hyper_parameters': '{"ntrees":␣
↪[50,100,200],"max_depth": [3,5,7]}', 'base_models': '[]',␣
↪'metalearner_algorithm': 'auto', 'metalearner_nfolds': '0',␣
↪'metalearner_transform': 'none', 'max_runtime_secs': '0.0', 'seed': '-1',␣
↪'score_training_samples': '10000', 'keep_levelone_frame': 'False', 'auc_type'␣
↪'auto', 'training_frame': 'Key_Frame__upload_b48705ec556a7ed359c534a5c054cf6.
↪hex', 'validation_frame': 'Key_Frame__upload_9e5dd1084f3171e1e206efdd8da4229.
↪hex', 'response_column': 'fractal_dimension_worst'}
```

14 Model Interpretability

SHAP analysis is performed on selected models to interpret the feature contributions to predictions. Partial dependence plots and feature importance summary plots are constructed.

[ ]:
```
#Calculating the shap values
train_df_best.reset_index(inplace=True,drop=True)
test_df_best.reset_index(inplace=True,drop=True)
test_df_best=test_df_best.drop(['diagnosis'],axis=1)
train_df_best=train_df_best.drop(['diagnosis'],axis=1)
explainer = shap.Explainer(rf.predict, train_df_best)

shap_lreg = explainer(test_df_best)
```

[ ]:
```
sample_feature_importance(shap_lreg,20)
```

[ ]:
```
shap.plots.bar(shap_lreg,max_display=13)
print("\n")
shap.summary_plot(shap_lreg,max_display=10)
```

Some of the best features obtained from KBest method are:-
'concavity_se', 'concave points_se', 'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smoothness_worst', 'compactness_worst', 'concavity_worst', 'concave points_worst', 'symmetry_worst', 'fractal_dimension_worst'.

The 12 best features obtained from the SHAP summary plot are:-
'concave points_worst','area_worst','perimeter worst','radius worst','concavity_worst' ,'concavity mean','concave points_mean','texture_worst', 'texture _mean','area se'.

15 Questions and Answers

- How well did it work?
  The EDA process was carried out succesfully on the given dataset. Major parameters such as anomaly detection and feature selection were succesfully implemented. Informed by the learnings from performing EDA, a suitable model was chosen which was agnostics to certain characteristics of the dataset. The model did a commendable job in fitting the data and gave accurate predictions with minimum recall/sensitivity. The model suggested by AutoML process also exhibited robust performance, for the generated set of hyperparameters. Model

inference of initial model was carried out by Shapley analysis, which yielded similar feature importances as the KBest approach.

- What did you learn?
  From my data science exercises, I learned a great deal about the entire data science workflow, from data acquisition to model interpretation. One of the most important lessons I learned was the importance of domain knowledge in identifying relevant features and building effective models.
  - Learnt the importance of data preparation and cleaning, as these steps are critical to ensuring the accuracy and relevance of the insights derived from the data.
  - Gained valuable experience in data visualization and presentation skills.

  - Learnt how to use various machine learning algorithms and techniques, such as decision trees, random forests, and logistic regression to build predictive models.
  - Implemented the AutoML process to enhance model building process and get insights into model interpretability.

16 References

- Feature Importance
- Isolation Forest *Hyperparameter Tuning

MIT License 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.

[ ]:

```
```

Model Interpretability

Due Mar 29 by 11:59pm **Points** 100 **Submitting** a website url or a file upload
Available until Apr 15 at 11:59pm

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

Interpret your models.

1. Fit a linear model and interpret the regression coefficients

2. Fit a tree-based model and interpret the nodes

3. Use auto ml to find the best model

4. Run SHAP analysis on the models from steps 1, 2, and 3, interpret the SHAP values and compare them with the other model interpretability methods.

Grading: 80% if meet the above requirements

Up to 20% for presentation and explanation.

Model_Interpretability (2)

December 16, 2023

0.1 # Model Interpretability

The purpose of this notebook is to implement interpretation of linear and tree-based machine learning models.
The interpretation of linear models such as Logistic Regression follows the calculation of predictor coefficients and significances. Incase of tree based models, the decision nodes and end nodes are observed.
Model agnostic methods such as SHAP analysis is performed to ascertain feature importance, for a given model.
Finally, AutoML functionality of the h2o library is used to train several models and obtain various explanations of the best performing model.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
import statsmodels.api as sm
from sklearn.model_selection import␣
↪train_test_split,cross_val_score,StratifiedKFold,RandomizedSearchCV
from sklearn import tree
import pylab as py
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import␣
↪classification_report,confusion_matrix,ConfusionMatrixDisplay,accuracy_score,precision_scor
from sklearn.tree import DecisionTreeClassifier
!pip install h2o
import h2o
import shap
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: h2o in /usr/local/lib/python3.9/dist-packages
(3.40.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-
packages (from h2o) (2.27.1)
Requirement already satisfied: tabulate in /usr/local/lib/python3.9/dist-
packages (from h2o) (0.8.10)
```

```
Requirement already satisfied: future in /usr/local/lib/python3.9/dist-packages
(from h2o) (0.18.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from requests->h2o) (2022.12.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from requests->h2o) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from requests->h2o) (1.26.15)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-
packages (from requests->h2o) (3.4)
```

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. 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.

[2]:
```
#importing the dataset and dropping 'id' column
dataset=pd.read_csv("https://raw.githubusercontent.com/SKashyap1999/DataScience/
↪main/Breast_Cancer-Wisconsin.csv")
```

```
dataset=dataset.drop(["id"],axis=1)
```

2 Exploratory Data Analysis

[3]:
```
dataset.describe()
```

[3]:
```
       radius_mean  texture_mean  perimeter_mean    area_mean  \
count   569.000000    569.000000      569.000000   569.000000
mean     14.127292     19.289649       91.969033   654.889104
std       3.524049      4.301036       24.298981   351.914129
min       6.981000      9.710000       43.790000   143.500000
25%      11.700000     16.170000       75.170000   420.300000
50%      13.370000     18.840000       86.240000   551.100000
75%      15.780000     21.800000      104.100000   782.700000
max      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  fractal_dimension_mean  …  radius_worst  \
count     569.000000              569.000000  …    569.000000
mean        0.181162                0.062798  …     16.269190
std         0.027414                0.007060  …      4.833242
min         0.106000                0.049960  …      7.930000
25%         0.161900                0.057700  …     13.010000
50%         0.179200                0.061540  …     14.970000
75%         0.195700                0.066120  …     18.790000
max         0.304000                0.097440  …     36.040000

       texture_worst  perimeter_worst   area_worst  smoothness_worst  \
count     569.000000       569.000000   569.000000        569.000000
mean       25.677223       107.261213   880.583128          0.132369
std         6.146258        33.602542   569.356993          0.022832
min        12.020000        50.410000   185.200000          0.071170
25%        21.080000        84.110000   515.300000          0.116600
50%        25.410000        97.660000   686.500000          0.131300
75%        29.720000       125.400000  1084.000000          0.146000
max        49.540000       251.200000  4254.000000          0.222600
```

```
       compactness_worst  concavity_worst  concave points_worst  \
count         569.000000       569.000000            569.000000
mean            0.254265         0.272188              0.114606
std             0.157336         0.208624              0.065732
min             0.027290         0.000000              0.000000
25%             0.147200         0.114500              0.064930
50%             0.211900         0.226700              0.099930
75%             0.339100         0.382900              0.161400
max             1.058000         1.252000              0.291000

       symmetry_worst  fractal_dimension_worst
count      569.000000               569.000000
mean         0.290076                 0.083946
std          0.061867                 0.018061
min          0.156500                 0.055040
25%          0.250400                 0.071460
50%          0.282200                 0.080040
75%          0.317900                 0.092080
max          0.663800                 0.207500

[8 rows x 30 columns]
```

[4]:
```
dataset[dataset["concavity_worst"]==0.0]['diagnosis']
```

[4]:
```
101    B
140    B
174    B
175    B
192    B
314    B
391    B
473    B
538    B
550    B
557    B
561    B
568    B
Name: diagnosis, dtype: object
```

It is observed that for few samples, the values relating to concavity is zero. All the samples with afforementioned trend has a similar target value.
To prevent any effect on generalization by the models, these samples(rows in dataset) are dropped.

[5]:
```
lst=[l for l in dataset.describe().loc['min'].index if dataset.describe().
 ↪loc['min'][l]==0]
for a in lst:
  if(a=='diagnosis'):
    continue
```

```
  dataset=dataset[dataset[a]!=0.0]
```

[6]:
```
#Checking for null-values
print(dataset.isnull().sum())
```

```
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
```

[7]:
```
# Encoding the categorical target feature
def dummy(d):
  if(d=="M"):
    return 1
  else:
    return 0
dataset['diagnosis']=dataset['diagnosis'].apply(dummy)

df_target=dataset['diagnosis']
```

```
df_predictor=dataset.drop(['diagnosis'],axis=1)
```

[8]:
```
# Scaling and Splitting operations.
from sklearn.preprocessing import MinMaxScaler;

X_train, X_test, y_train, y_test = train_test_split(df_predictor, df_target,␣
 ↪test_size=0.1, random_state=100)
scaler=MinMaxScaler()
fit=scaler.fit(X_train)
X_train=fit.transform(X_train)
X_test=fit.transform(X_test)
train_df_pred=pd.DataFrame(X_train,columns=df_predictor.columns)
train_df_target=pd.DataFrame(y_train,columns=['diagnosis'])
test_df_pred=pd.DataFrame(X_test,columns=df_predictor.columns)
test_df_target=pd.DataFrame(y_test,columns=['diagnosis'])

train_df_pred=train_df_pred.reset_index(drop = True)
train_df_target=train_df_target.reset_index(drop = True)
test_df_pred=test_df_pred.reset_index(drop = True)
test_df_target=test_df_target.reset_index(drop = True)

pred_scaled_array=np.concatenate((X_train,X_test),axis=0)
target_scaled_array=np.concatenate((y_train,y_test),axis=0)
df_scaled_pred=pd.DataFrame(pred_scaled_array,columns=df_predictor.columns)
df_scaled_target=pd.DataFrame(target_scaled_array,columns=['diagnosis'])
```

[9]:
```
def cf_display(model,test,pred):
  cf = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(test,model.
 ↪predict(pred)),display_labels=['Benign','Malignant'])
  cf.plot()
  plt.show()

def metric_scores(test,pred):
  accuracy = accuracy_score(test,pred)
  precision=precision_score(test,pred)
  recall=recall_score(test,pred)
  print("Accuracy:", accuracy)
  print("Precision: ",precision)
  print("Recall ",recall)


def pdp(feature,model,train,index,shap_values):
  shap.partial_dependence_plot(
            feature,
            model.predict,
            train,
            ice=False,
```

```
            model_expected_value=True,
            feature_expected_value=True,
            shap_values=shap_values[index:index+1,:])
def sample_feature_importance(shap_vals,idx):
        return shap.plots.waterfall(shap_vals[idx],max_display=10)
```

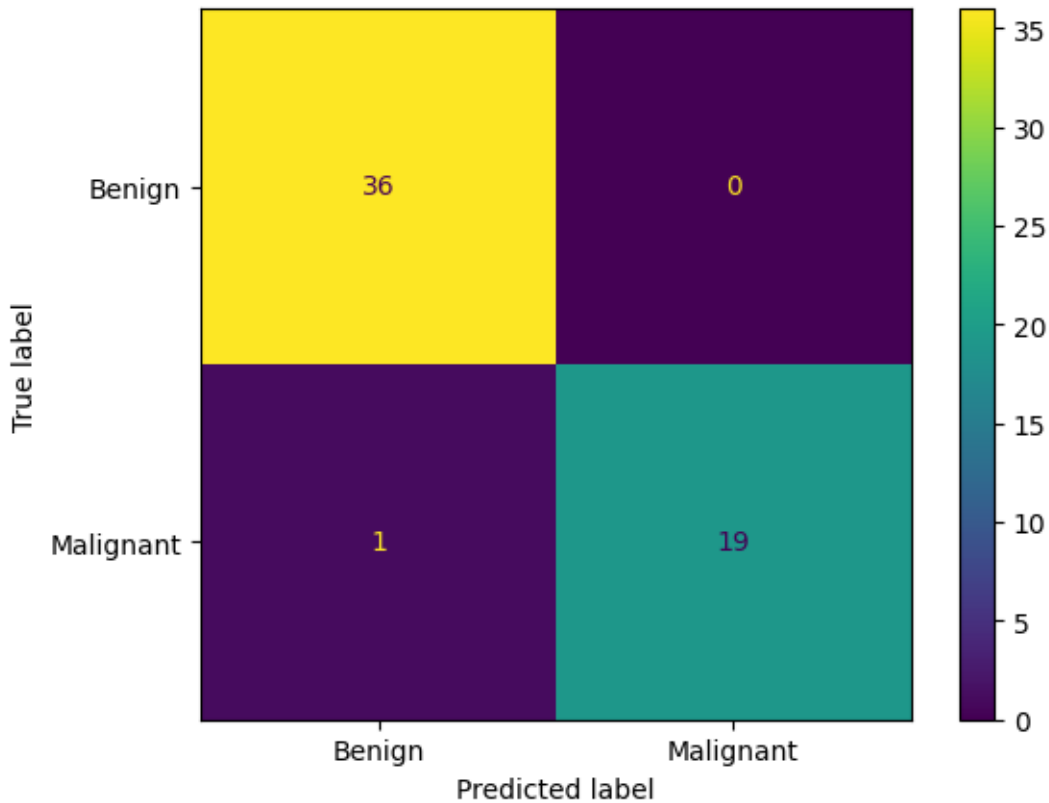
#Logistic Regression

The logistic regression model uses a linear function to predict the probability of the dependent variable being equal to 1 given a set of independent variables. The linear function is then transformed using the logistic function to obtain a predicted probability between 0 and 1.

[10]:
```
model=LogisticRegression(solver='liblinear',random_state=0)
lreg_fit=model.fit(train_df_pred,train_df_target)

y_pred=model.predict(X_test)
metric_scores(y_test,y_pred)
cf_display(model,y_test,X_test)
```

```
Accuracy: 0.9821428571428571
Precision:  1.0
Recall  0.95
```

```
A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
X does not have valid feature names, but LogisticRegression was fitted with
feature names
X does not have valid feature names, but LogisticRegression was fitted with
feature names
```



[11]:
```
#Fitting another logistic regression model provided by the statsmodel library,␣
↪to interpret the regression coefficients
log_reg=sm.Logit(df_scaled_target,df_scaled_pred).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.043790
         Iterations 17
```

[12]:
```
#Summarizing the regression results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:              diagnosis   No. Observations:                  556
Model:                          Logit   Df Residuals:                      526
Method:                           MLE   Df Model:                           29
Date:                Thu, 30 Mar 2023   Pseudo R-squ.:                  0.9341
Time:                        03:40:45   Log-Likelihood:                -24.347
converged:                       True   LL-Null:                       -369.57
Covariance Type:            nonrobust   LLR p-value:                 9.766e-127
==============================================================================
===========
```

```
                            coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-----------
radius_mean              -613.2096    357.996     -1.713      0.087   -1314.868
88.449
texture_mean                7.2716     11.183      0.650      0.516     -14.647
29.191
perimeter_mean            154.2401    286.928      0.538      0.591    -408.129
716.609
area_mean                 452.3395    239.088      1.892      0.058     -16.265
920.944
smoothness_mean            18.4999     12.426      1.489      0.137      -5.854
42.854
compactness_mean          -43.5413     29.379     -1.482      0.138    -101.123
14.040
concavity_mean             54.3995     38.656      1.407      0.159     -21.365
130.164
concave points_mean        64.6109     31.262      2.067      0.039       3.339
125.883
symmetry_mean             -12.5495      7.061     -1.777      0.076     -26.388
1.289
fractal_dimension_mean    -30.7175     13.686     -2.245      0.025     -57.541
-3.894
radius_se                -231.9179    113.599     -2.042      0.041    -454.569
-9.267
texture_se                -11.9732      8.473     -1.413      0.158     -28.579
4.633
perimeter_se               30.5349     75.952      0.402      0.688    -118.329
179.398
area_se                   516.2740    241.068      2.142      0.032      43.789
988.759
smoothness_se              -9.5804     13.838     -0.692      0.489     -36.702
17.541
compactness_se             25.0799     22.410      1.119      0.263     -18.843
69.003
concavity_se              -91.5797     39.312     -2.330      0.020    -168.630
-14.529
concave points_se          59.1464     25.532      2.317      0.021       9.105
109.188
symmetry_se               -31.6735     16.432     -1.928      0.054     -63.879
0.532
fractal_dimension_se      -55.4965     33.922     -1.636      0.102    -121.982
10.989
radius_worst              236.2193    196.310      1.203      0.229    -148.541
620.980
texture_worst              18.2711     12.558      1.455      0.146      -6.343
42.885
```

```
perimeter_worst          -148.6346    130.087     -1.143      0.253    -403.601
106.331
area_worst                 3.0726    231.144      0.013      0.989    -449.961
456.106
smoothness_worst         -17.0149     12.842     -1.325      0.185     -42.186
8.156
compactness_worst        -16.4575     33.734     -0.488      0.626     -82.575
49.660
concavity_worst           31.4377     33.243      0.946      0.344     -33.717
96.593
concave points_worst     -25.6028     18.806     -1.361      0.173     -62.461
11.256
symmetry_worst            36.8654     15.522      2.375      0.018       6.443
67.288
fractal_dimension_worst   67.4195     34.380      1.961      0.050       0.036
134.803
==============================================================================
===========

Possibly complete quasi-separation: A fraction 0.79 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

[13]:
```python
# Observing the feature coefficients
l=zip(log_reg.params.index,log_reg.params)
print("Coefficient estimates:\n")
for a in l:
  print(str(a[0])+" => "+str(round(a[1],3)))
```

```
Coefficient estimates:

radius_mean => -613.21
texture_mean => 7.272
perimeter_mean => 154.24
area_mean => 452.34
smoothness_mean => 18.5
compactness_mean => -43.541
concavity_mean => 54.399
concave points_mean => 64.611
symmetry_mean => -12.55
fractal_dimension_mean => -30.718
radius_se => -231.918
texture_se => -11.973
perimeter_se => 30.535
area_se => 516.274
smoothness_se => -9.58
compactness_se => 25.08
concavity_se => -91.58
```

```
concave points_se => 59.146
symmetry_se => -31.673
fractal_dimension_se => -55.497
radius_worst => 236.219
texture_worst => 18.271
perimeter_worst => -148.635
area_worst => 3.073
smoothness_worst => -17.015
compactness_worst => -16.458
concavity_worst => 31.438
concave points_worst => -25.603
symmetry_worst => 36.865
fractal_dimension_worst => 67.419
```

[14]:
```
# Obtaining features of significance and respective regression coefficients,␣
↪based on p-values
l=sorted(zip(log_reg.pvalues.index,log_reg.pvalues),key=lambda x:
↪x[1],reverse=False)
imp_features=[a for a in l if a[1]<0.05]
imp_cols=[l[0] for l in imp_features]
```

[15]:
```
imp_features_coefs=[log_reg.params.loc[l] for l in imp_cols]
```

[16]:
```
imp_features_coefs
feat_coefs=list(zip(imp_cols,imp_features_coefs))
```

[17]:
```
feat_coefs
```

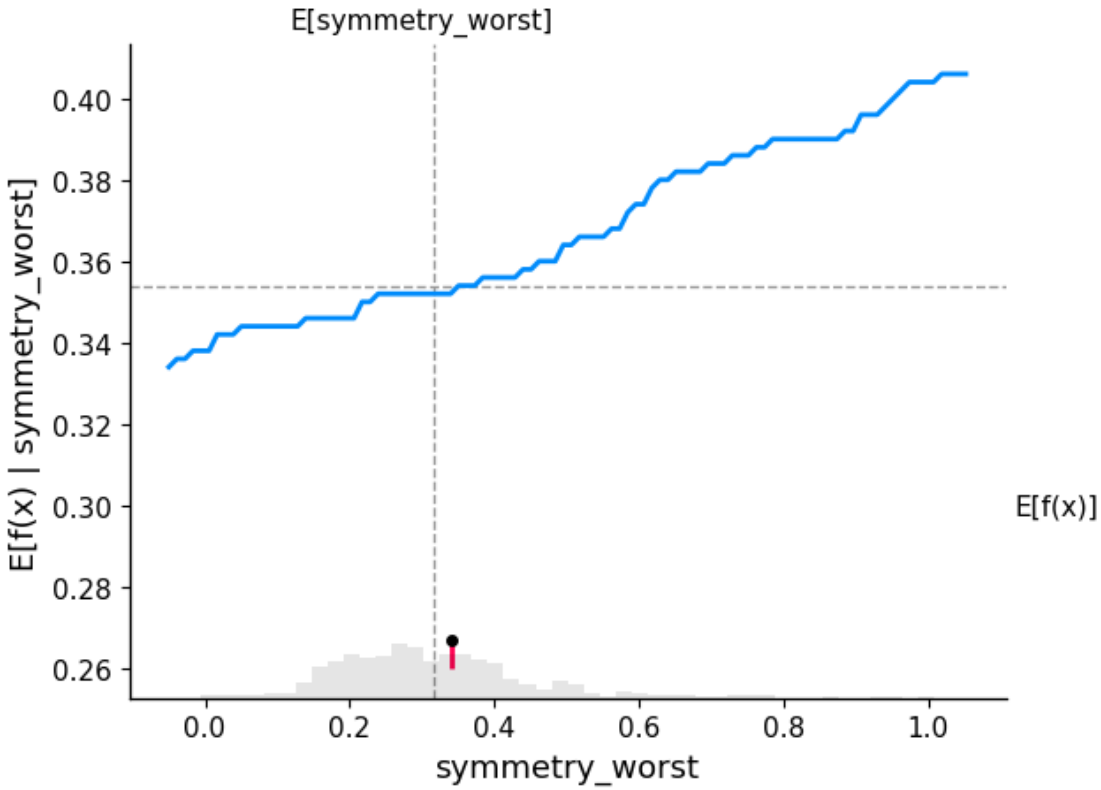
[17]:
```
[('symmetry_worst', 36.865386021058974),
 ('concavity_se', -91.57965788023972),
 ('concave points_se', 59.14639910510369),
 ('fractal_dimension_mean', -30.717544238358087),
 ('area_se', 516.2740280355208),
 ('concave points_mean', 64.61093578395679),
 ('radius_se', -231.91791410884727),
 ('fractal_dimension_worst', 67.41945488178855)]
```

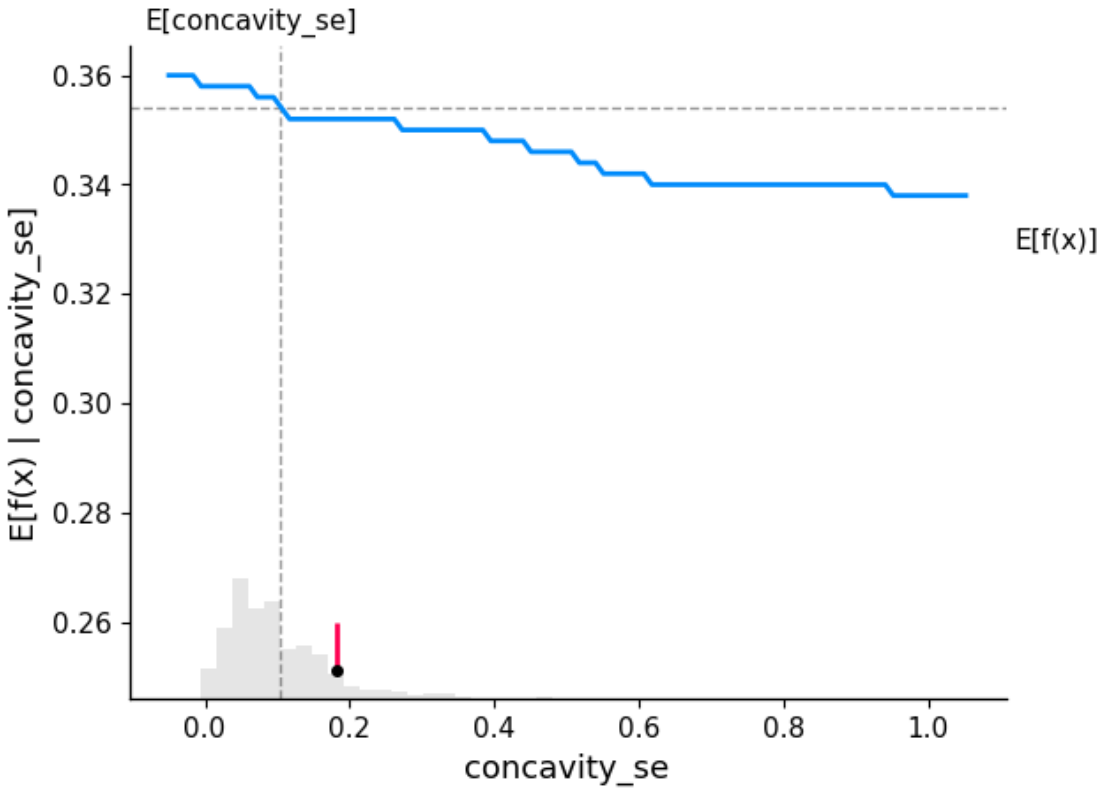
2.0.1 Feature Importance Analysis

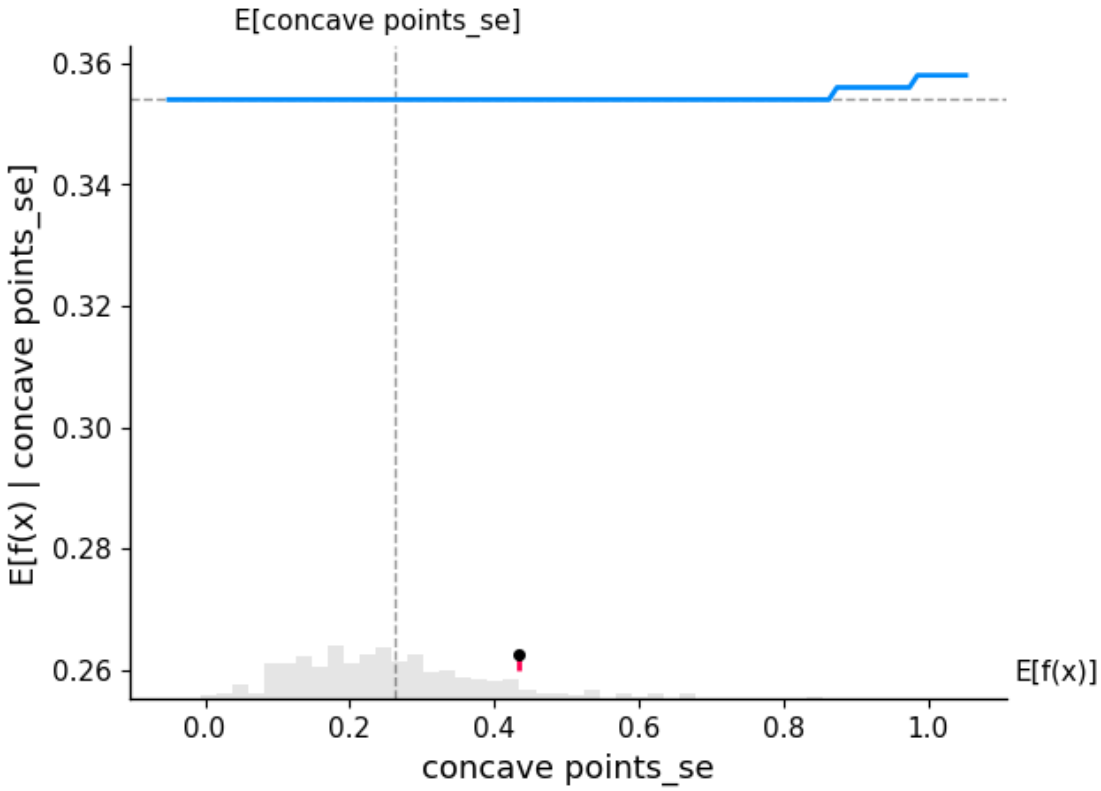
For each of the models, SHAP analysis is performed, by computing the shapley values of every feature in a sample.
Partial dependence plots and feature importance summary plots are constructed.

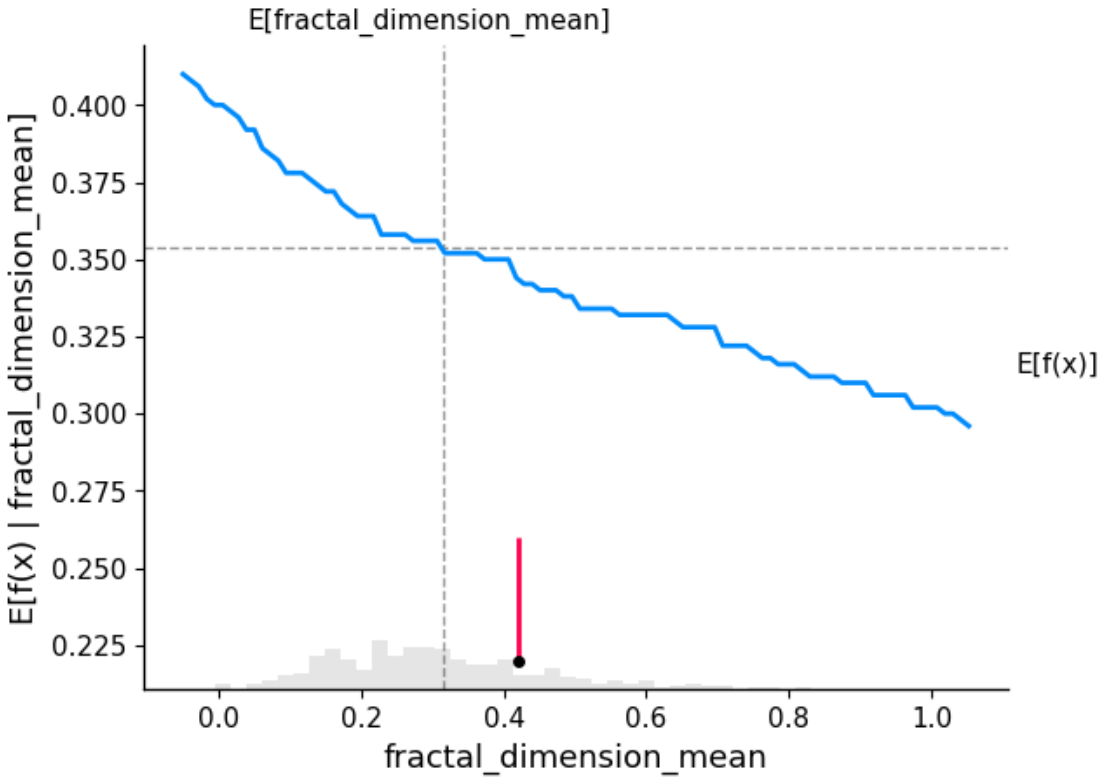
[18]:
```
#Calculating the shap values
explainer = shap.Explainer(model.predict, train_df_pred)
shap_lreg = explainer(test_df_pred)
```

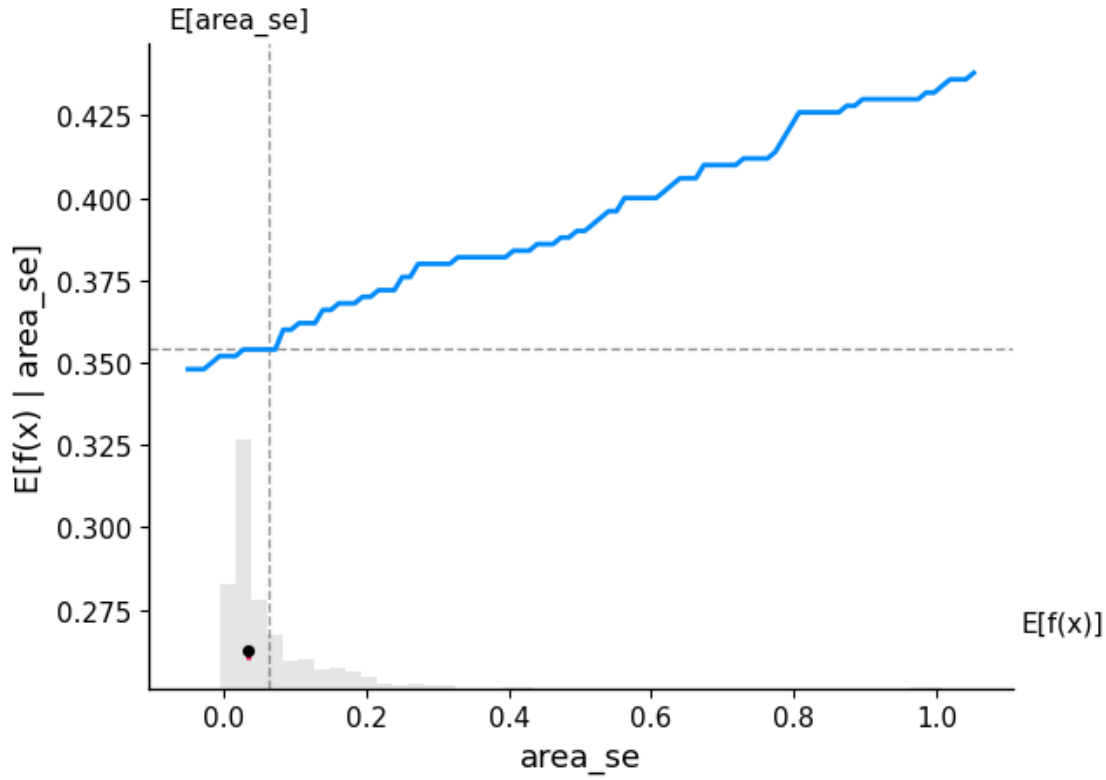
[19]:
```
# PDP for each of five previously estimated significant features
for feat in imp_features[:5]:
  pdp(feat[0],model,train_df_pred,0,shap_lreg)
  print("\n")
```



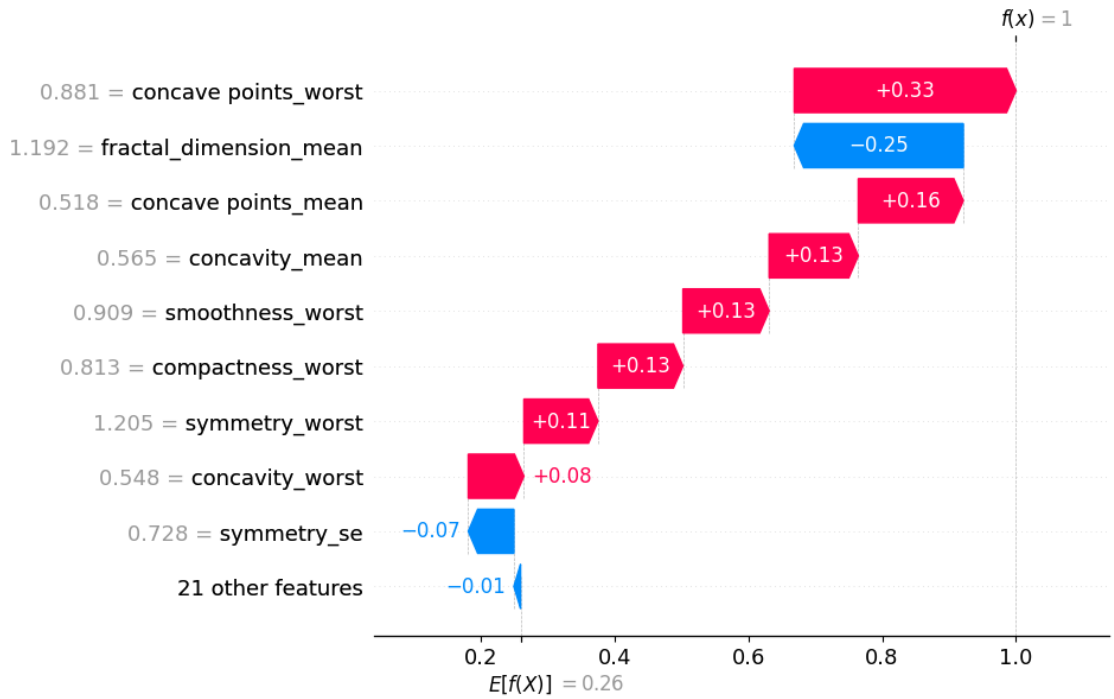

E[concavity_se]
0.36
0.34
0.32
0.30
0.28
0.26
E[f(x) | concavity_se]
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
concavity_se


E[concave points_se]
0.36
0.34
0.32
0.30
0.28
0.26
E[f(x) | concave points_se]
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
concave points_se

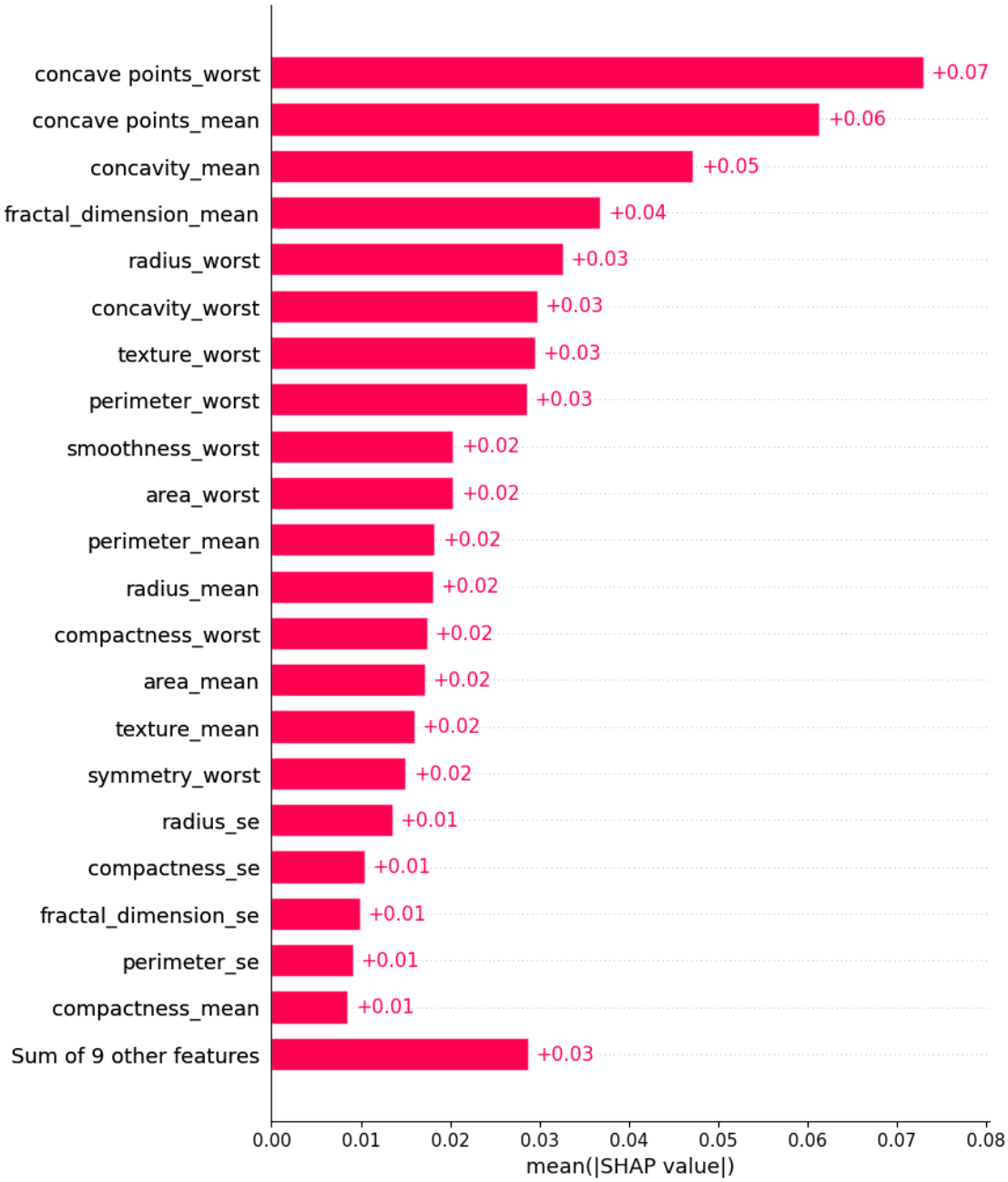

E[fractal_dimension_mean]
E[f(x) | fractal_dimension_mean]
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
E[f(x)]
0.0
0.2
0.4
0.6
0.8
1.0
fractal_dimension_mean



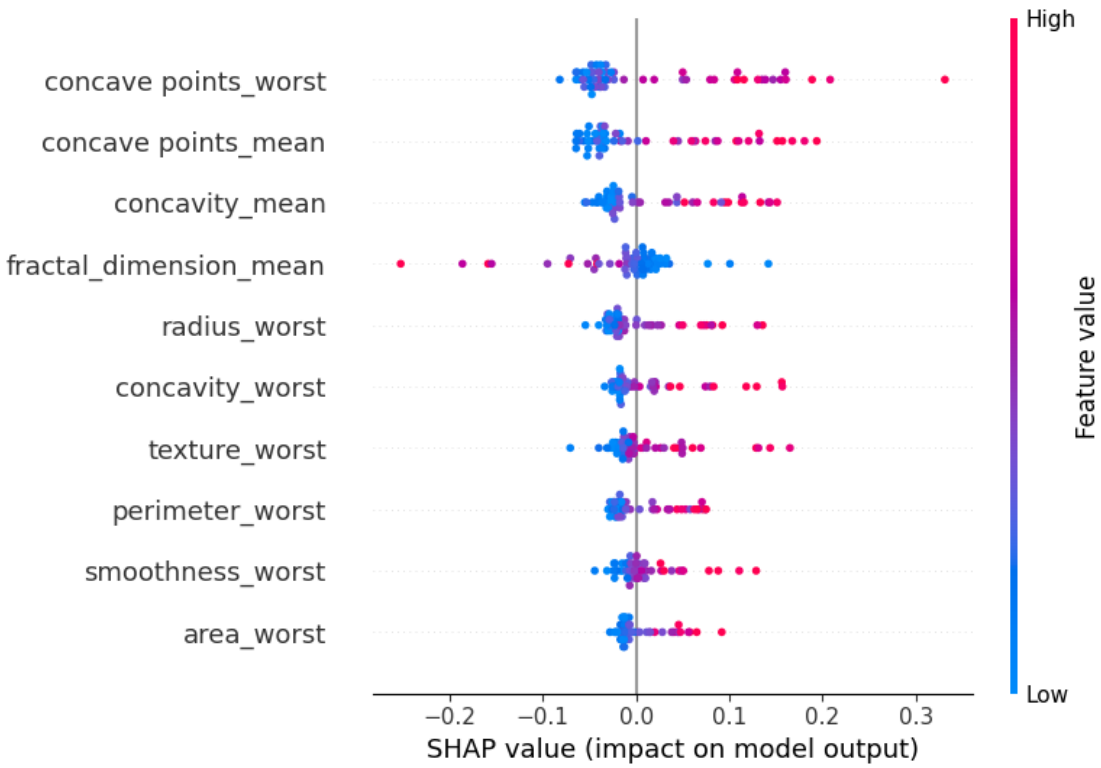
[20]:
```
#Visualization of shap values for the 20th sample
sample_feature_importance(shap_lreg,20)
```



[21]:
```
# Plots for overall feature importance of the model
shap.plots.bar(shap_lreg,max_display=22)
print("\n\n")
shap.summary_plot(shap_lreg,max_display=10)
```




```
No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will be
ignored
```



Regression summary obtained from the statsmodel summary assumed **symmetry_worst,concavity_se,concave points_se,fractal_dimension_mean, area_se, concave points_mean, radius_se and fractal_dimension_worst** to be most significant features.

SHAP analysis of a fitted logistic regression model gives **concave points_worst, concave points_mean, concavity_mean, radius_worst concavity worst, fractal_dimension_mean, perimeter_worst, texture_worst, area_worst, texture_mean, smoothness_worst, perimeter_mean** features to be significant.

3 Decision Tree

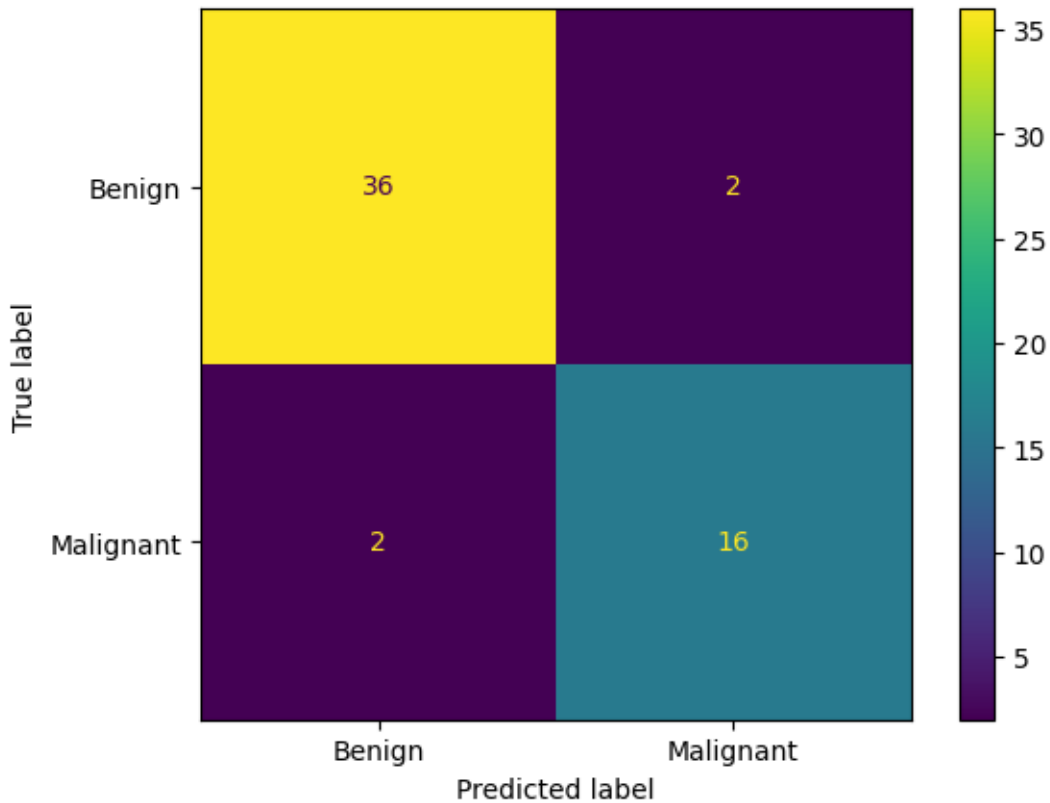
A decision tree is a tree-based model because it partitions the input space into a tree-like structure based on the values of input features. Each internal node of the tree represents a decision based on a feature, and each leaf node represents an output or prediction. Decision trees can handle both regression and classification problems and are widely used in machine learning for their ability to handle complex data structures and feature interactions.

[22]:
```
Xtrain,Xtest,ytrain,ytest=train_test_split(df_predictor,df_target,test_size=0.
↪1,shuffle=True)
```

[23]:
```
# Fitting a decision tree classifier and obtaining performance metrics
dt = DecisionTreeClassifier(max_depth=5
```

```
                                ,random_state=10)
dt.fit(pd.DataFrame(Xtrain,columns=df_predictor.columns), pd.
 ↪DataFrame(ytrain,columns=['diagnosis']))
y_pred = dt.predict(Xtest)
metric_scores(ytest,y_pred)
cf_display(dt,ytest,Xtest)
```

```
Accuracy: 0.9285714285714286
Precision:  0.8888888888888888
Recall  0.8888888888888888
```



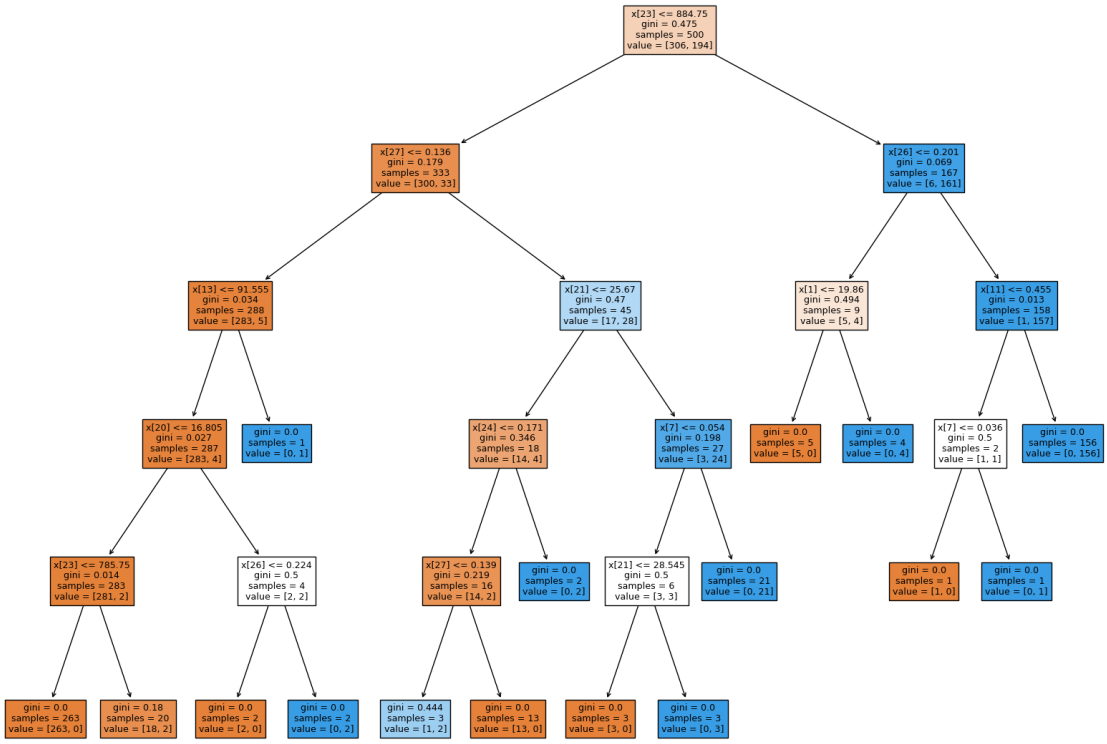
3.0.1 Interpreting the nodes of a Decision Tree

We can obtain the feature importances from interpreting the nodes by traversing the tree from root to node.
A rough estimate can be inferred by recording the number of times a given feature is used for performing a split.

[24]:
```
plt.figure(figsize=(20,15))
tree.plot_tree(dt,filled=True)
```

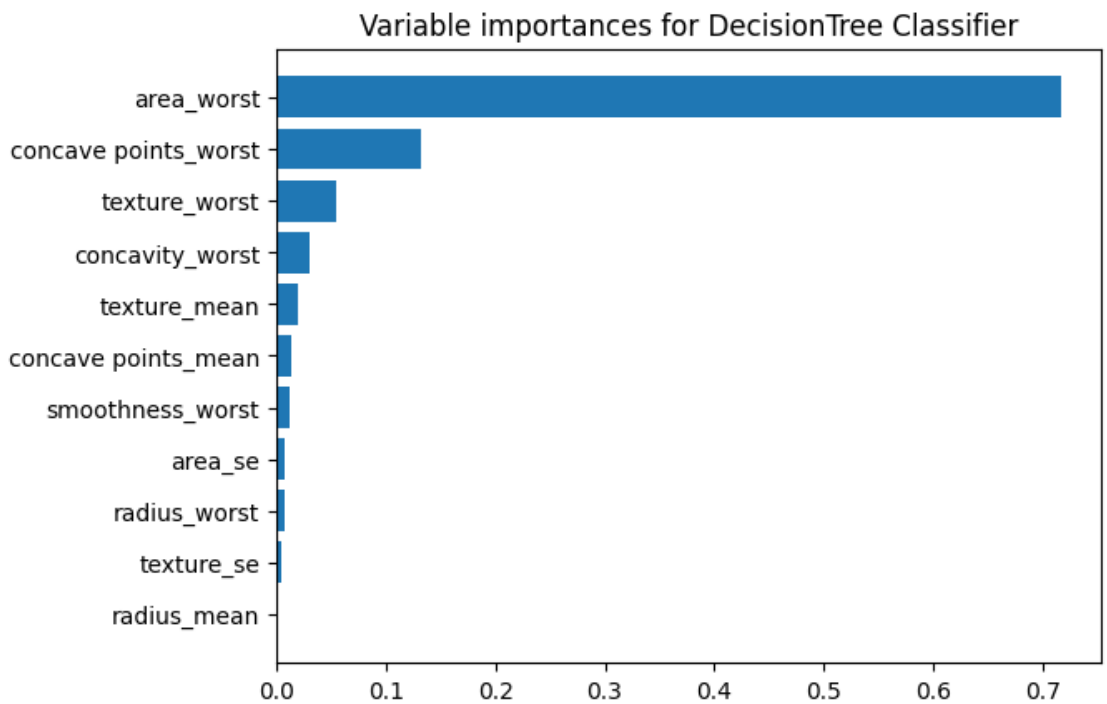
[24]:
```
[Text(0.6041666666666666, 0.9166666666666666, 'x[23] <= 884.75\ngini =
0.475\nsamples = 500\nvalue = [306, 194]'),
 Text(0.375, 0.75, 'x[27] <= 0.136\ngini = 0.179\nsamples = 333\nvalue = [300,
33]'),
 Text(0.20833333333333334, 0.5833333333333334, 'x[13] <= 91.555\ngini =
0.034\nsamples = 288\nvalue = [283, 5]'),
 Text(0.16666666666666666, 0.4166666666666667, 'x[20] <= 16.805\ngini =
0.027\nsamples = 287\nvalue = [283, 4]'),
 Text(0.08333333333333333, 0.25, 'x[23] <= 785.75\ngini = 0.014\nsamples =
283\nvalue = [281, 2]'),
 Text(0.041666666666666664, 0.08333333333333333, 'gini = 0.0\nsamples =
263\nvalue = [263, 0]'),
 Text(0.125, 0.08333333333333333, 'gini = 0.18\nsamples = 20\nvalue = [18, 2]'),
 Text(0.25, 0.25, 'x[26] <= 0.224\ngini = 0.5\nsamples = 4\nvalue = [2, 2]'),
 Text(0.20833333333333334, 0.08333333333333333, 'gini = 0.0\nsamples = 2\nvalue
= [2, 0]'),
 Text(0.2916666666666667, 0.08333333333333333, 'gini = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(0.25, 0.4166666666666667, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.5416666666666666, 0.5833333333333334, 'x[21] <= 25.67\ngini =
0.47\nsamples = 45\nvalue = [17, 28]'),
 Text(0.4583333333333333, 0.4166666666666667, 'x[24] <= 0.171\ngini =
0.346\nsamples = 18\nvalue = [14, 4]'),
 Text(0.4166666666666667, 0.25, 'x[27] <= 0.139\ngini = 0.219\nsamples =
16\nvalue = [14, 2]'),
 Text(0.375, 0.08333333333333333, 'gini = 0.444\nsamples = 3\nvalue = [1, 2]'),
 Text(0.4583333333333333, 0.08333333333333333, 'gini = 0.0\nsamples = 13\nvalue
= [13, 0]'),
 Text(0.5, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.625, 0.4166666666666667, 'x[7] <= 0.054\ngini = 0.198\nsamples =
27\nvalue = [3, 24]'),
 Text(0.5833333333333334, 0.25, 'x[21] <= 28.545\ngini = 0.5\nsamples = 6\nvalue
= [3, 3]'),
 Text(0.5416666666666666, 0.08333333333333333, 'gini = 0.0\nsamples = 3\nvalue =
[3, 0]'),
 Text(0.625, 0.08333333333333333, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
 Text(0.6666666666666666, 0.25, 'gini = 0.0\nsamples = 21\nvalue = [0, 21]'),
 Text(0.8333333333333334, 0.75, 'x[26] <= 0.201\ngini = 0.069\nsamples =
167\nvalue = [6, 161]'),
 Text(0.75, 0.5833333333333334, 'x[1] <= 19.86\ngini = 0.494\nsamples = 9\nvalue
= [5, 4]'),
 Text(0.7083333333333334, 0.4166666666666667, 'gini = 0.0\nsamples = 5\nvalue =
[5, 0]'),
 Text(0.7916666666666666, 0.4166666666666667, 'gini = 0.0\nsamples = 4\nvalue =
[0, 4]'),
 Text(0.9166666666666666, 0.5833333333333334, 'x[11] <= 0.455\ngini =
0.013\nsamples = 158\nvalue = [1, 157]'),
```

```
 Text(0.875, 0.4166666666666667, 'x[7] <= 0.036\ngini = 0.5\nsamples = 2\nvalue
= [1, 1]'),
 Text(0.8333333333333334, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.9166666666666666, 0.25, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.9583333333333334, 0.4166666666666667, 'gini = 0.0\nsamples = 156\nvalue
= [0, 156]')]
```



[25]:
```
# Obtaining and plotting feature importances from the model fit
l=list(zip(df_predictor.columns,dt.feature_importances_))
l=sorted(l,key=lambda x:x[1],reverse=True)
x=[x[0] for x in l[10::-1]]
y=[x[1] for x in l[10::-1]]
plt.barh(x, y)
plt.title("Variable importances for DecisionTree Classifier")
```

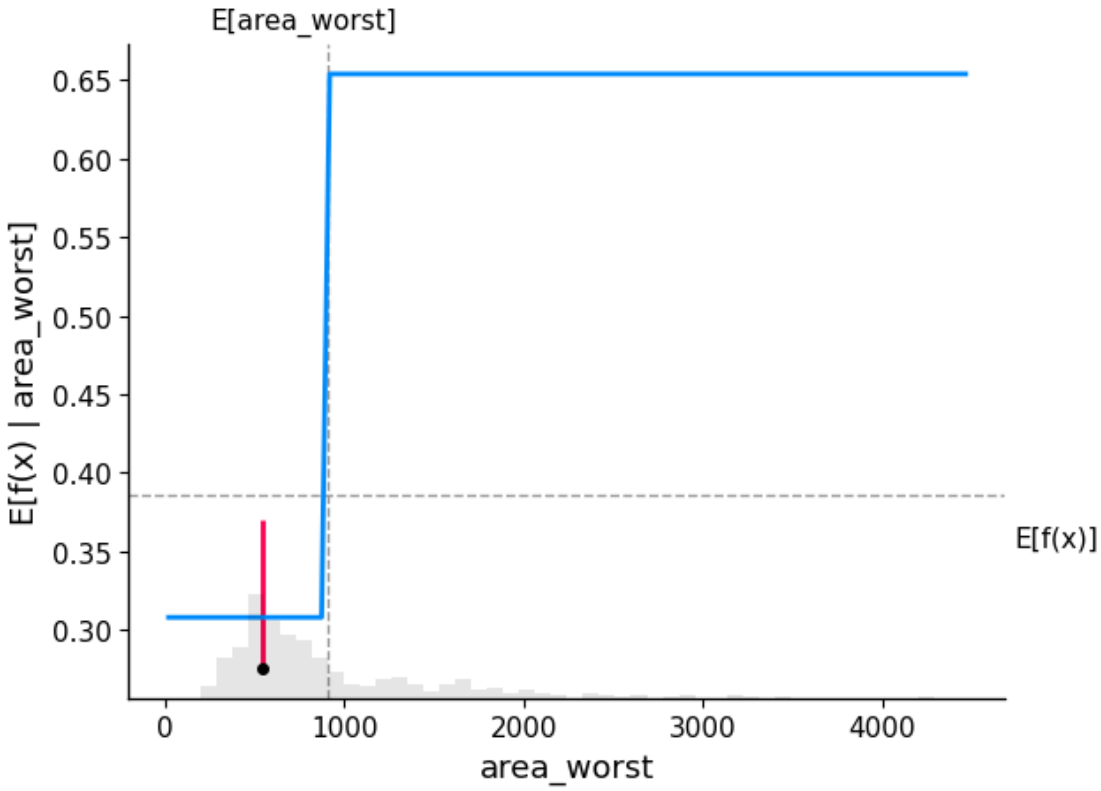
[25]: Text(0.5, 1.0, 'Variable importances for DecisionTree Classifier')

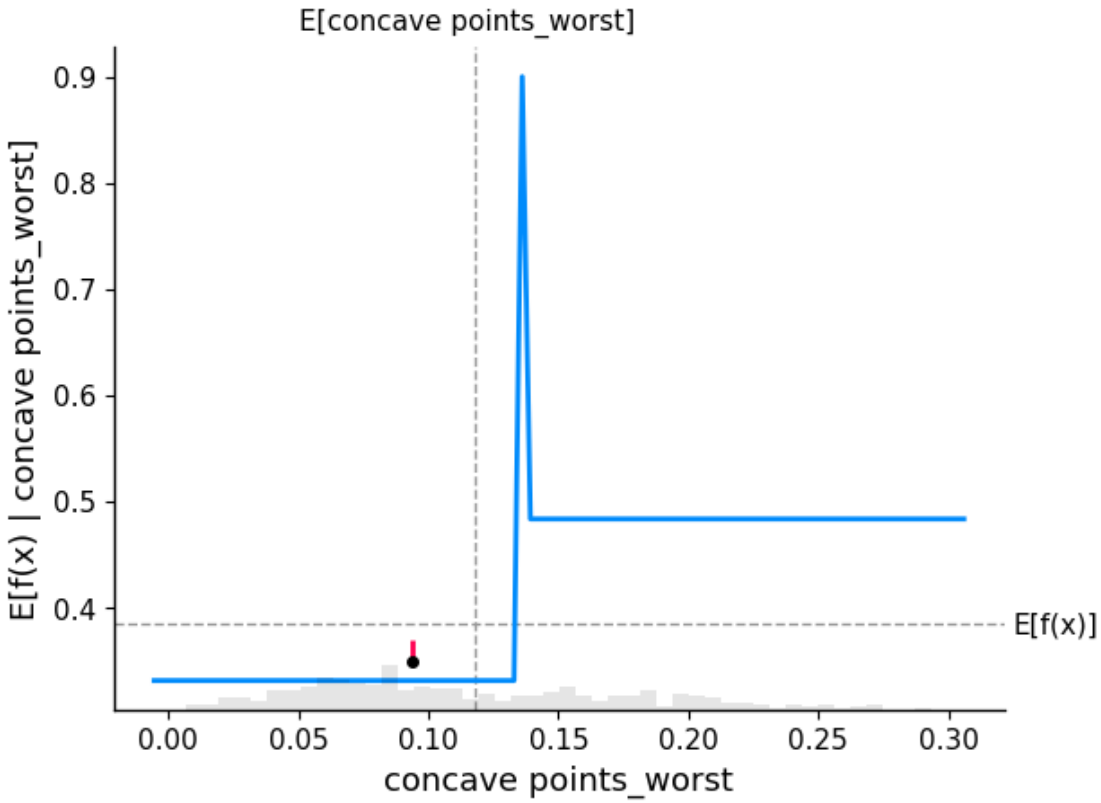


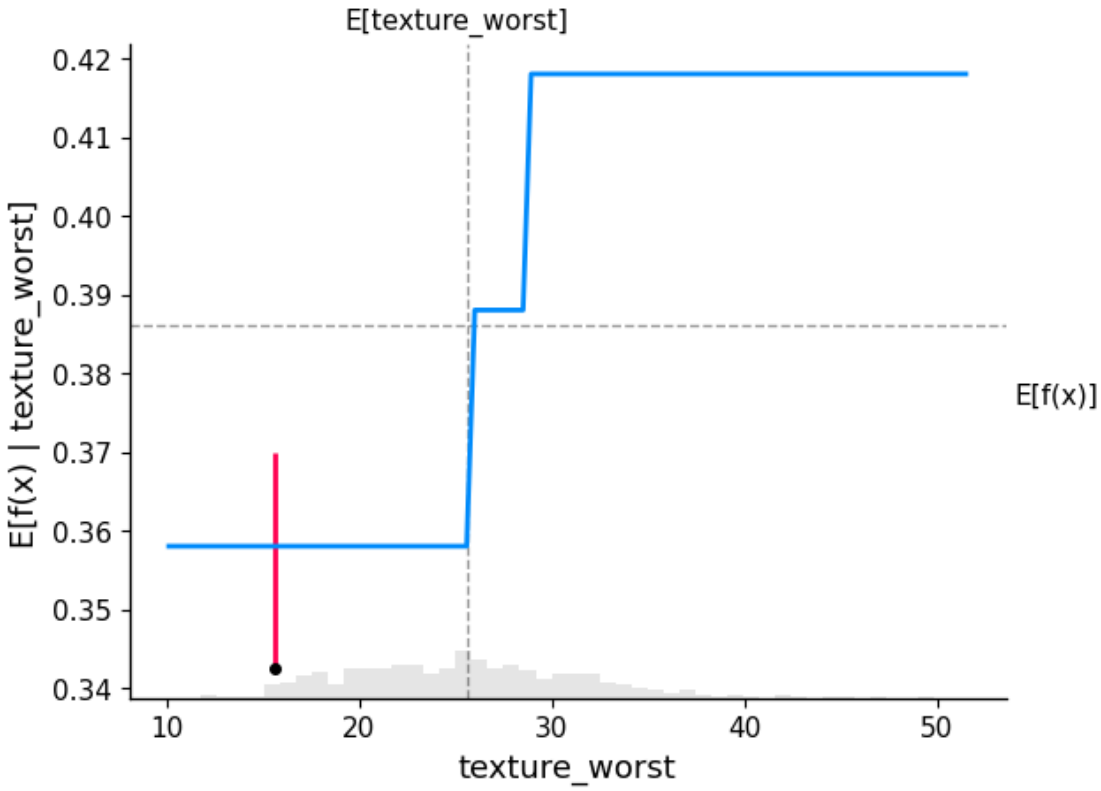
3.0.2 Feature Importance Analysis

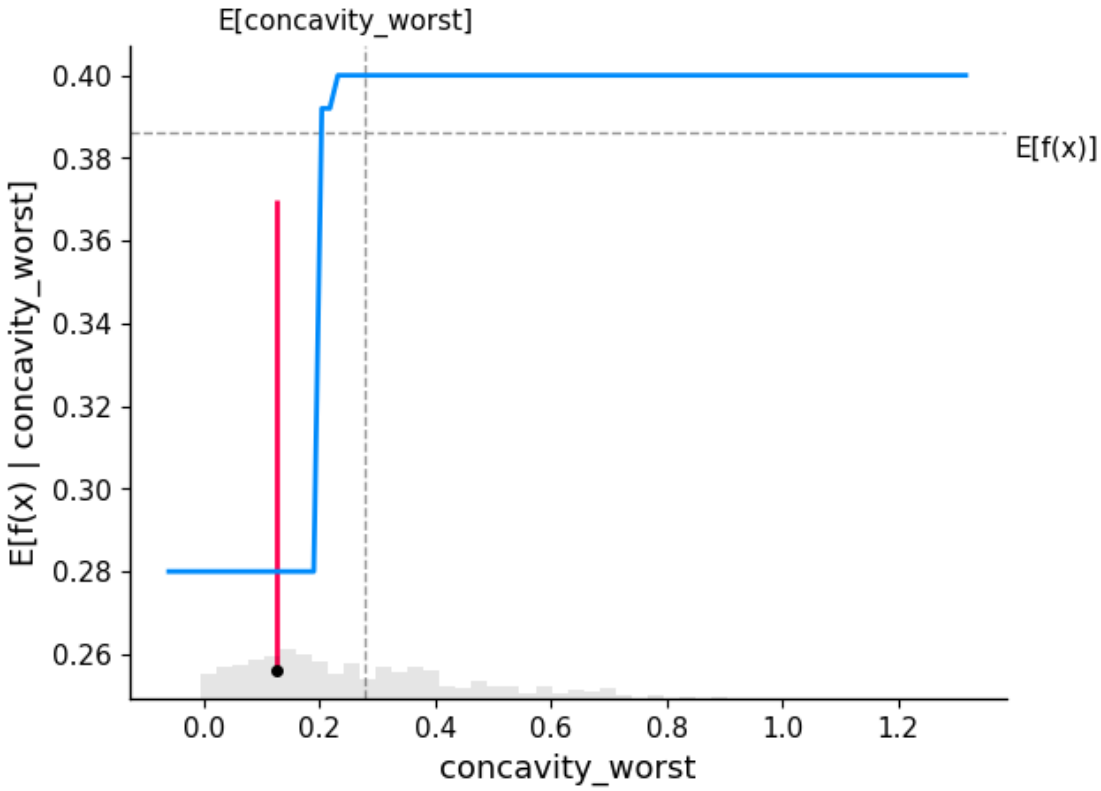
[26]:
```
explainer = shap.Explainer(dt.predict, Xtrain)
shap_dt = explainer(Xtest)
```

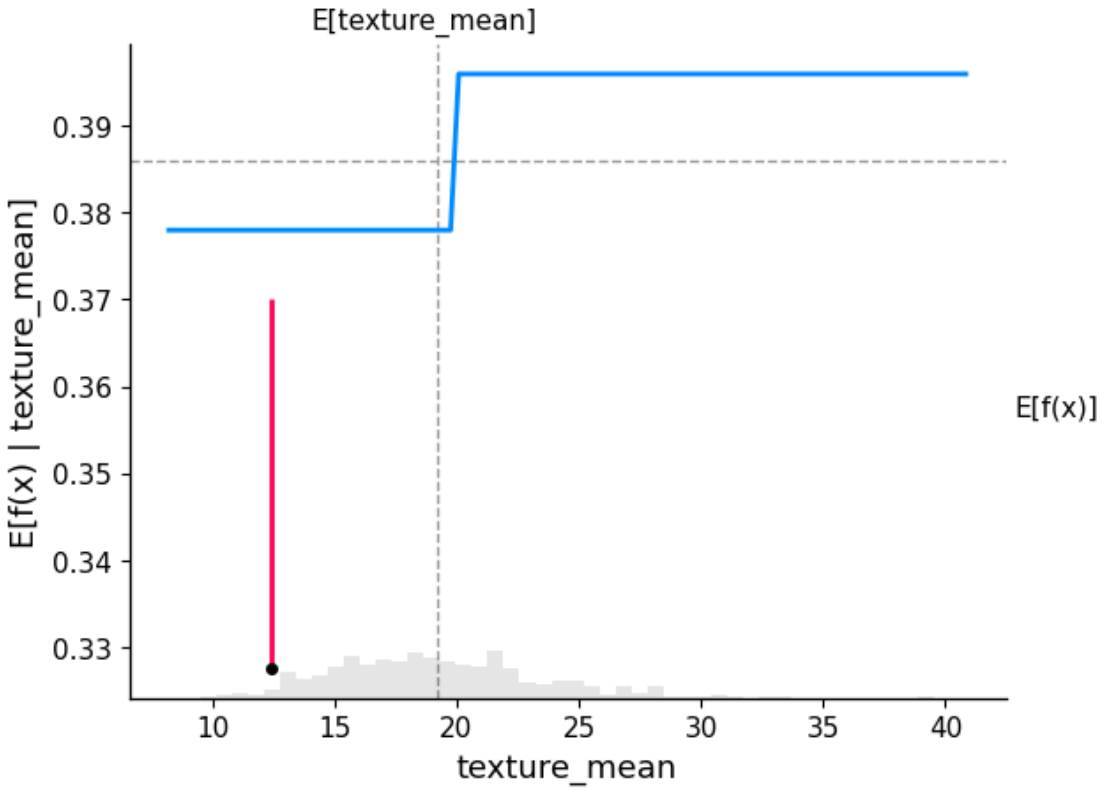
[27]:
```
for feat in x[10::-1][:5]:
  pdp(feat,dt,Xtrain,0,shap_dt)
  print("\n")
```


E[area_worst]
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
E[f(x) | area_worst]
E[f(x)]
0
1000
2000
3000
4000
area_worst

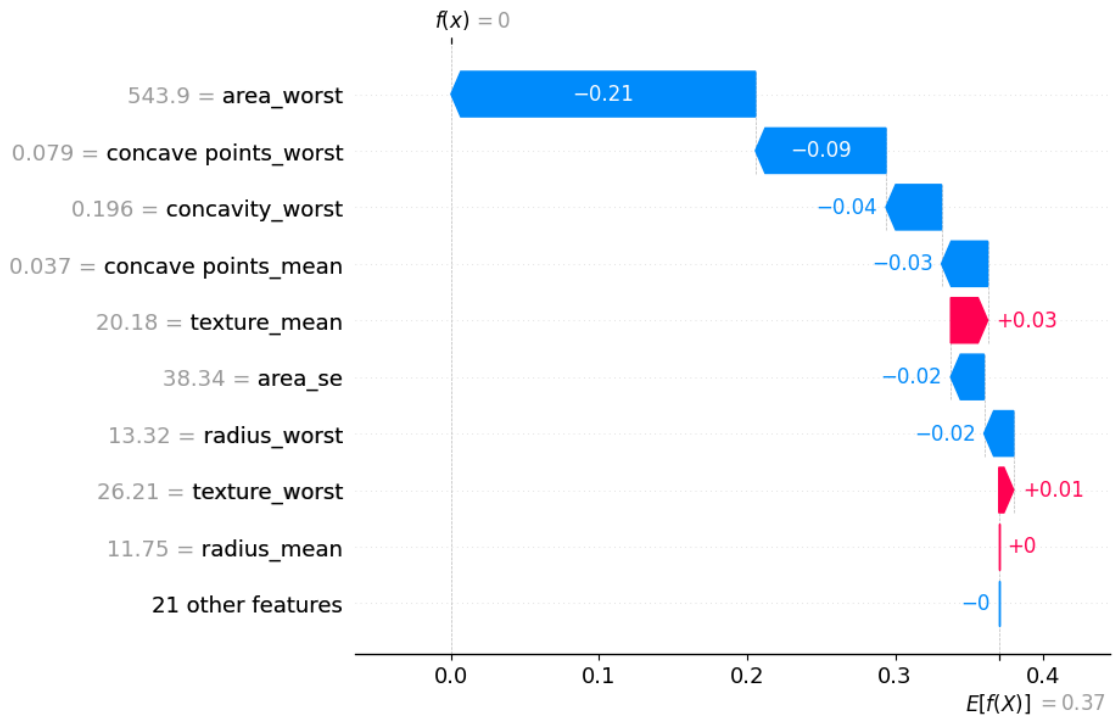

E[concave points_worst]
E[f(x) | concave points_worst]
0.9
0.8
0.7
0.6
0.5
0.4
E[f(x)]
0.00
0.05
0.10
0.15
0.20
0.25
0.30
concave points_worst


E[texture_worst]
0.42
0.41
0.40
0.39
0.38
0.37
0.36
0.35
0.34
E[f(x) | texture_worst]
E[f(x)]
10
20
30
40
50
texture_worst

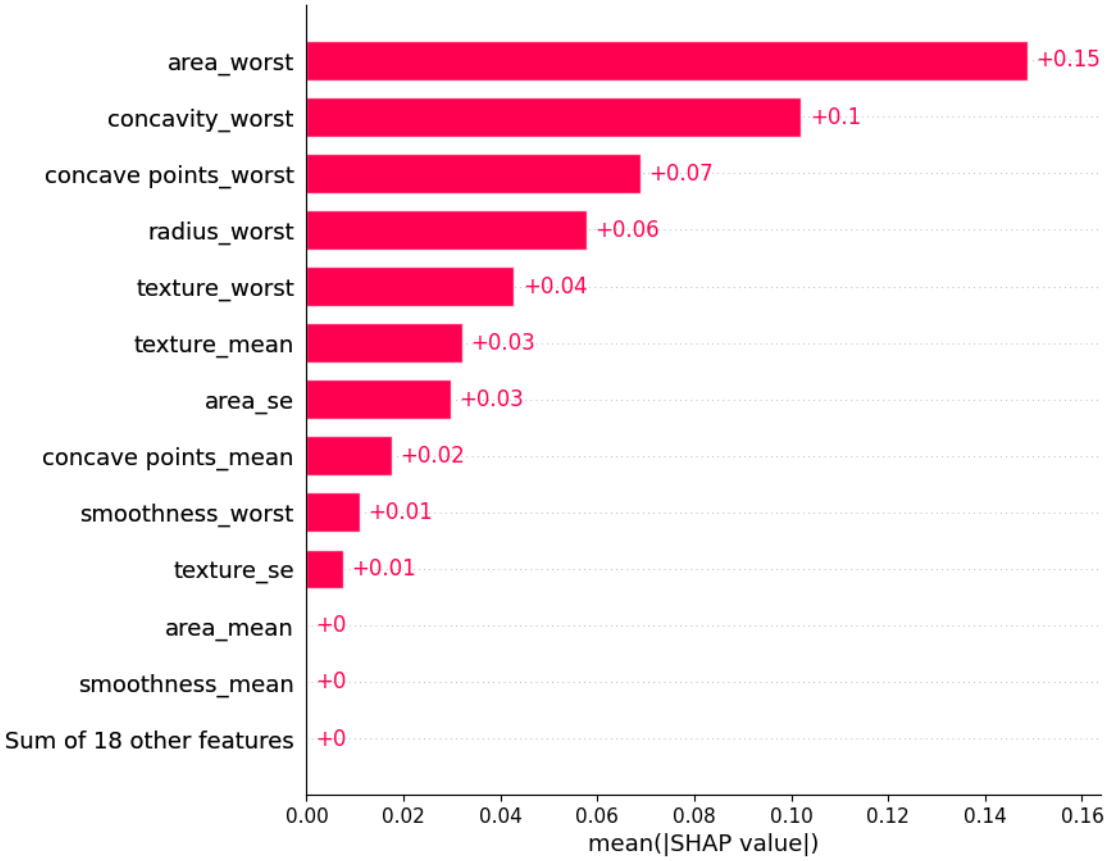

E[concavity_worst]
E[f(x)]
E[f(x) | concavity_worst]
0.40
0.38
0.36
0.34
0.32
0.30
0.28
0.26
0.0
0.2
0.4
0.6
0.8
1.0
1.2
concavity_worst



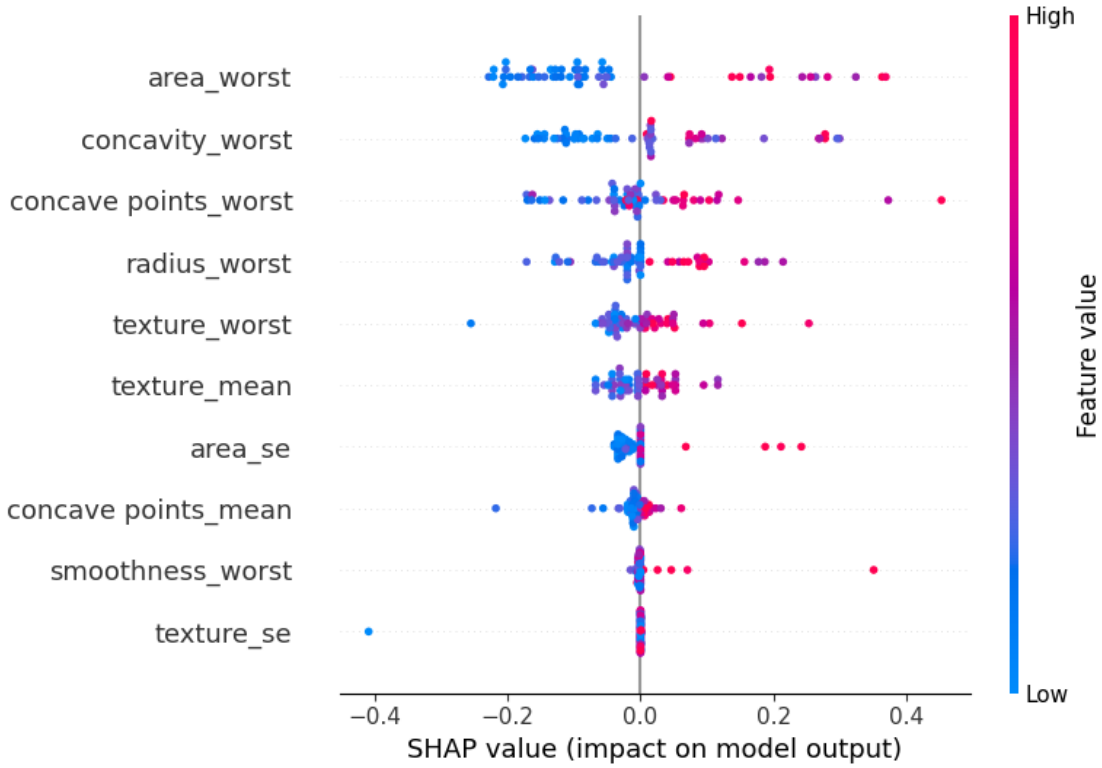
[28]:
```
sample_feature_importance(shap_dt,30)
```



[29]:
```
shap.plots.bar(shap_dt,max_display=13)
print("\n")
shap.summary_plot(shap_dt,max_display=10)
```




```
No data for colormapping provided via 'c'. Parameters 'vmin', 'vmax' will be
ignored
```



The SHAP analysis results in the **radius_worst, concave points worst, concavity_mean, texture_worst, texture mean, area worst, radius se, fractal_dimension_mean, symmetry_mean, concave points_mean and fractal_dimension_worst** features being considered as significant.

Feature importances calculated from the inbuilt sklearn method(feature_importances_) give **radius_worst, concave points_worst, texture_worst, texture_mean, concavity_mean, area_worst, radius_se and area_se** as significant features.

4 H2O AutoML

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.

[30]:
```
h2o.init()
```

```
Checking whether there is an H2O instance running at http://localhost:54321…
not found.
Attempting to start a local H2O server…
  Java Version: openjdk version "11.0.18" 2023-01-17; OpenJDK Runtime
```

```
Environment (build 11.0.18+10-post-Ubuntu-0ubuntu120.04.1); OpenJDK 64-Bit
Server VM (build 11.0.18+10-post-Ubuntu-0ubuntu120.04.1, mixed mode, sharing)
  Starting server from /usr/local/lib/python3.9/dist-
packages/h2o/backend/bin/h2o.jar
  Ice root: /tmp/tmpjug3f4ip
  JVM stdout: /tmp/tmpjug3f4ip/h2o_unknownUser_started_from_python.out
  JVM stderr: /tmp/tmpjug3f4ip/h2o_unknownUser_started_from_python.err
  Server is running at http://127.0.0.1:54321
Connecting to H2O server at http://127.0.0.1:54321 … successful.

--------------------------  ----------------------------------
H2O_cluster_uptime:         03 secs
H2O_cluster_timezone:       Etc/UTC
H2O_data_parsing_timezone:  UTC
H2O_cluster_version:        3.40.0.2
H2O_cluster_version_age:    20 days
H2O_cluster_name:           H2O_from_python_unknownUser_qpvg54
H2O_cluster_total_nodes:    1
H2O_cluster_free_memory:    3.172 Gb
H2O_cluster_total_cores:    2
H2O_cluster_allowed_cores:  2
H2O_cluster_status:         locked, healthy
H2O_connection_url:         http://127.0.0.1:54321
H2O_connection_proxy:       {"http": null, "https": null}
H2O_internal_security:      False
Python_version:             3.9.16 final
--------------------------  ----------------------------------
```

[31]:
```python
#Loading the dataset as a "H2O Frame"
df=h2o.H2OFrame(dataset)
```

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

[32]:
```python
# Categorizing the diagnosis columns; imparting train-test split
df['diagnosis']=df['diagnosis'].asfactor()
train,test=df.split_frame(ratios=[0.8],seed=1)
```

[33]:
```python
# Initializing the AutoML object and performin training
from h2o.automl import H2OAutoML
aml = H2OAutoML(max_models =25,exclude_algos=["DeepLearning"],sort_metric='auc',
                seed =1)
aml.train(training_frame=train,y='diagnosis',x=[l for l in df.columns if l!
 ↪='diagnosis'])
```

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

[33]:
```
Model Details
=============
H2OGradientBoostingEstimator : Gradient Boosting Machine
Model Key: GBM_2_AutoML_1_20230330_34123


Model Summary:
    number_of_trees    number_of_internal_trees    model_size_in_bytes
min_depth    max_depth    mean_depth    min_leaves    max_leaves    mean_leaves
--  -----------------  --------------------------  ---------------------
-----------  -----------  ------------  ------------  ------------
-------------
    79                 79                          21482                  4
7            6.96202       7             23            17.0253

ModelMetricsBinomial: gbm
** Reported on train data. **

MSE: 0.00010498357732962206
RMSE: 0.010246149390362317
LogLoss: 0.003121161887400136
Mean Per-Class Error: 0.0
AUC: 1.0
AUCPR: 1.0
Gini: 1.0

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.871381005907102
       0    1    Error    Rate
-----  ---  ---  -------  -----------
0      267  0    0        (0.0/267.0)
1      0    173  0        (0.0/173.0)
Total  267  173  0        (0.0/440.0)

Maximum Metrics: Maximum metrics at their respective thresholds
metric                        threshold    value    idx
---------------------------  -----------  -------  -----
max f1                        0.871381     1        157
max f2                        0.871381     1        157
max f0point5                  0.871381     1        157
max accuracy                  0.871381     1        157
max precision                 0.999763     1        0
max recall                    0.871381     1        157
max specificity               0.999763     1        0
max absolute_mcc              0.871381     1        157
max min_per_class_accuracy    0.871381     1        157
max mean_per_class_accuracy   0.871381     1        157
max tns                       0.999763     267      0
```

```
max fns                      0.999763     172      0
max fps                      0.000154244  267      399
max tps                      0.871381     173      157
max tnr                      0.999763     1        0
max fnr                      0.999763     0.99422  0
max fpr                      0.000154244  1        399
max tpr                      0.871381     1        157

Gains/Lift Table: Avg response rate: 39.32 %, avg score: 39.29 %
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        0.0113636                   0.999745           2.54335  2.54335
1                0.999752     1                          0.999752
0.0289017       0.0289017                  154.335  154.335            0.0289017
2        0.0204545                   0.999726           2.54335  2.54335
1                0.999738     1                          0.999746
0.0231214       0.0520231                  154.335  154.335            0.0520231
3        0.0318182                   0.999713           2.54335  2.54335
1                0.999718     1                          0.999736
0.0289017       0.0809249                  154.335  154.335            0.0809249
4        0.0409091                   0.999696           2.54335  2.54335
1                0.999704     1                          0.999729
0.0231214       0.104046                   154.335  154.335            0.104046
5        0.05                        0.999689           2.54335  2.54335
1                0.999692     1                          0.999722
0.0231214       0.127168                   154.335  154.335            0.127168
6        0.1                         0.999635           2.54335  2.54335
1                0.999671     1                          0.999696
0.127168        0.254335                   154.335  154.335            0.254335
7        0.15                        0.999561           2.54335  2.54335
1                0.999597     1                          0.999663
0.127168        0.381503                   154.335  154.335            0.381503
8        0.2                         0.999416           2.54335  2.54335
1                0.999504     1                          0.999624
0.127168        0.508671                   154.335  154.335            0.508671
9        0.3                         0.998164           2.54335  2.54335
1                0.998971     1                          0.999406
0.254335        0.763006                   154.335  154.335            0.763006
10       0.4                         0.022874           2.36994  2.5
0.931818         0.919326     0.982955                   0.979386
0.236994        1                          136.994  150                0.988764
```

```
11       0.5                        0.00320371         0        2
0                0.00721835   0.786364                    0.784952            0
1                         -100     100                0.82397
12       0.6                        0.00132076         0        1.66667
0                0.00211619   0.655303                    0.65448             0
1                         -100     66.6667            0.659176
13       0.7                        0.000579689        0        1.42857
0                0.000859929  0.561688                    0.561105            0
1                         -100     42.8571            0.494382
14       0.8                        0.000349227        0        1.25
0                0.000453368  0.491477                    0.491024            0
1                         -100     25                 0.329588
15       0.9                        0.000244564        0        1.11111
0                0.000288435  0.436869                    0.436498            0
1                         -100     11.1111            0.164794
16       1                          0.000154244        0        1
0                0.000195744  0.393182                    0.392868            0
1                         -100     0                  0

ModelMetricsBinomial: gbm
** Reported on cross-validation data. **

MSE: 0.02377438427112442
RMSE: 0.15418944280048624
LogLoss: 0.08713824830154107
Mean Per-Class Error: 0.02959450975298218
AUC: 0.9949340780671561
AUCPR: 0.9933137476110027
Gini: 0.9898681561343121

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.40509056893512435
       0    1    Error    Rate
-----  ---  ---  -------  ------------
0      262  5    0.0187   (5.0/267.0)
1      7    166  0.0405   (7.0/173.0)
Total  269  171  0.0273   (12.0/440.0)

Maximum Metrics: Maximum metrics at their respective thresholds
metric                       threshold    value     idx
---------------------------  -----------  --------  -----
max f1                       0.405091     0.965116  159
max f2                       0.104887     0.967008  175
max f0point5                 0.635961     0.977081  152
max accuracy                 0.405091     0.972727  159
max precision                0.999999     1         0
max recall                   0.00882449   1         218
max specificity              0.999999     1         0
```

```
max absolute_mcc             0.405091     0.942772  159
max min_per_class_accuracy   0.250509     0.966292  165
max mean_per_class_accuracy  0.405091     0.970405  159
max tns                      0.999999     267       0
max fns                      0.999999     170       0
max fps                      5.46454e-07  267       399
max tps                      0.00882449   173       218
max tnr                      0.999999     1         0
max fnr                      0.999999     0.982659  0
max fpr                      5.46454e-07  1         399
max tpr                      0.00882449   1         218

Gains/Lift Table: Avg response rate: 39.32 %, avg score: 38.25 %
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        0.0113636                   0.999998           2.54335   2.54335
1                0.999999     1                          0.999999
0.0289017       0.0289017                  154.335   154.335
0.0289017
2        0.0204545                   0.999997           2.54335   2.54335
1                0.999998     1                          0.999998
0.0231214       0.0520231                  154.335   154.335
0.0520231
3        0.0318182                   0.999996           2.54335   2.54335
1                0.999996     1                          0.999998
0.0289017       0.0809249                  154.335   154.335
0.0809249
4        0.0409091                   0.999967           2.54335   2.54335
1                0.999988     1                          0.999995
0.0231214       0.104046                   154.335   154.335            0.104046
5        0.05                        0.999943           2.54335   2.54335
1                0.999949     1                          0.999987
0.0231214       0.127168                   154.335   154.335            0.127168
6        0.1                         0.999507           2.54335   2.54335
1                0.999863     1                          0.999925
0.127168        0.254335                   154.335   154.335            0.254335
7        0.15                        0.997276           2.54335   2.54335
1                0.998353     1                          0.999401
0.127168        0.381503                   154.335   154.335            0.381503
8        0.2                         0.994823           2.54335   2.54335
1                0.995713     1                          0.998479
```

```
0.127168        0.508671                    154.335   154.335            0.508671
9        0.3                         0.985304           2.54335   2.54335
1                0.992167     1                           0.996375
0.254335        0.763006                    154.335   154.335            0.763006
10       0.4                         0.254481           2.02312   2.41329
0.795455         0.761293     0.948864                    0.937605
0.202312        0.965318                    102.312   141.329            0.93161
11       0.5                         0.0126341          0.231214  1.97688
0.0909091        0.0617928    0.777273                    0.762442
0.0231214       0.988439                    -76.8786  97.6879            0.804919
12       0.6                         0.00526646         0.115607  1.66667
0.0454545        0.0072003    0.655303                    0.636569
0.0115607       1                           -88.4393  66.6667            0.659176
13       0.7                         0.00259611         0         1.42857
0                0.00389672   0.561688                    0.546187            0
1                         -100      42.8571            0.494382
14       0.8                         0.000793232        0         1.25
0                0.0016703    0.491477                    0.478122            0
1                         -100      25                 0.329588
15       0.9                         6.9248e-05         0         1.11111
0                0.000260396  0.436869                    0.425027            0
1                         -100      11.1111            0.164794
16       1                           3e-07              0         1
0                1.44018e-05  0.393182                    0.382525            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                 0.975      0.0259131   0.931818      0.977273
0.977273      1             0.988636
auc                      0.99164    0.0102964   0.984496      0.998326
0.97703       1             0.998347
err                      0.025      0.0259131   0.0681818     0.0227273
0.0227273     0             0.0113636
err_count                2.2        2.28035     6             2             2
0             1
f0point5                 0.965235   0.0360015   0.909091      0.952381
0.988372      1             0.976331
f1                       0.971907   0.0247772   0.933333      0.969697
0.971429      1             0.985075
f2                       0.979118   0.0207199   0.958904      0.987654
0.955056      1             0.993976
lift_top_group           2.58842    0.369322    2.04651       2.75
2.44444       3.03448       2.66667
```

```
logloss                   0.0924407  0.0775578   0.198483      0.0661143
0.14392       0.00602794    0.0476581
max_per_class_error       0.0441126  0.0427452   0.111111      0.0357143
0.0555556     0             0.0181818
mcc                       0.949987   0.0500608   0.867428      0.952661
0.953659      1             0.976187
mean_per_class_accuracy   0.975618   0.0260498   0.932817      0.982143
0.972222      1             0.990909
mean_per_class_error      0.0243819  0.0260498   0.0671835     0.0178571
0.0277778     0             0.00909091
mse                       0.0246855  0.0221945   0.0606817     0.0209884
0.0253275     0.000533405   0.0158964
pr_auc                    0.99159    0.00924122  0.984724      0.997142
0.978885      1             0.997201
precision                 0.961076   0.0449092   0.893617      0.941176      1
1             0.970588
r2                        0.898287   0.0881186   0.757148      0.9093
0.895227      0.997586      0.932175
recall                    0.984238   0.0244183   0.976744      1
0.944444      1             1
rmse                      0.139907   0.0799345   0.246336      0.144874
0.159146      0.0230956     0.126081
specificity               0.966999   0.0461148   0.888889      0.964286      1
1             0.981818

Scoring History:
    timestamp            duration    number_of_trees    training_rmse
training_logloss    training_auc    training_pr_auc    training_lift
training_classification_error
--  -------------------  ----------  -----------------  ---------------
------------------  --------------  -----------------  ---------------
-------------------------------
    2023-03-30 03:41:53  7.539 sec   0                  0.488457         0.67015
0.5             0.393182           1                 0.606818
    2023-03-30 03:41:53  7.591 sec   5                  0.321495
0.380209            0.997056        0.995924           2.54335
0.0204545
    2023-03-30 03:41:54  7.672 sec   10                 0.227199
0.242659            0.997987        0.997131           2.54335
0.0181818
    2023-03-30 03:41:54  7.748 sec   15                 0.169515         0.16216
0.998853        0.998309           2.54335          0.0136364
    2023-03-30 03:41:54  7.812 sec   20                 0.134876
0.114412            0.999372        0.999056           2.54335
0.0113636
    2023-03-30 03:41:54  7.876 sec   25                 0.107289
0.0805677           0.999805        0.999702           2.54335
```

```
0.00681818
    2023-03-30 03:41:54  7.933 sec   30                 0.0876514
0.0577178           0.999935        0.9999             2.54335
0.00454545
    2023-03-30 03:41:54  7.998 sec   35                 0.0715472
0.042069            1               1                  2.54335          0
    2023-03-30 03:41:54  8.083 sec   40                 0.0592301
0.0308401           1               1                  2.54335          0
    2023-03-30 03:41:54  8.154 sec   45                 0.0464237
0.0218777           1               1                  2.54335          0
    2023-03-30 03:41:54  8.208 sec   50                 0.0379476
0.0160455           1               1                  2.54335          0
    2023-03-30 03:41:54  8.269 sec   55                 0.0289267
0.0116662           1               1                  2.54335          0
    2023-03-30 03:41:54  8.324 sec   60                 0.0239101
0.00875218          1               1                  2.54335          0
    2023-03-30 03:41:54  8.380 sec   65                 0.0204718
0.00670105          1               1                  2.54335          0
    2023-03-30 03:41:54  8.438 sec   70                 0.016236
0.00513545          1               1                  2.54335          0
    2023-03-30 03:41:54  8.496 sec   75                 0.0127319
0.00389243          1               1                  2.54335          0
    2023-03-30 03:41:54  8.538 sec   79                 0.0102461
0.00312116          1               1                  2.54335          0

Variable Importances:
variable                 relative_importance    scaled_importance
percentage
-----------------------  ---------------------  ---------------------
----------------------
concave points_mean      149.2587890625         1.0
0.2819895185708784
radius_worst             139.19215393066406     0.9325558300783167
0.2629709695642504
perimeter_worst          103.71368408203125     0.6948581368873535
0.19594271149592235
concave points_worst     46.58530044555664      0.3121109365697032
0.08801201274399656
area_worst               26.539287567138672     0.17780720139720363
0.050139767120432666
texture_worst            15.092900276184082     0.10111900525914186
0.028514499611391533
texture_mean             9.081515312194824      0.06084409078511329
0.017157395868376914
concavity_worst          8.238534927368164      0.05519631359141199
0.0155647818965295
compactness_worst        5.458521842956543      0.03657085708146047
```

```
0.010312598382125438
smoothness_worst         3.304093360900879      0.02213667537874327
0.006242310432811632
---                      ---                    ---                    ---
smoothness_se            0.7584584355354309     0.005081499322749009
0.0014329295476402375
texture_se               0.6773902177810669     0.004538360668981572
0.0012797701401471232
smoothness_mean          0.6361203789710999     0.004261862118583405
0.0012018004470347983
compactness_mean         0.5840967893600464     0.00391331587927773
0.0011035140608133007
concavity_se             0.5729116201400757     0.0038383777849100536
0.0010823823036599407
fractal_dimension_se     0.5426474809646606     0.0036356149234028335
0.001025205301959466
fractal_dimension_worst  0.5044553279876709     0.003379736169348375
0.000953050175291133
radius_mean              0.4351959526538849     0.002915714078798085
0.0008222008093706042
concave points_se        0.1939942091703415     0.0012997171582914905
0.00036650661574492754
symmetry_mean            0.056754227727651596   0.0003802404406743952
0.00010722381880695135
[30 rows x 4 columns]


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

[34]:
```
#Invoking the explain method for multi-model inferences
explain_model = aml.explain(frame = test, figsize = (8,6))
```

5 Leaderboard

Leaderboard shows models with their metrics. When provided with H2OAutoML object, the leaderboard shows 5-fold cross-validated metrics by default (depending on the H2OAutoML settings), otherwise it shows metrics computed on the frame. At most 20 models are shown by default.

```
model_id                                                    auc    logloss    ␣
↪aucpr    mean_per_class_error      rmse        mse    training_time_ms    ␣
↪predict_time_per_row_ms  algo
```

```
-----------------------------------------------------  --------  ---------
--------  ----------------------  --------  ---------  ------------------
-------------------------  ---------------
StackedEnsemble_BestOfFamily_1_AutoML_1_20230330_34123  0.998335  0.0616396  0.997053  0.0128205  0.13187  0.0173898  6085  0.203598  StackedEnsemble
GLM_1_AutoML_1_20230330_34123  0.998335  0.0621704  0.996921  0.019314  0.131881  0.0173926  328  0.048644  GLM
StackedEnsemble_AllModels_1_AutoML_1_20230330_34123  0.998002  0.0628689  0.996387  0.019314  0.135626  0.0183945  5874  0.367201  StackedEnsemble
GBM_4_AutoML_1_20230330_34123  0.997336  0.0848648  0.995244  0.025641  0.155039  0.0240372  792  0.088599  GBM
XGBoost_grid_1_AutoML_1_20230330_34123_model_8  0.996004  0.0986087  0.992904  0.0258075  0.172385  0.0297167  326  0.081996  XGBoost
GBM_grid_1_AutoML_1_20230330_34123_model_5  0.996004  0.0958196  0.993038  0.025641  0.162333  0.0263521  1211  0.115128  GBM
GBM_grid_1_AutoML_1_20230330_34123_model_4  0.995338  0.0905848  0.991399  0.032301  0.167283  0.0279835  419  0.106672  GBM
GBM_grid_1_AutoML_1_20230330_34123_model_2  0.995005  0.116364  0.991051  0.0384615  0.175709  0.0308738  905  0.094684  GBM
XGBoost_grid_1_AutoML_1_20230330_34123_model_7  0.994672  0.104563  0.989525  0.0258075  0.172499  0.0297559  381  0.049432  XGBoost
GBM_3_AutoML_1_20230330_34123  0.994339  0.093541  0.990498  0.0384615  0.160691  0.0258215  950  0.083654  GBM
XGBoost_grid_1_AutoML_1_20230330_34123_model_1  0.994339  0.10279  0.989351  0.0387945  0.173297  0.0300318  268  0.046579  XGBoost
GBM_2_AutoML_1_20230330_34123  0.994339  0.109833  0.99002  0.044955  0.175989  0.030972  1030  0.078596  GBM
XGBoost_2_AutoML_1_20230330_34123  0.994339  0.0976623  0.988686  0.0194805  0.173391  0.0300644  309  0.054377  XGBoost
XGBoost_grid_1_AutoML_1_20230330_34123_model_6  0.994339  0.132771  0.988231  0.0258075  0.186416  0.034751  200  0.054117  XGBoost
```

```
GBM_grid_1_AutoML_1_20230330_34123_model_3              0.994006  0.11189    0.
↪989079                0.0387945  0.184857  0.0341719                  574       ␣
↪           0.085927  GBM
GBM_grid_1_AutoML_1_20230330_34123_model_1              0.994006  0.100963   0.
↪988375                0.025974   0.188452  0.035514                  1035       ␣
↪           0.088671  GBM
XGBoost_grid_1_AutoML_1_20230330_34123_model_4          0.993673  0.112138   0.
↪988117                0.044955   0.178924  0.0320139                  235       ␣
↪           0.047419  XGBoost
XGBoost_grid_1_AutoML_1_20230330_34123_model_2          0.993673  0.0904661  0.
↪989755                0.0258075  0.156355  0.0244467                  281       ␣
↪           0.052576  XGBoost
GBM_5_AutoML_1_20230330_34123                           0.99334   0.15535    0.
↪987441                0.0324675  0.211359  0.0446726                  804       ␣
↪           0.074495  GBM
DRF_1_AutoML_1_20230330_34123                           0.99334   0.112207   0.
↪987215                0.0387945  0.187529  0.0351672                  240       ␣
↪           0.055003  DRF
[20 rows x 10 columns]
```

6 Confusion Matrix

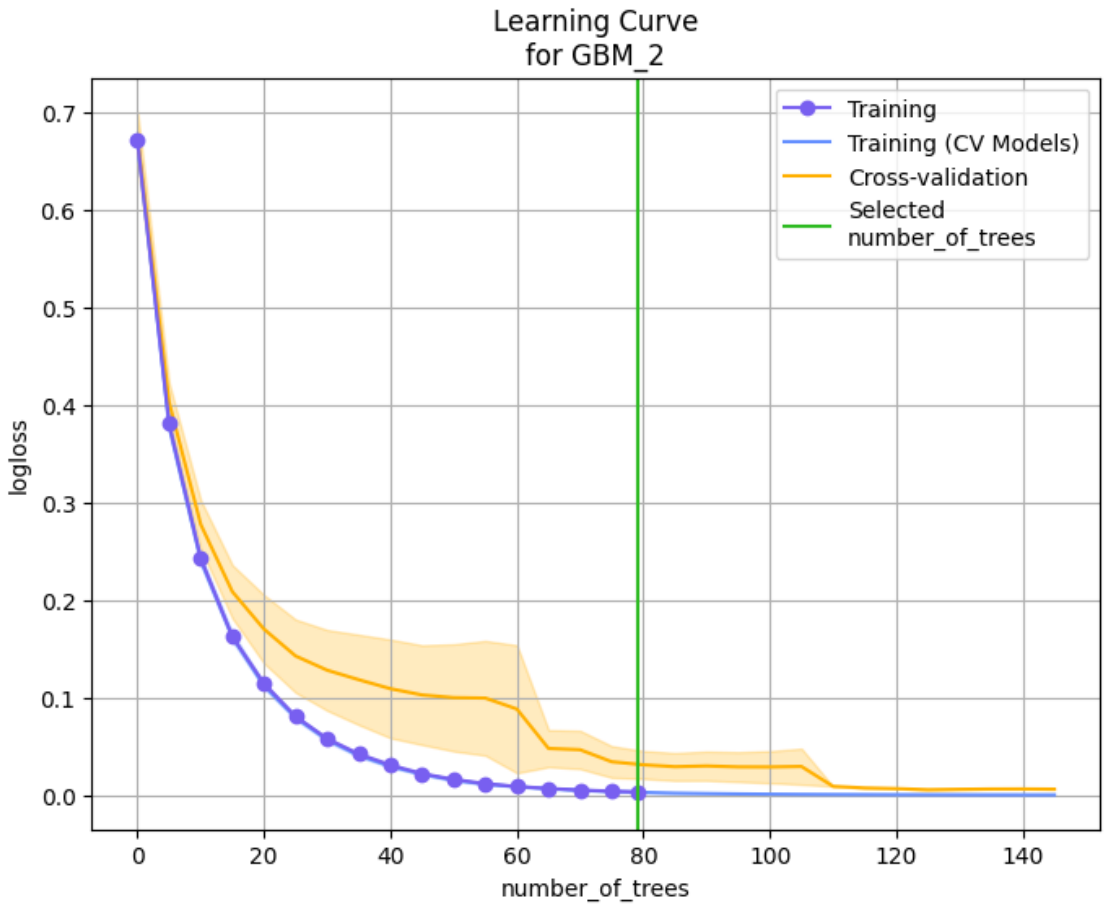
Confusion matrix shows a predicted class vs an actual class.

6.1 GBM_2_AutoML_1_20230330_34123

```
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.3614137239233675
       0    1    Error    Rate
-----  ---  ---  -------  -----------
0      76   1    0.013    (1.0/77.0)
1      3    36   0.0769   (3.0/39.0)
Total  79   37   0.0345   (4.0/116.0)
```

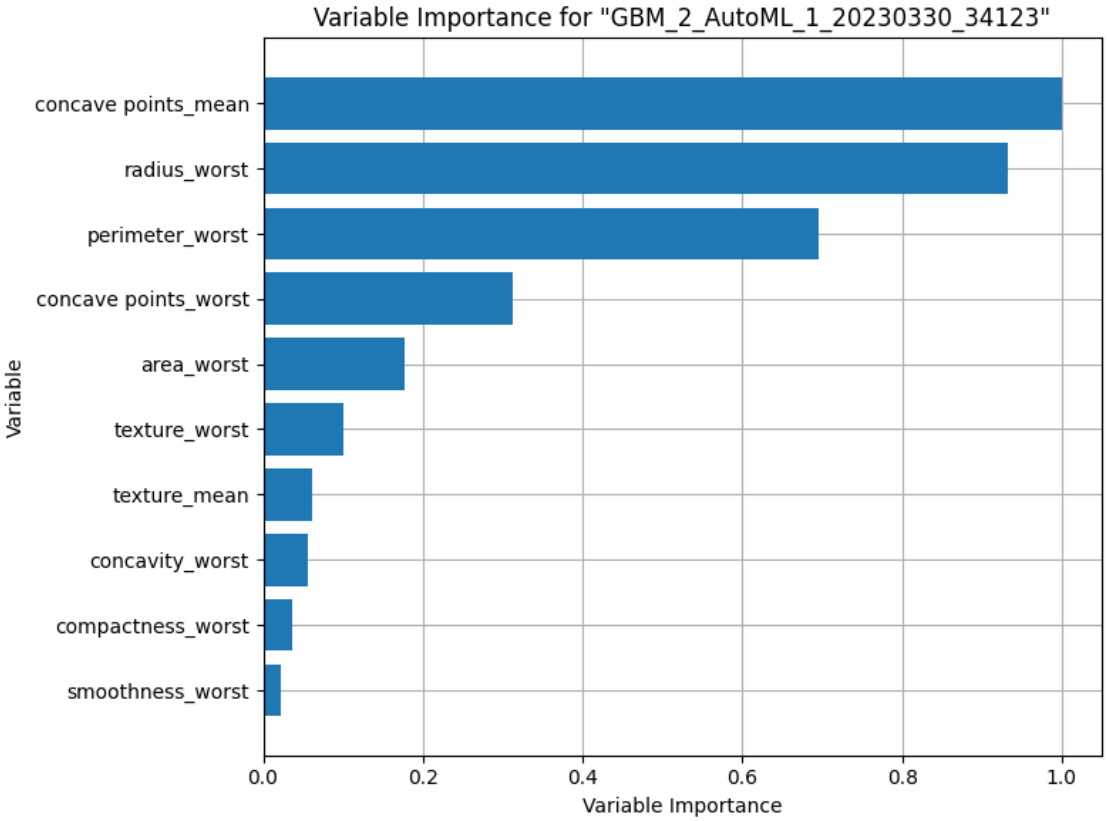
7 Learning Curve Plot

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.



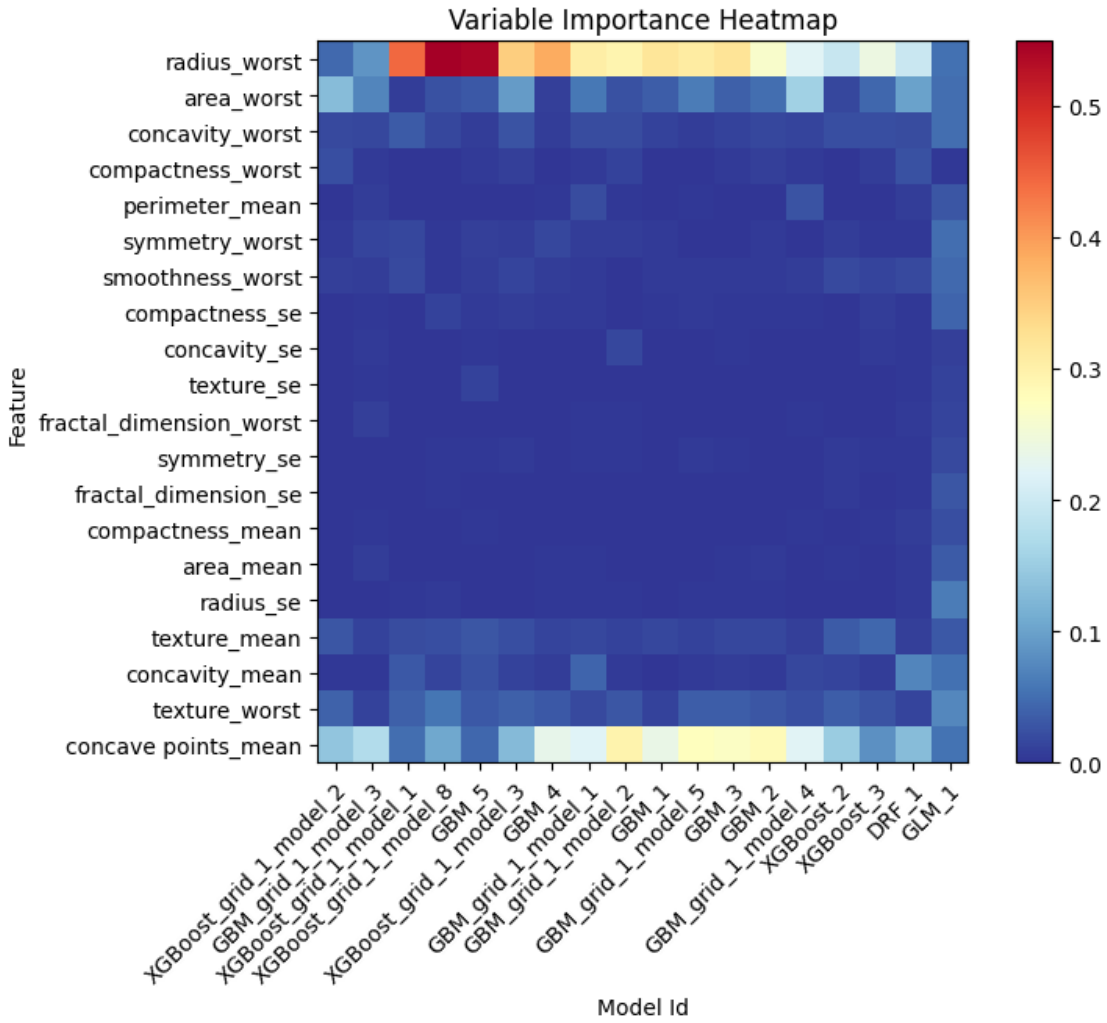
8 Variable Importance

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



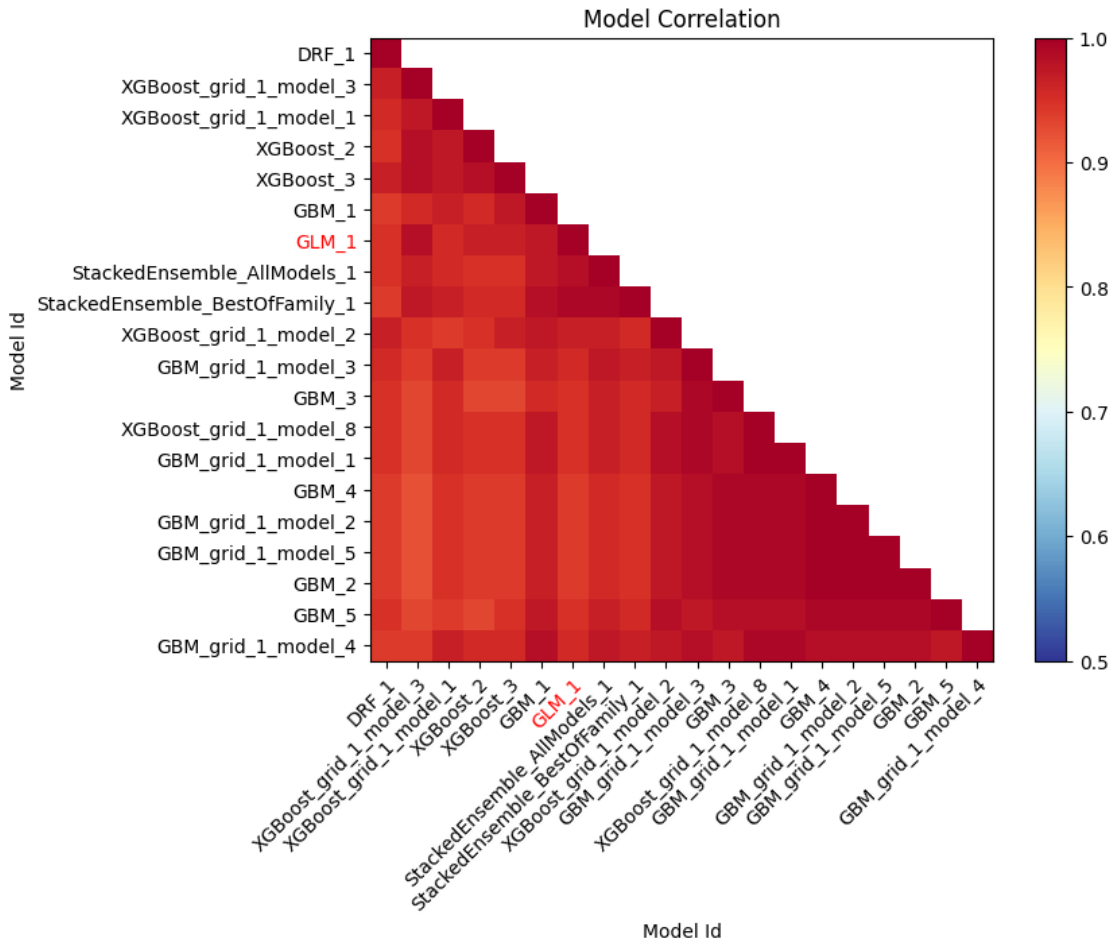
9 Variable Importance Heatmap

Variable importance heatmap shows variable importance across multiple models. Some models in H2O return variable importance for one-hot (binary indicator) encoded versions of categorical columns (e.g. Deep Learning, XGBoost). In order for the variable importance of categorical columns to be compared across all model types we compute a summarization of the the variable importance across all one-hot encoded features and return a single variable importance for the original categorical feature. By default, the models and variables are ordered by their similarity.



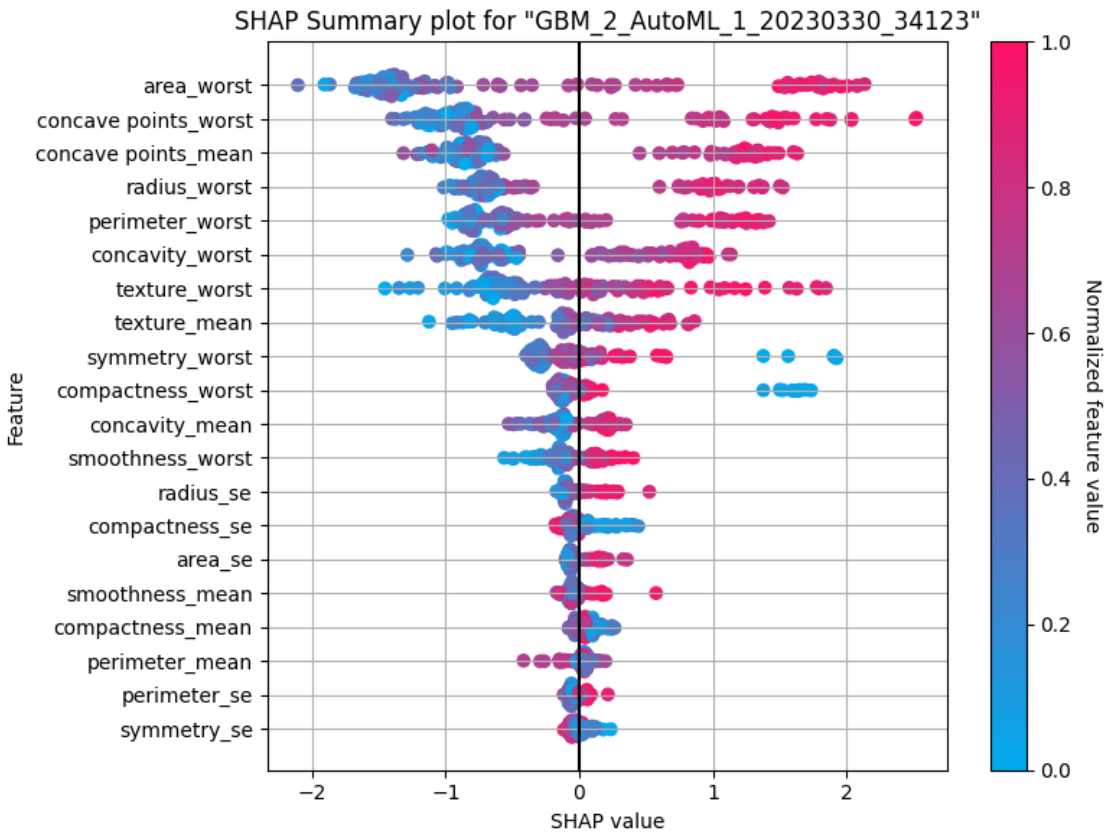
10 Model Correlation

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.



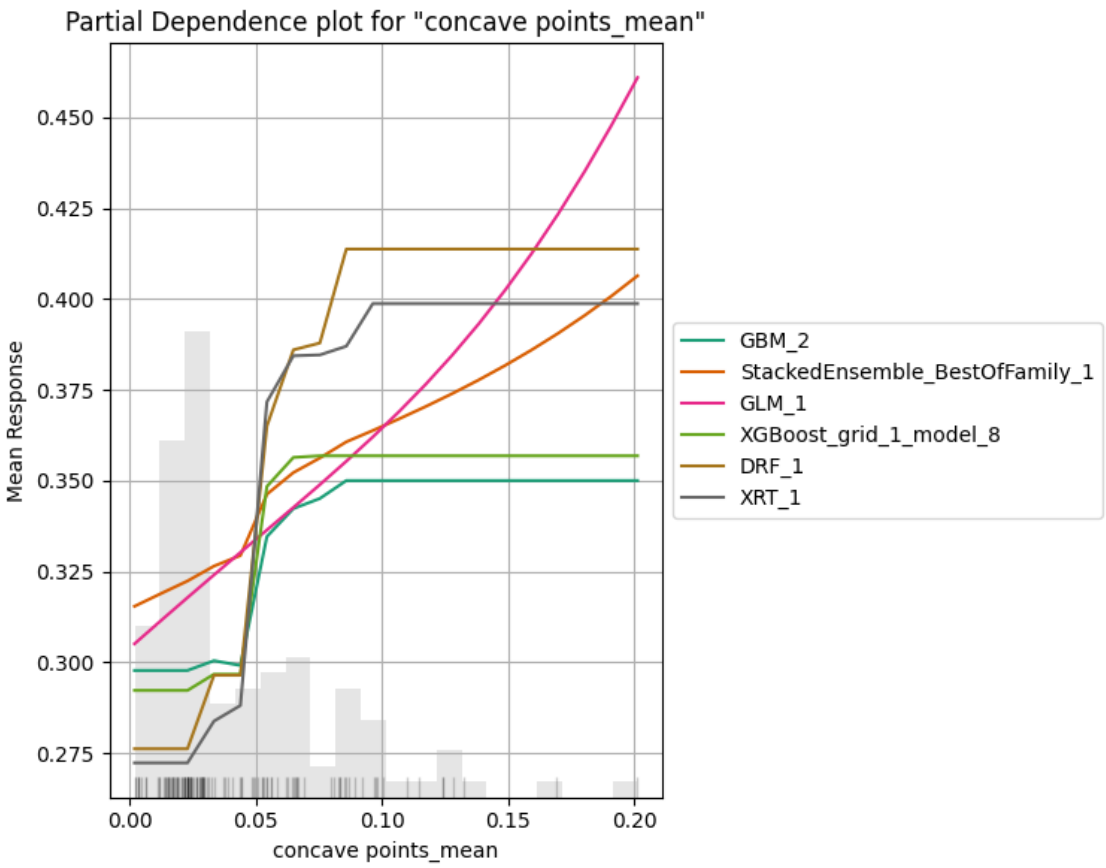
11 SHAP Summary

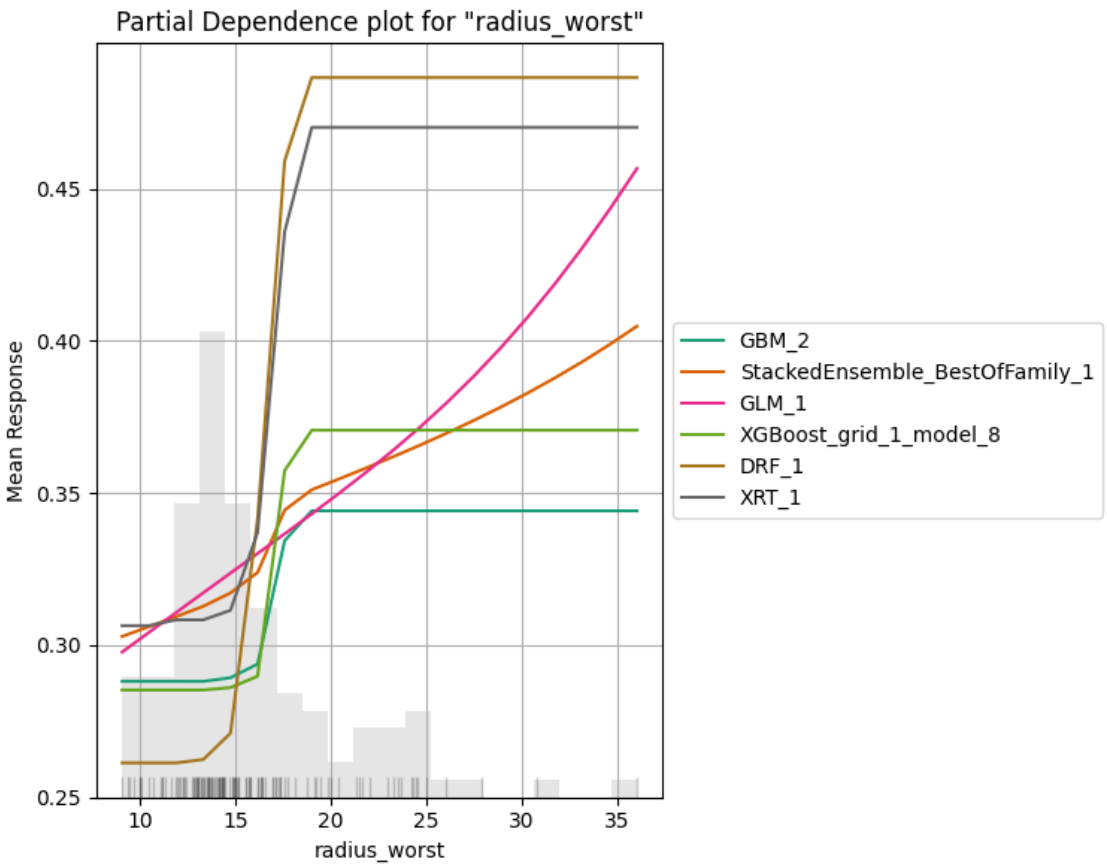
SHAP summary plot shows the contribution of the features for each instance (row of data). The sum of the feature contributions and the bias term is equal to the raw prediction of the model, i.e., prediction before applying inverse link function.

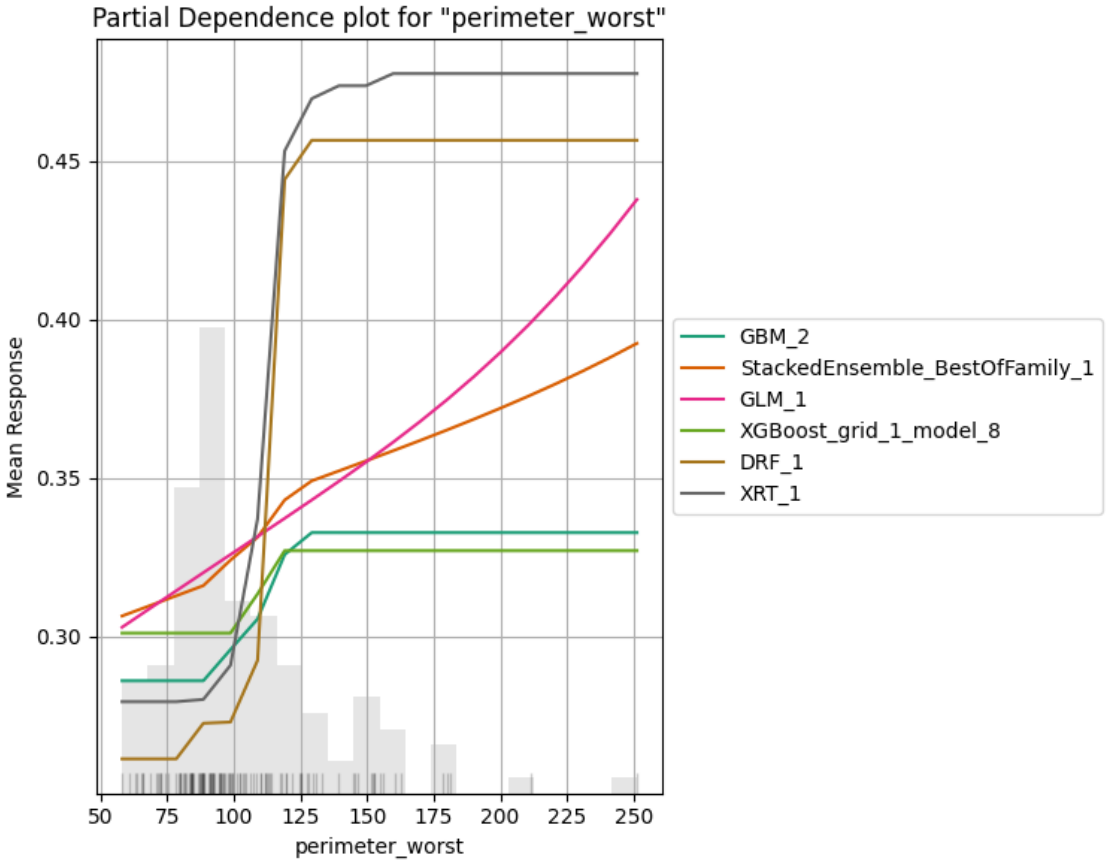


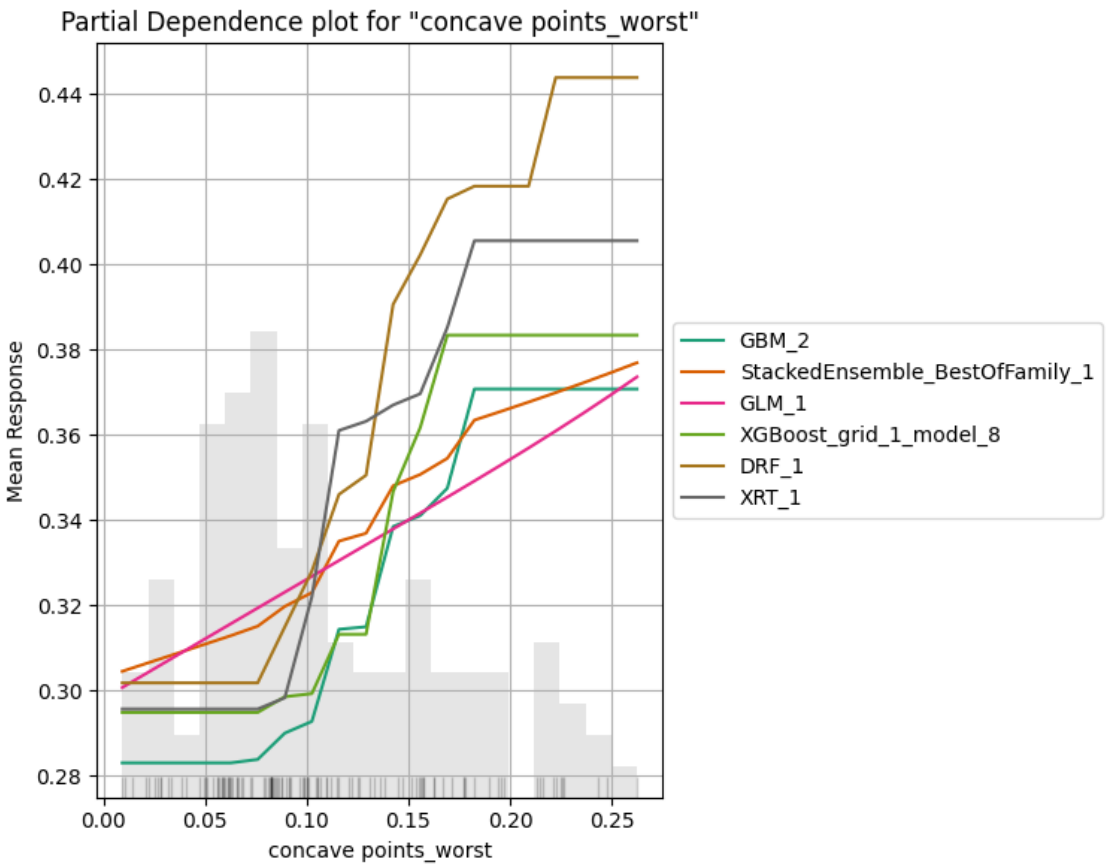
12 Partial Dependence Plots

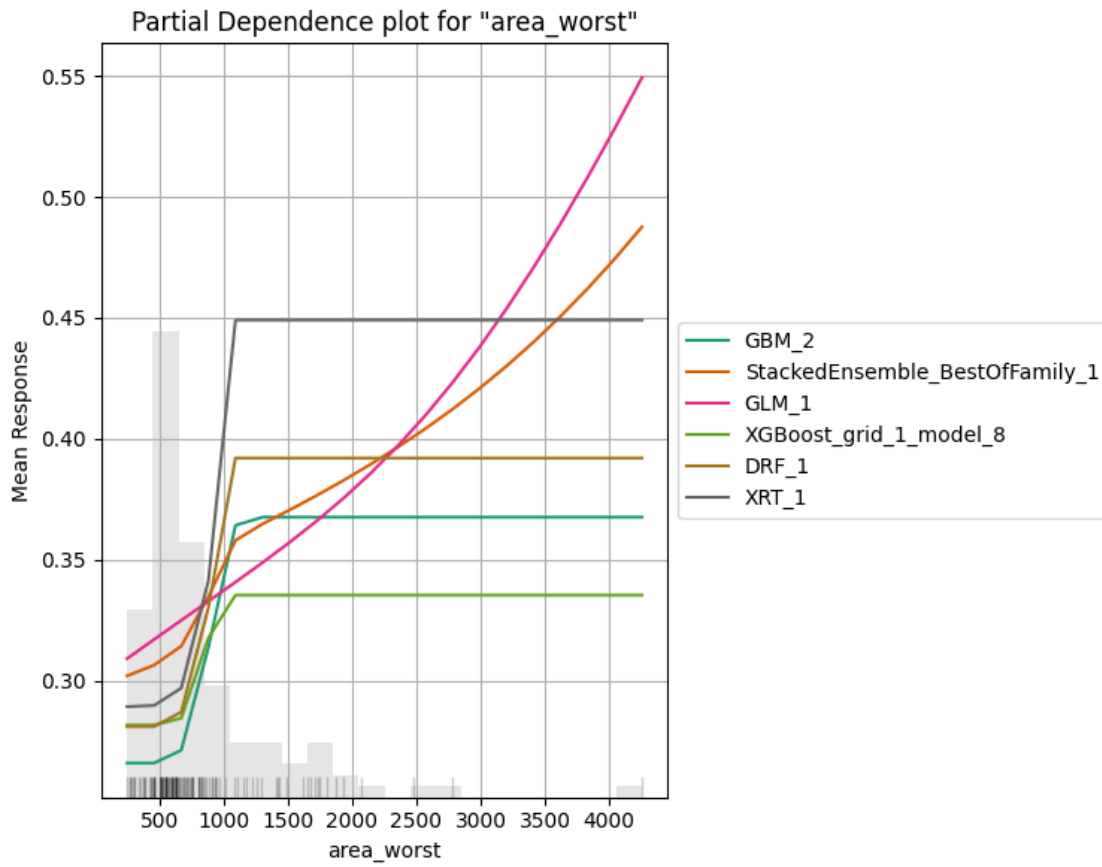
Partial dependence plot (PDP) gives a graphical depiction of the marginal effect of a variable on the response. The effect of a variable is measured in change in the mean response. PDP assumes independence between the feature for which is the PDP computed and the rest.


Partial Dependence plot for "concave points_mean"
Mean Response
0.450
0.425
0.400
0.375
0.350
0.325
0.300
0.275
0.00
0.05
0.10
0.15
0.20
concave points_mean
GBM_2
StackedEnsemble_BestOfFamily_1
GLM_1
XGBoost_grid_1_model_8
DRF_1
XRT_1


Partial Dependence plot for "radius_worst"
Mean Response
0.25
0.30
0.35
0.40
0.45
10
15
20
25
30
35
radius_worst
GBM_2
StackedEnsemble_BestOfFamily_1
GLM_1
XGBoost_grid_1_model_8
DRF_1
XRT_1


Partial Dependence plot for "perimeter_worst"
Mean Response
0.45
0.40
0.35
0.30
50
75
100
125
150
175
200
225
250
perimeter_worst
GBM_2
StackedEnsemble_BestOfFamily_1
GLM_1
XGBoost_grid_1_model_8
DRF_1
XRT_1


Partial Dependence plot for "concave points_worst"
Mean Response
0.44
0.42
0.40
0.38
0.36
0.34
0.32
0.30
0.28
0.00
0.05
0.10
0.15
0.20
0.25
concave points_worst
GBM_2
StackedEnsemble_BestOfFamily_1
GLM_1
XGBoost_grid_1_model_8
DRF_1
XRT_1



[35]:
```
#Observing the performance of best model
aml.leader.model_performance(test_data=test)
```

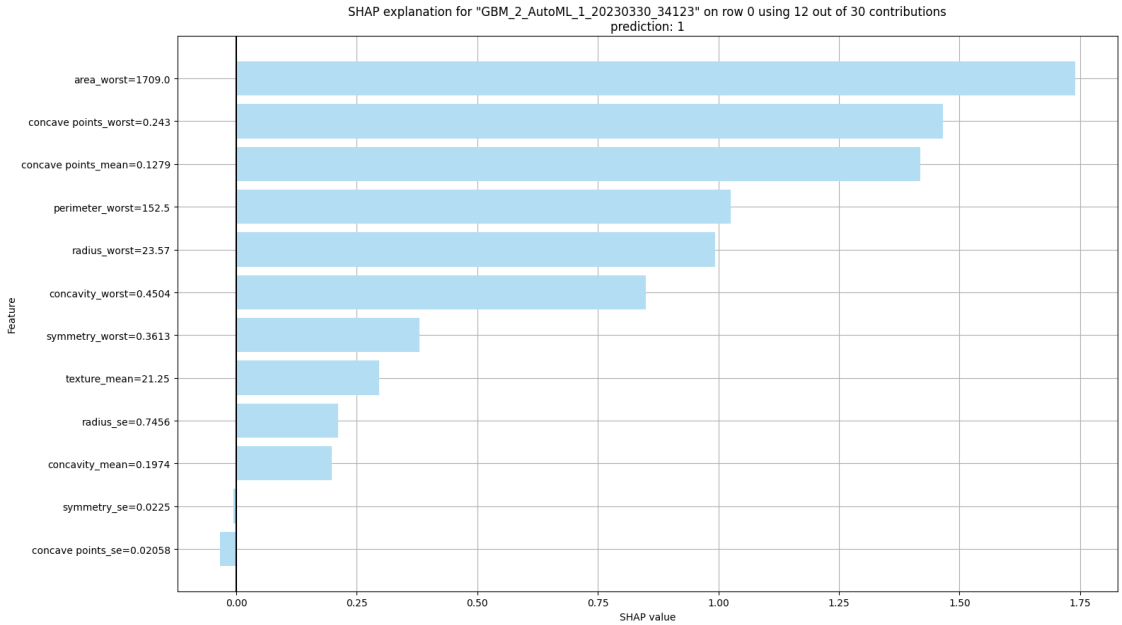
[35]:
```
ModelMetricsBinomial: gbm
** Reported on test data. **

MSE: 0.030972004811896832
RMSE: 0.1759886496678034
LogLoss: 0.10983339525988686
Mean Per-Class Error: 0.04495504495504496
AUC: 0.9943389943389944
AUCPR: 0.9900199157118534
Gini: 0.9886779886779888

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.3614137239233675
       0    1    Error    Rate
-----  ---  ---  -------  -----------
```

```
0      76   1    0.013    (1.0/77.0)
1      3    36   0.0769   (3.0/39.0)
Total  79   37   0.0345   (4.0/116.0)

Maximum Metrics: Maximum metrics at their respective thresholds
metric                        threshold    value     idx
---------------------------  -----------  --------  -----
max f1                        0.361414     0.947368  36
max f2                        0.096091     0.959596  41
max f0point5                  0.663264     0.977654  34
max accuracy                  0.663264     0.965517  34
max precision                 0.999832     1         0
max recall                    0.0104303    1         47
max specificity               0.999832     1         0
max absolute_mcc              0.663264     0.923644  34
max min_per_class_accuracy    0.138864     0.948718  39
max mean_per_class_accuracy   0.096091     0.961205  41
max tns                       0.999832     77        0
max fns                       0.999832     38        0
max fps                       0.000145912  77        115
max tps                       0.0104303    39        47
max tnr                       0.999832     1         0
max fnr                       0.999832     0.974359  0
max fpr                       0.000145912  1         115
max tpr                       0.0104303    1         47

Gains/Lift Table: Avg response rate: 33.62 %, avg score: 31.10 %
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        0.0172414                   0.999731           2.97436   2.97436
1                0.999781     1                          0.999781
0.0512821       0.0512821                  197.436   197.436
0.0512821
2        0.0258621                   0.999713           2.97436   2.97436
1                0.999728     1                          0.999763
0.025641        0.0769231                  197.436   197.436
0.0769231
3        0.0344828                   0.999665           2.97436   2.97436
1                0.999679     1                          0.999742
0.025641        0.102564                   197.436   197.436            0.102564
4        0.0431034                   0.99964            2.97436   2.97436
```

```
1                0.999649     1                           0.999724
0.025641        0.128205                    197.436   197.436            0.128205
5        0.0517241                    0.999631           2.97436   2.97436
1                0.999634     1                           0.999709
0.025641        0.153846                    197.436   197.436            0.153846
6        0.103448                    0.999427           2.97436   2.97436
1                0.999535     1                           0.999622
0.153846        0.307692                    197.436   197.436            0.307692
7        0.155172                    0.999089           2.97436   2.97436
1                0.999278     1                           0.999507
0.153846        0.461538                    197.436   197.436            0.461538
8        0.206897                    0.997978           2.97436   2.97436
1                0.99851      1                           0.999258
0.153846        0.615385                    197.436   197.436            0.615385
9        0.301724                    0.554957           2.97436   2.97436
1                0.913293     1                           0.97224
0.282051        0.897436                    197.436   197.436            0.897436
10       0.405172                    0.0170397          0.74359   2.4048
0.25             0.163536     0.808511                   0.765763
0.0769231       0.974359                    -25.641   140.48             0.857476
11       0.5                         0.00251416         0.270396  2
0.0909091        0.00496243   0.672414                   0.621473
0.025641        1                           -72.9604  100                0.753247
12       0.603448                    0.000821886        0         1.65714
0                0.00156189   0.557143                   0.515202            0
1                         -100      65.7143            0.597403
13       0.698276                    0.000544298        0         1.4321
0                0.000673076  0.481481                   0.445328            0
1                         -100      43.2099            0.454545
14       0.801724                    0.000313328        0         1.24731
0                0.000402058  0.419355                   0.387918            0
1                         -100      24.7312            0.298701
15       0.896552                    0.000233672        0         1.11538
0                0.00026367   0.375                      0.346916            0
1                         -100      11.5385            0.155844
16       1                           0.000145912        0         1
0                0.000195512  0.336207                   0.311049            0
1                         -100      0                  0
```

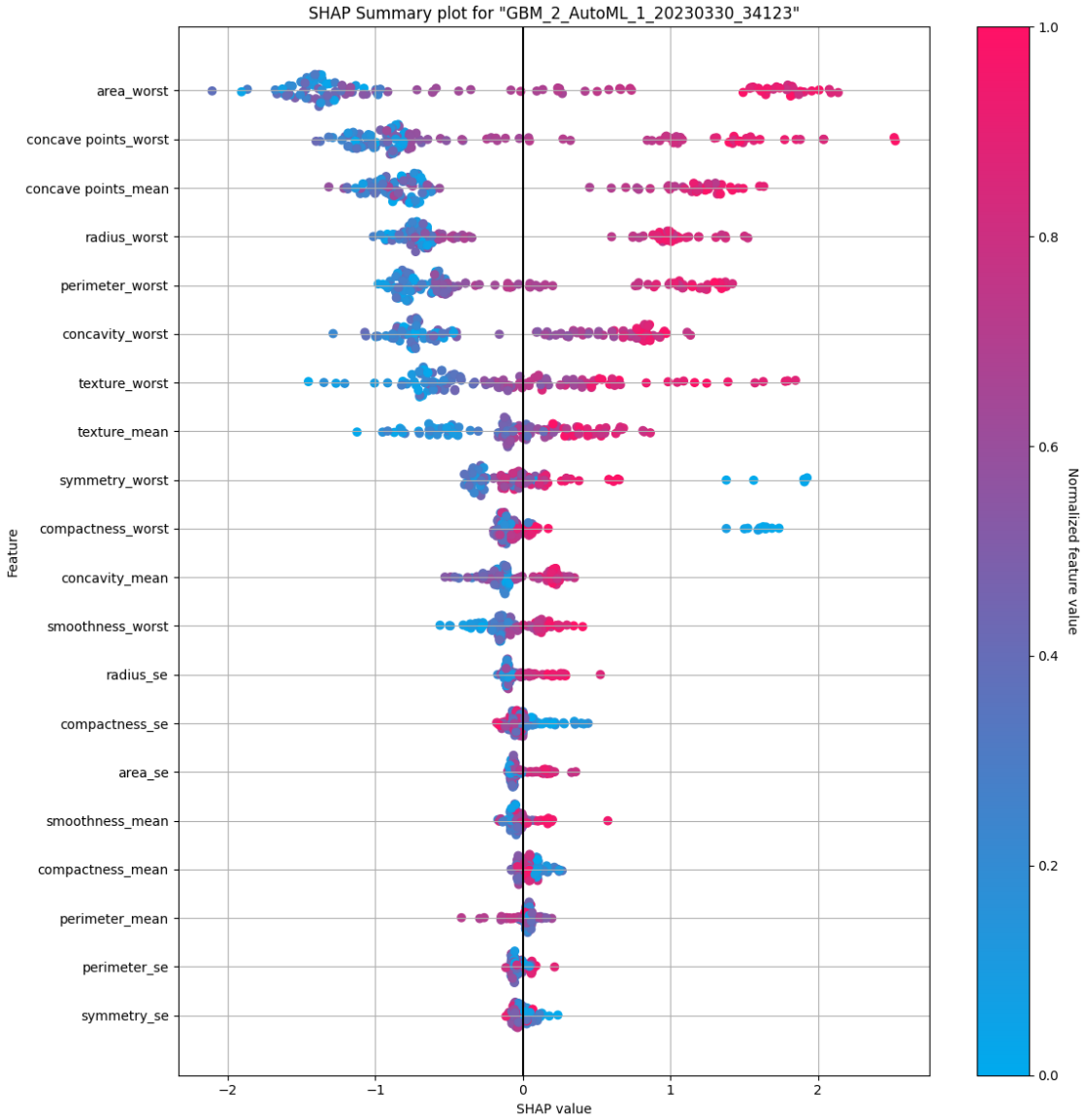
[36]:
```
aml.leader.shap_explain_row_plot(test,0)
```

[36]: <h2o.plot._plot_result._MObject at 0x7fda8dafbaf0>



[37]:
```
aml.leader.shap_summary_plot(test)
```

[37]: <h2o.plot._plot_result._MObject at 0x7fda8d7e0fa0>



The significant features assumed by GBM model are **concave points_mean, radius_worst, perimeter_worst, concave points_worst, area_worst, texture_worst, texture_mean, concavity_ worst, compactness_worst and smoothness worst**.

13 Conclusion

The set of important features obtained from a model-agnostic method(SHAP summary plot, which is a representation of the shap values) are compared to the significant feature investigation performed by other model-dependent interpretation methods.

14 References

ajhalthor/model-interpretability

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.