Project: **Breast Cancer Prediction**

https://github.com/uic-ds-fall-2023/class-project-masterminds

Part 1 - Project introduction

This dataset contains characteristics derived from digitized imaging of fine needle aspirates of a breast tumor cell mass. The goal of this analysis is to train a machine leanring algorightms to accurately distinguish between a benign and malignant tumor to aid in clinical diagnosis.

Ten real-valued features were computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)

b) texture (standard deviation of gray-scale values)

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

f) compactness (perimeter^2 / area - 1.0)

g) concavity (severity of concave portions of the contour)

h) concave points (number of concave portions of the contour)

i) symmetry

j) fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 5 is Mean Perimeter, field 15 is Perimeter SE, field 23 is Perimeter worst.

Number of instances: 569

Number of attributes: 32

Missing attribute values: none

Class distribution: 357 benign, 212 malignant

Original Dataset is available at the UCA Machine Learning Repository

Null Hypothesis: The distribution of each feature is the same for benign and malignant tumors.

Alternative Hypothesis: There are significant differences in the distribution of at least one feature between benign and malignant tumors.

In [ ]:
```python
# Package usage
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import mannwhitneyu
import sys
```

In [ ]:
```python
cancerDf = pd.read_csv('./datasets/wdbc.csv')
```

Part 2: Data Cleaning and Exploratory Data Analysis:

Original size: Size of the dataframe: (569, 33)

Listed below are are the different properties accounted for when describing the characteristics of the cell nuclei present in the image (digitized image of a fine need aspirate (FNA) of a breast mass):

In [ ]:
```python
cancerDf.dtypes
```

Out[ ]:
```
id                           int64
diagnosis                   object
radius_mean                float64
texture_mean               float64
perimeter_mean             float64
area_mean                  float64
smoothness_mean            float64
compactness_mean           float64
concavity_mean             float64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
concave points_se          float64
symmetry_se                float64
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
Unnamed: 32                float64
dtype: object
```

In [ ]:
```python
# We drop unnecessary columns
cancerDf.drop(['id','Unnamed: 32'], axis = 1, inplace = True)
```

```
# Replace 'M' with 'malignant' and 'B' with 'benign' in the "diagnosis" column
cancerDf['diagnosis'] = cancerDf['diagnosis'].replace({'M': 'Malignant', 'B': 'Benign'

# Here we are reordering our data based on the average properties of our images, based
# This allows us to make further insights on the varying characteristics that benign a
# Group by 'Diagnosis' and their means, respectively
cancerDf.groupby('diagnosis').mean()
```

Out[ ]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_
diagnosis						
Benign	12.146524	17.914762	78.075406	462.790196	0.092478	0.08
Malignant	17.462830	21.604906	115.365377	978.376415	0.102898	0.14

2 rows × 30 columns

In [ ]:
```
# Group by 'Diagnosis' and display descriptive statistics for selected features
selected_features = ['radius_mean', 'texture_mean', 'area_mean']
grouped_stats = cancerDf.groupby('diagnosis')[selected_features].describe()
print(grouped_stats)
```

```
          radius_mean                                                        \
                count       mean       std     min     25%     50%    75%
diagnosis
Benign          357.0  12.146524  1.780512   6.981  11.080  12.200  13.37
Malignant       212.0  17.462830  3.203971  10.950  15.075  17.325  19.59

                 texture_mean            ...                 area_mean  \
            max         count       mean  ...     75%    max      count
diagnosis                                 ...
Benign    17.85         357.0  17.914762  ...  19.760  33.81      357.0
Malignant 28.11         212.0  21.604906  ...  23.765  39.28      212.0


                 mean         std    min    25%    50%      75%     max
diagnosis
Benign     462.790196  134.287118  143.5  378.2  458.4   551.10   992.1
Malignant  978.376415  367.937978  361.6  705.3  932.0  1203.75  2501.0

[2 rows x 24 columns]
```

This code performs a groupby operation on the 'Diagnosis' column and calculates descriptive statistics (count, mean, standard deviation, minimum, 25th percentile, median (50th percentile), 75th percentile, and maximum) for the selected features ('radius_mean', 'texture_mean', 'area_mean') for each diagnosis category ('benign' and 'malignant').

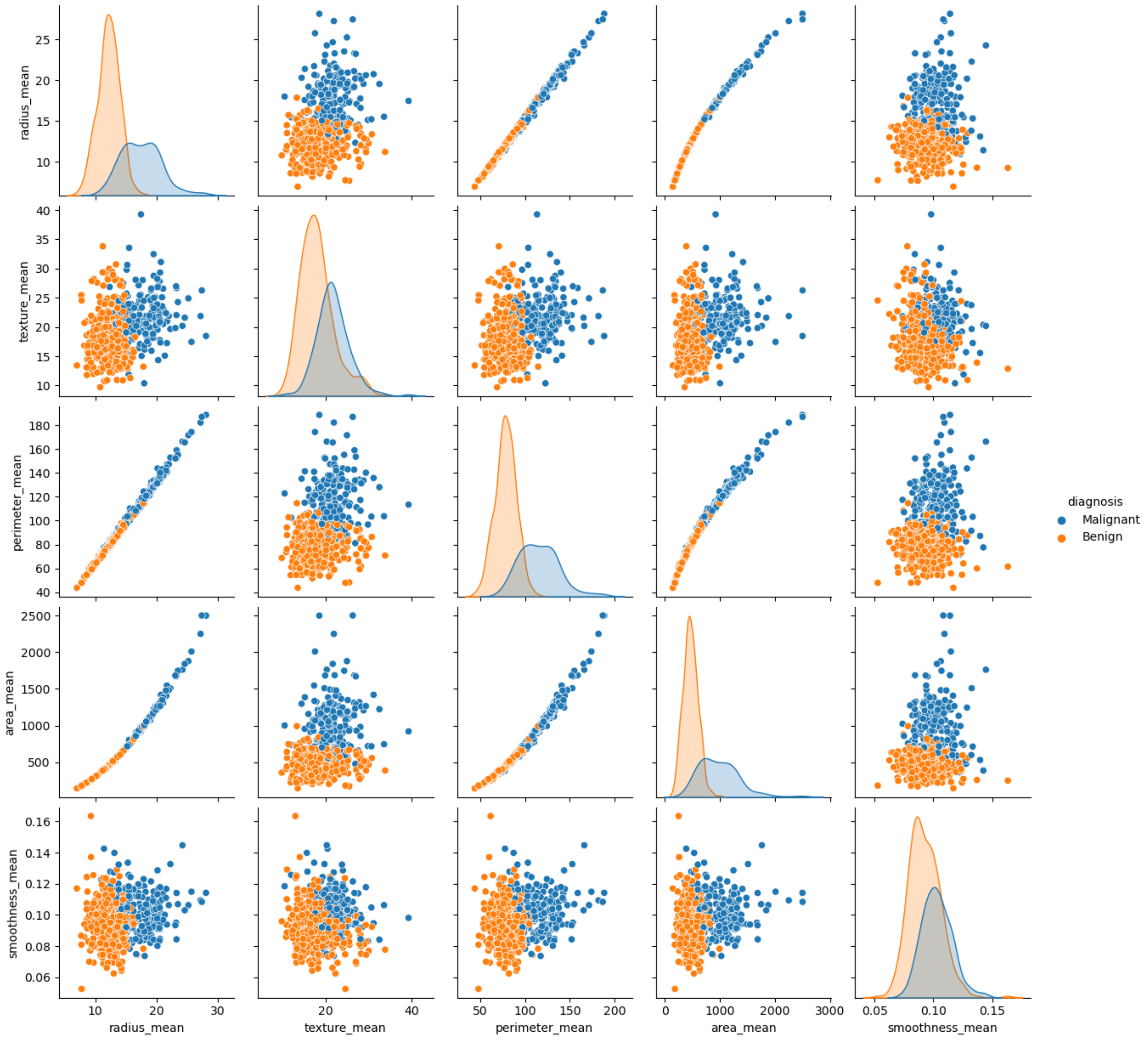
In [ ]:
```
# Create dataframes for Benign and Malignant components of Diagnosis
benign_df = cancerDf[cancerDf['diagnosis'] == 0]
malignant_df = cancerDf[cancerDf['diagnosis'] == 1]
# Create a list of features related to mean tumor characteristics
features_means = list(cancerDf.columns[1:11])
```

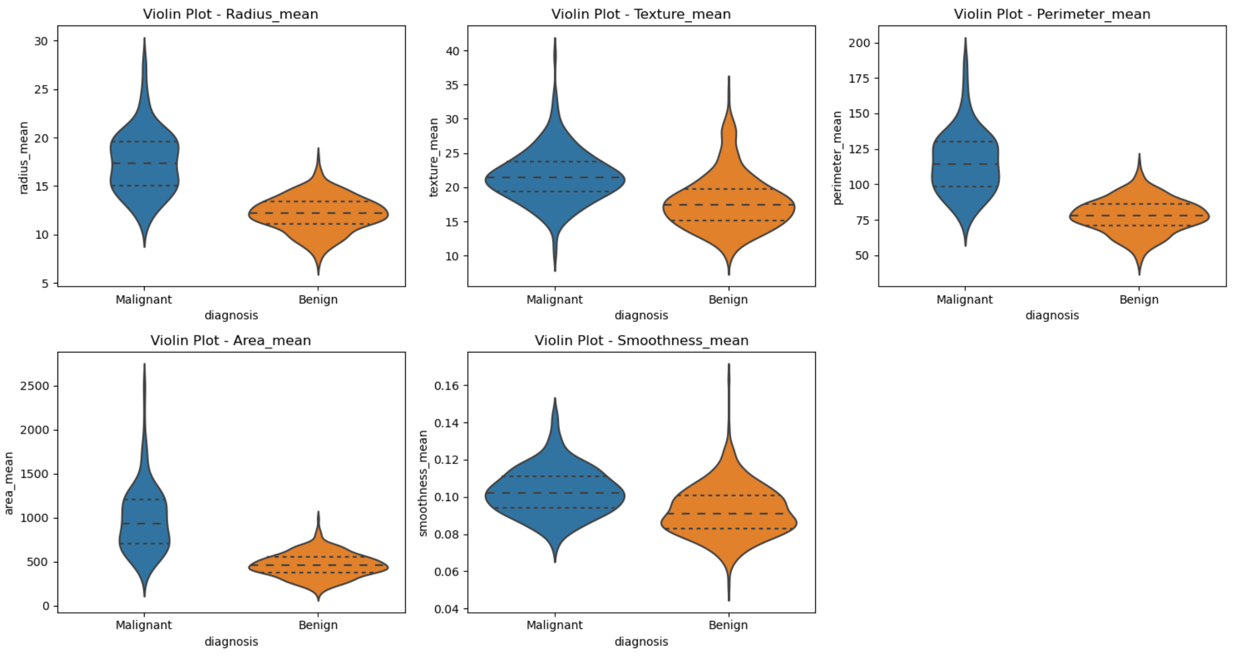
Import functions used to plot graphs

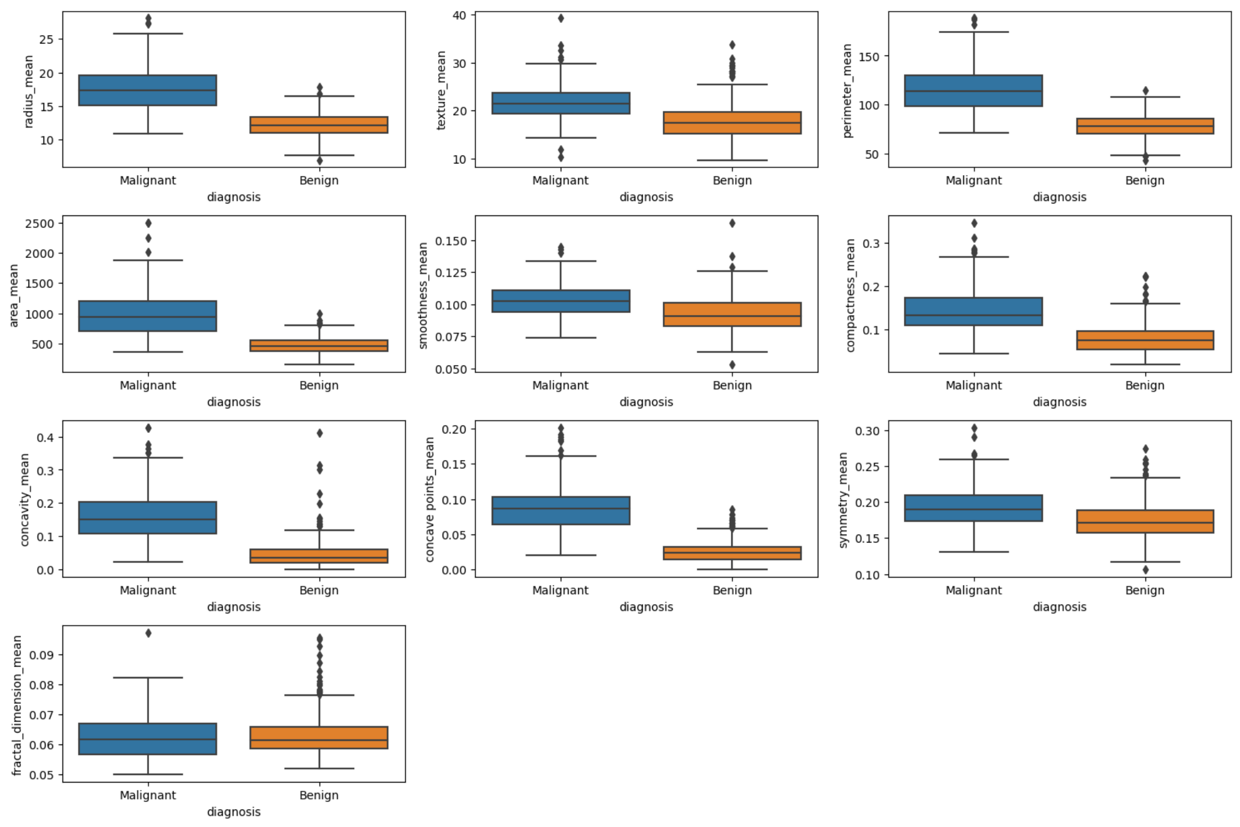
In [ ]:
```
import plotFunctions
```

In [ ]:
```
# Pairplot for selected features
selected_features = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'sm

# function to plot
plotFunctions.plot_data_1(cancerDf, selected_features)
```





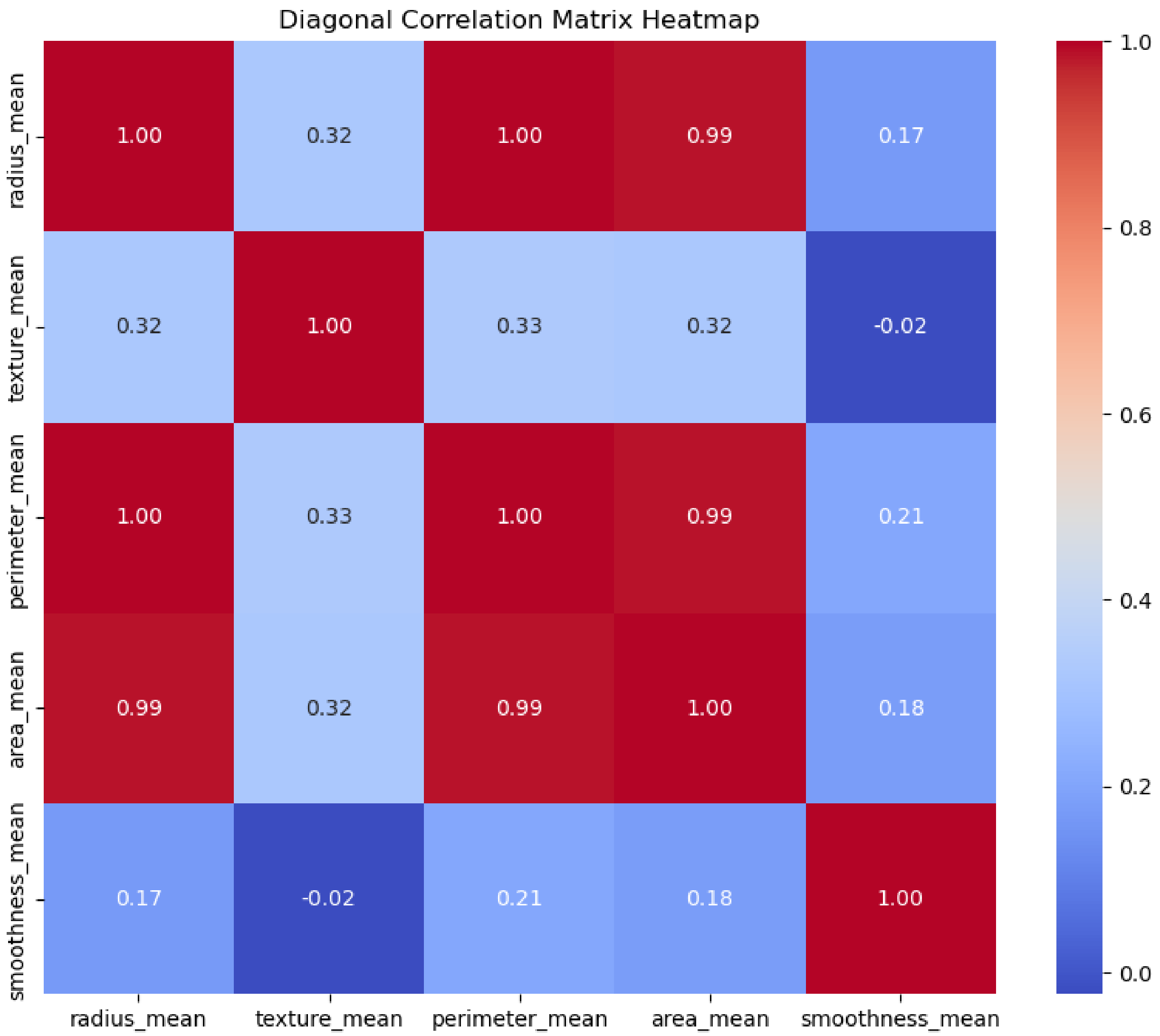


These visualizations help in understanding the distribution of individual features, relationships between features, and differences between malignant and benign tumors. The pairplot provides a comprehensive view of pairwise relationships, while the violin plots and boxplots focus on specific features to highlight their distributions and statistical measures.

```
In [ ]: # Assuming cancerDf is your DataFrame
        selected_features = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'sm
```

The code is a part of exploratory data analysis aiming to visualize and compare the distributions of selected tumor features for benign and malignant cases.The rounded p-values are included in the plot titles to provide information about the statistical significance of differences in feature distributions between the two tumor types.

```
In [ ]: # Select features for the correlation matrix
        selected_features = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'sm
        # Calculate the correlation matrix
        correlation_matrix = cancerDf[selected_features].corr()
        # Create a diagonal correlation matrix with a heatmap
        plt.figure(figsize=(10, 8))
        sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', square=True)
        plt.title('Diagonal Correlation Matrix Heatmap')
        plt.show()
```



A correlation matrix visually represents relationships between variables, using colors to indicate the strength and direction of correlation.

In our breast cancer dataset, the Pearson Correlation Coefficient is employed. Diagonal values are 1.0, as features are correlated with themselves. While correlations like radius mean with area mean and perimeter mean are observed, they don't directly align with our hypothesis.

Nevertheless, these insights help refine our understanding of the dataset, guiding future analyses.

Machine Learning / Statistics Analysis:

The following will feature the three model types that we utilized to further understand the characteristics of benign and malignant masses.

1. Decision Tress Classification:

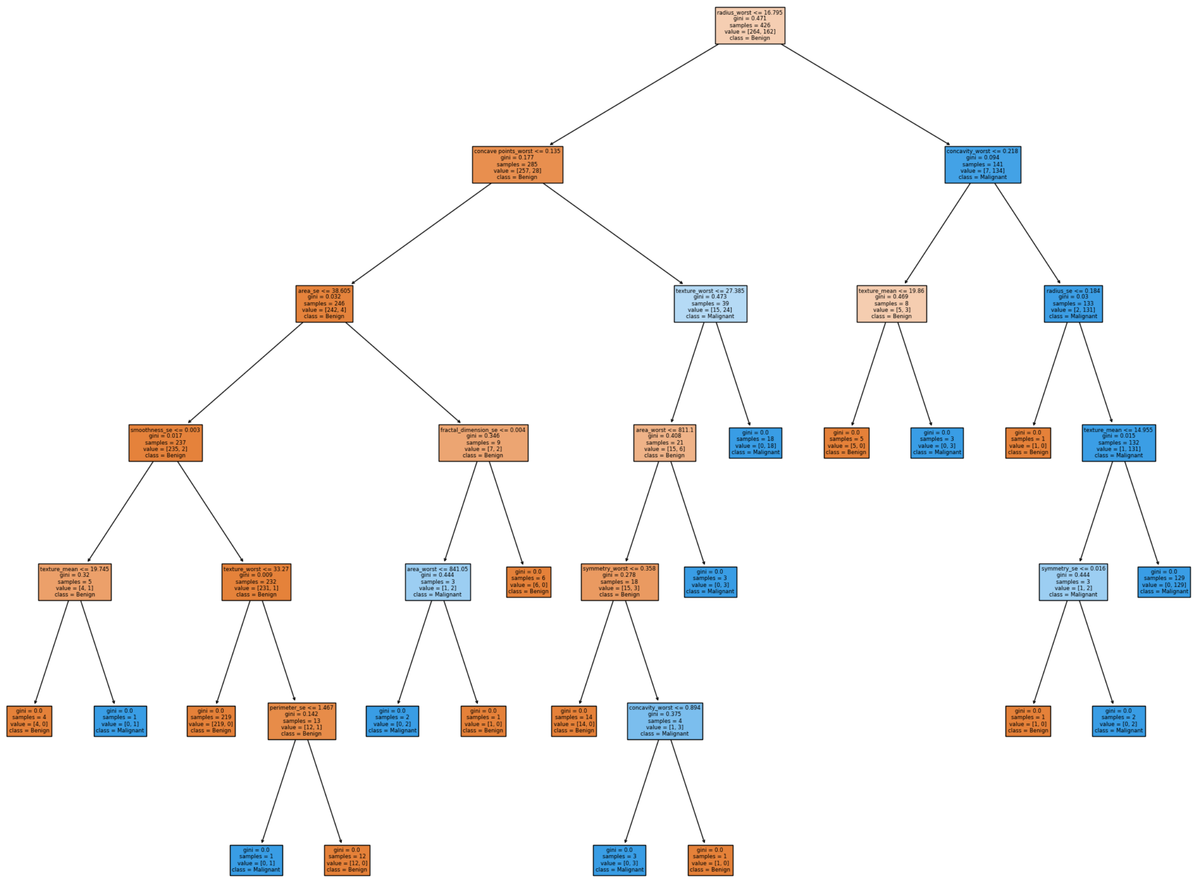
In [ ]:
```
# the following function performs a train-test split on the data, fits a Decision Tree
# calculates and returns the training accuracy, test accuracy, as well as the training
train, test, x_train, y_train = plotFunctions.decision_tree(cancerDf)
print("Train data accuracy:", train)
print("Test data accuracy:", test)
```

```
Train data accuracy: 1.0
Test data accuracy: 0.951048951048951
```

In [ ]:
```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()
clf.fit(x_train, y_train)
X = cancerDf.drop('diagnosis', axis=1)

fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clf,
                   feature_names=list(X.columns),
                   class_names=['Benign','Malignant'],
                   filled=True)
```


radius_worst <= 16.795 gini = 0.471 samples = 426 value = [264, 162] class = Benign
concave points_worst <= 0.135 gini = 0.177 samples = 285 value = [257, 28] class = Benign
concavity_worst <= 0.218 gini = 0.094 samples = 141 value = [7, 134] class = Malignant
area_se <= 38.605 gini = 0.032 samples = 246 value = [242, 4] class = Benign
texture_worst <= 27.385 gini = 0.473 samples = 39 value = [15, 24] class = Malignant
texture_mean <= 19.86 gini = 0.469 samples = 8 value = [5, 3] class = Benign
radius_se <= 0.184 gini = 0.03 samples = 133 value = [2, 131] class = Malignant
smoothness_se <= 0.003 gini = 0.017 samples = 237 value = [235, 2] class = Benign
fractal_dimension_se <= 0.004 gini = 0.346 samples = 9 value = [7, 2] class = Benign
area_worst <= 811.1 gini = 0.408 samples = 21 value = [15, 6] class = Benign
gini = 0.0 samples = 18 value = [0, 18] class = Malignant
gini = 0.0 samples = 5 value = [5, 0] class = Benign
gini = 0.0 samples = 3 value = [0, 3] class = Malignant
gini = 0.0 samples = 1 value = [1, 0] class = Benign
texture_mean <= 14.955 gini = 0.015 samples = 132 value = [1, 131] class = Malignant
texture_mean <= 19.745 gini = 0.32 samples = 5 value = [4, 1] class = Benign
texture_worst <= 33.27 gini = 0.009 samples = 232 value = [231, 1] class = Benign
area_worst <= 841.05 gini = 0.444 samples = 3 value = [1, 2] class = Malignant
gini = 0.0 samples = 6 value = [6, 0] class = Benign
symmetry_worst <= 0.358 gini = 0.278 samples = 18 value = [15, 3] class = Benign
gini = 0.0 samples = 3 value = [0, 3] class = Malignant
symmetry_se <= 0.016 gini = 0.444 samples = 3 value = [1, 2] class = Malignant
gini = 0.0 samples = 129 value = [0, 129] class = Malignant
gini = 0.0 samples = 4 value = [4, 0] class = Benign
gini = 0.0 samples = 1 value = [0, 1] class = Malignant
gini = 0.0 samples = 219 value = [219, 0] class = Benign
perimeter_se <= 1.467 gini = 0.142 samples = 13 value = [12, 1] class = Benign
gini = 0.0 samples = 2 value = [0, 2] class = Malignant
gini = 0.0 samples = 1 value = [1, 0] class = Benign
gini = 0.0 samples = 14 value = [14, 0] class = Benign
concavity_worst <= 0.894 gini = 0.375 samples = 4 value = [1, 3] class = Malignant
gini = 0.0 samples = 1 value = [1, 0] class = Benign
gini = 0.0 samples = 2 value = [0, 2] class = Malignant
gini = 0.0 samples = 1 value = [0, 1] class = Malignant
gini = 0.0 samples = 12 value = [12, 0] class = Benign
gini = 0.0 samples = 3 value = [0, 3] class = Malignant
gini = 0.0 samples = 1 value = [1, 0] class = Benign

Decision trees can offer insights into which features are most important for making a prediction. In the context of breast cancer prediction, this means identifying which characteristics of a tumor (like size, texture, perimeter, etc.) are most indicative of whether it is benign or malignant.

With the data provided above, we can see that the accuracy for:

- Training Data came out to 100%!
- Test Data came out to around 88%

While this may sound like good news, it may indicate that the model may be overfitting the training data, meaning it has learned the training data too well, including its noise and outliers, and may not generalize well to new, unseen data.

Due to these results and concerns about overfitting, our approach was to shift towards another Machine Learning algoritm.

2. K-Nearest Neighbors (KNN) Supervised Machine Learning Algorithm:

In [ ]:
```
accuracy, confusion_mat, classification_rep = plotFunctions.knn_neighbor(cancerDf, 4)
print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{confusion_mat}")
print(f"Classification Report:\n{classification_rep}")
```

```
Accuracy: 0.956140350877193
Confusion Matrix:
[[69  2]
 [ 3 40]]
Classification Report:
              precision    recall  f1-score   support

      Benign       0.96      0.97      0.97        71
   Malignant       0.95      0.93      0.94        43

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

The KNN model demonstrates strong performance in distinguishing between tumor types, with an accuracy of 96%.

High precision, recall, and F1-score values, along with the confusion matrix details, validate the model's effectiveness.

After evaluating various K-values, 4 was selected for its balanced performance.

The results support our hypothesis that the model accurately predicts benign or malignant masses based on FNA image features, offering confidence in early cancer diagnosis.

Benefits:

KNN makes minimal assumptions about the underlying data distribution. This flexibility is advantageous when the relationship between features and outcomes is complex and not easily modeled by parametric methods.

Drawbacks:

The choice of k and the distance metric can have a huge impact on the performance of the algorithm, and these hyperparameters must be carefully tuned to achieve good results.

3. Logistic Regression Binary Classification:

Correlation Analysis (Statistical Techniques): Model Chosen: Linear Regression, Pearson correlation

Benefits of Linear Regression: It captures linear correlations between variables, that is, how one variable affects another linearly. It can be used to forecast the values of one variable based on the values of another. It gives coefficients for each input feature, which may be simply translated as the amount of change in the output variable for a unit change in the associated input feature.

Drawbacks of Linear Regression: Because it presupposes a linear relationship between variables, it is incapable of capturing non-linear relationships. Because of the slope computation, it is sensitive to outliers, exactly like Pearson.

Benefits of Pearson Correlation: It represents the linear relationship between two variables. It generates coefficients ranging from -1 to 1, with -1 indicating maximum negative correlation, 0 indicating no association, and 1 indicating maximum positive correlation. This makes the strength and direction of the association between two variables interpretable and simple to understand. Coefficients for a large number of data points are incredibly simple to calculate, making it scalable for large datasets.

Drawbacks of Pearson Correlation: It can only record linear correlations between variables, not nonlinear ones. It computes means and standard deviations, making it susceptible to outliers.

```
In [ ]: plotFunctions.train_logistic_regression(cancerDf)
```

```
Out[ ]: 0.9766081871345029
```

Conclusion:

The data and visualizations demonstrates significant differences in the distribution of these features between benign and malignant breast tumors. This supports the Alternative Hypothesis, indicating that the distribution of at least one feature (in this case, all three examined features) differs significantly between benign and malignant tumors.

Citation:

1. O. L. Mangasarian and W. H. Wolberg: "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18.

1. William H. Wolberg and O.L. Mangasarian: "Multisurface method of pattern separation for medical diagnosis applied to breast cytology", Proceedings of the National Academy of Sciences, U.S.A., Volume 87, December 1990, pp 9193-9196.

1. O. L. Mangasarian, R. Setiono, and W.H. Wolberg: "Pattern recognition via linear programming: Theory and application to medical diagnosis", in: "Large-scale numerical optimization", Thomas F. Coleman and Yuying Li, editors, SIAM Publications, Philadelphia 1990, pp 22-30.

1. K. P. Bennett & O. L. Mangasarian: "Robust linear programming discrimination of two linearly inseparable sets", Optimization Methods and Software 1, 1992, 23-34 (Gordon & Breach Science Publishers).

2. Kristiansen, Søren L. “Nearest Neighbors with Keras and Coreml.” Medium, Medium, 20 May 2019, https://medium.com/@sorenlind/nearest-neighbors-with-keras-and-coreml-755e76fedf36.

3. Goyal, Anant. “Logistic Regression with Keras.” MarkTechPost, 9 Apr. 2021, https://www.marktechpost.com/2021/04/08/logistic-regression-with-keras/.