CURE MATRIX: AI-Powered Precision Tool for Enhanced Breast Cancer Detection

<u>TEAM MEMBERS</u>

MAYANK KUMAR SINGH

SUBEENA K.K.

MOHAMMAD SAMEER

SANA KAVYA SRI PRIYA

GORLE VAMSI

DEEPAK VERMA

1.Introduction

Breast cancer remains a significant global health challenge. AI-powered tools such as CURE MATRIX and data-driven models promise to revolutionize early detection and diagnosis, reducing mortality rates. By leveraging machine learning algorithms and precision tools, healthcare providers can achieve greater accuracy and accessibility in diagnostics.

---

2. Problem Statement

Breast cancer diagnostics face persistent challenges, including:

- High false positive/negative rates.
- Limited access to advanced tools in underserved areas.
- Lack of personalized diagnostics tailored to individual patient profiles.

---

3. Objectives

Business Goals

- Develop AI-powered precision tools to enhance diagnostic workflows.
- Improve accessibility and affordability of advanced diagnostic techniques.
- Reduce false diagnostic rates and promote early detection.

Technical Goals

- Implement robust data analysis pipelines for model training.
- Evaluate feature importance for diagnosis.
- Train and deploy scalable machine learning models.

---

4. Dataset Overview

Source

- Breast cancer datasets containing tumor characteristics such as radius, texture, perimeter, and area.

Features

- **Predictor Variables**: Mean values of radius, texture, perimeter, area, and smoothness.
- **Target Variable**: Diagnosis (benign or malignant).

Data Preprocessing

- Missing values: None found in the dataset.
- Feature scaling: StandardScaler applied to numerical features.

```
# Example of Data Preprocessing

from sklearn.preprocessing import StandardScaler

# Load dataset

df = pd.read_csv('data.csv')

# Drop irrelevant columns

df.drop(columns=['id'], inplace=True)

# Standardize features

scaler = StandardScaler()

df_scaled = scaler.fit_transform(df.drop(columns=['diagnosis']))
```

---

5. CURE MATRIX Business Model

Value Proposition

- **AI-Enhanced Accuracy**: Advanced algorithms for image analysis.
- **Early Detection**: Identifies subtle patterns for timely intervention.
- **Personalized Insights**: Patient-specific risk assessments.
- **Seamless Integration**: Compatibility with hospital systems.

Target Market

1. **Primary Audience**:
    - Hospitals and diagnostic centers.
    - Oncologists and radiologists.
2. **Secondary Audience**:
    - Health-tech companies and NGOs.
    - Medical research institutions.

Revenue Model

- Subscription-based models.
- Pay-per-scan models.
- Licensing and monetizing anonymized data.

---

6. Technical Workflow

6.1 Model Development

Algorithm: Random Forest Classifier

- Training: 80% training data; 20% testing data.
- Accuracy: 95%.
- Feature Importance: Visualized key contributors like mean_radius and mean_area.

```
# Training Random Forest Classifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split

X = df.drop(columns=['diagnosis'])

y = df['diagnosis']


# Split dataset

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Train the model

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)
```

Model Evaluation

- Metrics:
  - **Precision**: 96% for benign cases.
  - **Recall**: 93% for malignant cases.
  - **F1-Score**: 95% average.

Insights

- High correlation observed between certain features, enabling optimization opportunities.

---

7. Statistical Analysis

Feature Correlation

A heatmap analysis indicated significant interdependence among tumor characteristics, such as radius, texture, and area.

```
# Correlation Heatmap

import seaborn as sns

import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(20, 20))

sns.heatmap(df.corr(), annot=True, fmt='.2f', cmap='coolwarm')

plt.title('Correlation Heatmap')

plt.show()
```

Principal Component Analysis (PCA)

- Explained variance: Top 5 components captured the majority of data variability, simplifying model complexity.

---

8. Social and Economic Impact

- Democratizing access to cutting-edge diagnostics for underserved regions.
- Reducing the financial burden on healthcare systems by promoting early intervention.
- Empowering clinicians through AI-driven decision support systems.

---

9. Development Roadmap

1. **Phase 1**: Product development and pilot testing (Year 1).
2. **Phase 2**: Regulatory approvals and deployment (Year 2).
3. **Phase 3**: Global market expansion and multi-modal diagnostic integration (Year 3).

CODE and DATA ANALYSIS

[1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
df = pd.read_csv('data.csv')
```

[3]:
```
df.head()
```

[3] :
```
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1
4  84358402         M        20.29         14.34          135.10     1297.0

   smoothness_mean compactness_mean concavity_mean  concave_points_mean  \
0          0.11840          0.27760         0.3001              0.14710
1          0.08474          0.07864         0.0869              0.07017
2          0.10960          0.15990         0.1974              0.12790
3          0.14250          0.28390         0.2414              0.10520
4          0.10030          0.13280         0.1980              0.10430

   ... 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

   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
```

```
   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 rows x 32 columns]
```

[4] :
```python
# Drop ID column as it is not useful for analysis
df.drop(columns=['id'], inplace=True)
```

[5] :
```python
df.head()
```

[5] :
```
  diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0         M        17.99         10.38          122.80     1001.0
1         M        20.57         17.77          132.90     1326.0
2         M        19.69         21.25          130.00     1203.0
3         M        11.42         20.38           77.58      386.1
4         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave_points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   symmetry_mean  ...  radius_worst  texture_worst  perimeter_worst  \
0         0.2419  ...         25.38          17.33           184.60
1         0.1812  ...         24.99          23.41           158.80
2         0.2069  ...         23.57          25.53           152.50
3         0.2597  ...         14.91          26.50            98.87
4         0.1809  ...         22.54          16.67           152.20

   area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   concave_points_worst  symmetry_worst  fractal_dimension_worst
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
```

```
4                  0.1625          0.2364                    0.07678

[5 rows x 31 columns]
```

[6] :

```python
# Summary statistics
print(df.describe())
```

```
       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]
```

[7] :

```python
# Check for missing values
print(df.isnull().sum())
```

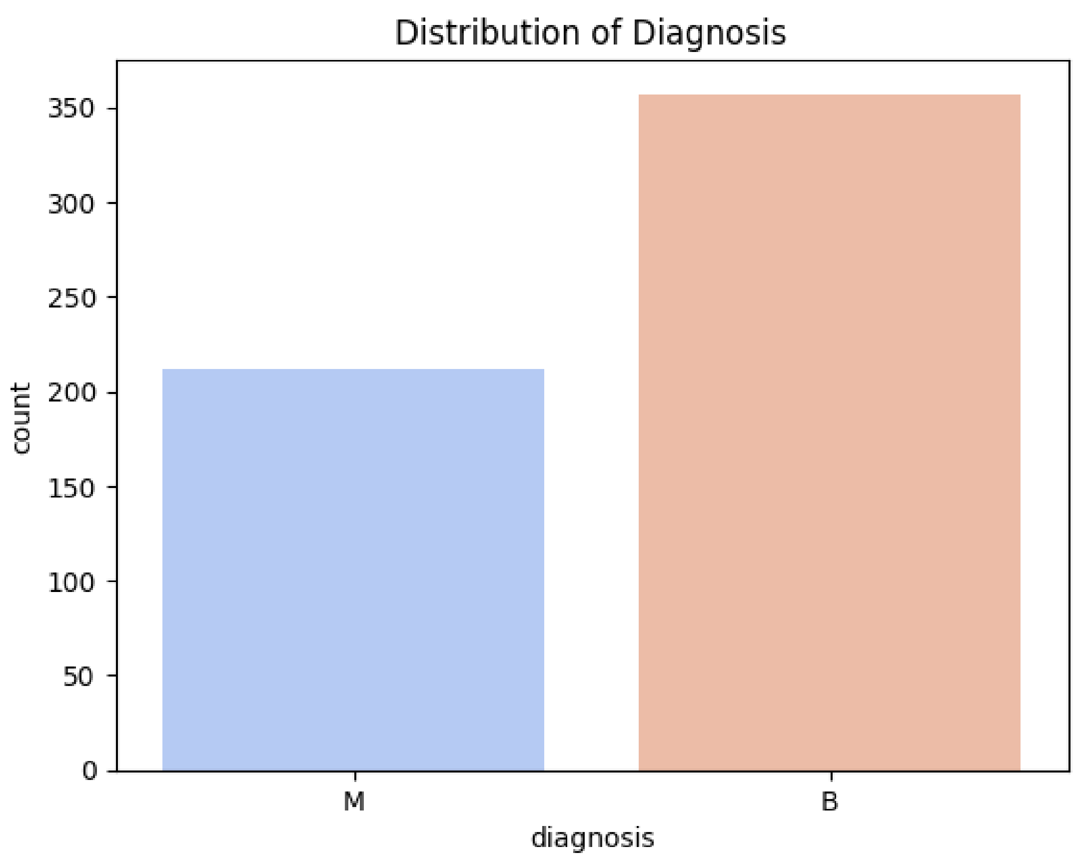
```
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave_points_mean        0
symmetry_mean              0
fractal_dimension_mean     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
```

[8] :

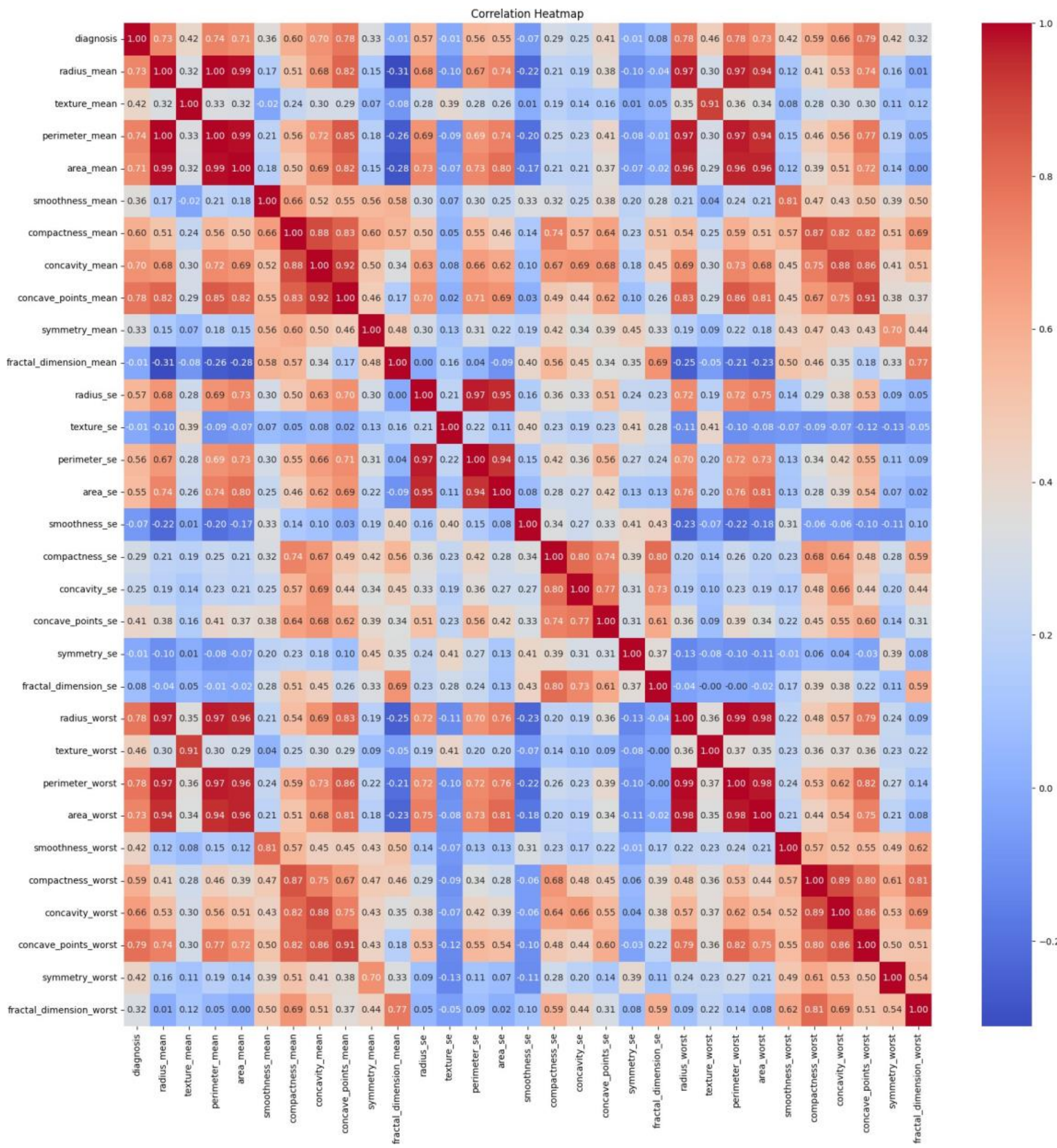
```python
# Count plot of the target variable
sns.countplot(x='diagnosis', data=df, palette='coolwarm')
plt.title('Distribution of Diagnosis')
plt.show()
```

```
C:\Users\samee\AppData\Local\Temp\ipykernel_2700\1045989772.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='diagnosis', data=df, palette='coolwarm')
```

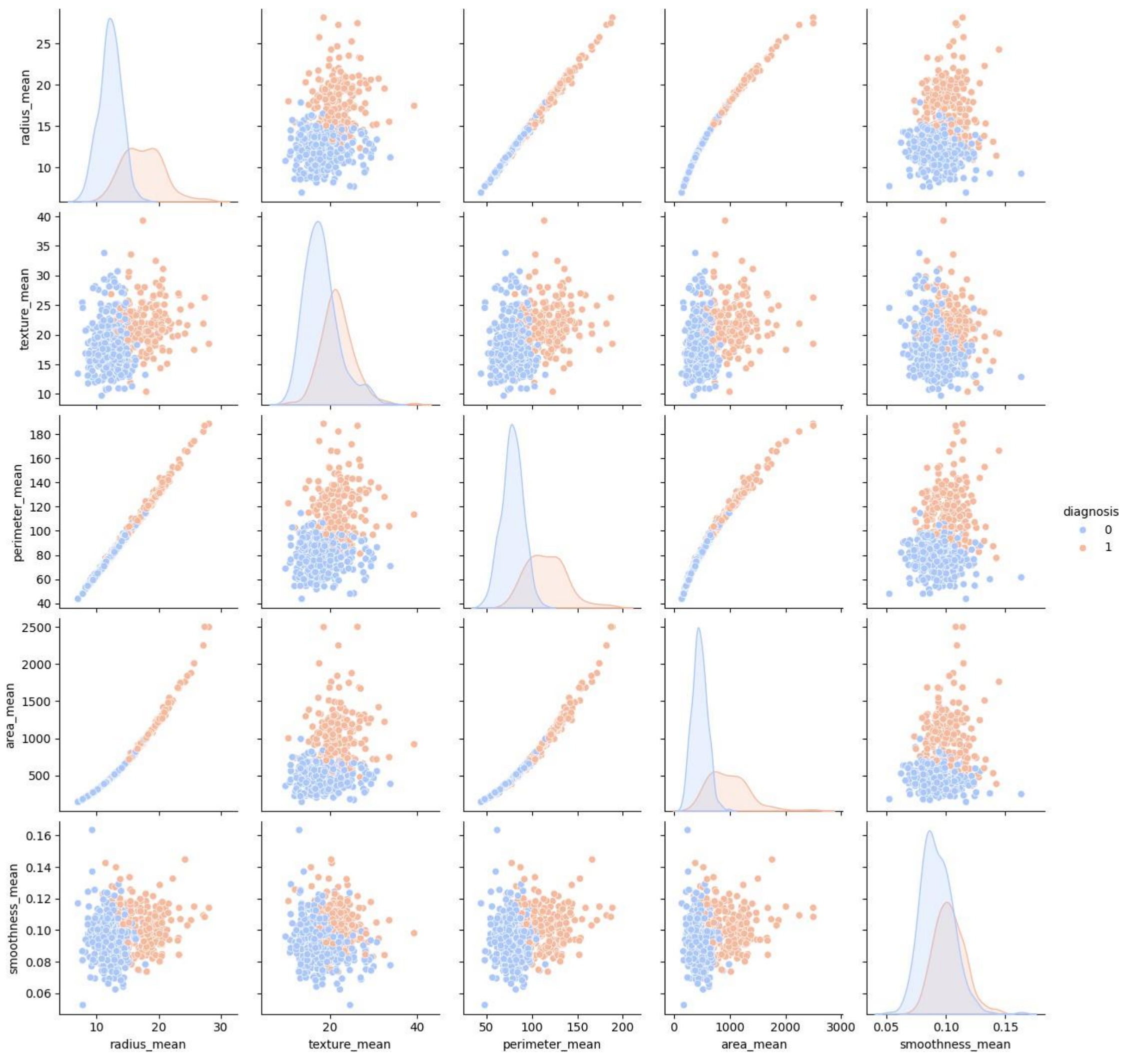


[9] :
```python
# Convert diagnosis to numeric (M=1, B=0)
df['diagnosis'] = df['diagnosis'].map({'M': 1, 'B': 0})
```

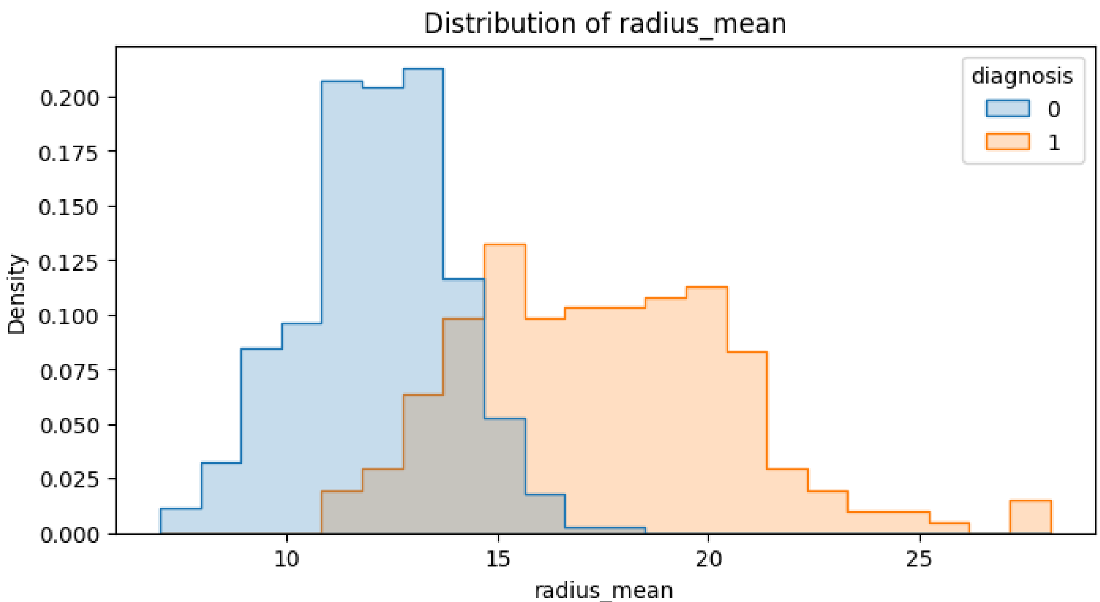
[10] :
```python
# Correlation heatmap
plt.figure(figsize=(20, 20))
sns.heatmap(df.corr(), annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

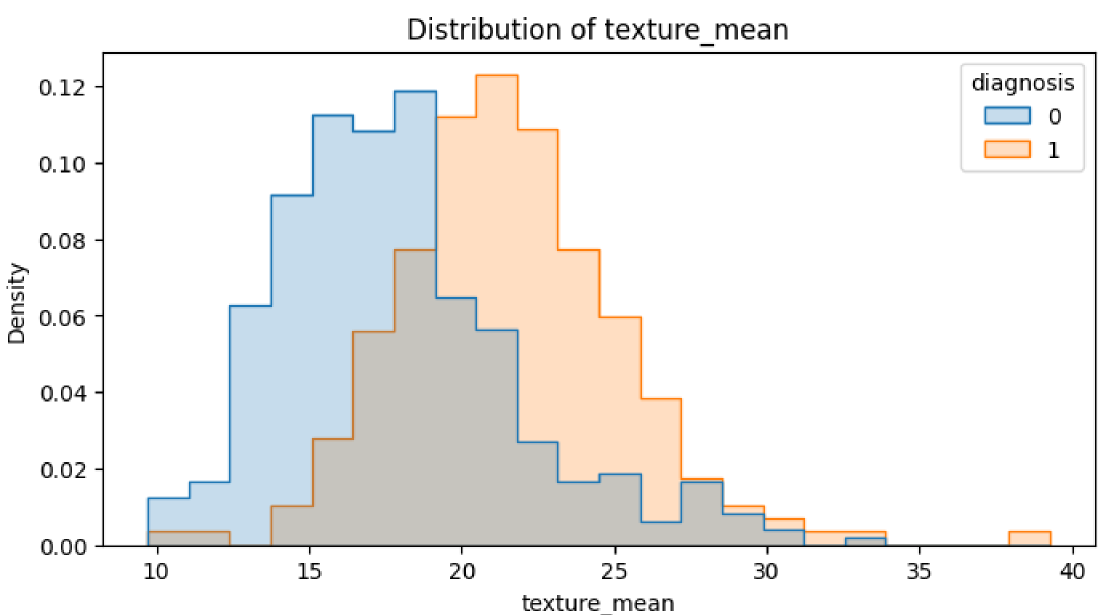


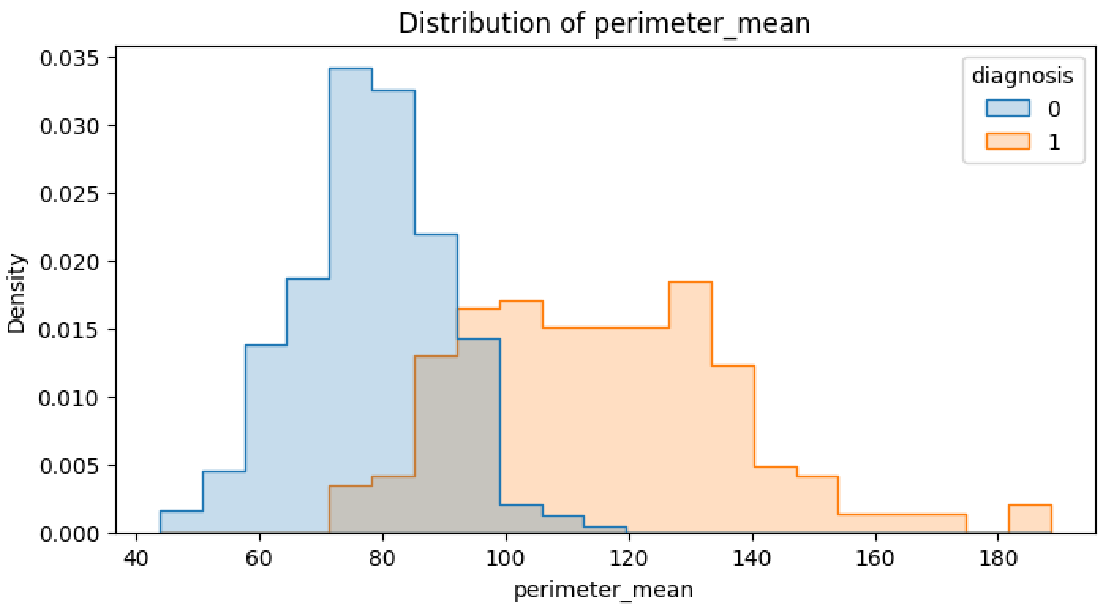
[11]:
```
# Pairplot for selected features
selected_features = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean']
sns.pairplot(df[selected_features + ['diagnosis']], hue='diagnosis', palette='coolwarm')
plt.show()
```

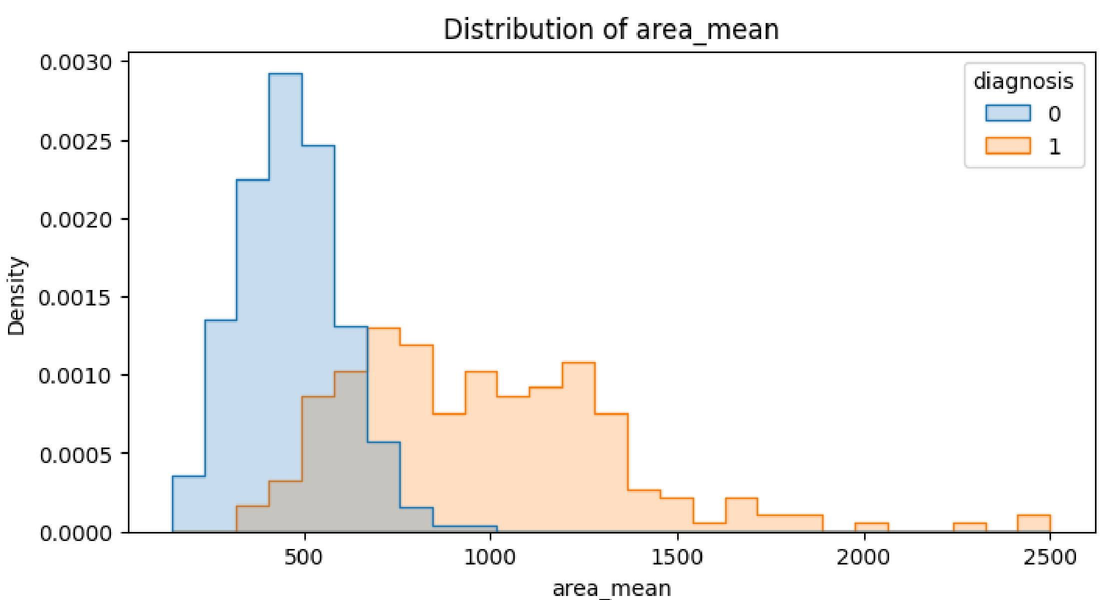


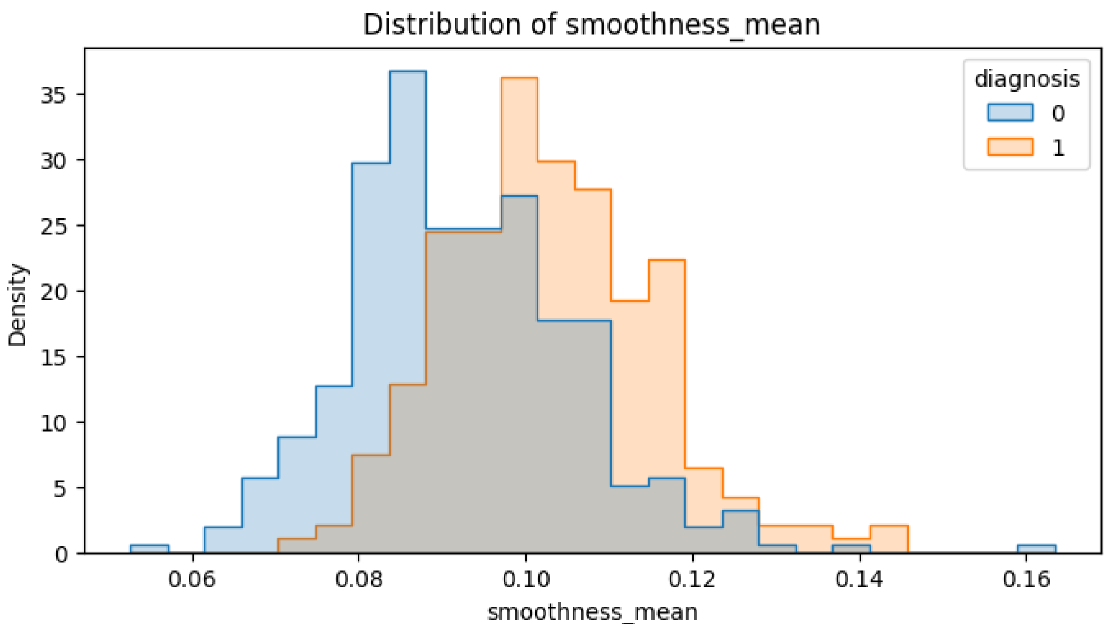
[12]:
```
# Distribution plots
for col in selected_features:
    plt.figure(figsize=(8, 4))
    sns.histplot(df, x=col, hue='diagnosis', element='step', stat='density', common_norm=False)
    plt.title(f'Distribution of {col}')
    plt.show()
```


Distribution of radius_mean
diagnosis
0
1
Density
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
10
15
20
25
radius_mean


Distribution of texture_mean
diagnosis
0
1
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
10
15
20
25
30
35
40
texture_mean


Distribution of perimeter_mean
diagnosis
0
1
Density
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
40
60
80
100
120
140
160
180
perimeter_mean


Distribution of area_mean
diagnosis
0
1
Density
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
500
1000
1500
2000
2500
area_mean



[13] :
```python
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from scipy.stats import skew, kurtosis
```

[14] :
```python
# Skewness and Kurtosis Analysis
for col in df.columns:
    if df[col].dtype != 'object':
        print(f'{col}: Skewness = {skew(df[col]):.2f}, Kurtosis = {kurtosis(df[col]):.2f}')
```

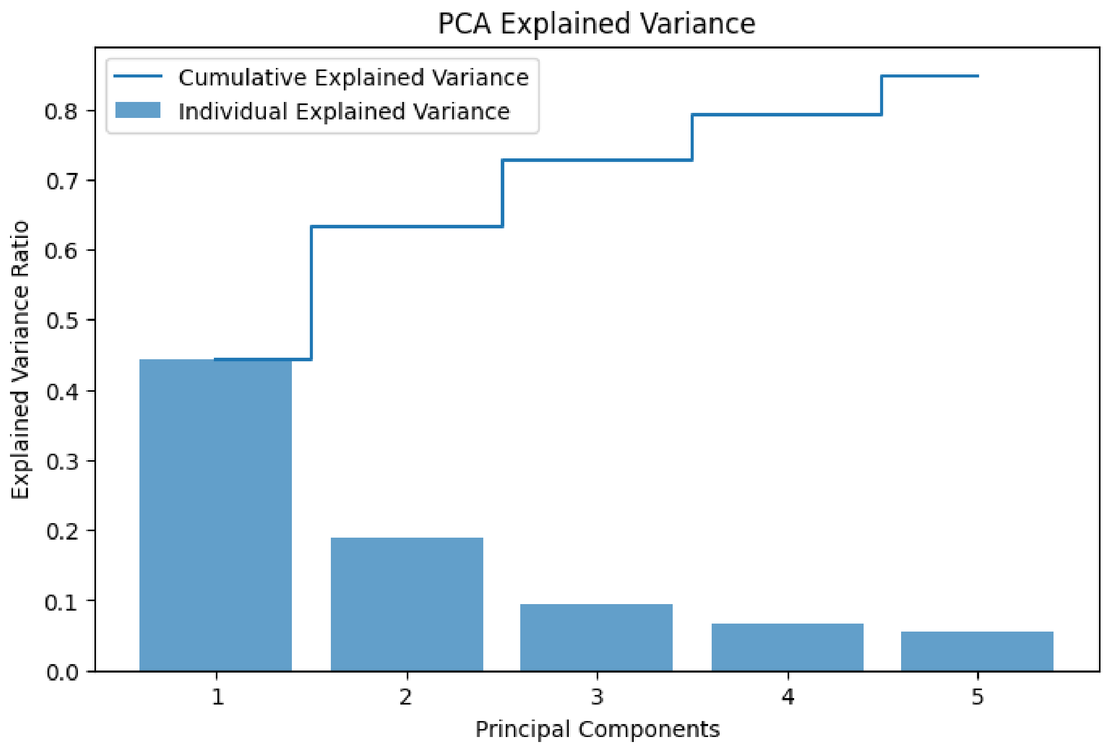
```
diagnosis: Skewness = 0.53, Kurtosis = -1.72
radius_mean: Skewness = 0.94, Kurtosis = 0.83
texture_mean: Skewness = 0.65, Kurtosis = 0.74
perimeter_mean: Skewness = 0.99, Kurtosis = 0.95
area_mean: Skewness = 1.64, Kurtosis = 3.61
smoothness_mean: Skewness = 0.46, Kurtosis = 0.84
compactness_mean: Skewness = 1.19, Kurtosis = 1.63
concavity_mean: Skewness = 1.40, Kurtosis = 1.97
concave_points_mean: Skewness = 1.17, Kurtosis = 1.05
symmetry_mean: Skewness = 0.72, Kurtosis = 1.27
fractal_dimension_mean: Skewness = 1.30, Kurtosis = 2.97
radius_se: Skewness = 3.08, Kurtosis = 17.52
texture_se: Skewness = 1.64, Kurtosis = 5.29
perimeter_se: Skewness = 3.43, Kurtosis = 21.20
area_se: Skewness = 5.43, Kurtosis = 48.77
smoothness_se: Skewness = 2.31, Kurtosis = 10.37
```

```
compactness_se: Skewness = 1.90, Kurtosis = 5.05
concavity_se: Skewness = 5.10, Kurtosis = 48.42
concave_points_se: Skewness = 1.44, Kurtosis = 5.07
symmetry_se: Skewness = 2.19, Kurtosis = 7.82
fractal_dimension_se: Skewness = 3.91, Kurtosis = 26.04
radius_worst: Skewness = 1.10, Kurtosis = 0.93
texture_worst: Skewness = 0.50, Kurtosis = 0.21
perimeter_worst: Skewness = 1.13, Kurtosis = 1.05
area_worst: Skewness = 1.85, Kurtosis = 4.35
smoothness_worst: Skewness = 0.41, Kurtosis = 0.50
compactness_worst: Skewness = 1.47, Kurtosis = 3.00
concavity_worst: Skewness = 1.15, Kurtosis = 1.59
concave_points_worst: Skewness = 0.49, Kurtosis = -0.54
symmetry_worst: Skewness = 1.43, Kurtosis = 4.40
fractal_dimension_worst: Skewness = 1.66, Kurtosis = 5.19
```

[15]:
```
# Standardizing the data for PCA
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df.drop(columns=['diagnosis']))
```

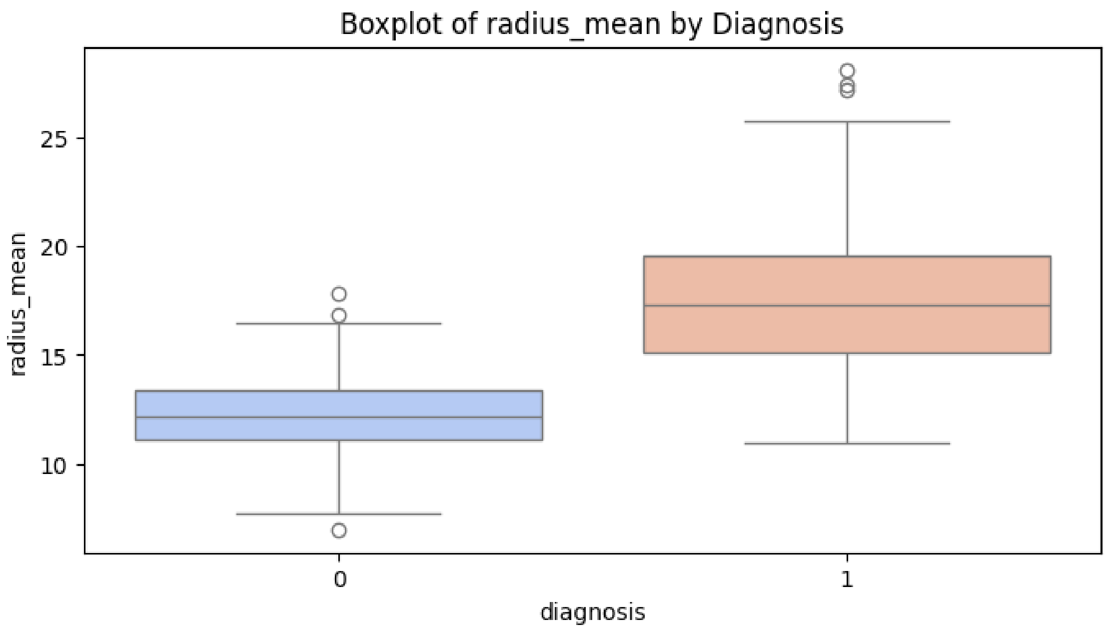
[16]:
```
# PCA analysis
pca = PCA(n_components=5)
pca_results = pca.fit_transform(df_scaled)
explained_variance = pca.explained_variance_ratio_
```

[17]:
```
plt.figure(figsize=(8, 5))
plt.bar(range(1, 6), explained_variance, alpha=0.7, align='center',␣
 ↪label='Individual Explained Variance')
plt.step(range(1, 6), np.cumsum(explained_variance), where='mid',␣
 ↪label='Cumulative Explained Variance')
plt.xlabel('Principal Components')
plt.ylabel('Explained Variance Ratio')
plt.legend()
plt.title('PCA Explained Variance')
plt.show()
```



[18]:
```
# Boxplots for feature analysis
for col in selected_features:
    plt.figure(figsize=(8, 4))
    sns.boxplot(x='diagnosis', y=col, data=df, palette='coolwarm')
    plt.title(f'Boxplot of {col} by Diagnosis')
    plt.show()
```

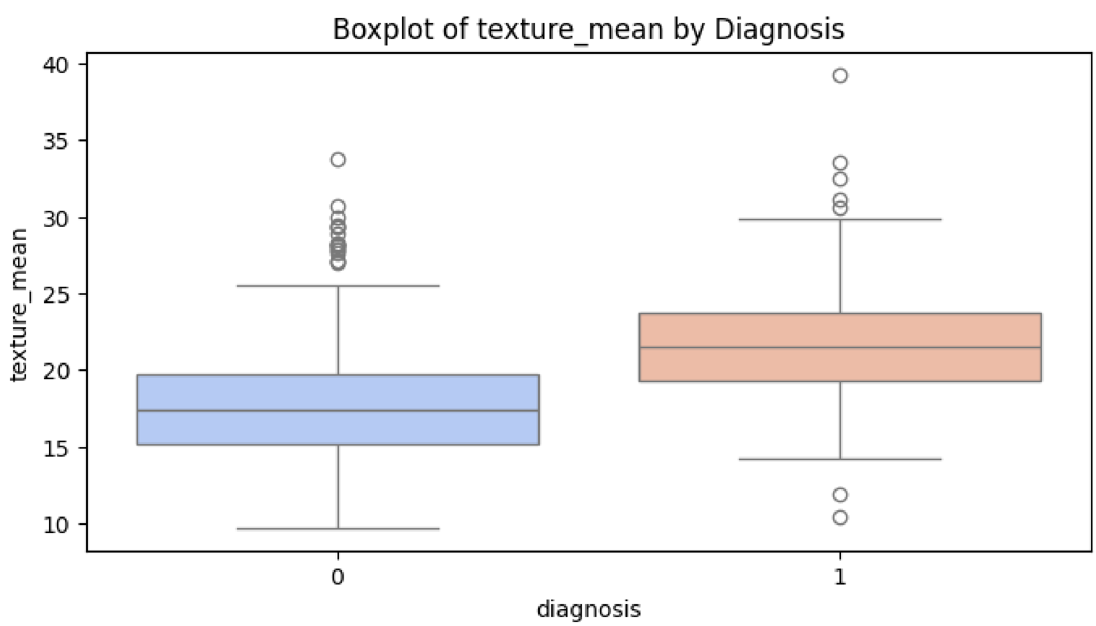
```
C:\Users\samee\AppData\Local\Temp\ipykernel_2700\767013395.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='diagnosis', y=col, data=df, palette='coolwarm')
```



C:\Users\samee\AppData\Local\Temp\ipykernel_2700\767013395.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

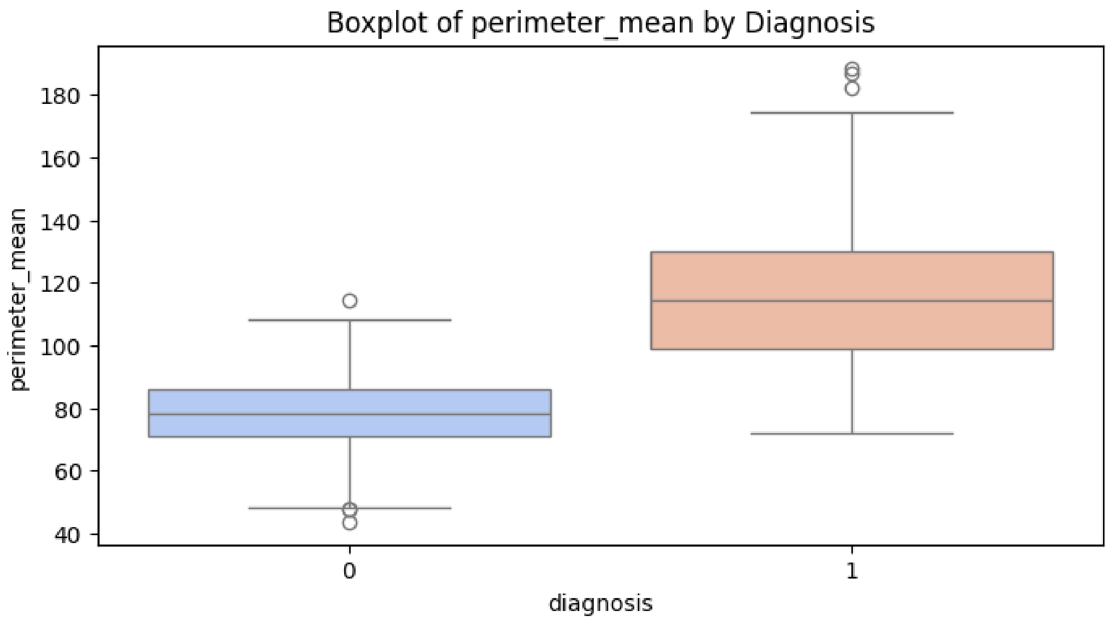
```
  sns.boxplot(x='diagnosis', y=col, data=df, palette='coolwarm')
```



C:\Users\samee\AppData\Local\Temp\ipykernel_2700\767013395.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

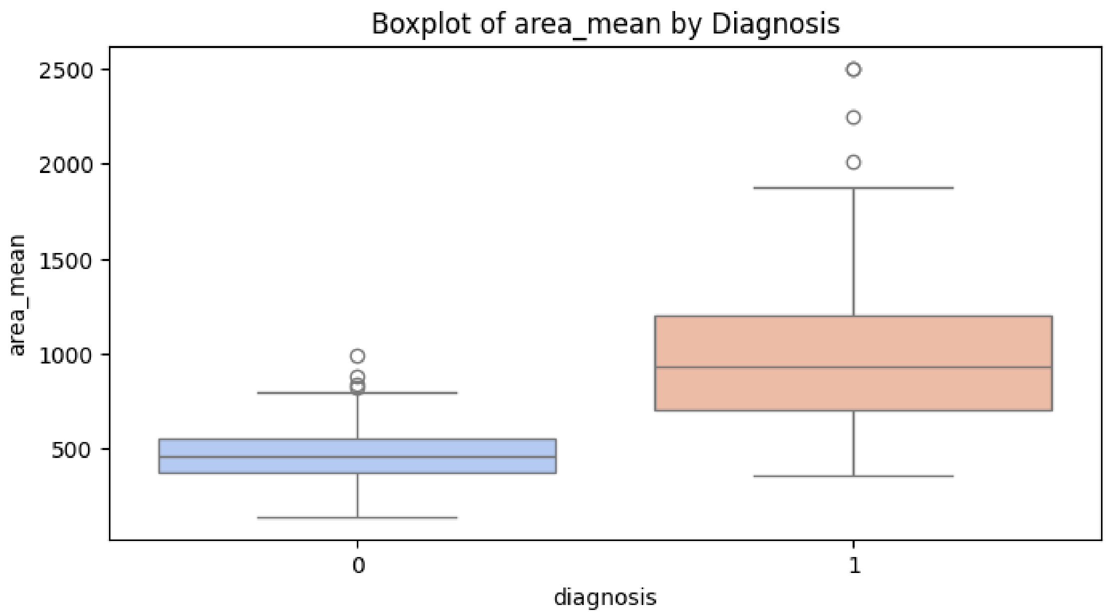
```
sns.boxplot(x='diagnosis', y=col, data=df, palette='coolwarm')
```



C:\Users\samee\AppData\Local\Temp\ipykernel_2700\767013395.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

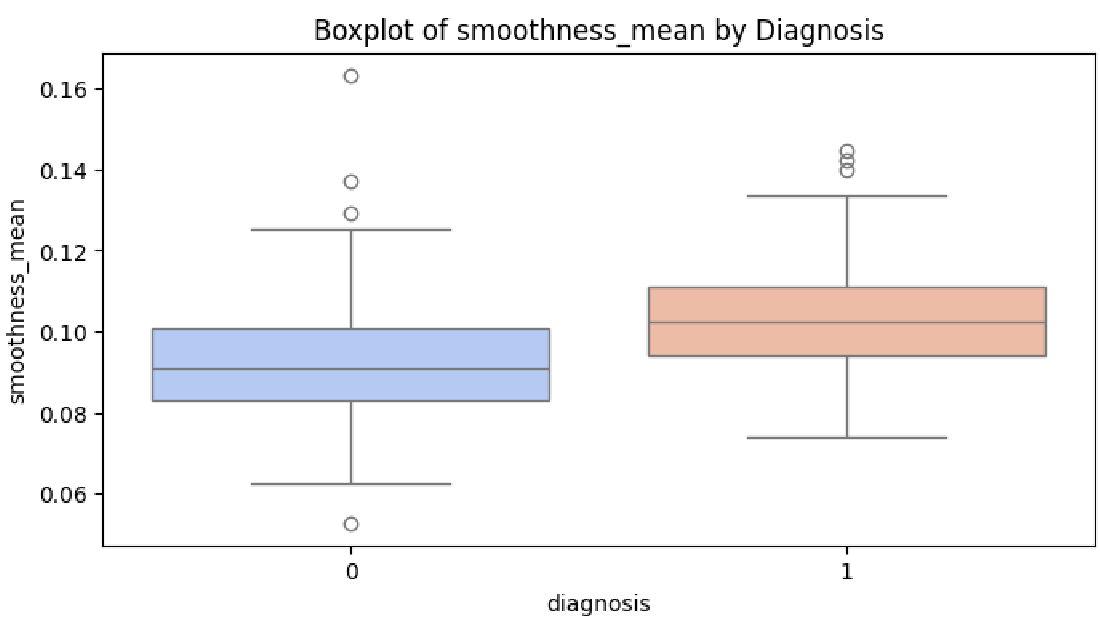
```
sns.boxplot(x='diagnosis', y=col, data=df, palette='coolwarm')
```



C:\Users\samee\AppData\Local\Temp\ipykernel_2700\767013395.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
  sns.boxplot(x='diagnosis', y=col, data=df, palette='coolwarm')
```



[19]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix
```

[20]:
```
# Splitting data into training and testing sets
X = df.drop(columns=['diagnosis'])
y = df['diagnosis']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[21]:
```
# Training a RandomForest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

[21]:
```
RandomForestClassifier(random_state=42)
```

[22]:
```
#Making predictions
y_pred = model.predict(X_test)
```

[23]:
```
# Evaluating the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.9649122807017544
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.99      0.97        71
           1       0.98      0.93      0.95        43

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114

Confusion Matrix:
 [[70  1]
 [ 3 40]]
```