

Student Name : Rahaf Barakat - id : 253002370

Breast Cancer Dataset

```
In [82]: from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
In [83]: import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Dataset Link : <https://www.kaggle.com/datasets/utkarshx27/breast-cancer-dataset-used-royston-and-altman>

```
In [84]: df = pd.read_csv('/content/drive/MyDrive/My Project 1/gbseg.csv')
```

```
In [85]: df.head()
```

Out[85]:

	Unnamed: 0	pid	age	meno	size	grade	nodes	pgr	er	hormon	rfstime	status
0	1	132	49	0	18	2	2	0	0	0	1838	0
1	2	1575	55	1	20	3	16	0	0	0	403	1
2	3	1140	56	1	40	3	3	0	0	0	1603	0
3	4	769	45	0	25	3	1	0	4	0	177	0
4	5	130	65	1	30	2	5	0	36	1	1855	0

```
In [86]: df.dtypes
```

Out[86]:

	0
Unnamed: 0	int64
pid	int64
age	int64
meno	int64
size	int64
grade	int64
nodes	int64
pgr	int64
er	int64
hormon	int64
rfstime	int64
status	int64

dtype: object

```
In [87]: df.columns
```

Out[87]: Index(['Unnamed: 0', 'pid', 'age', 'meno', 'size', 'grade', 'nodes', 'pgr',
'er', 'hormon', 'rfstime', 'status'],
dtype='object')

```
In [88]: df.isnull().sum()
```

Out[88]:

	0
Unnamed: 0	0
pid	0
age	0
meno	0
size	0
grade	0
nodes	0
pgr	0
er	0
hormon	0
rfstime	0
status	0

dtype: int64

```
In [89]: missing_percentage = df.isnull().mean() * 100

threshold = 30

# Filter columns with more than the threshold percentage of missing values
columns_to_drop = missing_percentage[missing_percentage > threshold].index

# Drop the columns
df = df.drop(columns=columns_to_drop)

# Display the resulting dataframe after removing columns with too many missing values
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")
```

Columns dropped: []
Remaining columns: Index(['Unnamed: 0', 'pid', 'age', 'meno', 'size', 'grade', 'nodes', 'pgr',
'er', 'hormon', 'rfstime', 'status'],
dtype='object')

```
In [90]: from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
    df[column] = label_encoder.fit_transform(df[column])
df.head()
```

Out[90]:

	Unnamed: 0	pid	age	meno	size	grade	nodes	pgr	er	hormon	rfstime	status
0	1	132	49	0	18	2	2	0	0	0	1838	0
1	2	1575	55	1	20	3	16	0	0	0	403	1
2	3	1140	56	1	40	3	3	0	0	0	1603	0
3	4	769	45	0	25	3	1	0	4	0	177	0
4	5	130	65	1	30	2	5	0	36	1	1855	0

```
In [91]: # Fill missing values in columns with mode
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean()

    # Get the mode (most frequent value)
    df[column] = df[column].fillna(mean_value)

# Fill missing values in categorical columns with mode
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0]

    # Get the mode (most frequentvalue)
    df[column] = df[column].fillna(mode_value)
df.isnull().sum()
```

Out[91]:

	0
Unnamed: 0	0
pid	0
age	0
meno	0
size	0
grade	0
nodes	0
pgr	0
er	0
hormon	0
rfstime	0
status	0

dtype: int64

```
In [101... import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import KNNImputer
```

```
In [102... data = df.copy()

# Define numeric columns to handle
numeric_cols = ['age', 'size', 'nodes', 'pgr', 'er', 'rfstime']

def detect_outliers_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return (column < lower_bound) | (column > upper_bound)

for col in numeric_cols:
    data[f'{col}_Outlier_IQR'] = detect_outliers_iqr(data[col])

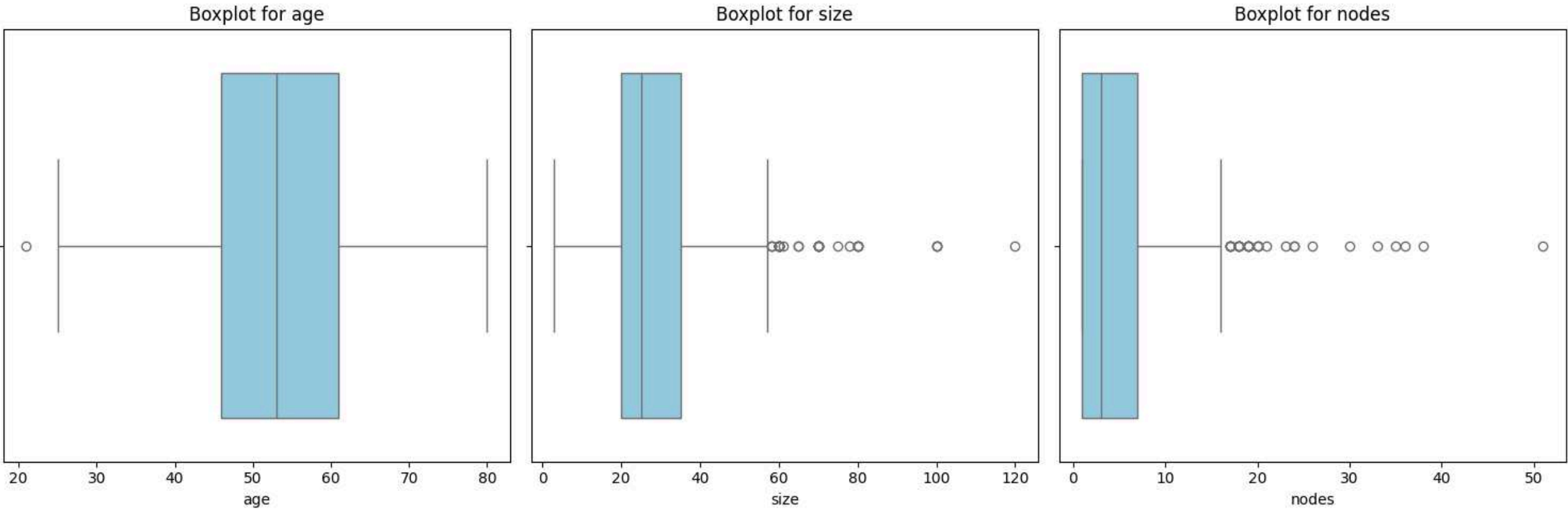
# Display rows with outliers
outliers_iqr = data[data.filter(like='_Outlier_IQR').any(axis=1)]
print("Number of outliers by IQR:", len(outliers_iqr))
display(outliers_iqr.head())
```

Number of outliers by IQR: 169

	Unnamed: 0	pid	age	meno	size	grade	nodes	pgr	er	hormon	rfstime	status	age_Outlier_IQR	size_Outlier_IQR	nodes_Outlier_IQR	pgr_Outlier_IQR	er_Outlier_IQR	rfstime_Outlier_IQR
14	15	820	32	0	57	3	24	0	13	0	448	1	False	False	True	False	False	False
26	27	1318	63	1	21	2	1	0	378	0	1356	0	False	False	False	False	True	False
40	41	129	50	0	60	3	4	0	0	0	446	1	False	True	False	False	False	False
43	44	1368	53	1	58	2	1	0	0	1	1483	0	False	True	False	False	False	False
57	58	382	70	1	80	3	8	0	0	0	1449	1	False	True	False	False	False	False

```
In [104... import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(15, 5))
for i, col in enumerate(numeric_cols[:3]): # first 3 numeric columns
    plt.subplot(1, 3, i+1)
    sns.boxplot(x=data[col], color='skyblue')
    plt.title(f"Boxplot for {col}")
plt.tight_layout()
plt.show()
```



```
In [105... # detect outliers in a column using Z-Score

def detect_outliers_zscore(column, threshold=3):
    mean = np.mean(column)
    std_dev = np.std(column)
    z_scores = (column - mean) / std_dev
    return np.abs(z_scores) > threshold

for col in numeric_cols:
    data[f'{col}_Outlier_Z'] = detect_outliers_zscore(data[col])

# Display rows with outliers by Z-Score
outliers_z = data[data.filter(like='_Outlier_Z').any(axis=1)]
print("Number of outliers by Z-Score:", len(outliers_z))
display(outliers_z.head())
```

Number of outliers by Z-Score: 43

	Unnamed: 0	pid	age	meno	size	grade	nodes	pgr	er	hormon	...	nodes_Outlier_IQR	pgr_Outlier_IQR	er_Outlier_IQR	rfstime_Outlier_IQR	age_Outlier_Z	size_Outlier_Z	nodes_Outlier_Z	pgr_Outlier_Z	er_Outlier_Z	rfstime_Outlier_Z
14	15	820	32	0	57	3	24	0	13	0	...	True	False	False	False	False	False	True	False	False	False
57	58	382	70	1	80	3	8	0	0	0	...	False	False	False	False	False	True	False	False	False	False
60	61	656	41	0	80	2	1	0	0	1	...	False	False	False	False	False	True	False	False	False	False
65	66	825	80	1	39	2	30	0	59	0	...	True	False	False	False	False	False	True	False	False	False
114	115	844	43	0	80	2	20	2	25	0	...	True	False	False	False	False	True	False	False	False	False

5 rows × 24 columns

In [106...

```
# Replace outliers with NaN
def replace_outliers_with_nan(column):
    mean = np.mean(column)
    std = np.std(column)
    z_scores = (column - mean) / std
    return column.where(np.abs(z_scores) <= 3, np.nan)

for col in numeric_cols:
    data[col] = replace_outliers_with_nan(data[col])

# Display missing values count after replacement
null_counts = data[numeric_cols].isnull().sum()
print("Missing values after replacing outliers with NaN:")
print(null_counts)
```

Missing values after replacing outliers with NaN:

age	1
size	10
nodes	10
pgr	14
er	14
rfstime	0

dtype: int64

In [119...

```
# Impute missing values using KNN

imputer = KNNImputer(n_neighbors=2)
data[numeric_cols] = pd.DataFrame(imputer.fit_transform(data[numeric_cols]), columns=numeric_cols)

print("Missing values after KNN Imputer:")
print(data[numeric_cols].isnull().sum())
```

Missing values after KNN Imputer:

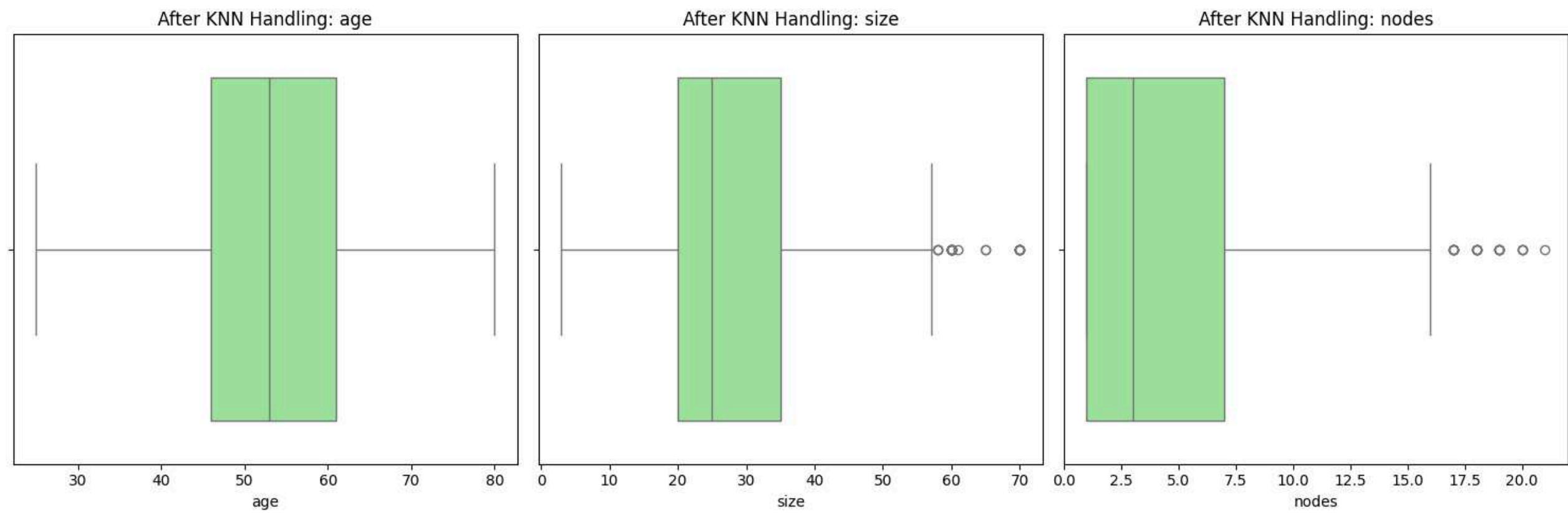
age	0
size	0
nodes	0
pgr	0
er	0
rfstime	0

dtype: int64

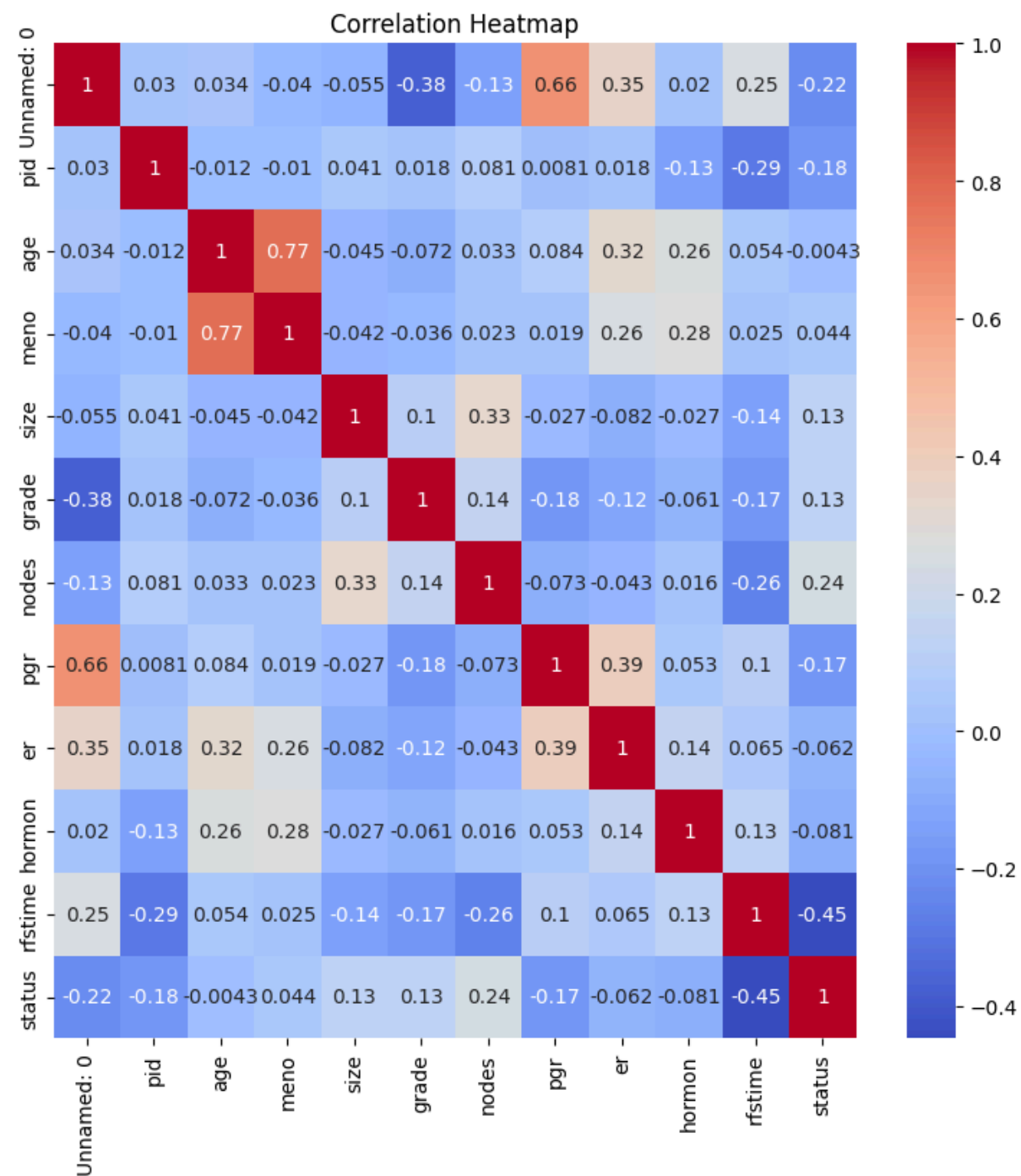
In [120...

```
# Boxplot after handling for comparison (first 3 numeric columns only)

plt.figure(figsize=(15, 5))
for i, col in enumerate(numeric_cols[:3]): # first 3 numeric columns
    plt.subplot(1, 3, i+1)
    sns.boxplot(x=data[col], color='lightgreen')
    plt.title(f"After KNN Handling: {col}")
plt.tight_layout()
plt.show()
```



```
In [121... # Correlation heatmap
plt.figure(figsize=(9, 9))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



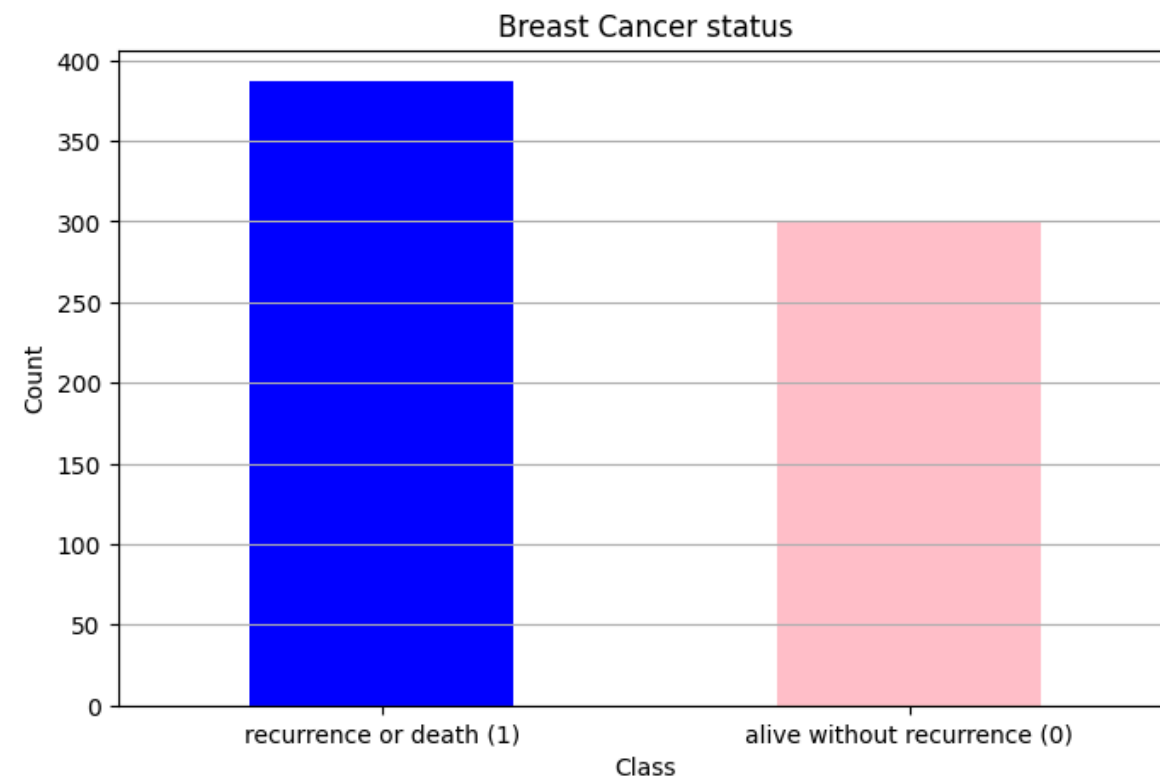
```
In [122...] df['status'].value_counts()
```

```
Out[122...] count
status
0      387
1      299
```

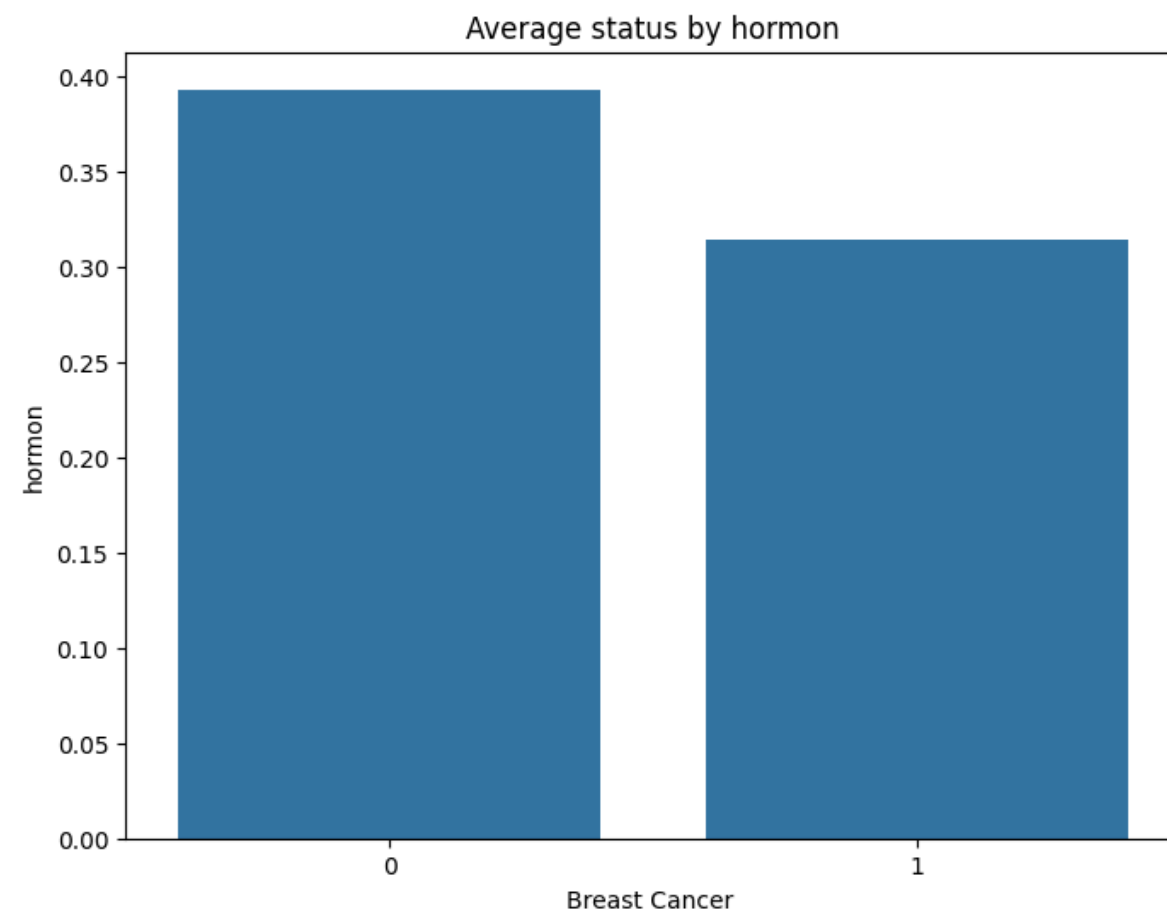
dtype: int64

```
In [123...] # Create a bar chart
plt.figure(figsize=(8, 5))
df['status'].value_counts().plot(kind='bar', color=['blue',
'pink'])
plt.xticks([0, 1], ['recurrence or death (1)', 'alive without recurrence (0)'], rotation=0)
plt.title('Breast Cancer status')
plt.xlabel('Class')
plt.ylabel('Count')
```

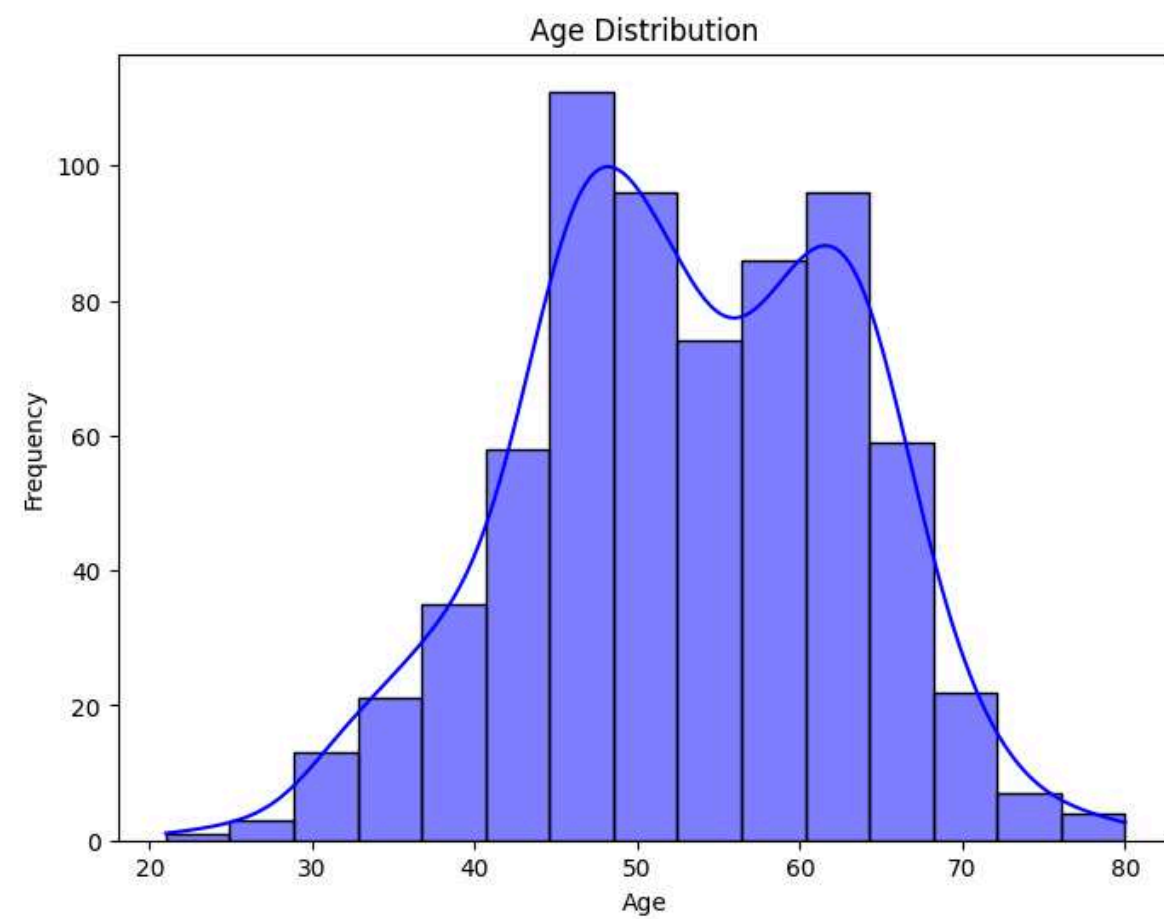
```
plt.grid(axis='y')
plt.show()
```



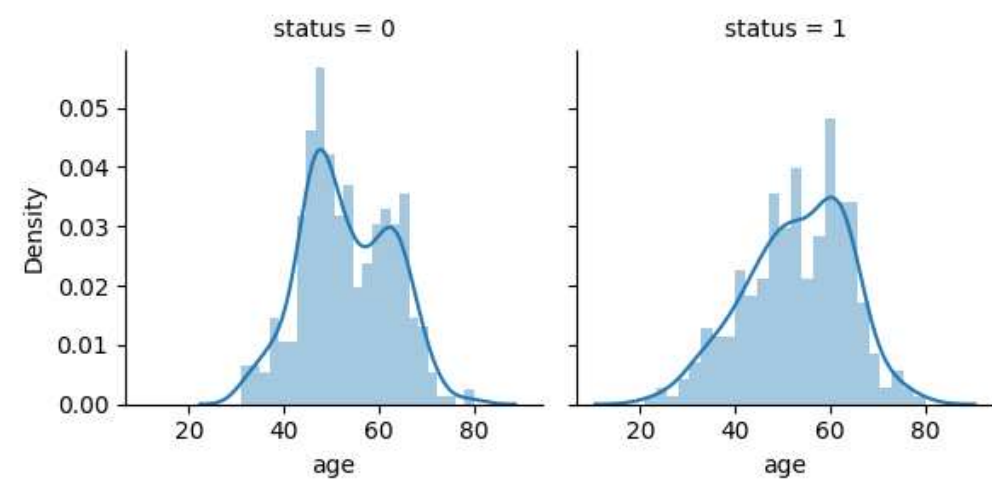
```
In [124... # Average Status by hormon
import seaborn as sns
plt.figure(figsize=(8,6))
sns.barplot(x='status', y='hormon', data=df, estimator='mean',
ci=None)
plt.title("Average status by hormon")
plt.xlabel("Breast Cancer ")
plt.ylabel("hormon")
plt.show()
```



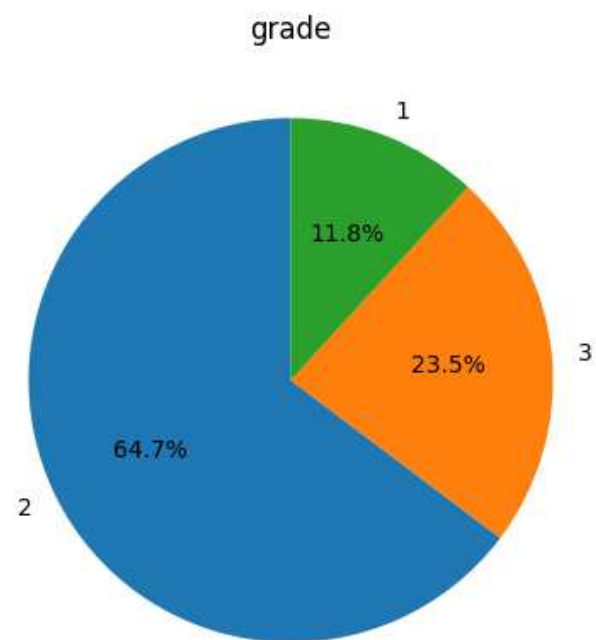
```
In [125... cat_cols = [col for col in df.columns if df[col].dtype=='object']
num_cols = [col for col in df.columns if df[col].dtype!='object']
for col in cat_cols:
    print(f'{col} has {df[col].unique()} values')
# 1. Distribution of Age
plt.figure(figsize=(8, 6))
sns.histplot(df['age'], bins=15, kde=True, color='blue')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



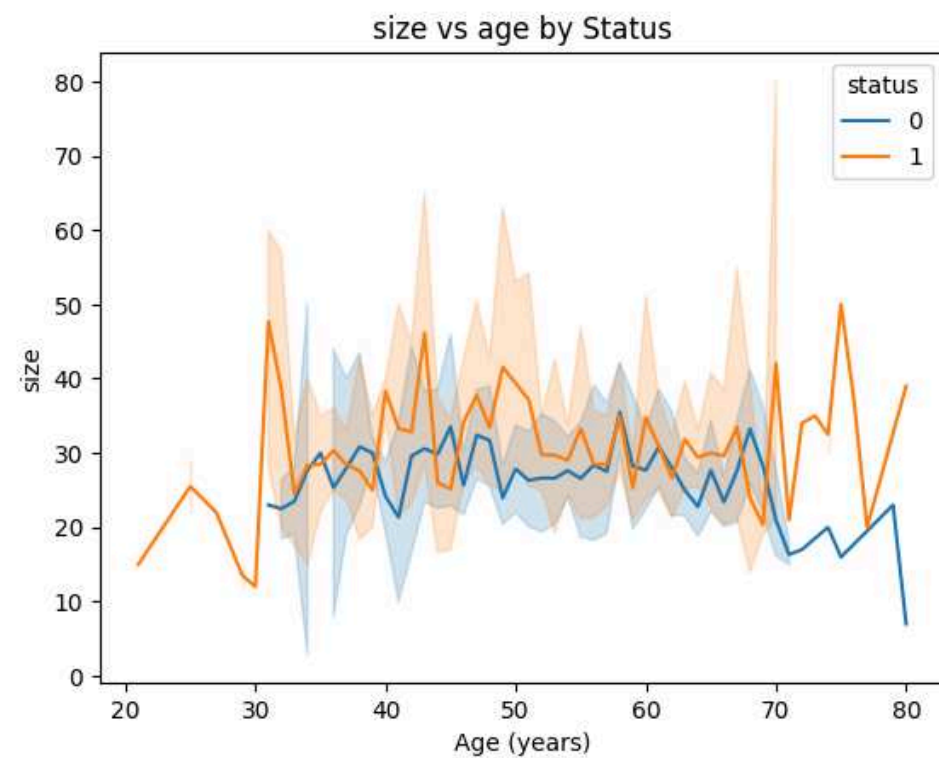
```
In [126... g = sns.FacetGrid(df,col="status")
g.map(sns.distplot,"age", bins=25)
plt.show()
```



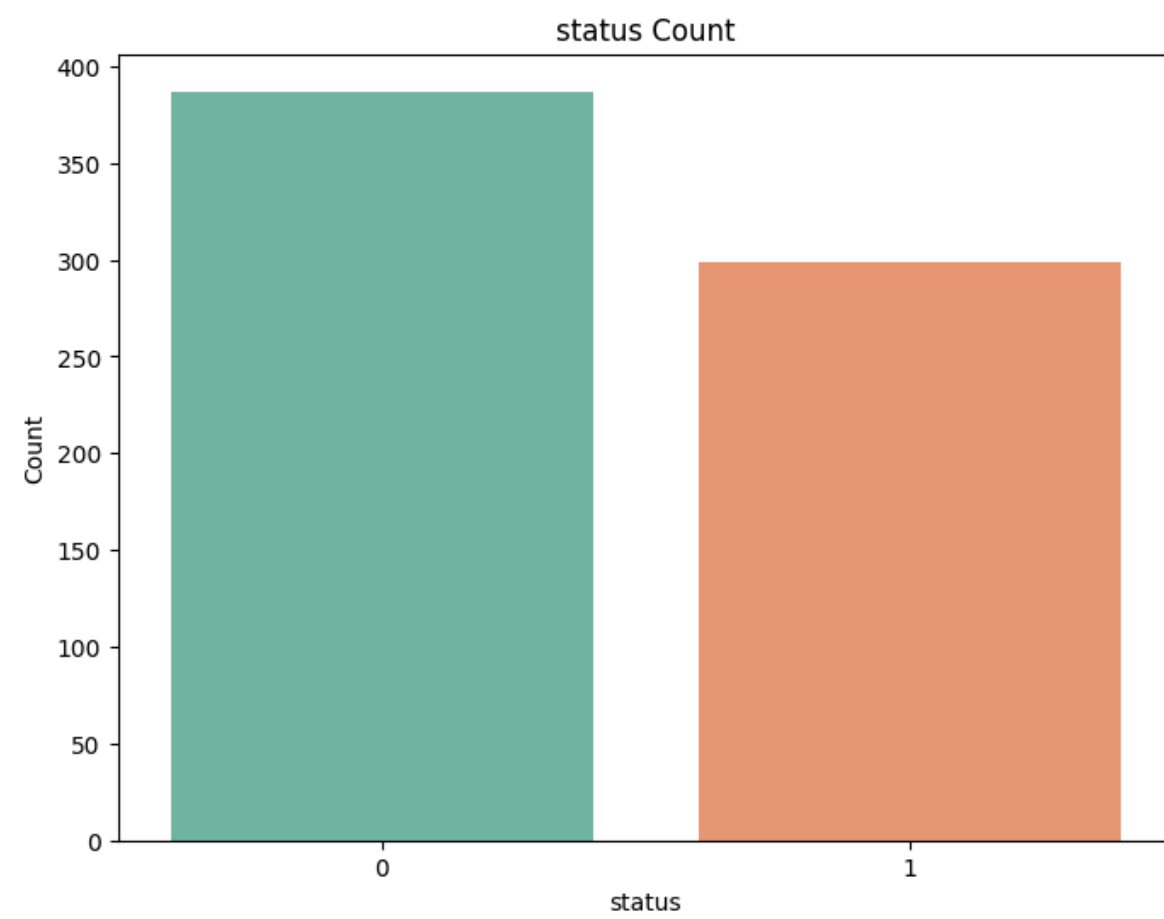
```
In [127... # grade of the breast cancer
fbs_counts = df['grade'].value_counts()
plt.figure(figsize=(5,5))
plt.pie(fbs_counts, labels=fbs_counts.index, autopct='%1.1f%%',
startangle=90)
plt.title("grade")
plt.show()
```



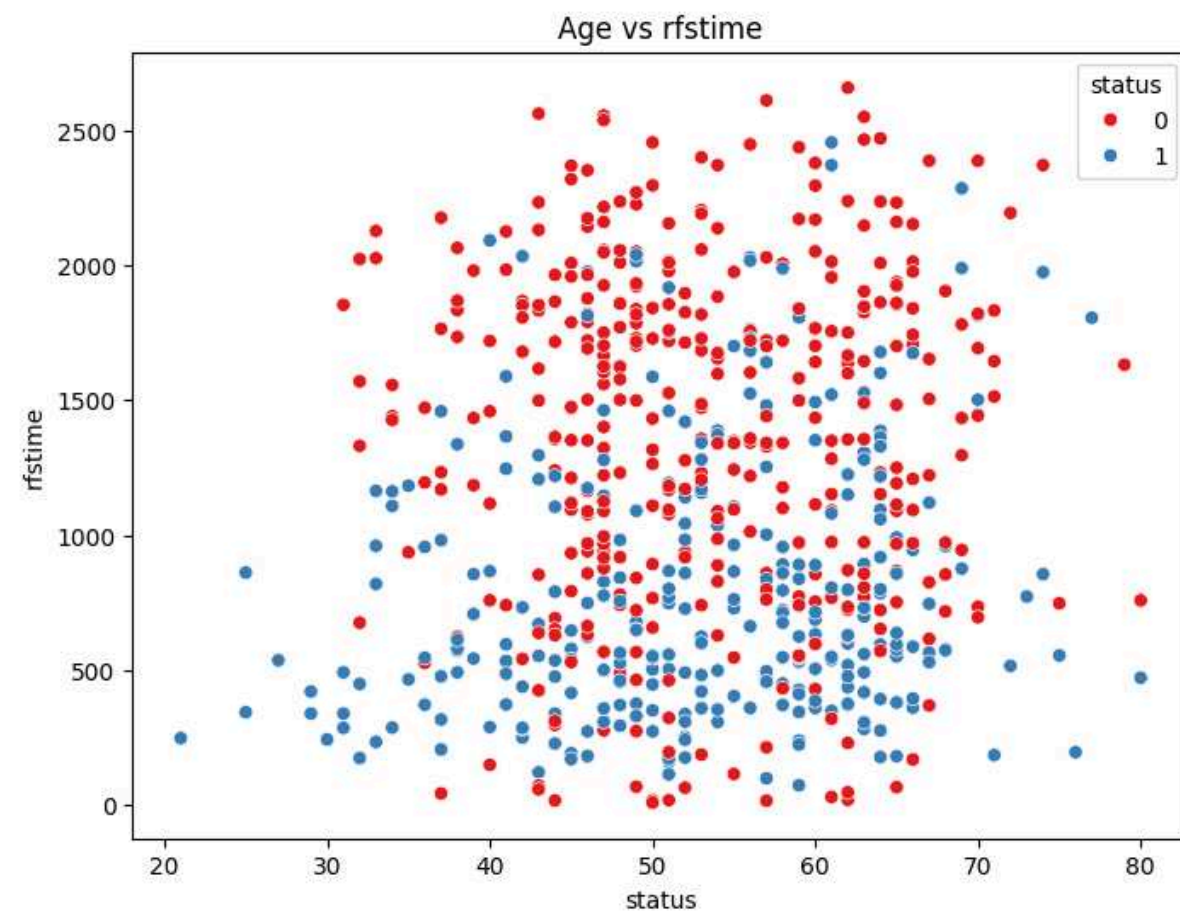
```
In [128... # Boxplot for Age vs size by Status
sns.lineplot(x='age', y='size', hue='status', data=df)
plt.title("size vs age by Status")
plt.xlabel("Age (years)")
plt.ylabel("size")
plt.show()
```



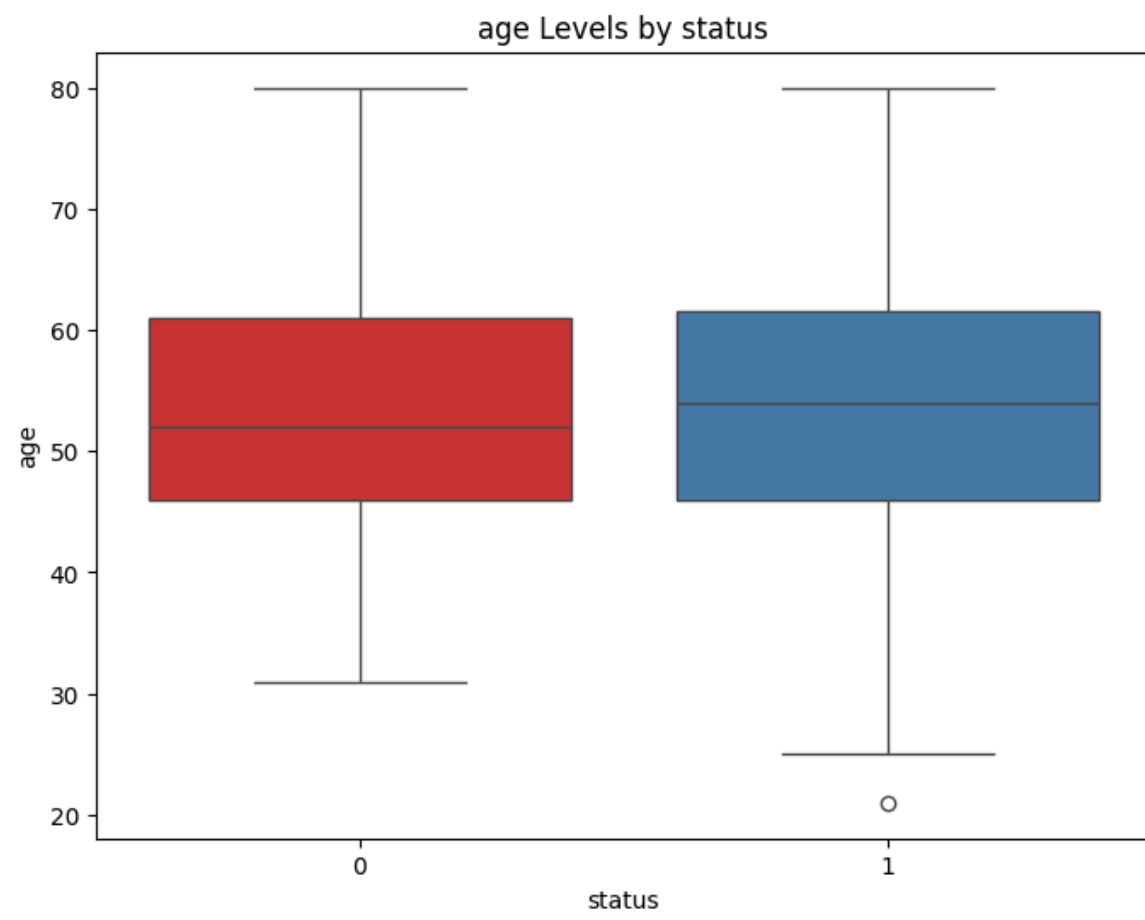
```
In [129... # Countplot for status count
plt.figure(figsize=(8, 6))
sns.countplot(x='status', data=df, palette='Set2')
plt.title('status Count')
plt.xlabel('status')
plt.ylabel('Count')
plt.show()
```



```
In [130... # Age vs rfstime
plt.figure(figsize=(8, 6))
sns.scatterplot(x='age', y='rfstime', data=df, hue='status',
palette='Set1')
plt.title('Age vs rfstime')
plt.xlabel('status')
plt.ylabel('rfstime')
plt.show()
```



```
In [131]: # Boxplot for Age by Status
plt.figure(figsize=(8, 6))
sns.boxplot(x='status', y='age', data=df, palette='Set1')
plt.title('age Levels by status')
plt.xlabel('status')
plt.ylabel('age')
plt.show()
```



Thank you