In [1]:
```
# LOADING AND PREPROCESSING

from sklearn.datasets import load_breast_cancer
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load the dataset
data = load_breast_cancer()
data
X = data.data
y = data.target

# Convert to a DataFrame for convenience
df = pd.DataFrame(X, columns=data.feature_names)
df['target'] = y

# Display the first few rows
print(df.head())
```

```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  ...  worst texture  worst perimeter  worst are
a  \
0                 0.07871  ...          17.33           184.60      2019.
0
1                 0.05667  ...          23.41           158.80      1956.
0
2                 0.05999  ...          25.53           152.50      1709.
0
3                 0.09744  ...          26.50            98.87       567.
7
4                 0.05883  ...          16.67           152.20      1575.
0

   worst smoothness  worst compactness  worst concavity  worst concave po
ints  \
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

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
```

In [2]:
```
df.duplicated().sum()
```

Out[2]:
```
0
```

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

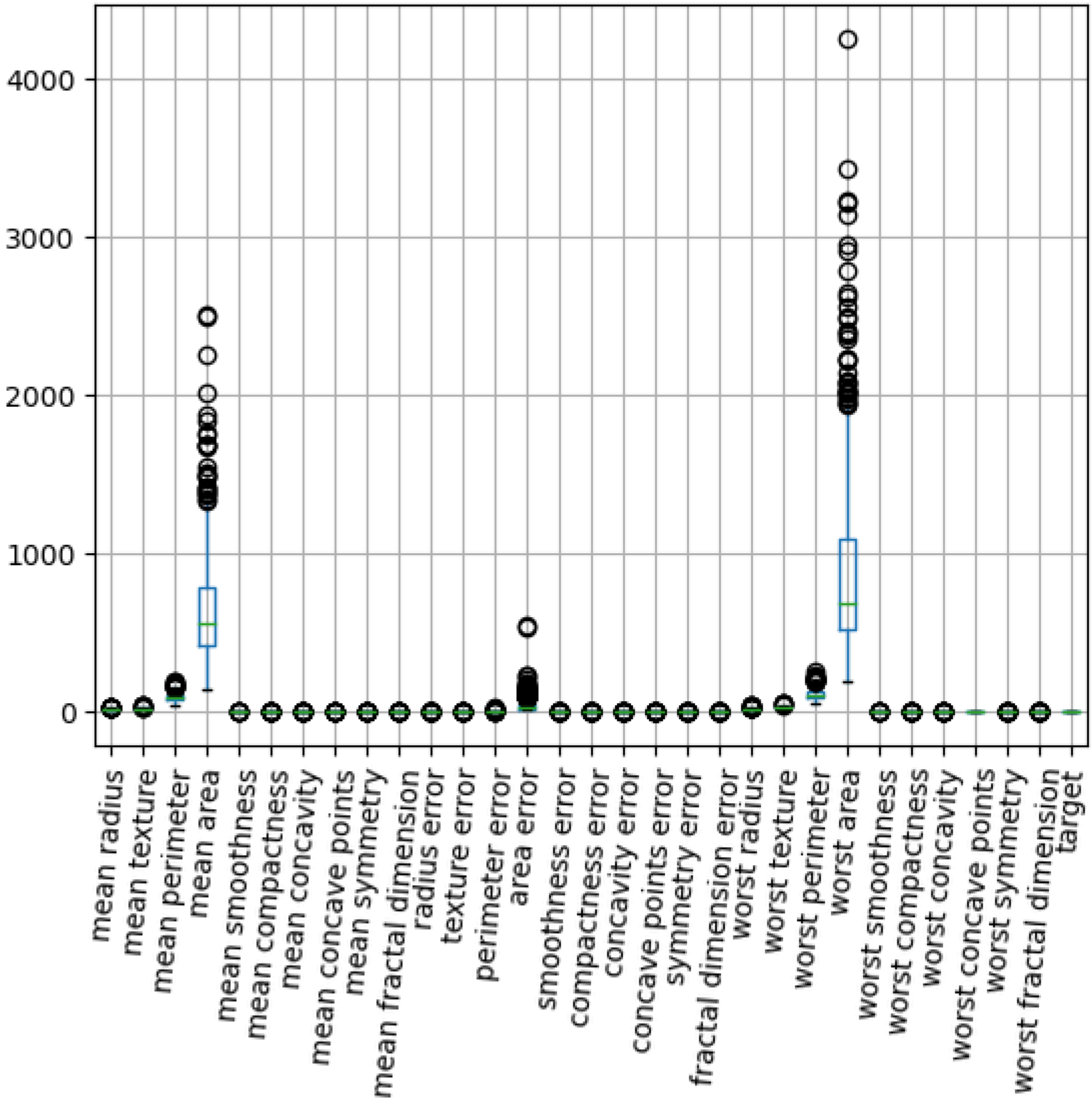
In [4]:
```
df.columns
```

Out[4]:
```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error', 'fractal dimension erro
r',
       'worst radius', 'worst texture', 'worst perimeter', 'worst area',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry', 'worst fractal dimensio
n',
       'target'],
      dtype='object')
```

In [5]:
```
# Checking Outliers

numeric=df.select_dtypes("number")
numeric.skew()
```

Out[5]:
```
mean radius                0.942380
mean texture               0.650450
mean perimeter             0.990650
mean area                  1.645732
mean smoothness            0.456324
mean compactness           1.190123
mean concavity             1.401180
mean concave points        1.171180
mean symmetry              0.725609
mean fractal dimension     1.304489
radius error               3.088612
texture error              1.646444
perimeter error            3.443615
area error                 5.447186
smoothness error           2.314450
compactness error          1.902221
concavity error            5.110463
concave points error       1.444678
symmetry error             2.195133
fractal dimension error    3.923969
worst radius               1.103115
worst texture              0.498321
worst perimeter            1.128164
worst area                 1.859373
worst smoothness           0.415426
worst compactness          1.473555
worst concavity            1.150237
worst concave points       0.492616
worst symmetry             1.433928
worst fractal dimension    1.662579
target                    -0.528461
dtype: float64
```

In [6]:
```python
import matplotlib.pyplot as plt
numeric.boxplot()
plt.xticks(rotation=85)
plt.show() # Outliers present
```



In [7]:
```
numeric.hist()
```

Out[7]:
```
array([[<Axes: title={'center': 'mean radius'}>,
        <Axes: title={'center': 'mean texture'}>,
        <Axes: title={'center': 'mean perimeter'}>,
        <Axes: title={'center': 'mean area'}>,
        <Axes: title={'center': 'mean smoothness'}>,
        <Axes: title={'center': 'mean compactness'}>],
       [<Axes: title={'center': 'mean concavity'}>,
        <Axes: title={'center': 'mean concave points'}>,
        <Axes: title={'center': 'mean symmetry'}>,
        <Axes: title={'center': 'mean fractal dimension'}>,
        <Axes: title={'center': 'radius error'}>,
        <Axes: title={'center': 'texture error'}>],
       [<Axes: title={'center': 'perimeter error'}>,
        <Axes: title={'center': 'area error'}>,
        <Axes: title={'center': 'smoothness error'}>,
        <Axes: title={'center': 'compactness error'}>,
        <Axes: title={'center': 'concavity error'}>,
        <Axes: title={'center': 'concave points error'}>],
       [<Axes: title={'center': 'symmetry error'}>,
        <Axes: title={'center': 'fractal dimension error'}>,
        <Axes: title={'center': 'worst radius'}>,
        <Axes: title={'center': 'worst texture'}>,
        <Axes: title={'center': 'worst perimeter'}>,
        <Axes: title={'center': 'worst area'}>],
       [<Axes: title={'center': 'worst smoothness'}>,
        <Axes: title={'center': 'worst compactness'}>,
        <Axes: title={'center': 'worst concavity'}>,
        <Axes: title={'center': 'worst concave points'}>,
        <Axes: title={'center': 'worst symmetry'}>,
        <Axes: title={'center': 'worst fractal dimension'}>],
       [<Axes: title={'center': 'target'}>, <Axes: >, <Axes: >, <Axes: >,
        <Axes: >, <Axes: >]], dtype=object)
```

In [8]:
```
# Remove Outliers

# Calculate IQR for each feature

# Define a function to calculate outlier bounds
def remove_outliers(df):
    # Create a copy to avoid modifying the original DataFrame
    df_clean = df.copy()

    # Identify outliers using IQR
    for i in df.columns[:-1]:  # Exclude the target column
        Q1 = df_clean[i].quantile(0.25)
        Q3 = df_clean[i].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Remove outliers
        df_clean = df_clean[(df_clean[i] >= lower_bound) & (df_clean[i] <=

    return df_clean

# Apply the IQR outlier removal
df_clean = remove_outliers(df)
print(df_clean)
print("Original dataset shape:",df.shape)
print("Cleaned dataset shape:",df_clean.shape)
```

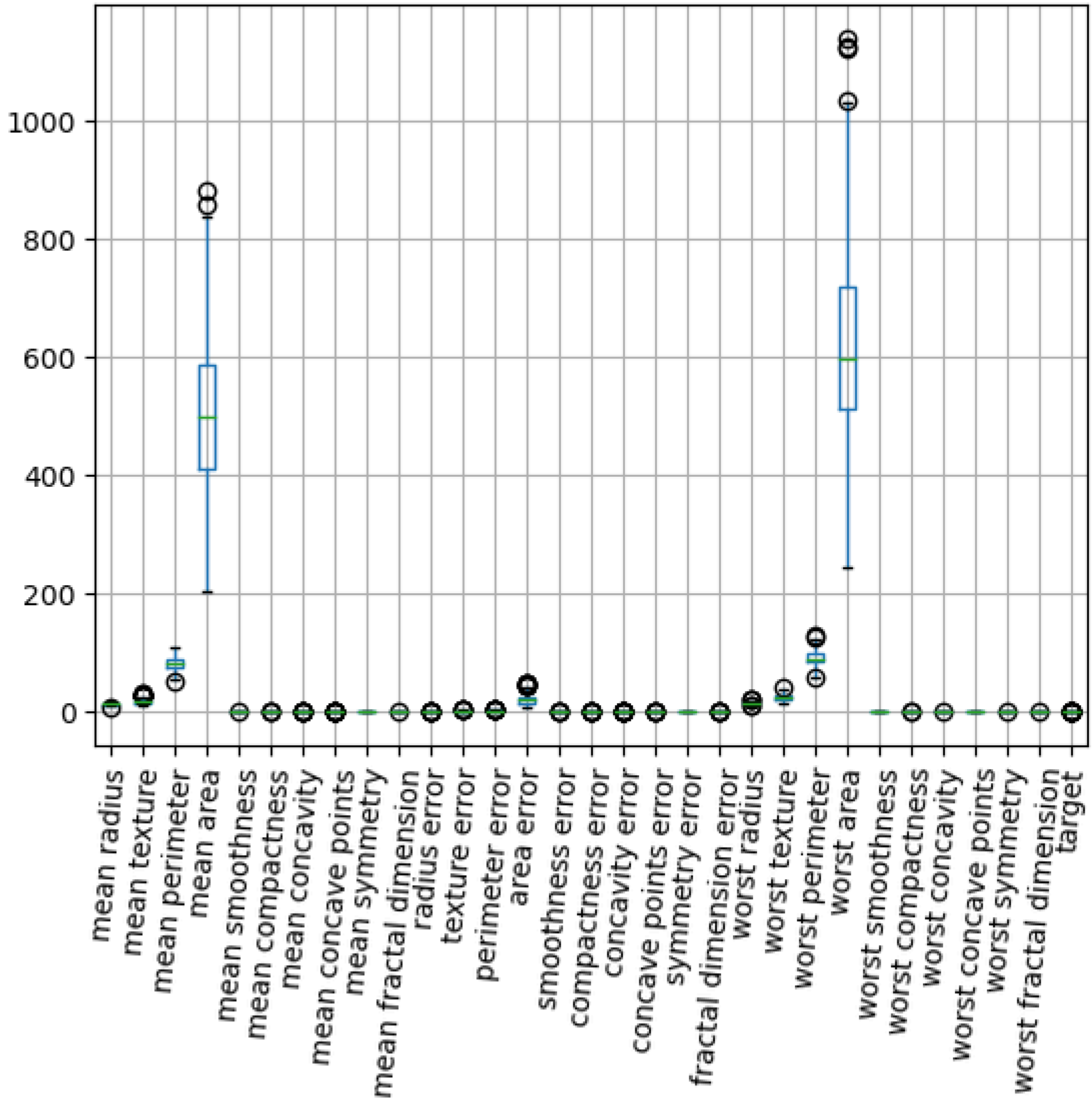
```
     mean radius  mean texture  mean perimeter  mean area  mean smoothnes
s  \
16        14.680         20.13           94.74      684.5          0.0986
7
19        13.540         14.36           87.46      566.3          0.0977
9
20        13.080         15.71           85.63      520.0          0.1075
0
21         9.504         12.44           60.34      273.9          0.1024
0
37        13.030         18.42           82.61      523.8          0.0898
3
..           ...           ...             ...        ...              
...
552       12.770         29.43           81.35      507.9          0.0827
6
554       12.880         28.92           82.50      514.3          0.0812
3
555       10.290         27.61           65.67      321.4          0.0903
0
560       14.050         27.15           91.38      600.4          0.0992
9
566       16.600         28.08          108.30      858.1          0.0845
5

     mean compactness  mean concavity  mean concave points  mean symmetry
\
16            0.07200         0.07395              0.05259         0.1586
19            0.08129         0.06664              0.04781         0.1885
20            0.12700         0.04568              0.03110         0.1967
21            0.06492         0.02956              0.02076         0.1815
37            0.03766         0.02562              0.02923         0.1467
..                ...             ...                  ...            ...
552           0.04234         0.01997              0.01499         0.1539
554           0.05824         0.06195              0.02343         0.1566
555           0.07658         0.05999              0.02738         0.1593
560           0.11260         0.04462              0.04304         0.1537
566           0.10230         0.09251              0.05302         0.1590

     mean fractal dimension  ...  worst texture  worst perimeter  worst a
rea  \
16                  0.05922  ...          30.88           123.40       113
8.0
19                  0.05766  ...          19.26            99.70        71
1.2
20                  0.06811  ...          20.49            96.09        63
0.5
21                  0.06905  ...          15.66            65.13        31
4.9
37                  0.05863  ...          22.81            84.46        54
5.9
..                      ...  ...            ...              ...         
...
552                 0.05637  ...          36.00            88.10        59
4.7
554                 0.05708  ...          35.74            88.84        59
5.7
555                 0.06127  ...          34.91            69.57        35
7.6
560                 0.06171  ...          33.17           100.20        70
6.7
```

```
566                 0.05648  ...          34.12           126.70      112
4.0

     worst smoothness  worst compactness  worst concavity  \
16            0.14640            0.18710          0.29140
19            0.14400            0.17730          0.23900
20            0.13120            0.27760          0.18900
21            0.13240            0.11480          0.08867
37            0.09701            0.04619          0.04833
..                ...                ...              ...
552           0.12340            0.10640          0.08653
554           0.12270            0.16200          0.24390
555           0.13840            0.17100          0.20000
560           0.12410            0.22640          0.13260
566           0.11390            0.30940          0.34030

     worst concave points  worst symmetry  worst fractal dimension  targe
t
16                0.16090          0.3029                  0.08216
0
19                0.12880          0.2977                  0.07259
1
20                0.07283          0.3184                  0.08183
1
21                0.06227          0.2450                  0.07773
1
37                0.05013          0.1987                  0.06169
1
..                    ...             ...                      ...
...
552               0.06498          0.2407                  0.06484
1
554               0.06493          0.2372                  0.07242
1
555               0.09127          0.2226                  0.08283
1
560               0.10480          0.2250                  0.08321
1
566               0.14180          0.2218                  0.07820
0

[277 rows x 31 columns]
Original dataset shape: (569, 31)
Cleaned dataset shape: (277, 31)
```

In [9]:
```
numeric2 = df_clean.select_dtypes("number")
numeric2.skew() # skewness has changed
```

Out[9]:
```
mean radius                -0.128294
mean texture                0.734160
mean perimeter             -0.093142
mean area                   0.212862
mean smoothness             0.215284
mean compactness            0.609782
mean concavity              1.082927
mean concave points         1.033365
mean symmetry               0.187767
mean fractal dimension      0.507242
radius error                1.033028
texture error               0.690570
perimeter error             1.061449
area error                  1.098519
smoothness error            0.618078
compactness error           0.915554
concavity error             0.813968
concave points error        0.417194
symmetry error              0.558661
fractal dimension error     0.802053
worst radius                0.043051
worst texture               0.457921
worst perimeter             0.102289
worst area                  0.442374
worst smoothness            0.131011
worst compactness           0.470541
worst concavity             0.516711
worst concave points        0.194376
worst symmetry              0.226846
worst fractal dimension     0.316991
target                     -3.127780
dtype: float64
```

In [15]:
```
numeric2.boxplot()
plt.xticks(rotation=85)
plt.show()
```



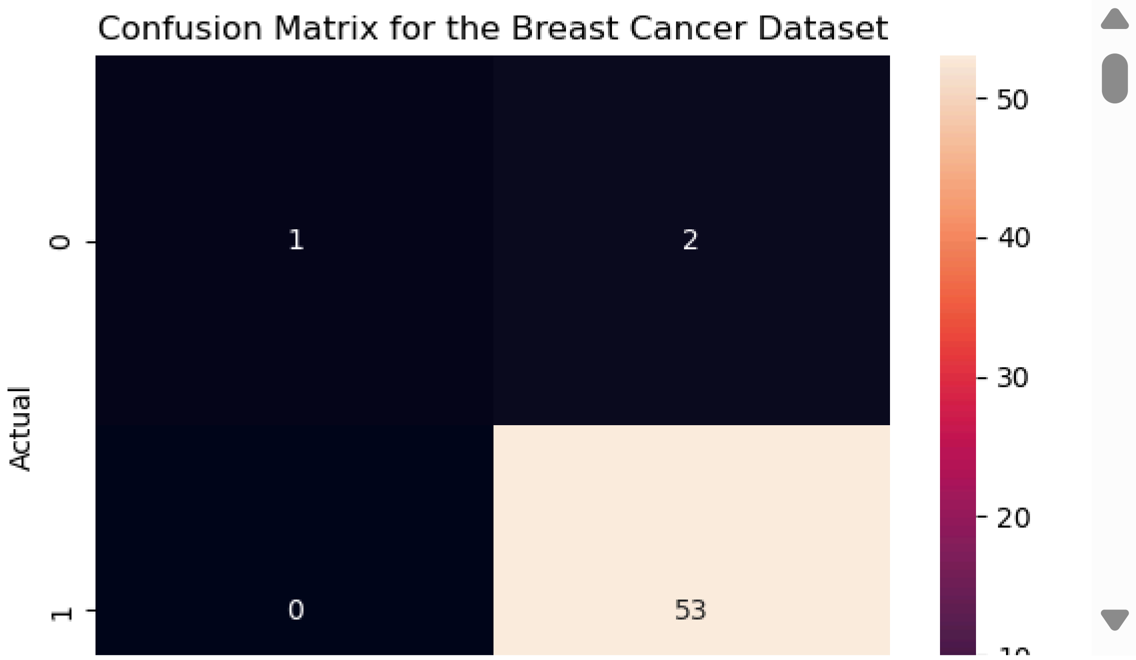

In [11]:
```
df_clean['target'].unique() # no need of label encoding
```

Out[11]:
```
array([0, 1])
```

In [12]:
```
# Separate features and target

X_clean = df_clean.drop('target', axis=1)
y_clean = df_clean['target']
```

In [13]:
```
# Split Data

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_clean, y_clean, test_

# Scale the features
x = StandardScaler()
X_train_scaled = x.fit_transform(X_train)
X_test_scaled = x.transform(X_test)
```

In [16]:
```
# Train and Evaluate models
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accura
import seaborn as sns

# Initialize classifiers

models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC(),
    "k-NN": KNeighborsClassifier()
}

# Train and evaluate each model

results = []

for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    class_report = classification_report(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)
    conf_matrix = confusion_matrix(y_test, y_pred)
    sns.heatmap(conf_matrix, annot=True)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix for the Breast Cancer Dataset')
    plt.show()

    results.append({
        "Model": name,
        "Accuracy": accuracy,
        "Confusion Matrix": conf_matrix,
        "Classification Report": class_report
    })


    # Print results
    for result in results:
        print(f"Model: {result['Model']}")
        print(f"{name} Accuracy: {accuracy:.4f}")
        print("Confusion Matrix:")
        print(result['Confusion Matrix'])
        print(f"{name} Classification Report:")
        print(class_report)
        print()
```




In [17]:

```
# Find the best model based on Accuracy Score

best_model = max(results, key=lambda x: x["Accuracy"])
worst_model = min(results, key=lambda x: x["Accuracy"])

print(f"Best Model based on Accuracy: {best_model['Model']} with Accuracy: 
print(f"Worst Model based on Accuracy: {worst_model['Model']} with Accuracy
```

```
Best Model based on Accuracy: SVM with Accuracy: 0.9821
Worst Model based on Accuracy: Decision Tree with Accuracy: 0.8929
```

In [ ]: