cancer-prediction

June 23, 2024

1 Enhancing Breast Cancer Diagnosis: Leveraging Machine Learning for Accurate Classification

Dr. DILEEP KUMAR SHETTY (Ph.D. ,M.Sc with KSET & Data Science)

The dataset describing the breast cancer, likely the Breast Cancer Wisconsin (Diagnostic) dataset. This dataset contains various features computed from breast cancer images and is commonly used for classification tasks, particularly to distinguish between malignant (cancerous) and benign (non-cancerous) tumors. Here is a detailed description of the dataset:

Columns Description 1. diagnosis: Diagnosis of the breast mass (M = malignant, B = benign). 2. radius_mean: Mean of distances from the center to points on the perimeter. 3. texture_mean: Standard deviation of gray-scale values. 4. perimeter_mean: Mean size of the core tumor. 5. area_mean: Mean area of the tumor. 6. smoothness_mean: Mean of local variation in radius lengths. 7. compactness_mean: Mean of perimeter^2 / area - 1.0. 8. concavity_mean: Mean of the severity of concave portions of the contour. 9. concave points_mean: Mean for the number of concave portions of the contour. 10. symmetry_mean: Mean symmetry. 11. fractal_dimension_mean: Mean "coastline approximation" - 1. 12. radius_se: Standard error of distances from the center to points on the perimeter. 13. texture_se: Standard error of gray-scale values. 14. perimeter_se: Standard error of the core tumor perimeter. 15. area_se: Standard error of the tumor area. 16. smoothness_se: Standard error of local variation in radius lengths. 17. compactness_se: Standard error of perimeter^2 / area - 1.0. 18. concavity_se: Standard error of the severity of concave portions of the contour. 19. concave points_se: Standard error for the number of concave portions of the contour. 20. symmetry_se: Standard error for symmetry. 21. fractal_dimension_se: Standard error for "coastline approximation" - 1. 22. radius_worst: "Worst" or largest mean value for radius. 23. texture_worst: "Worst" or largest mean value for texture. 24. perimeter_worst: "Worst" or largest mean value for perimeter. 25. area_worst: "Worst" or largest mean value for area. 26. smoothness_worst: "Worst" or largest mean value for smoothness. 27. compactness_worst: "Worst" or largest mean value for compactness. 28. concavity_worst: "Worst" or largest mean value for concavity. 29. concave points_worst: "Worst" or largest mean value for concave points. 30. symmetry_worst: "Worst" or largest mean value for symmetry. 31. fractal_dimension_worst: "Worst" or largest mean value for fractal dimension.

Summary • Total Observations: 569 • Total Features: 30 numeric features and 1 target label (diagnosis). Purpose The purpose of this dataset is to train machine learning models to predict whether a breast mass is malignant or benign based on the features derived from digitized images of fine needle aspirates (FNA) of breast masses. Use in Machine Learning This dataset is typically used for: • Classification tasks. • Testing different machine learning algorithms and models, such as Support Vector Machines (SVM), Decision Trees, Random Forests, Neural Networks, etc.

• Feature selection and dimensionality reduction techniques. • Understanding the importance of different features in predicting the diagnosis. This dataset is popular in the field of biomedical image analysis and is often used for educational purposes to demonstrate the application of machine learning in healthcare.

2 Project Objectives:

Evaluate Multiple Machine Learning Algorithms: The primary objective of the project was to evaluate and compare the performance of 15 different machine learning algorithms on a cancer dataset. This includes popular algorithms such as logistic regression, SVM, random forest, XGBoost, and Adaboost, among others. Predict Malignant vs. Benign Cancer: The core aim was to develop a predictive model that accurately distinguishes between malignant and benign cancer cases based on relevant features in the dataset. This predictive capability is crucial for early diagnosis and effective treatment planning. Optimize Feature Selection: Another objective was to explore the impact of feature selection techniques, such as backward model selection, on model performance. Identifying the most relevant features helps in building a more efficient and accurate predictive model. Achieve High Accuracy and Performance: The project aimed to achieve high accuracy, precision, recall, F1-score, and AUC score across different machine learning models. The goal was to identify the model or combination of models that best suit the task of cancer prediction.

3 Project Outcomes:

1. Identification of Top-Performing Model: The logistic regression model with backward model selection emerged as the top performer, achieving an impressive accuracy score of 97% and excellent performance across all evaluation metrics.
2. Demonstrated Importance of Feature Selection: The success of the logistic regression model highlighted the critical role of feature selection in enhancing predictive accuracy. Incorporating the most relevant features significantly contributed to the model's ability to differentiate between cancer types accurately.
3. Validation of Machine Learning Algorithms: The project validated the effectiveness of various machine learning algorithms in cancer prediction tasks. It showcased the strengths and weaknesses of each algorithm, providing valuable insights for future model development.
4. Real-World Applicability: The high accuracy scores and robust performance of the top-performing model indicate its potential for practical deployment in real-world scenarios. This includes aiding medical professionals in cancer diagnosis and treatment decisions.
5. Continuous Improvement and Validation: The project emphasized the importance of ongoing monitoring, validation, and refinement of predictive models. Continuous feedback, feature refinement, and domain expert input are crucial for improving accuracy and effectiveness over time.
6. Enhanced Understanding of Cancer Data: Through the project, a deeper understanding of the cancer dataset and its predictive features was achieved. This understanding contributes to improved insights into cancer characteristics and diagnostic patterns.
7. Contributions to Medical Diagnostics: The project outcomes contribute significantly to the field of medical diagnostics, particularly in cancer diagnosis. Accurate predictive models enhance patient outcomes, treatment planning, and overall healthcare effectiveness.

4 Libraries and modules commonly used in data analysis and machine learning in Python

[1]:
```
#Pandas is a powerful data manipulation library for Python.
import pandas as pd

#NumPy is a numerical computing library for Python.
import numpy as np

#Matplotlib is a plotting library for creating static, interactive, and␣
↪animated visualizations in Python.
import matplotlib.pyplot as plt

#ListedColormap is a class in Matplotlib used to create a colormap from a list␣
↪of colors.
from matplotlib.colors import ListedColormap

#Seaborn is a statistical data visualization library based on Matplotlib.
import seaborn as sns

#is_string_dtype is a function from Pandas used to check if a dtype is of␣
↪object type.
from pandas.api.types import is_string_dtype

#StandardScaler is a preprocessing technique used to standardize features by␣
↪removing the mean and scaling to unit variance.
from sklearn.preprocessing import StandardScaler

#train_test_split is a function in scikit-learn used for splitting a dataset␣
↪into training and testing sets.
from sklearn.model_selection import train_test_split
```

[2]:
```
#The metrics module in scikit-learn provides various metrics for evaluating␣
↪model performance.
from sklearn import metrics

#LogisticRegression is a class in scikit-learn used for logistic regression␣
↪modeling.
from sklearn.linear_model import LogisticRegression

#classification_report is a function in scikit-learn that generates a text␣
↪report showing the main classification metrics.
from sklearn.metrics import classification_report

#cohen_kappa_score is a function in scikit-learn used for calculating the␣
↪Cohen's kappa statistic.
```

```
from sklearn.metrics import cohen_kappa_score

#confusion_matrix is a function in scikit-learn that computes the confusion␣
↪matrix to evaluate the accuracy of a classification.
from sklearn.metrics import confusion_matrix

#roc_auc_score is a function in scikit-learn used for computing the area under␣
↪the ROC AUC.
from sklearn.metrics import roc_auc_score

#roc_curve is a function in scikit-learn used for generating receiver operating␣
↪characteristic (ROC) curves.
from sklearn.metrics import roc_curve

#SGDClassifier is a class in scikit-learn implementing linear classifiers with␣
↪Stochastic Gradient Descent training.
from sklearn.linear_model import SGDClassifier

#DecisionTreeClassifier is a class in scikit-learn for building decision tree␣
↪models.
from sklearn.tree import DecisionTreeClassifier

#GridSearchCV is a class in scikit-learn for hyperparameter tuning using grid␣
↪search.
from sklearn.model_selection import GridSearchCV

#The tree module in scikit-learn provides tools for working with decision trees.
from sklearn import tree

#export_graphviz is a function in scikit-learn for exporting decision tree␣
↪models to Graphviz format.
from sklearn.tree import export_graphviz
```

[3]:
```
#Statsmodels is a library for estimating and testing statistical models.
import statsmodels
import statsmodels.api as sm

#SVC is a class in scikit-learn implementing Support Vector Classification.
from sklearn.svm import SVC

#GaussianNB is a class in scikit-learn implementing Gaussian Naive Bayes␣
↪classification.
from sklearn.naive_bayes import GaussianNB

#KNeighborsClassifier is a class in scikit-learn for k-nearest neighbors␣
↪classification.
```

[4]:
```
#Ignore Warnings:
import warnings
from warnings import filterwarnings
filterwarnings('ignore')

#Adjust Figure Size for Matplotlib:
plt.rcParams['figure.figsize'] = [10,4]
```

[5]:
```
#Adjusting some display and print options for Pandas and NumPy
#max_columns to None, Pandas not to truncate the display of columns.
pd.options.display.max_columns = None

##max_rows to None, Pandas not to truncate the display of rows.
pd.options.display.max_rows = None

# To see the full numeric values without exponential notation.
np.set_printoptions(suppress=True)
```

[6]:
```
#The os.chdir function is used to change the current working directory to the␣
 ↪specified path.
import os
os.chdir(r"C:\DKS\Machine_Learning\Random_Forest")

##Load the Dataset
data= pd.read_csv('cancer.csv')
#The sample(15) method is used to display a random sample of 15 rows from the␣
 ↪loaded DataFrame
data.sample(15)
```

[6]:
```
           id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
427     90745         B       10.800         21.98           68.79      359.9
66     859464         B        9.465         21.01           60.11      269.4
371   9012568         B       15.190         13.21           97.65      711.8
299    892399         B       10.510         23.09           66.85      334.2
527  91813702         B       12.340         12.27           78.94      468.5
49     857156         B       13.490         22.30           86.91      561.0
94     862028         M       15.060         19.83          100.30      705.6
309    893548         B       13.050         13.84           82.71      530.6
524    917897         B        9.847         15.68           63.00      293.2
111     86408         B       12.630         20.76           82.15      480.4
470   9113778         B        9.667         18.49           61.49      289.1
62     858986         M       14.250         22.15           96.42      645.7
346    898678         B       12.060         18.90           76.66      445.3
97     862261         B        9.787         19.94           62.11      294.5
165   8712291         B       14.970         19.76           95.50      690.2

     smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
427          0.08801            0.05743        0.036140             0.014040
66           0.10440            0.07773        0.021720             0.015040
371          0.07963            0.06934        0.033930             0.026570
299          0.10150            0.06797        0.024950             0.018750
527          0.09003            0.06307        0.029580             0.026470
49           0.08752            0.07698        0.047510             0.033840
94           0.10390            0.15530        0.170000             0.088150
309          0.08352            0.03735        0.004559             0.008829
524          0.09492            0.08419        0.023300             0.024160
111          0.09933            0.12090        0.106500             0.060210
470          0.08946            0.06258        0.029480             0.015140
62           0.10490            0.20080        0.213500             0.086530
346          0.08386            0.05794        0.007510             0.008488
97           0.10240            0.05301        0.006829             0.007937
165          0.08421            0.05352        0.019470             0.019390

     symmetry_mean  fractal_dimension_mean  radius_se  texture_se  \
427         0.2016                 0.05977     0.3077      1.6210
66          0.1717                 0.06899     0.2351      2.0110
371         0.1721                 0.05544     0.1783      0.4125
299         0.1695                 0.06556     0.2868      1.1430
527         0.1689                 0.05808     0.1166      0.4957
49          0.1809                 0.05718     0.2338      1.3530
94          0.1855                 0.06284     0.4768      0.9644
309         0.1453                 0.05518     0.3975      0.8285
524         0.1387                 0.06891     0.2498      1.2160
111         0.1735                 0.07070     0.3424      1.8030
470         0.2238                 0.06413     0.3776      1.3500
62          0.1949                 0.07292     0.7036      1.2680
346         0.1555                 0.06048     0.2430      1.1520
97          0.1350                 0.06890     0.3350      2.0430
165         0.1515                 0.05266     0.1840      1.0650

     perimeter_se  area_se  smoothness_se  compactness_se  concavity_se  \
427        2.2400   20.200       0.006543        0.021480      0.029910
66         1.6600   14.200       0.010520        0.017550      0.017140
371        1.3380   17.720       0.005012        0.014850      0.015510
299        2.2890   20.560       0.010170        0.014430      0.018610
527        0.7714    8.955       0.003681        0.009169      0.008732
49         1.7350   20.200       0.004455        0.013820      0.020950
94         3.7060   47.140       0.009250        0.037150      0.048670
309        2.5670   33.010       0.004148        0.004711      0.002831
524        1.9760   15.240       0.008732        0.020420      0.010620
111        2.7110   20.480       0.012910        0.040420      0.051010
470        2.5690   22.730       0.007501        0.019890      0.027140
62         5.3730   60.780       0.009407        0.070560      0.068990
346        1.5590   18.020       0.007180        0.010960      0.005832
```

```
97        2.1320   20.050       0.011130        0.014630      0.005308
165       1.2860   16.640       0.003634        0.007983      0.008268

     concave points_se  symmetry_se  fractal_dimension_se  radius_worst  \
427           0.010450      0.01844              0.002690         12.76
66            0.009333      0.02279              0.004237         10.41
371           0.009155      0.01647              0.001767         16.20
299           0.012500      0.03464              0.001971         10.93
527           0.005740      0.01129              0.001366         13.61
49            0.011840      0.01641              0.001956         15.15
94            0.018510      0.01498              0.003520         18.23
309           0.004821      0.01422              0.002273         14.73
524           0.006801      0.01824              0.003494         11.24
111           0.022950      0.02144              0.005891         13.33
470           0.009883      0.01960              0.003913         11.14
62            0.018480      0.01700              0.006113         17.67
346           0.005495      0.01982              0.002754         13.64
97            0.005250      0.01801              0.005667         10.92
165           0.006432      0.01924              0.001520         15.98

     texture_worst  perimeter_worst  area_worst  smoothness_worst  \
427          32.04            83.69       489.5            0.1303
66           31.56            67.03       330.7            0.1548
371          15.73           104.50       819.1            0.1126
299          24.22            70.10       362.7            0.1143
527          19.27            87.22       564.9            0.1292
49           31.82            99.00       698.8            0.1162
94           24.23           123.50      1025.0            0.1551
309          17.40            93.96       672.4            0.1016
524          22.99            74.32       376.5            0.1419
111          25.47            89.00       527.4            0.1287
470          25.62            70.88       385.2            0.1234
62           29.51           119.10       959.5            0.1640
346          27.06            86.54       562.6            0.1289
97           26.29            68.81       366.1            0.1316
165          25.82           102.30       782.1            0.1045

     compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
427            0.16960          0.19270               0.07485          0.2965
66             0.16640          0.09412               0.06517          0.2878
371            0.17370          0.13620               0.08178          0.2487
299            0.08614          0.04158               0.03125          0.2227
527            0.20740          0.17910               0.10700          0.3110
49             0.17110          0.22820               0.12820          0.2871
94             0.42030          0.52030               0.21150          0.2834
309            0.05847          0.01824               0.03532          0.2107
524            0.22430          0.08434               0.06528          0.2502
```

```
111       0.22500       0.22160            0.11050       0.2226
470       0.15420       0.12770            0.06560       0.3174
62        0.62470       0.69220            0.17850       0.2844
346       0.13520       0.04506            0.05093       0.2880
97        0.09473       0.02049            0.02381       0.1934
165       0.09995       0.07750            0.05754       0.2646

     fractal_dimension_worst  Unnamed: 32
427                  0.07662          NaN
66                   0.09211          NaN
371                  0.06766          NaN
299                  0.06777          NaN
527                  0.07592          NaN
49                   0.06917          NaN
94                   0.08234          NaN
309                  0.06580          NaN
524                  0.09209          NaN
111                  0.08486          NaN
470                  0.08524          NaN
62                   0.11320          NaN
346                  0.08083          NaN
97                   0.08988          NaN
165                  0.06085          NaN
```

[7]:
```
# Dropping the 'id' and 'Unnamed: 32' columns from the DataFrame
# The 'id' column is typically an identifier that is not useful for modeling
# 'Unnamed: 32' might be an empty or irrelevant column that can be safely␣
 ↪removed
data = data.drop(['id', 'Unnamed: 32'], axis=1)

# Display the first few rows of the cleaned dataset to verify the changes
print(data.head())
```

```
  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  fractal_dimension_mean  radius_se  texture_se  perimeter_se  \
0         0.2419                 0.07871     1.0950      0.9053         8.589   
1         0.1812                 0.05667     0.5435      0.7339         3.398   
2         0.2069                 0.05999     0.7456      0.7869         4.585   
3         0.2597                 0.09744     0.4956      1.1560         3.445   
4         0.1809                 0.05883     0.7572      0.7813         5.438   

   area_se  smoothness_se  compactness_se  concavity_se  concave points_se  \
0   153.40       0.006399         0.04904       0.05373            0.01587   
1    74.08       0.005225         0.01308       0.01860            0.01340   
2    94.03       0.006150         0.04006       0.03832            0.02058   
3    27.23       0.009110         0.07458       0.05661            0.01867   
4    94.44       0.011490         0.02461       0.05688            0.01885   

   symmetry_se  fractal_dimension_se  radius_worst  texture_worst  \
0      0.03003              0.006193         25.38          17.33   
1      0.01389              0.003532         24.99          23.41   
2      0.02250              0.004571         23.57          25.53   
3      0.05963              0.009208         14.91          26.50   
4      0.01756              0.005115         22.54          16.67   

   perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           184.60      2019.0            0.1622             0.6656   
1           158.80      1956.0            0.1238             0.1866   
2           152.50      1709.0            0.1444             0.4245   
3            98.87       567.7            0.2098             0.8663   
4           152.20      1575.0            0.1374             0.2050   

   concavity_worst  concave points_worst  symmetry_worst  \
0           0.7119                0.2654          0.4601   
1           0.2416                0.1860          0.2750   
2           0.4504                0.2430          0.3613   
3           0.6869                0.2575          0.6638   
4           0.4000                0.1625          0.2364   

   fractal_dimension_worst  
0                  0.11890  
1                  0.08902  
2                  0.08758  
3                  0.17300  
4                  0.07678  
```

[8]:
```
# Display summary statistics
summary_stats = data.describe()
summary_stats
```

[8]:
```
       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_se  texture_se  \
count     569.000000              569.000000  569.000000  569.000000
mean        0.181162                0.062798    0.405172    1.216853
std         0.027414                0.007060    0.277313    0.551648
min         0.106000                0.049960    0.111500    0.360200
25%         0.161900                0.057700    0.232400    0.833900
50%         0.179200                0.061540    0.324200    1.108000
75%         0.195700                0.066120    0.478900    1.474000
max         0.304000                0.097440    2.873000    4.885000

       perimeter_se     area_se  smoothness_se  compactness_se  concavity_se  \
count    569.000000  569.000000     569.000000      569.000000    569.000000
mean       2.866059   40.337079       0.007041        0.025478      0.031894
std        2.021855   45.491006       0.003003        0.017908      0.030186
min        0.757000    6.802000       0.001713        0.002252      0.000000
25%        1.606000   17.850000       0.005169        0.013080      0.015090
50%        2.287000   24.530000       0.006380        0.020450      0.025890
75%        3.357000   45.190000       0.008146        0.032450      0.042050
max       21.980000  542.200000       0.031130        0.135400      0.396000

       concave points_se  symmetry_se  fractal_dimension_se  radius_worst  \
count         569.000000   569.000000            569.000000    569.000000
mean            0.011796     0.020542              0.003795     16.269190
std             0.006170     0.008266              0.002646      4.833242
min             0.000000     0.007882              0.000895      7.930000
25%             0.007638     0.015160              0.002248     13.010000
50%             0.010930     0.018730              0.003187     14.970000
```

```
75%              0.014710      0.023480              0.004558     18.790000
max              0.052790      0.078950              0.029840     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
```

[9]:
```
#The dtypes attribute in Pandas is used to display the data types of each␣
↪column in a DataFrame.
data.dtypes
```

[9]:
```
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
dtype: object
```

[10]:
```
# Check the info
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   diagnosis                569 non-null    object
 1   radius_mean              569 non-null    float64
 2   texture_mean             569 non-null    float64
 3   perimeter_mean           569 non-null    float64
 4   area_mean                569 non-null    float64
 5   smoothness_mean          569 non-null    float64
 6   compactness_mean         569 non-null    float64
 7   concavity_mean           569 non-null    float64
 8   concave points_mean      569 non-null    float64
 9   symmetry_mean            569 non-null    float64
 10  fractal_dimension_mean   569 non-null    float64
 11  radius_se                569 non-null    float64
 12  texture_se               569 non-null    float64
 13  perimeter_se             569 non-null    float64
 14  area_se                  569 non-null    float64
 15  smoothness_se            569 non-null    float64
 16  compactness_se           569 non-null    float64
 17  concavity_se             569 non-null    float64
 18  concave points_se        569 non-null    float64
 19  symmetry_se              569 non-null    float64
 20  fractal_dimension_se     569 non-null    float64
 21  radius_worst             569 non-null    float64
 22  texture_worst            569 non-null    float64
 23  perimeter_worst          569 non-null    float64
 24  area_worst               569 non-null    float64
 25  smoothness_worst         569 non-null    float64
 26  compactness_worst        569 non-null    float64
 27  concavity_worst          569 non-null    float64
 28  concave points_worst     569 non-null    float64
 29  symmetry_worst           569 non-null    float64
 30  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), object(1)
memory usage: 137.9+ KB
```

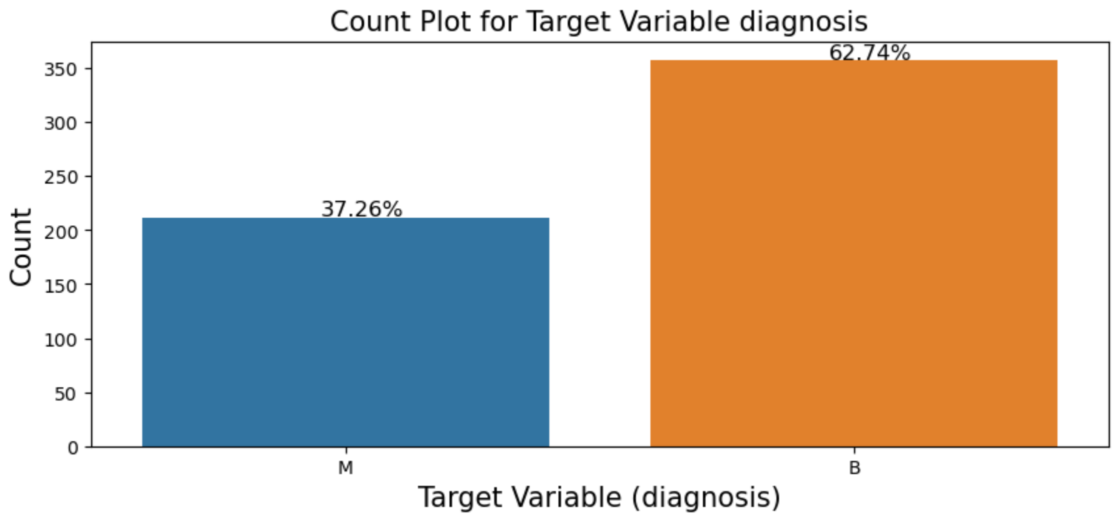
[11]:
```
#Splitting the DataFrame into feature variables (data_x) and the target␣
 ↪variable (data_y).
data_x = data.iloc[:, data.columns != 'diagnosis']
data_y = data.iloc[:,data.columns == 'diagnosis']
data_y.head(2)
```

[11]:
```
  diagnosis
0         M
1         M
```

[12]:
```
# Calculate the frequency of each class in the target variable
class_frequency = data_y.value_counts()

# Print the class frequencies
print(class_frequency)

# Calculate the percentage distribution of each class
class_percentage = data_y.value_counts(normalize=True) * 100

# Print the percentage distribution
print(class_percentage)
```

```
diagnosis
B            357
M            212
dtype: int64
diagnosis
B            62.741652
M            37.258348
dtype: float64
```

[13]:
```python
# Create a count plot for the target variable 'diagnosis'
sns.countplot(data=data_y, x="diagnosis")

# Calculate the percentage of each class and annotate the plot
# The coordinates (x, y) for the text annotations are chosen based on the␣
↪position of the bars
plt.text(x=-0.05, y=data_y.value_counts()[1]+1,
         s=str(round((class_frequency[1])*100/len(data_y), 2)) + '%',
         fontsize=12, color='black')
plt.text(x=0.95, y=data_y.value_counts()[0]+1,
         s=str(round((class_frequency[0])*100/len(data_y), 2)) + '%',
         fontsize=12, color='black')

# Add a title to the plot
plt.title('Count Plot for Target Variable diagnosis', fontsize=15)

# Label the x-axis
plt.xlabel('Target Variable (diagnosis)', fontsize=15)

# Label the y-axis
plt.ylabel('Count', fontsize=15)

# Display the plot
plt.show()
```

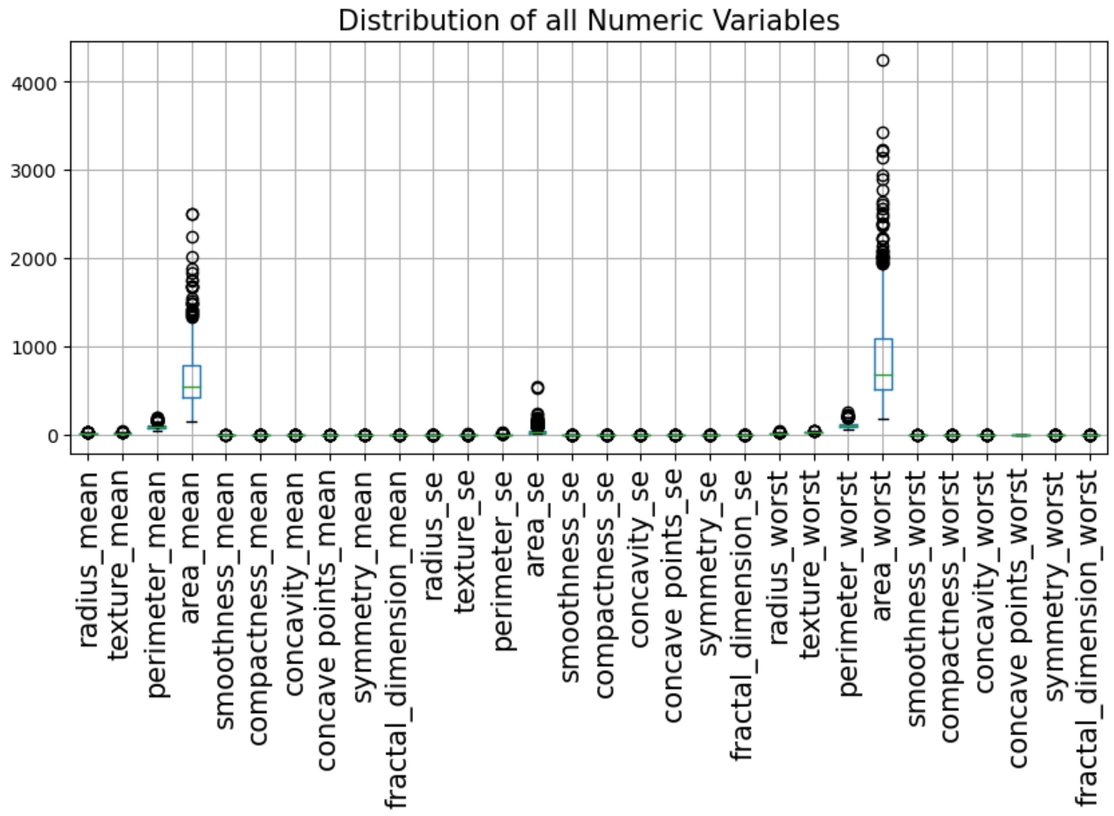


Interpretation In our study, the target variable is ”diagnosis,” which indicates whether a person has a malignant or benign tumor. Here, the value ’M’ denotes malignant, indicating a cancerous tumor, while the value ’B’ represents benign, indicating a non-cancerous tumor.

Our analysis reveals that 37.26

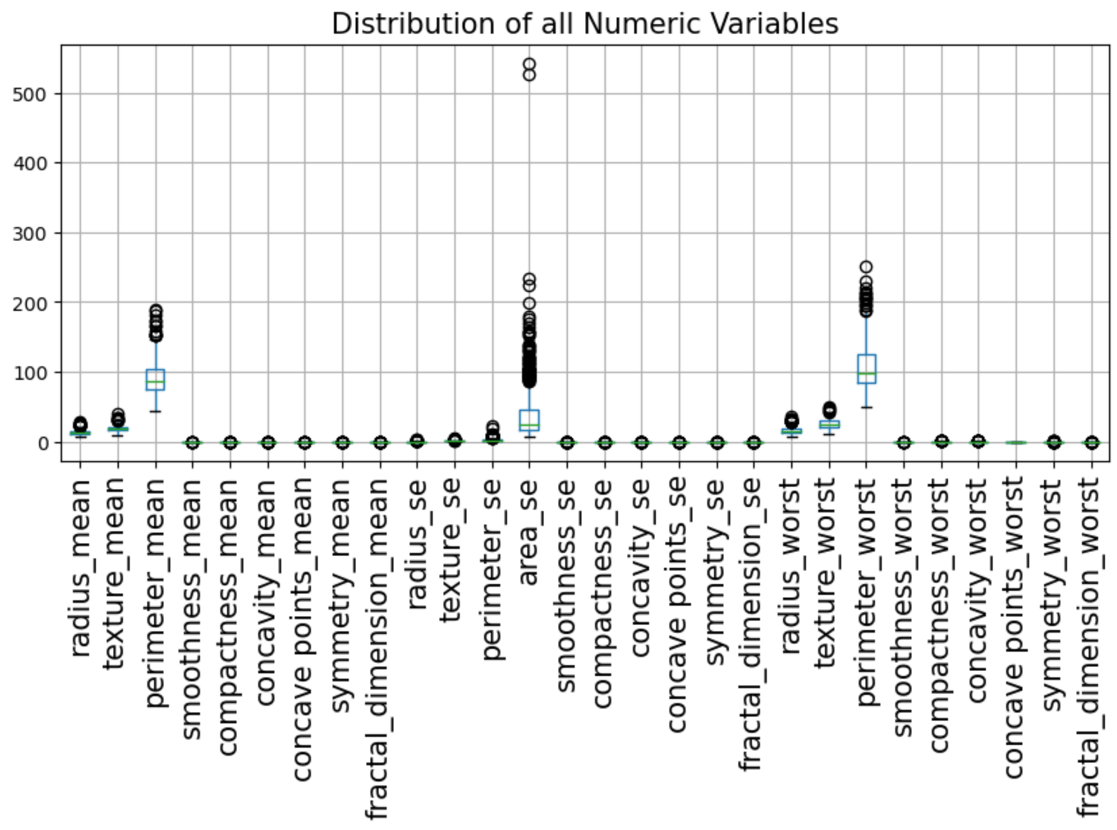
Additional Points Class Imbalance: The dataset exhibits a class imbalance, with a significantly higher number of benign cases compared to malignant ones. This imbalance should be considered when developing predictive models, as it might affect the model's performance and bias it towards the majority class. Feature Importance: The dataset contains various features derived from digitized images of breast masses, such as mean radius, texture, perimeter, area, and others. Understanding the importance of these features can help in identifying key indicators of malignancy. Potential Applications: The insights gained from this dataset can be used to develop machine learning models that aid in early detection and diagnosis of breast cancer, potentially improving patient outcomes. Model Evaluation: It is essential to use appropriate evaluation metrics, such as precision, recall, F1-score, and ROC-AUC, especially given the class imbalance, to ensure that the model performs well for both malignant and benign classifications.

[14]:
```python
# Create a boxplot for all numeric features in the dataset
data_x.boxplot()

# Add a title to the boxplot
plt.title('Distribution of all Numeric Variables', fontsize=15)

# Rotate x-axis labels for better readability and set their font size
plt.xticks(rotation='vertical', fontsize=15)

# Display the plot
plt.show()
```

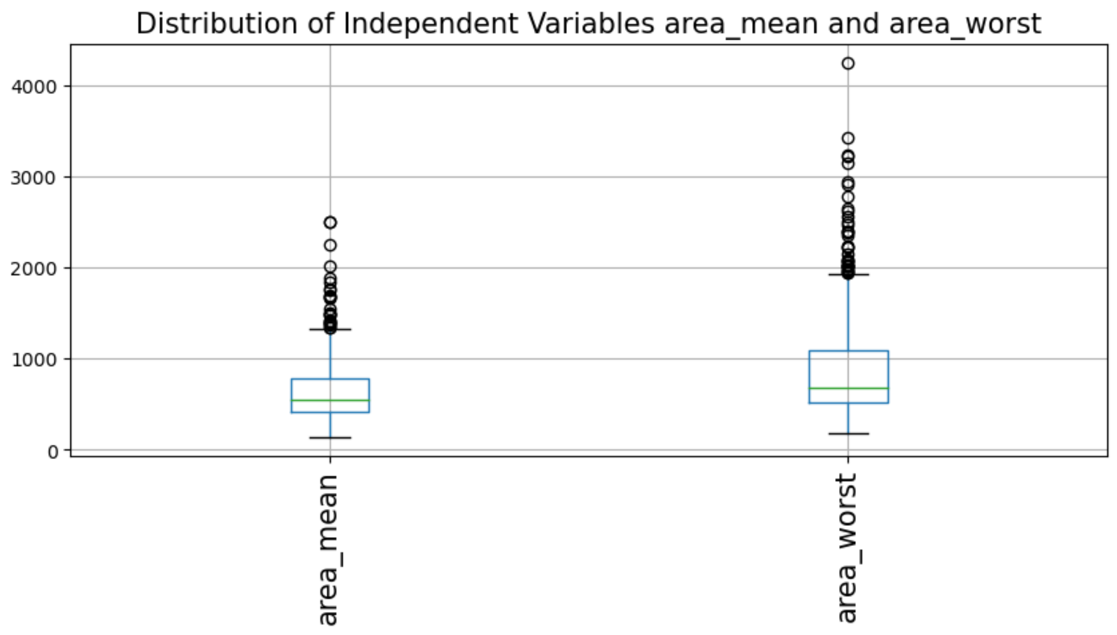


[15]:
```python
dataxn = data.drop(['area_mean', 'area_worst'], axis=1)
```

[16]:
```python
# Create a boxplot for all numeric features in the dataset
dataxn.boxplot()

# Add a title to the boxplot
plt.title('Distribution of all Numeric Variables', fontsize=15)

# Rotate x-axis labels for better readability and set their font size
plt.xticks(rotation='vertical', fontsize=15)

# Display the plot
plt.show()
```



[17]:
```python
variables = ['area_mean', 'area_worst']
data_x[variables].boxplot()
plt.title('Distribution of Independent Variables area_mean and area_worst',␣
 ↪fontsize = 15)
plt.xticks(rotation = 'vertical', fontsize = 15)
plt.show()
```



[18]:
```
# Calculate the total number of missing values for each column and sort in␣
  ↪descending order
Total = data.isnull().sum().sort_values(ascending=False)

# Calculate the percentage of missing values for each column and sort in␣
  ↪descending order
Percentage = (data.isnull().sum() * 100 / data.isnull().count()).
  ↪sort_values(ascending=False)

# Concatenate the total and percentage of missing values into a single DataFrame
Missing_Values = pd.concat([Total, Percentage], axis=1, keys=['Total',␣
  ↪'Percentage of missing observations'])

# Display the DataFrame showing the total and percentage of missing values for␣
  ↪each column
print(Missing_Values)
```

```
                         Total  Percentage of missing observations
diagnosis                    0                                 0.0
compactness_se               0                                 0.0
symmetry_worst               0                                 0.0
concave points_worst         0                                 0.0
concavity_worst              0                                 0.0
compactness_worst            0                                 0.0
```

```
smoothness_worst            0                                  0.0
area_worst                  0                                  0.0
perimeter_worst             0                                  0.0
texture_worst               0                                  0.0
radius_worst                0                                  0.0
fractal_dimension_se        0                                  0.0
symmetry_se                 0                                  0.0
concave points_se           0                                  0.0
concavity_se                0                                  0.0
smoothness_se               0                                  0.0
radius_mean                 0                                  0.0
area_se                     0                                  0.0
perimeter_se                0                                  0.0
texture_se                  0                                  0.0
radius_se                   0                                  0.0
fractal_dimension_mean      0                                  0.0
symmetry_mean               0                                  0.0
concave points_mean         0                                  0.0
concavity_mean              0                                  0.0
compactness_mean            0                                  0.0
smoothness_mean             0                                  0.0
area_mean                   0                                  0.0
perimeter_mean              0                                  0.0
texture_mean                0                                  0.0
fractal_dimension_worst     0                                  0.0
```

[19]:
```
# Generate descriptive statistics for the object (categorical) columns
# The 'include="object"' parameter ensures only the categorical columns are␣
 ↪included in the summary
categorical_summary = data.describe(include="object")

# Display the descriptive statistics for the categorical columns
print(categorical_summary)
```

```
       diagnosis
count        569
unique         2
top            B
freq         357
```

[20]:
```
# Replace 'M' with 0 in the 'diagnosis' column
data["diagnosis"] = data["diagnosis"].replace("M", 1)

# Replace 'B' with 1 in the 'diagnosis' column
data["diagnosis"] = data["diagnosis"].replace("B", 0)

# Display the first few rows of the modified DataFrame to verify the change
```

```
data.head()
```

```
[20]:    diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
      0          1        17.99         10.38          122.80     1001.0
      1          1        20.57         17.77          132.90     1326.0
      2          1        19.69         21.25          130.00     1203.0
      3          1        11.42         20.38           77.58      386.1
      4          1        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  fractal_dimension_mean  radius_se  texture_se  perimeter_se  \
      0         0.2419                 0.07871     1.0950      0.9053         8.589
      1         0.1812                 0.05667     0.5435      0.7339         3.398
      2         0.2069                 0.05999     0.7456      0.7869         4.585
      3         0.2597                 0.09744     0.4956      1.1560         3.445
      4         0.1809                 0.05883     0.7572      0.7813         5.438

         area_se  smoothness_se  compactness_se  concavity_se  concave points_se  \
      0   153.40       0.006399         0.04904       0.05373            0.01587
      1    74.08       0.005225         0.01308       0.01860            0.01340
      2    94.03       0.006150         0.04006       0.03832            0.02058
      3    27.23       0.009110         0.07458       0.05661            0.01867
      4    94.44       0.011490         0.02461       0.05688            0.01885

         symmetry_se  fractal_dimension_se  radius_worst  texture_worst  \
      0      0.03003              0.006193         25.38          17.33
      1      0.01389              0.003532         24.99          23.41
      2      0.02250              0.004571         23.57          25.53
      3      0.05963              0.009208         14.91          26.50
      4      0.01756              0.005115         22.54          16.67

         perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
      0           184.60      2019.0            0.1622             0.6656
      1           158.80      1956.0            0.1238             0.1866
      2           152.50      1709.0            0.1444             0.4245
      3            98.87       567.7            0.2098             0.8663
      4           152.20      1575.0            0.1374             0.2050

         concavity_worst  concave points_worst  symmetry_worst  \
      0           0.7119                0.2654          0.4601
      1           0.2416                0.1860          0.2750
```

```
2          0.4504                0.2430          0.3613
3          0.6869                0.2575          0.6638
4          0.4000                0.1625          0.2364

   fractal_dimension_worst
0                  0.11890
1                  0.08902
2                  0.08758
3                  0.17300
4                  0.07678
```

5 Univariate Analysis

6 1.radius_mean

[21]:
```
# Describe the 'radius_mean' column to generate summary statistics
radius_mean_description = data.radius_mean.describe()

# Display the descriptive statistics for the 'radius_mean' column
print(radius_mean_description)
```

```
count    569.000000
mean      14.127292
std        3.524049
min        6.981000
25%       11.700000
50%       13.370000
75%       15.780000
max       28.110000
Name: radius_mean, dtype: float64
```

The radius_mean feature has a range of values from approximately 6.98 to 28.11, with an average radius of around 14.13 units. The data is fairly spread out, as indicated by the standard deviation of 3.52. Most of the tumor radii (50%) fall between 11.70 and 15.78 units, with the median at 13.37 units. The distribution of values appears to be moderately spread around the mean, with some larger radii extending up to 28.11 units. This information can help in understanding the typical size and variability of tumor radii in this dataset, which is crucial for further analysis and modeling.

7 Skewness and Kurtosis

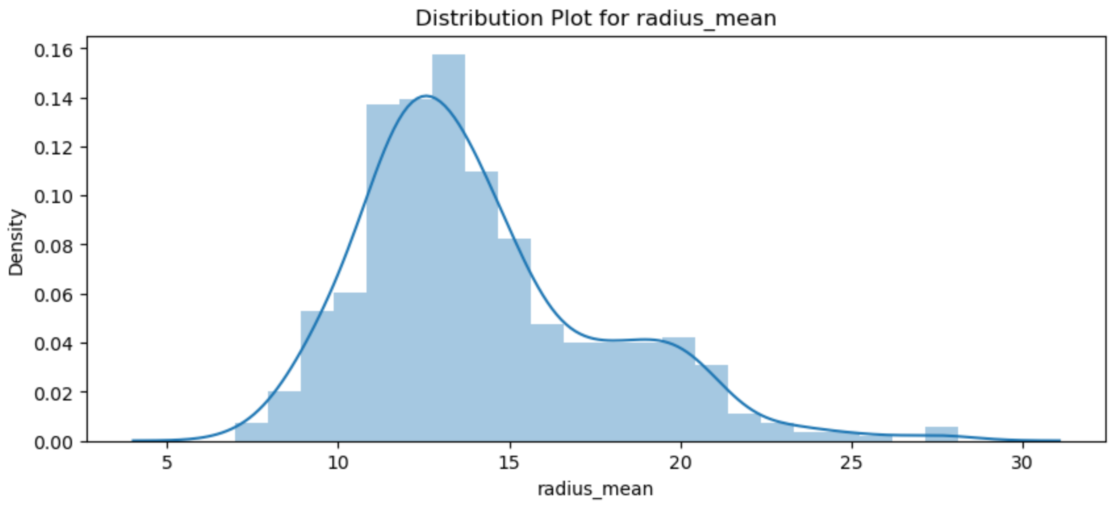
[22]:
```
# Calculate the skewness of the 'radius_mean' column
skewness = data['radius_mean'].skew()

# Calculate the kurtosis of the 'radius_mean' column
kurtosis = data['radius_mean'].kurt()
```

```
# Print the calculated skewness and kurtosis
print("Skewness: %f" % skewness)
print("Kurtosis: %f" % kurtosis)
```

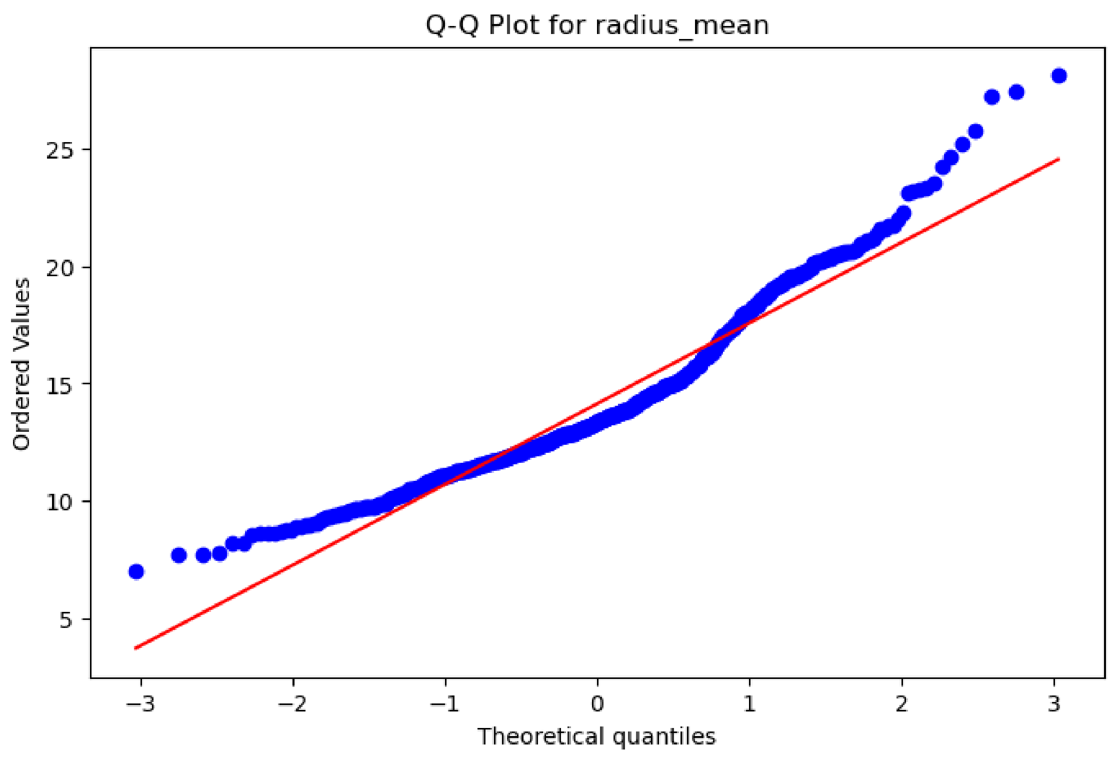
```
Skewness: 0.942380
Kurtosis: 0.845522
```

The distribution of radius_mean is moderately skewed to the right and has lighter tails, suggesting most of the data points are clustered around the mean with some larger values extending the right tail. This information is valuable for understanding the shape and characteristics of the radius_mean distribution, which can impact statistical analyses and modeling techniques.

[23]:
```
# Create a distribution plot (histogram with KDE curve) for the 'radius_mean'␣
 ↪column
sns.distplot(data.radius_mean)

# Add a title to the plot
plt.title("Distribution Plot for radius_mean")

# Display the plot
plt.show()
```



[24]:
```
# q-q plot:q-q plot is used to compare the quantiles of two distributions
# p-p plot:p-p plot is the way to visual comparison of cdf of the two␣
 ↪distributions
import scipy.stats as stats
plt.figure(figsize = (8,5))
stats.probplot(data["radius_mean"],plot=plt)
plt.title("Q-Q Plot for radius_mean")
plt.show()
```



[25]:
```
import numpy as np
from scipy.stats import jarque_bera


# Perform Jarque-Bera test
statistic, p_value = jarque_bera(data.radius_mean)

# Display the results
print(f"Jarque-Bera statistic: {statistic}")
print(f"P-value: {p_value}")

# Check the null hypothesis
if p_value < 0.05:
    print("The radius_mean does not come from a normal distribution (reject the␣
 ↪null hypothesis).")
else:
    print("The radius_mean comes from a normal distribution (fail to reject the␣
 ↪null hypothesis).")
```

```
Jarque-Bera statistic: 100.01344990455239
P-value: 1.915822613520449e-22
The radius_mean does not come from a normal distribution (reject the null
hypothesis).
```

The confirmation of non-normal distribution for radius_mean is supported by the density plot, Q-Q plot, and Jarque-Bera test.

8 Multivariate Analysis

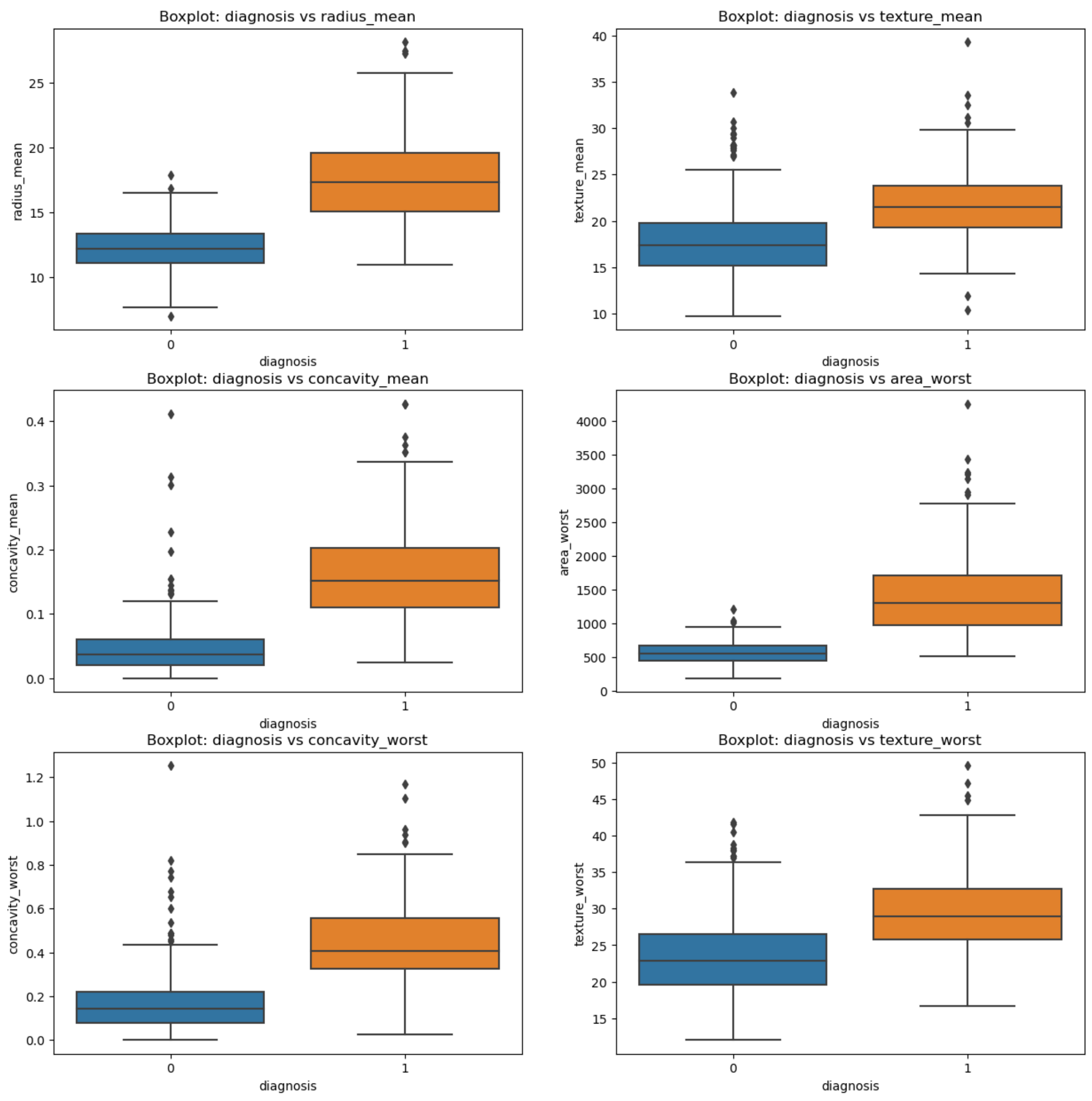
9 1.Box Plots for Target Variable (diagnosis) with Different Features

[26]:
```
data.dtypes
```

[26]:
```
diagnosis                    int64
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
dtype: object
```

[27]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

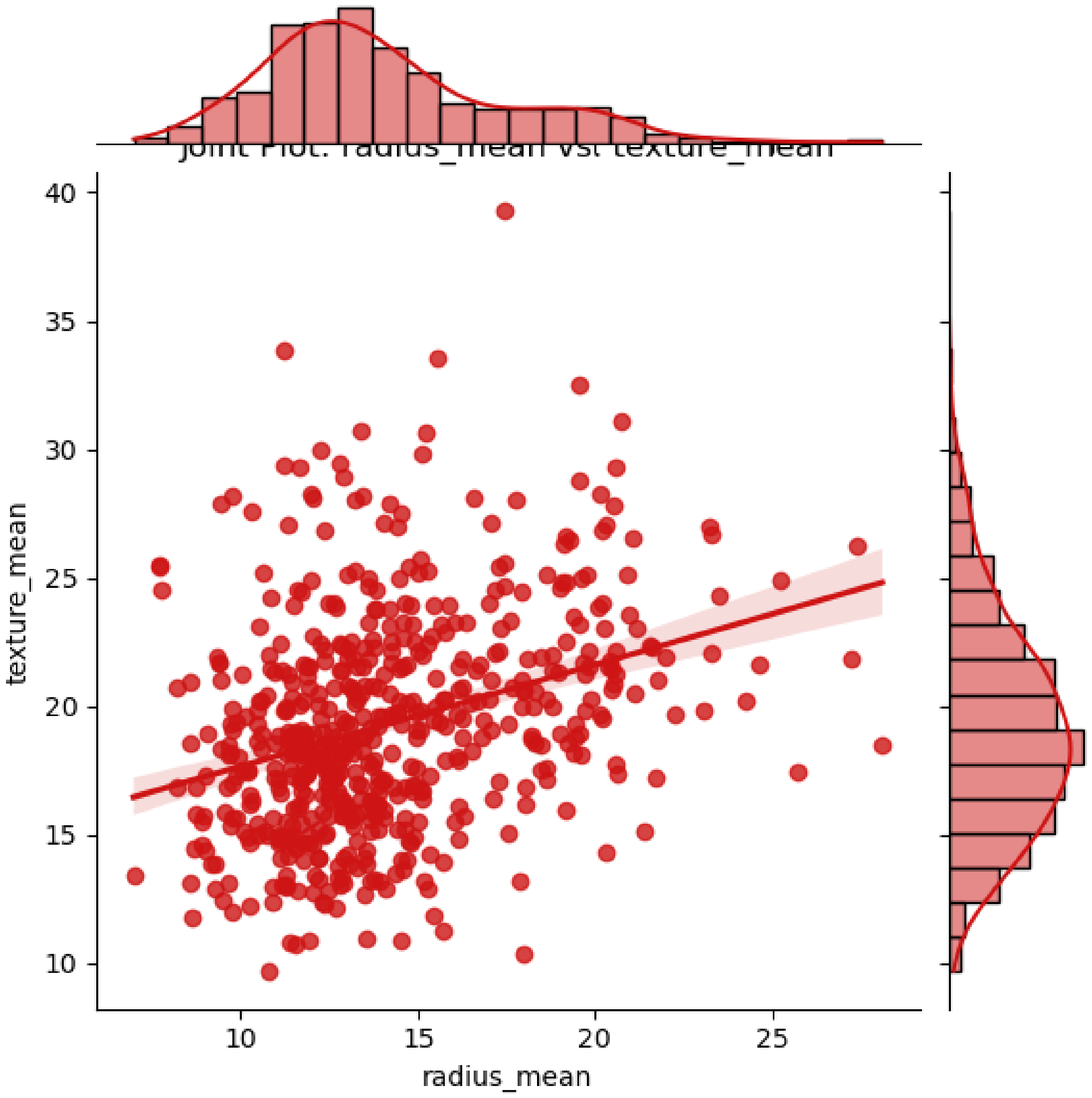
```
# Set up the figure with subplots
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(15, 15))

# Boxplot for 'diagnosis' vs 'radius_mean  '
sns.boxplot(x='diagnosis', y='radius_mean', data=data, ax=axes[0, 0])
axes[0, 0].set_title('Boxplot: diagnosis vs radius_mean')

# Boxplot for 'diagnosis' vs 'texture_mean'
sns.boxplot(x='diagnosis', y='texture_mean', data=data, ax=axes[0, 1])
axes[0, 1].set_title('Boxplot: diagnosis vs texture_mean')

# Boxplot for 'diagnosis' vs 'concavity_mean'
sns.boxplot(x='diagnosis', y='concavity_mean', data=data, ax=axes[1, 0])
axes[1, 0].set_title('Boxplot: diagnosis vs concavity_mean')

# Boxplot for 'diagnosis' vs 'area_worst'
sns.boxplot(x='diagnosis', y='area_worst', data=data, ax=axes[1, 1])
axes[1, 1].set_title('Boxplot: diagnosis vs area_worst')

# Boxplot for 'diagnosis' vs 'concavity_worst'
sns.boxplot(x='diagnosis', y='concavity_worst', data=data, ax=axes[2, 0])
axes[2, 0].set_title('Boxplot: diagnosis vs concavity_worst')

# Boxplot for 'diagnosis' vs 'texture_worst'
sns.boxplot(x='diagnosis', y='texture_worst', data=data, ax=axes[2, 1])
axes[2, 1].set_title('Boxplot: diagnosis vs texture_worst')

# For example, if using matplotlib
plt.savefig('my_plot.png', bbox_inches='tight')
```



10 2. Analysis of radius_mean with texture_mean

[28]:
```
# Selecting the columns 'radius_mean' and 'texture_mean' as x
x = data[['radius_mean', 'texture_mean']]

# Create a joint plot (scatter plot with regression line) for 'radius_mean' vs.␣
↪'texture_mean'
sns.jointplot(x=x.loc[:, 'radius_mean'], y=x.loc[:, 'texture_mean'],␣
↪kind="reg", color="#ce1414")

# Add a title to the plot
plt.title("Joint Plot: radius_mean vs. texture_mean")
```

```
# Display the plot
plt.show()
```



[29]:
```
# Calculate the correlation matrix for the features
correlation_matrix = data_x.corr()

# Display the correlation matrix
correlation_matrix
```

[29]:
```
                         radius_mean  texture_mean  perimeter_mean  area_mean  \
radius_mean                 1.000000      0.323782        0.997855   0.987357
```

```
texture_mean                 0.323782      1.000000         0.329533   0.321086
perimeter_mean               0.997855      0.329533         1.000000   0.986507
area_mean                    0.987357      0.321086         0.986507   1.000000
smoothness_mean              0.170581     -0.023389         0.207278   0.177028
compactness_mean             0.506124      0.236702         0.556936   0.498502
concavity_mean               0.676764      0.302418         0.716136   0.685983
concave points_mean          0.822529      0.293464         0.850977   0.823269
symmetry_mean                0.147741      0.071401         0.183027   0.151293
fractal_dimension_mean      -0.311631     -0.076437        -0.261477  -0.283110
radius_se                    0.679090      0.275869         0.691765   0.732562
texture_se                  -0.097317      0.386358        -0.086761  -0.066280
perimeter_se                 0.674172      0.281673         0.693135   0.726628
area_se                      0.735864      0.259845         0.744983   0.800086
smoothness_se               -0.222600      0.006614        -0.202694  -0.166777
compactness_se               0.206000      0.191975         0.250744   0.212583
concavity_se                 0.194204      0.143293         0.228082   0.207660
concave points_se            0.376169      0.163851         0.407217   0.372320
symmetry_se                 -0.104321      0.009127        -0.081629  -0.072497
fractal_dimension_se        -0.042641      0.054458        -0.005523  -0.019887
radius_worst                 0.969539      0.352573         0.969476   0.962746
texture_worst                0.297008      0.912045         0.303038   0.287489
perimeter_worst              0.965137      0.358040         0.970387   0.959120
area_worst                   0.941082      0.343546         0.941550   0.959213
smoothness_worst             0.119616      0.077503         0.150549   0.123523
compactness_worst            0.413463      0.277830         0.455774   0.390410
concavity_worst              0.526911      0.301025         0.563879   0.512606
concave points_worst         0.744214      0.295316         0.771241   0.722017
symmetry_worst               0.163953      0.105008         0.189115   0.143570
fractal_dimension_worst      0.007066      0.119205         0.051019   0.003738

                         smoothness_mean  compactness_mean  concavity_mean  \
radius_mean                     0.170581          0.506124        0.676764
texture_mean                   -0.023389          0.236702        0.302418
perimeter_mean                  0.207278          0.556936        0.716136
area_mean                       0.177028          0.498502        0.685983
smoothness_mean                 1.000000          0.659123        0.521984
compactness_mean                0.659123          1.000000        0.883121
concavity_mean                  0.521984          0.883121        1.000000
concave points_mean             0.553695          0.831135        0.921391
symmetry_mean                   0.557775          0.602641        0.500667
fractal_dimension_mean          0.584792          0.565369        0.336783
radius_se                       0.301467          0.497473        0.631925
texture_se                      0.068406          0.046205        0.076218
perimeter_se                    0.296092          0.548905        0.660391
area_se                         0.246552          0.455653        0.617427
smoothness_se                   0.332375          0.135299        0.098564
compactness_se                  0.318943          0.738722        0.670279
```

```
concavity_se                      0.248396          0.570517         0.691270
concave points_se                 0.380676          0.642262         0.683260
symmetry_se                       0.200774          0.229977         0.178009
fractal_dimension_se              0.283607          0.507318         0.449301
radius_worst                      0.213120          0.535315         0.688236
texture_worst                     0.036072          0.248133         0.299879
perimeter_worst                   0.238853          0.590210         0.729565
area_worst                        0.206718          0.509604         0.675987
smoothness_worst                  0.805324          0.565541         0.448822
compactness_worst                 0.472468          0.865809         0.754968
concavity_worst                   0.434926          0.816275         0.884103
concave points_worst              0.503053          0.815573         0.861323
symmetry_worst                    0.394309          0.510223         0.409464
fractal_dimension_worst           0.499316          0.687382         0.514930

                         concave points_mean  symmetry_mean  \
radius_mean                         0.822529       0.147741
texture_mean                        0.293464       0.071401
perimeter_mean                      0.850977       0.183027
area_mean                           0.823269       0.151293
smoothness_mean                     0.553695       0.557775
compactness_mean                    0.831135       0.602641
concavity_mean                      0.921391       0.500667
concave points_mean                 1.000000       0.462497
symmetry_mean                       0.462497       1.000000
fractal_dimension_mean              0.166917       0.479921
radius_se                           0.698050       0.303379
texture_se                          0.021480       0.128053
perimeter_se                        0.710650       0.313893
area_se                             0.690299       0.223970
smoothness_se                       0.027653       0.187321
compactness_se                      0.490424       0.421659
concavity_se                        0.439167       0.342627
concave points_se                   0.615634       0.393298
symmetry_se                         0.095351       0.449137
fractal_dimension_se                0.257584       0.331786
radius_worst                        0.830318       0.185728
texture_worst                       0.292752       0.090651
perimeter_worst                     0.855923       0.219169
area_worst                          0.809630       0.177193
smoothness_worst                    0.452753       0.426675
compactness_worst                   0.667454       0.473200
concavity_worst                     0.752399       0.433721
concave points_worst                0.910155       0.430297
symmetry_worst                      0.375744       0.699826
fractal_dimension_worst             0.368661       0.438413
```

```
                         fractal_dimension_mean  radius_se  texture_se  \
radius_mean                           -0.311631   0.679090   -0.097317
texture_mean                          -0.076437   0.275869    0.386358
perimeter_mean                        -0.261477   0.691765   -0.086761
area_mean                             -0.283110   0.732562   -0.066280
smoothness_mean                        0.584792   0.301467    0.068406
compactness_mean                       0.565369   0.497473    0.046205
concavity_mean                         0.336783   0.631925    0.076218
concave points_mean                    0.166917   0.698050    0.021480
symmetry_mean                          0.479921   0.303379    0.128053
fractal_dimension_mean                 1.000000   0.000111    0.164174
radius_se                              0.000111   1.000000    0.213247
texture_se                             0.164174   0.213247    1.000000
perimeter_se                           0.039830   0.972794    0.223171
area_se                               -0.090170   0.951830    0.111567
smoothness_se                          0.401964   0.164514    0.397243
compactness_se                         0.559837   0.356065    0.231700
concavity_se                           0.446630   0.332358    0.194998
concave points_se                      0.341198   0.513346    0.230283
symmetry_se                            0.345007   0.240567    0.411621
fractal_dimension_se                   0.688132   0.227754    0.279723
radius_worst                          -0.253691   0.715065   -0.111690
texture_worst                         -0.051269   0.194799    0.409003
perimeter_worst                       -0.205151   0.719684   -0.102242
area_worst                            -0.231854   0.751548   -0.083195
smoothness_worst                       0.504942   0.141919   -0.073658
compactness_worst                      0.458798   0.287103   -0.092439
concavity_worst                        0.346234   0.380585   -0.068956
concave points_worst                   0.175325   0.531062   -0.119638
symmetry_worst                         0.334019   0.094543   -0.128215
fractal_dimension_worst                0.767297   0.049559   -0.045655

                         perimeter_se   area_se  smoothness_se  \
radius_mean                  0.674172  0.735864      -0.222600
texture_mean                 0.281673  0.259845       0.006614
perimeter_mean               0.693135  0.744983      -0.202694
area_mean                    0.726628  0.800086      -0.166777
smoothness_mean              0.296092  0.246552       0.332375
compactness_mean             0.548905  0.455653       0.135299
concavity_mean               0.660391  0.617427       0.098564
concave points_mean          0.710650  0.690299       0.027653
symmetry_mean                0.313893  0.223970       0.187321
fractal_dimension_mean       0.039830 -0.090170       0.401964
radius_se                    0.972794  0.951830       0.164514
texture_se                   0.223171  0.111567       0.397243
perimeter_se                 1.000000  0.937655       0.151075
area_se                      0.937655  1.000000       0.075150
```

```
smoothness_se                0.151075  0.075150       1.000000
compactness_se               0.416322  0.284840       0.336696
concavity_se                 0.362482  0.270895       0.268685
concave points_se            0.556264  0.415730       0.328429
symmetry_se                  0.266487  0.134109       0.413506
fractal_dimension_se         0.244143  0.127071       0.427374
radius_worst                 0.697201  0.757373      -0.230691
texture_worst                0.200371  0.196497      -0.074743
perimeter_worst              0.721031  0.761213      -0.217304
area_worst                   0.730713  0.811408      -0.182195
smoothness_worst             0.130054  0.125389       0.314457
compactness_worst            0.341919  0.283257      -0.055558
concavity_worst              0.418899  0.385100      -0.058298
concave points_worst         0.554897  0.538166      -0.102007
symmetry_worst               0.109930  0.074126      -0.107342
fractal_dimension_worst      0.085433  0.017539       0.101480

                         compactness_se  concavity_se  concave points_se  \
radius_mean                    0.206000      0.194204           0.376169
texture_mean                   0.191975      0.143293           0.163851
perimeter_mean                 0.250744      0.228082           0.407217
area_mean                      0.212583      0.207660           0.372320
smoothness_mean                0.318943      0.248396           0.380676
compactness_mean               0.738722      0.570517           0.642262
concavity_mean                 0.670279      0.691270           0.683260
concave points_mean            0.490424      0.439167           0.615634
symmetry_mean                  0.421659      0.342627           0.393298
fractal_dimension_mean         0.559837      0.446630           0.341198
radius_se                      0.356065      0.332358           0.513346
texture_se                     0.231700      0.194998           0.230283
perimeter_se                   0.416322      0.362482           0.556264
area_se                        0.284840      0.270895           0.415730
smoothness_se                  0.336696      0.268685           0.328429
compactness_se                 1.000000      0.801268           0.744083
concavity_se                   0.801268      1.000000           0.771804
concave points_se              0.744083      0.771804           1.000000
symmetry_se                    0.394713      0.309429           0.312780
fractal_dimension_se           0.803269      0.727372           0.611044
radius_worst                   0.204607      0.186904           0.358127
texture_worst                  0.143003      0.100241           0.086741
perimeter_worst                0.260516      0.226680           0.394999
area_worst                     0.199371      0.188353           0.342271
smoothness_worst               0.227394      0.168481           0.215351
compactness_worst              0.678780      0.484858           0.452888
concavity_worst                0.639147      0.662564           0.549592
concave points_worst           0.483208      0.440472           0.602450
symmetry_worst                 0.277878      0.197788           0.143116
```

```
fractal_dimension_worst        0.590973      0.439329           0.310655

                         symmetry_se  fractal_dimension_se  radius_worst  \
radius_mean                -0.104321             -0.042641      0.969539
texture_mean                0.009127              0.054458      0.352573
perimeter_mean             -0.081629             -0.005523      0.969476
area_mean                  -0.072497             -0.019887      0.962746
smoothness_mean             0.200774              0.283607      0.213120
compactness_mean            0.229977              0.507318      0.535315
concavity_mean              0.178009              0.449301      0.688236
concave points_mean         0.095351              0.257584      0.830318
symmetry_mean               0.449137              0.331786      0.185728
fractal_dimension_mean      0.345007              0.688132     -0.253691
radius_se                   0.240567              0.227754      0.715065
texture_se                  0.411621              0.279723     -0.111690
perimeter_se                0.266487              0.244143      0.697201
area_se                     0.134109              0.127071      0.757373
smoothness_se               0.413506              0.427374     -0.230691
compactness_se              0.394713              0.803269      0.204607
concavity_se                0.309429              0.727372      0.186904
concave points_se           0.312780              0.611044      0.358127
symmetry_se                 1.000000              0.369078     -0.128121
fractal_dimension_se        0.369078              1.000000     -0.037488
radius_worst               -0.128121             -0.037488      1.000000
texture_worst              -0.077473             -0.003195      0.359921
perimeter_worst            -0.103753             -0.001000      0.993708
area_worst                 -0.110343             -0.022736      0.984015
smoothness_worst           -0.012662              0.170568      0.216574
compactness_worst           0.060255              0.390159      0.475820
concavity_worst             0.037119              0.379975      0.573975
concave points_worst       -0.030413              0.215204      0.787424
symmetry_worst              0.389402              0.111094      0.243529
fractal_dimension_worst     0.078079              0.591328      0.093492

                         texture_worst  perimeter_worst  area_worst  \
radius_mean                   0.297008         0.965137    0.941082
texture_mean                  0.912045         0.358040    0.343546
perimeter_mean                0.303038         0.970387    0.941550
area_mean                     0.287489         0.959120    0.959213
smoothness_mean               0.036072         0.238853    0.206718
compactness_mean              0.248133         0.590210    0.509604
concavity_mean                0.299879         0.729565    0.675987
concave points_mean           0.292752         0.855923    0.809630
symmetry_mean                 0.090651         0.219169    0.177193
fractal_dimension_mean       -0.051269        -0.205151   -0.231854
radius_se                     0.194799         0.719684    0.751548
texture_se                    0.409003        -0.102242   -0.083195
```

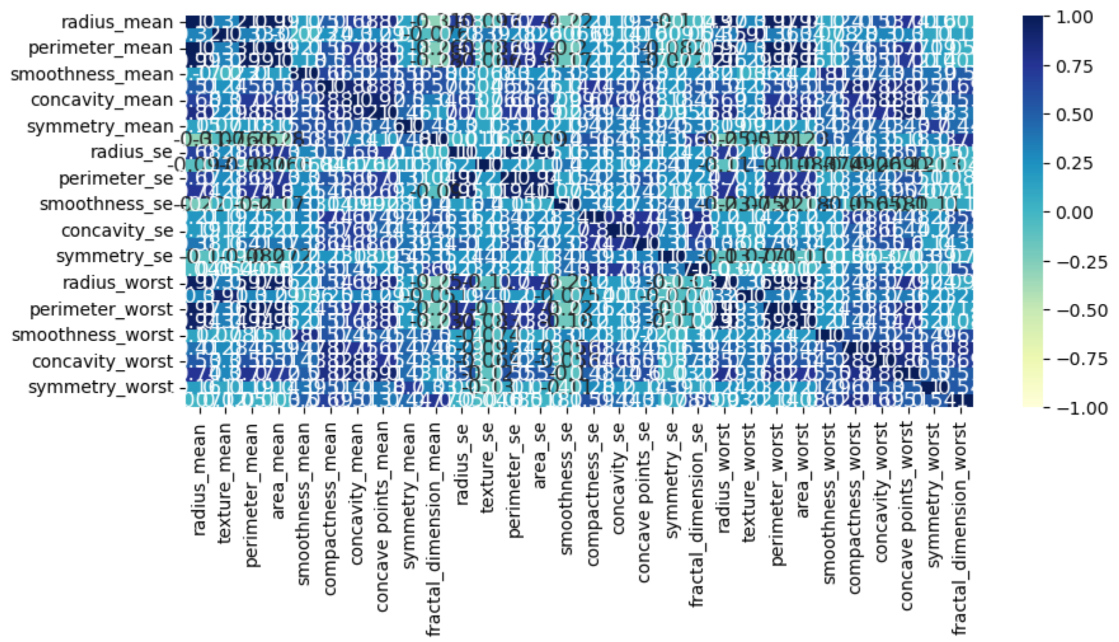
```
perimeter_se                0.200371        0.721031    0.730713
area_se                     0.196497        0.761213    0.811408
smoothness_se              -0.074743       -0.217304   -0.182195
compactness_se              0.143003        0.260516    0.199371
concavity_se                0.100241        0.226680    0.188353
concave points_se           0.086741        0.394999    0.342271
symmetry_se                -0.077473       -0.103753   -0.110343
fractal_dimension_se       -0.003195       -0.001000   -0.022736
radius_worst                0.359921        0.993708    0.984015
texture_worst               1.000000        0.365098    0.345842
perimeter_worst             0.365098        1.000000    0.977578
area_worst                  0.345842        0.977578    1.000000
smoothness_worst            0.225429        0.236775    0.209145
compactness_worst           0.360832        0.529408    0.438296
concavity_worst             0.368366        0.618344    0.543331
concave points_worst        0.359755        0.816322    0.747419
symmetry_worst              0.233027        0.269493    0.209146
fractal_dimension_worst     0.219122        0.138957    0.079647

                         smoothness_worst  compactness_worst  concavity_worst  \
radius_mean                      0.119616           0.413463         0.526911
texture_mean                     0.077503           0.277830         0.301025
perimeter_mean                   0.150549           0.455774         0.563879
area_mean                        0.123523           0.390410         0.512606
smoothness_mean                  0.805324           0.472468         0.434926
compactness_mean                 0.565541           0.865809         0.816275
concavity_mean                   0.448822           0.754968         0.884103
concave points_mean              0.452753           0.667454         0.752399
symmetry_mean                    0.426675           0.473200         0.433721
fractal_dimension_mean           0.504942           0.458798         0.346234
radius_se                        0.141919           0.287103         0.380585
texture_se                      -0.073658          -0.092439        -0.068956
perimeter_se                     0.130054           0.341919         0.418899
area_se                          0.125389           0.283257         0.385100
smoothness_se                    0.314457          -0.055558        -0.058298
compactness_se                   0.227394           0.678780         0.639147
concavity_se                     0.168481           0.484858         0.662564
concave points_se                0.215351           0.452888         0.549592
symmetry_se                     -0.012662           0.060255         0.037119
fractal_dimension_se             0.170568           0.390159         0.379975
radius_worst                     0.216574           0.475820         0.573975
texture_worst                    0.225429           0.360832         0.368366
perimeter_worst                  0.236775           0.529408         0.618344
area_worst                       0.209145           0.438296         0.543331
smoothness_worst                 1.000000           0.568187         0.518523
compactness_worst                0.568187           1.000000         0.892261
concavity_worst                  0.518523           0.892261         1.000000
```

```
concave points_worst            0.547691          0.801080         0.855434
symmetry_worst                  0.493838          0.614441         0.532520
fractal_dimension_worst         0.617624          0.810455         0.686511

                         concave points_worst  symmetry_worst  \
radius_mean                          0.744214        0.163953
texture_mean                         0.295316        0.105008
perimeter_mean                       0.771241        0.189115
area_mean                            0.722017        0.143570
smoothness_mean                      0.503053        0.394309
compactness_mean                     0.815573        0.510223
concavity_mean                       0.861323        0.409464
concave points_mean                  0.910155        0.375744
symmetry_mean                        0.430297        0.699826
fractal_dimension_mean               0.175325        0.334019
radius_se                            0.531062        0.094543
texture_se                          -0.119638       -0.128215
perimeter_se                         0.554897        0.109930
area_se                              0.538166        0.074126
smoothness_se                       -0.102007       -0.107342
compactness_se                       0.483208        0.277878
concavity_se                         0.440472        0.197788
concave points_se                    0.602450        0.143116
symmetry_se                         -0.030413        0.389402
fractal_dimension_se                 0.215204        0.111094
radius_worst                         0.787424        0.243529
texture_worst                        0.359755        0.233027
perimeter_worst                      0.816322        0.269493
area_worst                           0.747419        0.209146
smoothness_worst                     0.547691        0.493838
compactness_worst                    0.801080        0.614441
concavity_worst                      0.855434        0.532520
concave points_worst                 1.000000        0.502528
symmetry_worst                       0.502528        1.000000
fractal_dimension_worst              0.511114        0.537848

                         fractal_dimension_worst
radius_mean                             0.007066
texture_mean                            0.119205
perimeter_mean                          0.051019
area_mean                               0.003738
smoothness_mean                         0.499316
compactness_mean                        0.687382
concavity_mean                          0.514930
concave points_mean                     0.368661
symmetry_mean                           0.438413
fractal_dimension_mean                  0.767297
```

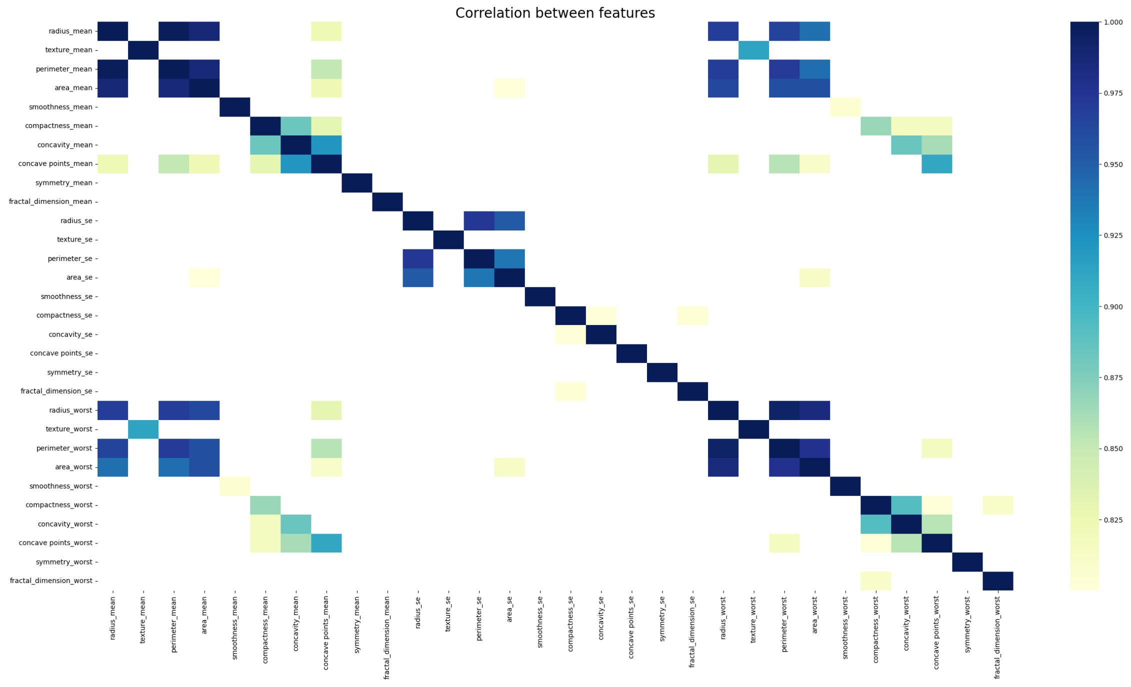
```
radius_se                    0.049559
texture_se                  -0.045655
perimeter_se                 0.085433
area_se                      0.017539
smoothness_se                0.101480
compactness_se               0.590973
concavity_se                 0.439329
concave points_se            0.310655
symmetry_se                  0.078079
fractal_dimension_se         0.591328
radius_worst                 0.093492
texture_worst                0.219122
perimeter_worst              0.138957
area_worst                   0.079647
smoothness_worst             0.617624
compactness_worst            0.810455
concavity_worst              0.686511
concave points_worst         0.511114
symmetry_worst               0.537848
fractal_dimension_worst      1.000000
```

[30]:
```python
corr=data_x.corr()
sns.heatmap(corr, cmap = 'YlGnBu', vmax = 1.0, vmin = -1.0, annot = True,␣
 ↪annot_kws = {"size": 12})
```

[30]:
```
<Axes: >
```



```
[31]: plt.figure(figsize=(30,15))
      sns.heatmap(corr[(corr>=0.8)|(corr<=-0.8)],cmap="YlGnBu",vmax=1)
      plt.title("Correlation between features",fontsize=20)
      plt.show()
```



```
[33]: drop_list=['perimeter_mean','compactness_mean','concave␣
       ↪points_mean','radius_se','perimeter_se','radius_worst','perimeter_worst','compactness_worst
       ↪points_worst','compactness_se','concave␣
       ↪points_se','texture_worst','area_worst']
      data_dummy=data.drop(drop_list,axis=1)
      data_dummy.head()
```

```
[33]:    diagnosis  radius_mean  texture_mean  area_mean  smoothness_mean  \
      0          1        17.99         10.38     1001.0          0.11840
      1          1        20.57         17.77     1326.0          0.08474
      2          1        19.69         21.25     1203.0          0.10960
      3          1        11.42         20.38      386.1          0.14250
      4          1        20.29         14.34     1297.0          0.10030

         concavity_mean  symmetry_mean  fractal_dimension_mean  texture_se  area_se  \
      0          0.3001         0.2419                 0.07871      0.9053   153.40
      1          0.0869         0.1812                 0.05667      0.7339    74.08
      2          0.1974         0.2069                 0.05999      0.7869    94.03
      3          0.2414         0.2597                 0.09744      1.1560    27.23
      4          0.1980         0.1809                 0.05883      0.7813    94.44
```

```
   smoothness_se  concavity_se  symmetry_se  fractal_dimension_se  \
0       0.006399       0.05373      0.03003              0.006193
1       0.005225       0.01860      0.01389              0.003532
2       0.006150       0.03832      0.02250              0.004571
3       0.009110       0.05661      0.05963              0.009208
4       0.011490       0.05688      0.01756              0.005115

   smoothness_worst  concavity_worst  symmetry_worst  fractal_dimension_worst
0            0.1622           0.7119          0.4601                  0.11890
1            0.1238           0.2416          0.2750                  0.08902
2            0.1444           0.4504          0.3613                  0.08758
3            0.2098           0.6869          0.6638                  0.17300
4            0.1374           0.4000          0.2364                  0.07678
```

[34]:
```python
X = data_dummy.drop(['diagnosis'], axis = 1)
y = pd.DataFrame(data_dummy['diagnosis'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 1)
```

[36]:
```python
def get_test_report(model):
    return(classification_report(y_test,y_pred))
```

[37]:
```python
def kappa_score(model):
    return(cohen_kappa_score(y_test,y_pred))
```

[38]:
```python
def plot_confusion_matrix(model):
    cm = confusion_matrix(y_test, y_pred)
    conf_matrix = pd.DataFrame(data = cm,columns = ['Predicted:0','Predicted:
 ↪1'], index = ['Actual:0','Actual:1'])
    sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap =␣
 ↪ListedColormap(['lightskyblue']),cbar = False, linewidths = 0.1, annot_kws =␣
 ↪{'size':25})
    plt.xticks(fontsize = 20)
    plt.yticks(fontsize = 20)
    plt.show()
```

[39]:
```python
def plot_roc(model):
    fpr,tpr,_=roc_curve(y_test,y_pred_prob)
    plt.plot(fpr,tpr)
    plt.xlim([0.0,1.0])
    plt.ylim([0.0,1.0])
    plt.plot([0,1],[0,1],"r--")
    plt.title("ROC Curve",fontsize=15)
    plt.xlabel("False positive",fontsize=15)
    plt.ylabel("True positive",fontsize=15)
    plt.text(x=0.02,y=0.9,s=("AUC Score:
↪",round(roc_auc_score(y_test,y_pred_prob),4)))
    plt.grid(True)
```

```
[40]: score_card=pd.DataFrame(columns=["Model","AUC Score","Precision Score","Recall␣
↪Score","Accuracy Score","Kappa Score","f1-Score"])
def update_score_card(model_name):
    global score_card
    score_card=score_card.append({"Model":model_name,"AUC Score":
↪roc_auc_score(y_test,y_pred_prob),"Precision Score":metrics.
↪precision_score(y_test,y_pred),"Recall Score":metrics.
↪accuracy_score(y_test,y_pred),'Accuracy Score': metrics.
↪accuracy_score(y_test, y_pred),"Kappa Score":
↪cohen_kappa_score(y_test,y_pred),"f1-Score":metrics.
↪f1_score(y_test,y_pred)},ignore_index=True)
    return(score_card)
```

After completing data cleaning and certain exploratory data analysis (EDA) steps, we partitioned the data into two sets: a training set comprising 80% of the observations and a test set with 20% of the observations to assess the model's accuracy.

In this phase, we applied various machine learning models, namely Logistic Regression, Decision Tree, Naive Bayes, and Support Vector Machine. Subsequently, we compared the accuracy of these different models, selecting the best-performing ones for deployment.

```
[41]: #SGDC Classifier with constant(intercept term alpha)
SGD = SGDClassifier(loss = 'log', random_state = 10)
Log_Reg_with_SGD = SGD.fit(X_train, y_train)
```

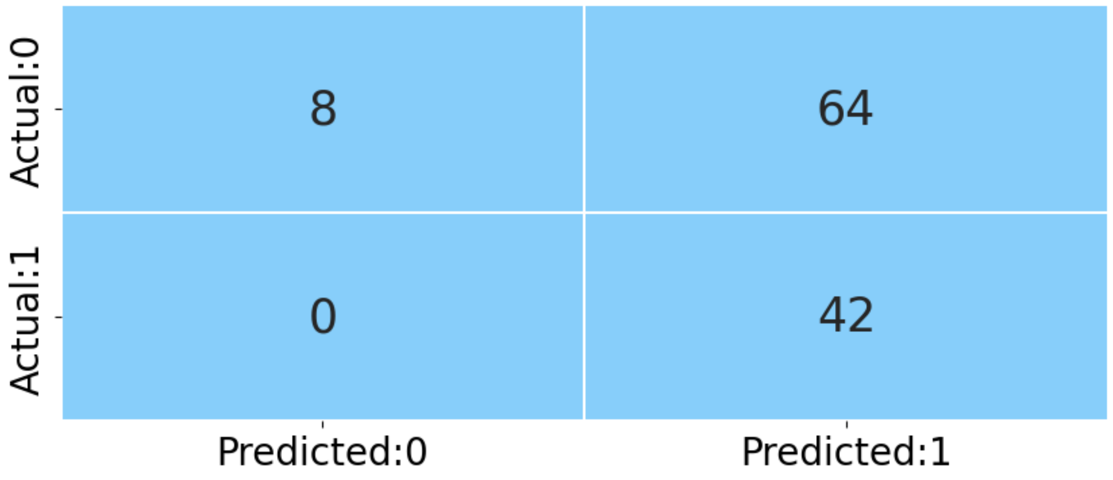
```
[42]: y_pred_prob =Log_Reg_with_SGD.predict_proba(X_test)[:,1]
y_pred_prob
```

```
[42]: array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 1., 0., 1., 1., 1., 1., 1.,
       0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

```
[43]: y_pred =Log_Reg_with_SGD.predict(X_test)
y_pred
```

```
[43]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1], dtype=int64)
```

[44]:
```python
plot_confusion_matrix(Log_Reg_with_SGD)
```



The confusion matrix reveals a 22.93% false negative rate and a 7.3% false positive rate, leading to an overall accuracy of 69.72%. This accuracy is comparatively lower than that of the previous model.

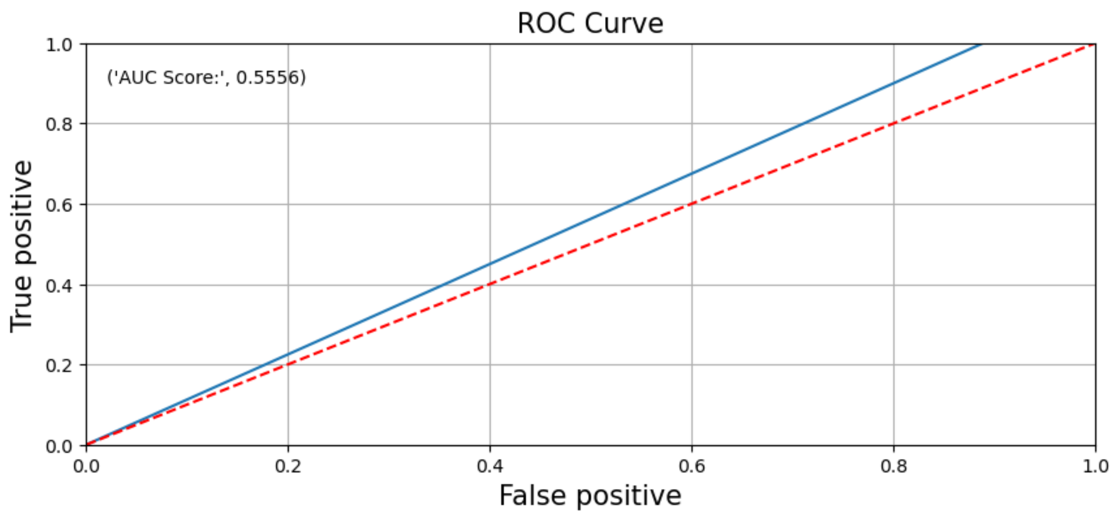
[45]:
```python
test_report = get_test_report(Log_Reg_with_SGD)
print(test_report)
```

```
              precision    recall  f1-score   support

           0       1.00      0.11      0.20        72
           1       0.40      1.00      0.57        42

    accuracy                           0.44       114
   macro avg       0.70      0.56      0.38       114
weighted avg       0.78      0.44      0.34       114
```

[46]:
```python
kappa_value = kappa_score(Log_Reg_with_SGD)
print(kappa_value)
```

```
0.08433734939759036
```

[47]:
```python
plot_roc(Log_Reg_with_SGD)
```



An Area Under the Curve (AUC) score of 0.6427 on the Receiver Operating Characteristic (ROC) curve suggests a moderate discriminatory performance of the model. The ROC curve illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity) across various threshold values.

[48]:
```
update_score_card(model_name = 'Logistic Regression (SGD)')
```

[48]:
```
                       Model  AUC Score  Precision Score  Recall Score  \
0  Logistic Regression (SGD)   0.555556         0.396226      0.438596

   Accuracy Score  Kappa Score  f1-Score
0        0.438596     0.084337  0.567568
```

11 Decision Tree Classifiaction

[49]:
```
tuned_parameters=[{"criterion":["gini","entropy"],"min_samples_split":
  ↪[10,20,30],"max_depth":[3,5,7,9],"min_samples_leaf":
  ↪[15,20,25,30,35],"max_leaf_nodes":[5,10,15,20,25]}]
```

[50]:
```
decision_tree_classification=DecisionTreeClassifier(random_state=10)
grid=GridSearchCV(estimator=decision_tree_classification,param_grid=tuned_parameters,cv=10)
dt_grid=grid.fit(X_train,y_train)
print("Best parameters for DT:",dt_grid.best_params_,"\n")
```

```
Best parameters for DT: {'criterion': 'gini', 'max_depth': 3, 'max_leaf_nodes':
5, 'min_samples_leaf': 20, 'min_samples_split': 10}
```

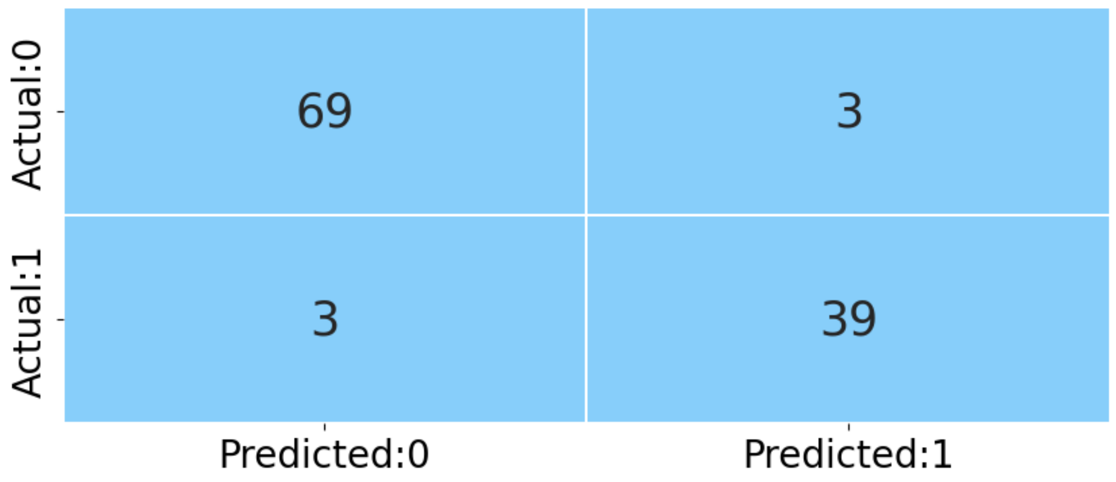
```
[51]: dt_grid_model=DecisionTreeClassifier(criterion=dt_grid.best_params_.
        ↪get("criterion"),max_depth=dt_grid.best_params_.
        ↪get("max_depth"),max_leaf_nodes=dt_grid.best_params_.
        ↪get("max_leaf_nodes"),min_samples_leaf=dt_grid.best_params_.
        ↪get("min_samples_leaf"),min_samples_split=dt_grid.best_params_.
        ↪get("min_samples_split"))
```

```
[52]: decision_tree_grid=dt_grid_model.fit(X_train,y_train)
```

```
[53]: y_pred_prob=decision_tree_grid.predict_proba(X_test)[:,1]
```

```
[54]: y_pred=decision_tree_grid.predict(X_test)
```

```
[55]: plot_confusion_matrix(decision_tree_grid)
```



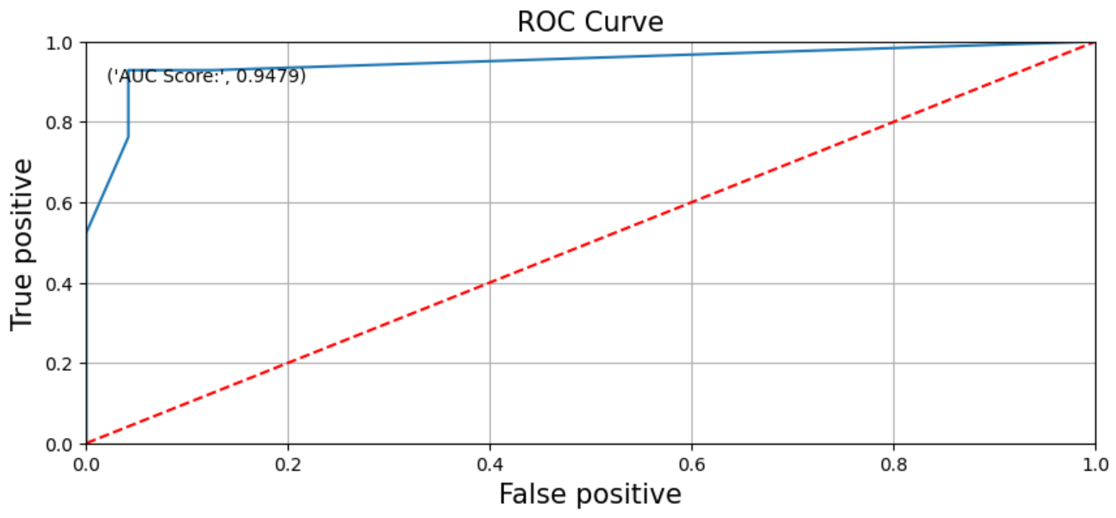
```
[56]: test_report = get_test_report(decision_tree_grid)

      # print the performace measures
      print(test_report)
```

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        72
           1       0.93      0.93      0.93        42

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114
```

[57]:
```
kappa_value = kappa_score(decision_tree_grid)

# print the kappa value
print(kappa_value)
```

```
0.8869047619047619
```

[58]:
```
plot_roc(decision_tree_grid)
```



[59]:
```
update_score_card(model_name = 'decision_tree_grid')
```

[59]:
```
                       Model  AUC Score  Precision Score  Recall Score  \
0  Logistic Regression (SGD)   0.555556         0.396226      0.438596
1         decision_tree_grid   0.947917         0.928571      0.947368

   Accuracy Score  Kappa Score  f1-Score
0        0.438596     0.084337  0.567568
1        0.947368     0.886905  0.928571
```

[60]:
```
from sklearn.naive_bayes import GaussianNB
```

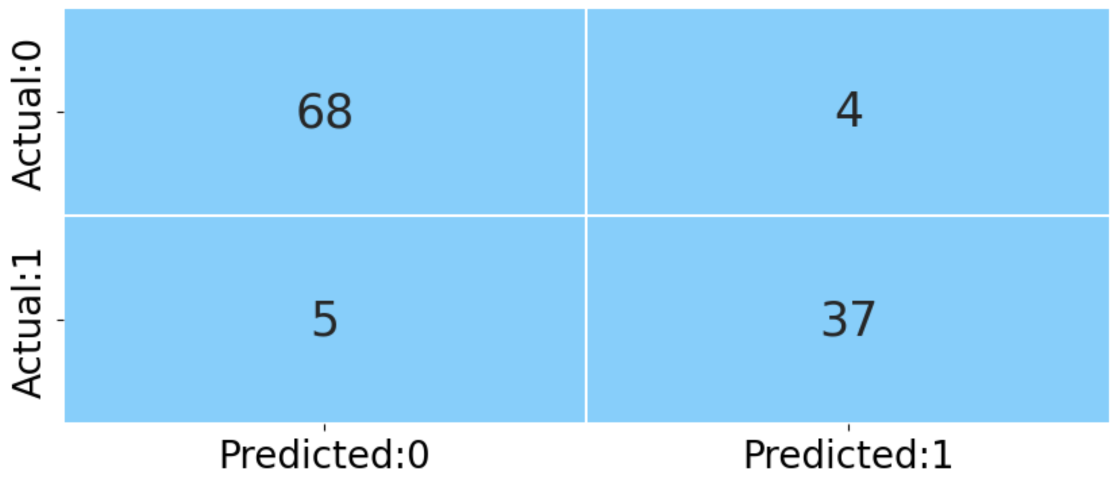
[61]:
```
Naive_Bayes_Model =GaussianNB().fit(X_train, y_train)
```

[62]:
```
y_pred_prob =Naive_Bayes_Model .predict_proba(X_test)[:,1]
```

[63]:
```
y_pred = Naive_Bayes_Model.predict(X_test)
y_pred[0:11]
```

[63]:
```
array([1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0], dtype=int64)
```

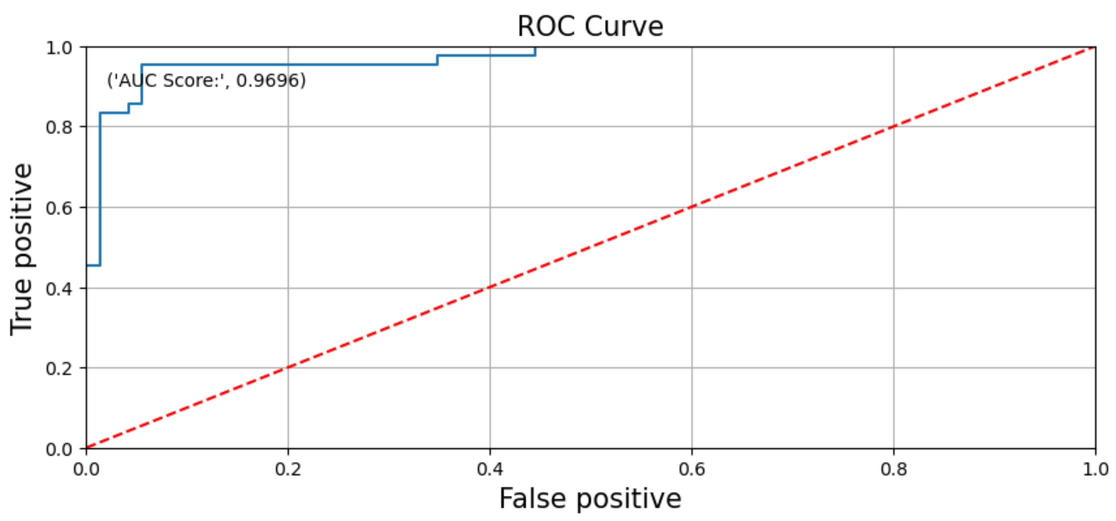
```
[64]: plot_confusion_matrix(Naive_Bayes_Model)
```



```
[65]: test_report = get_test_report(Naive_Bayes_Model)
      print(test_report)
```

```
              precision    recall  f1-score   support

           0       0.93      0.94      0.94        72
           1       0.90      0.88      0.89        42

    accuracy                           0.92       114
   macro avg       0.92      0.91      0.91       114
weighted avg       0.92      0.92      0.92       114
```

```
[66]: plot_roc(Naive_Bayes_Model)
```

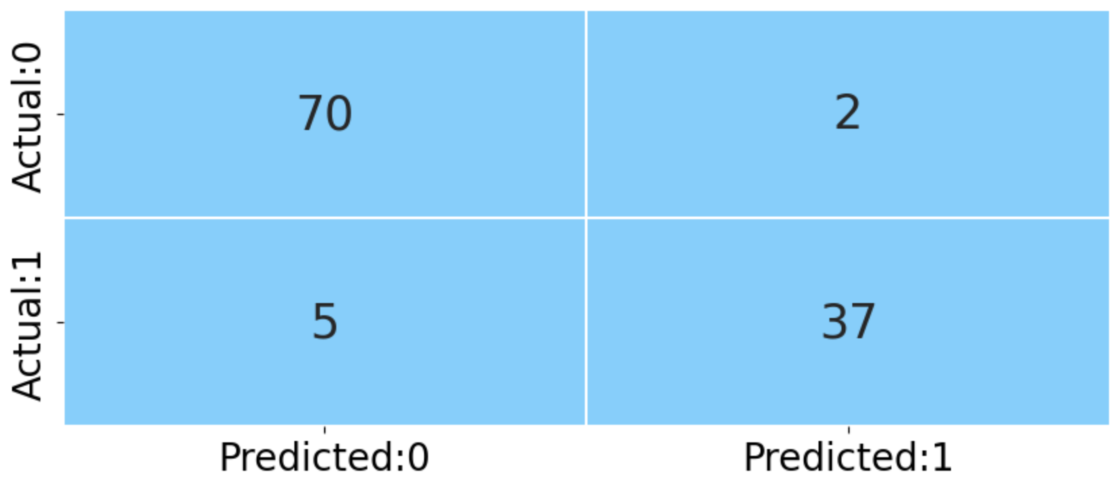


[67]:
```
update_score_card(model_name = 'Naive_Bayes_Model')
```

[67]:
```
                     Model  AUC Score  Precision Score  Recall Score  \
0  Logistic Regression (SGD)   0.555556         0.396226      0.438596
1         decision_tree_grid   0.947917         0.928571      0.947368
2          Naive_Bayes_Model   0.969577         0.902439      0.921053

   Accuracy Score  Kappa Score  f1-Score
0        0.438596     0.084337  0.567568
1        0.947368     0.886905  0.928571
2        0.921053     0.829511  0.891566
```

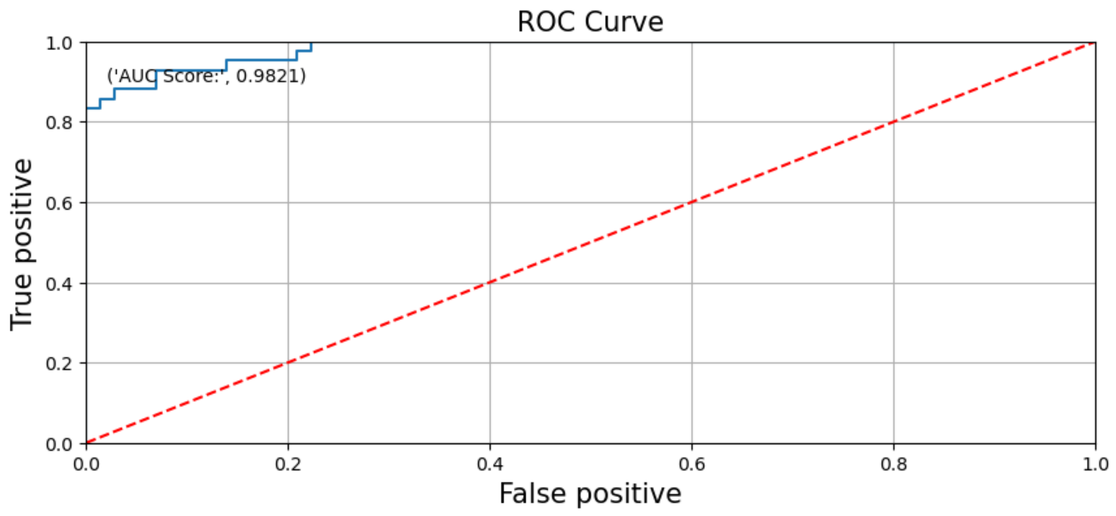
[68]:
```
from sklearn.svm import SVC
```

[69]:
```
svc_linear = SVC(kernel='linear', probability=True)  # Specify␣
 ↪'probability=True' to enable probability estimates
svm_linear=svc_linear.fit(X_train, y_train)
y_pred_prob =svm_linear.predict_proba(X_test)[:,1]
y_pred =svm_linear .predict(X_test)
plot_confusion_matrix(svm_linear)
test_report = get_test_report(svm_linear)
print(test_report)
plot_roc(svm_linear)
update_score_card(model_name = 'svm_linear')
```

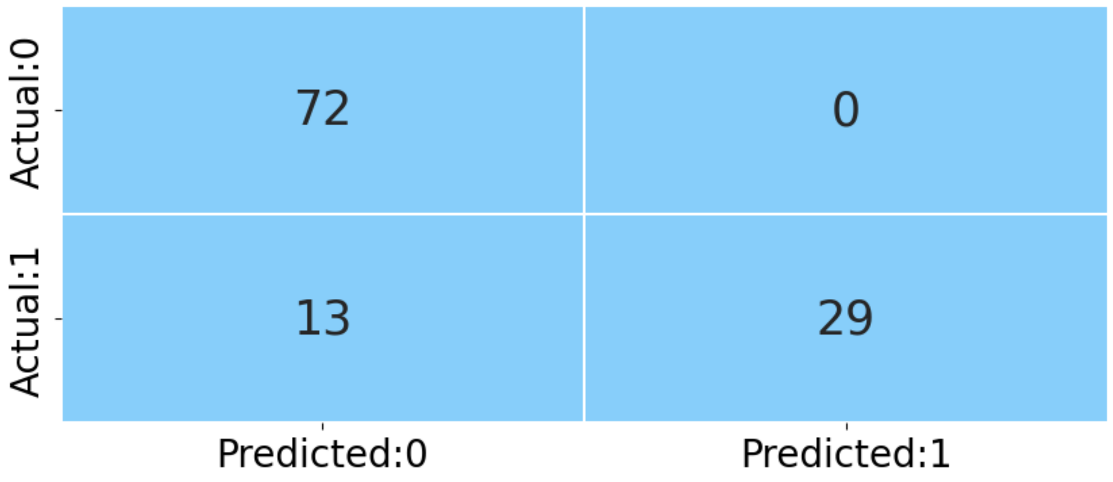


```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95        72
           1       0.95      0.88      0.91        42

    accuracy                           0.94       114
   macro avg       0.94      0.93      0.93       114
weighted avg       0.94      0.94      0.94       114
```

[69]:
```
                       Model  AUC Score  Precision Score  Recall Score  \
0  Logistic Regression (SGD)   0.555556         0.396226      0.438596
1          decision_tree_grid   0.947917         0.928571      0.947368
2           Naive_Bayes_Model   0.969577         0.902439      0.921053
3                  svm_linear   0.982143         0.948718      0.938596

   Accuracy Score  Kappa Score  f1-Score
0        0.438596     0.084337  0.567568
1        0.947368     0.886905  0.928571
2        0.921053     0.829511  0.891566
3        0.938596     0.866062  0.913580
```



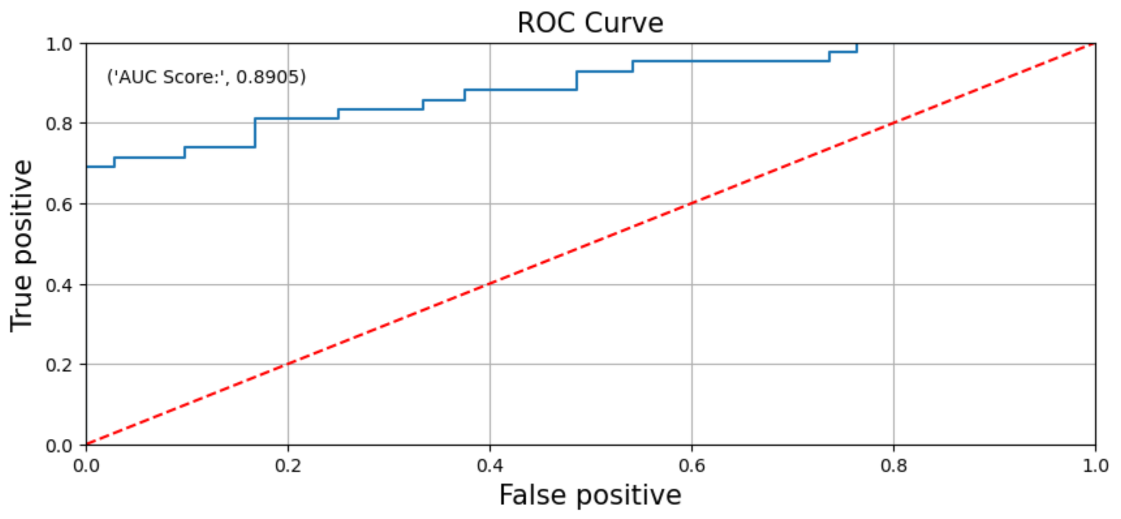
```
[70]: svc_poly = SVC(kernel='poly', probability=True)  # Specify 'probability=True'␣
       ↪to enable probability estimates
      svm_poly=svc_poly.fit(X_train, y_train)
      y_pred_prob =svm_poly.predict_proba(X_test)[:,1]
      y_pred =svm_poly .predict(X_test)
      plot_confusion_matrix(svm_poly)
      test_report = get_test_report(svm_poly)
      print(test_report)
      plot_roc(svm_poly)
      update_score_card(model_name = 'svm_poly')
```



```
              precision    recall  f1-score   support
```

```
           0       0.85      1.00      0.92        72
           1       1.00      0.69      0.82        42

    accuracy                           0.89       114
   macro avg       0.92      0.85      0.87       114
weighted avg       0.90      0.89      0.88       114
```

[70]:
```
                        Model  AUC Score  Precision Score  Recall Score  \
0  Logistic Regression (SGD)   0.555556         0.396226      0.438596
1          decision_tree_grid   0.947917         0.928571      0.947368
2           Naive_Bayes_Model   0.969577         0.902439      0.921053
3                  svm_linear   0.982143         0.948718      0.938596
4                    svm_poly   0.890542         1.000000      0.885965

   Accuracy Score  Kappa Score  f1-Score
0        0.438596     0.084337  0.567568
1        0.947368     0.886905  0.928571
2        0.921053     0.829511  0.891566
3        0.938596     0.866062  0.913580
4        0.885965     0.738070  0.816901
```



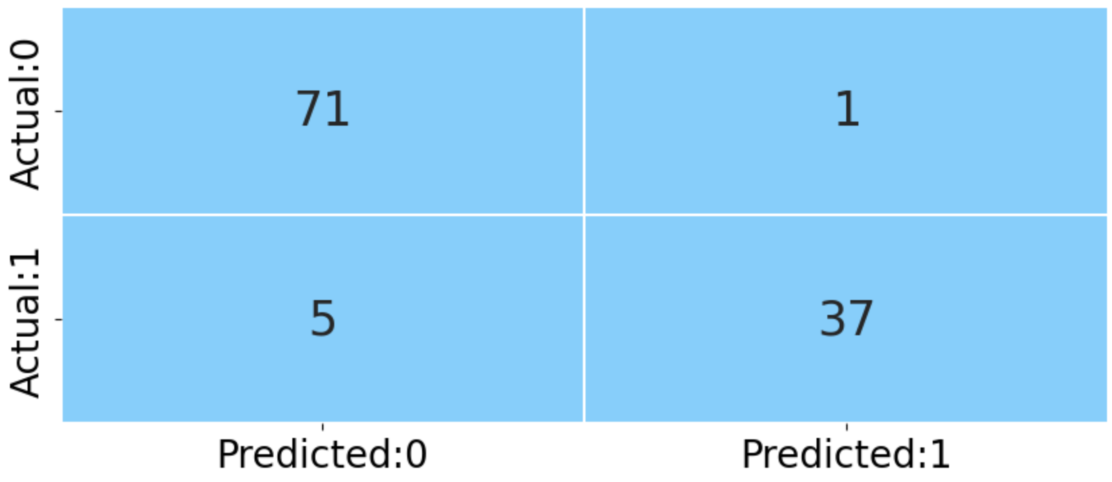
[71]:
```
from sklearn.ensemble import RandomForestClassifier
#intantiate the regressor
rf_cls = RandomForestClassifier(n_estimators=100, random_state=10)

# fit the regressor with training dataset
rf_cls.fit(X_train, y_train)
```

[71]:
```
RandomForestClassifier(random_state=10)
```

[72]:
```python
# predict the values on test dataset using predict()
y_pred = rf_cls.predict(X_test)
y_pred
```

[72]:
```
array([1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 0], dtype=int64)
```

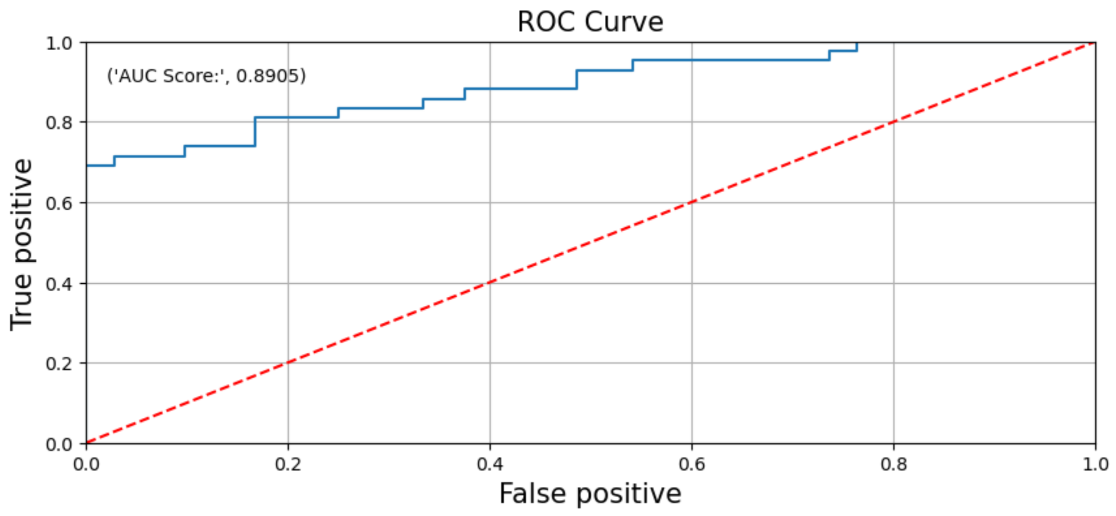
[73]:
```python
plot_confusion_matrix(rf_cls)
test_report = get_test_report(rf_cls)
print(test_report)
plot_roc(rf_cls)
update_score_card(model_name = 'Random_Forest_cls')
```



```
              precision    recall  f1-score   support

           0       0.93      0.99      0.96        72
           1       0.97      0.88      0.93        42

    accuracy                           0.95       114
   macro avg       0.95      0.93      0.94       114
weighted avg       0.95      0.95      0.95       114
```

[73]:
```
                      Model  AUC Score  Precision Score  Recall Score  \
0  Logistic Regression (SGD)   0.555556         0.396226      0.438596
1         decision_tree_grid   0.947917         0.928571      0.947368
2            Naive_Bayes_Model   0.969577         0.902439      0.921053
```

```
3                svm_linear   0.982143         0.948718       0.938596
4                  svm_poly   0.890542         1.000000       0.885965
5                    rf_cls   0.890542         0.973684       0.947368

   Accuracy Score  Kappa Score  f1-Score
0        0.438596     0.084337  0.567568
1        0.947368     0.886905  0.928571
2        0.921053     0.829511  0.891566
3        0.938596     0.866062  0.913580
4        0.885965     0.738070  0.816901
5        0.947368     0.884615  0.925000
```



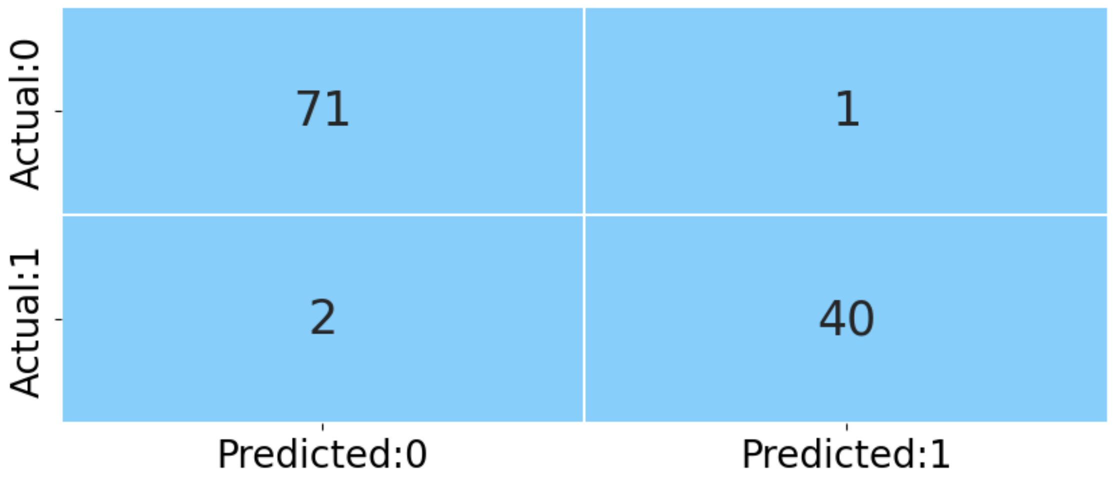
[74]:
```
X = data_dummy.drop(['diagnosis'], axis = 1)
X=sm.add_constant(X)
y = pd.DataFrame(data_dummy['diagnosis'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 1)
Log_Reg_Full_Model=sm.Logit(y_train,X_train).fit()
print(Log_Reg_Full_Model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.051659
         Iterations 16
                           Logit Regression Results
==============================================================================
Dep. Variable:              diagnosis   No. Observations:                  455
Model:                          Logit   Df Residuals:                      437
Method:                           MLE   Df Model:                           17
Date:                Sun, 16 Jun 2024   Pseudo R-squ.:                  0.9218
Time:                        18:54:19   Log-Likelihood:                -23.505
```

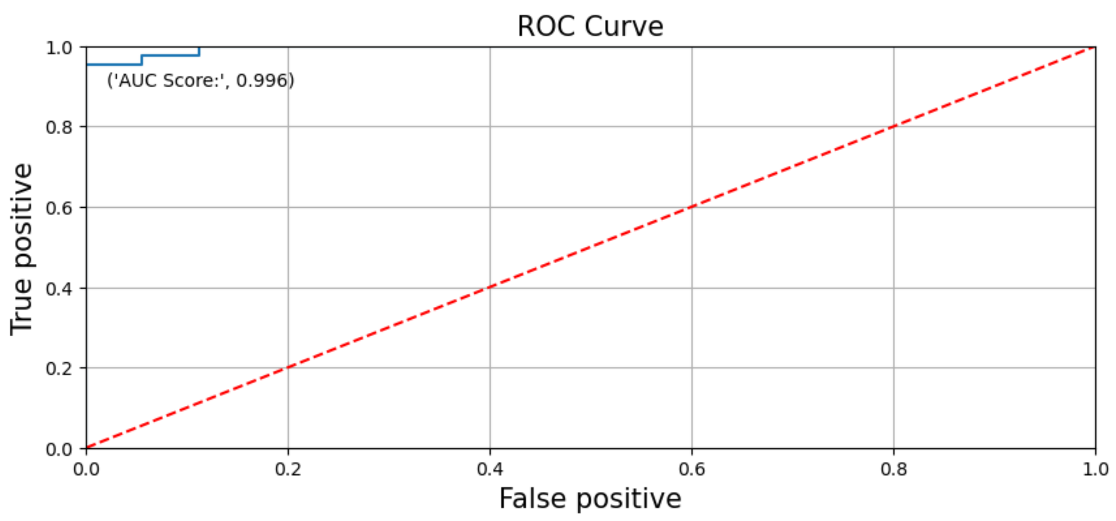
```
converged:                       True   LL-Null:                       -300.69
Covariance Type:            nonrobust   LLR p-value:                6.388e-107
================================================================================
===========
                              coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-----------
const                     -34.9582     30.804     -1.135      0.256     -95.334
25.417
radius_mean                -0.8439      4.237     -0.199      0.842      -9.149
7.461
texture_mean                0.3894      0.165      2.358      0.018       0.066
0.713
area_mean                   0.0176      0.048      0.364      0.716      -0.077
0.112
smoothness_mean            91.6007    103.554      0.885      0.376    -111.361
294.562
concavity_mean             68.8361     32.214      2.137      0.033       5.697
131.975
symmetry_mean             -17.6204     33.892     -0.520      0.603     -84.048
48.807
fractal_dimension_mean   -220.5953    207.075     -1.065      0.287    -626.455
185.265
texture_se                  0.3302      1.088      0.303      0.762      -1.803
2.463
area_se                     0.2683      0.090      2.991      0.003       0.092
0.444
smoothness_se             559.4614    355.875      1.572      0.116    -138.041
1256.963
concavity_se              -24.9318     74.681     -0.334      0.738    -171.304
121.440
symmetry_se              -230.6517    133.613     -1.726      0.084    -492.529
31.226
fractal_dimension_se    -2210.4455    940.846     -2.349      0.019   -4054.469
-366.422
smoothness_worst           -9.4006     59.282     -0.159      0.874    -125.591
106.790
concavity_worst             2.8429     11.443      0.248      0.804     -19.584
25.270
symmetry_worst             35.7199     16.709      2.138      0.033       2.970
68.469
fractal_dimension_worst   243.2878    125.553      1.938      0.053      -2.791
489.367
================================================================================
===========

Possibly complete quasi-separation: A fraction 0.65 of observations can be
```

```
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

[75]:
```
y_pred_prob=Log_Reg_Full_Model.predict(X_test)
y_pred=["0" if x<0.5 else "1" for x in y_pred_prob]
y_pred=np.array(y_pred,dtype=np.float32)
y_pred[0:5]
plot_confusion_matrix(Log_Reg_Full_Model)
plot_roc(Log_Reg_Full_Model)
update_score_card(model_name="Logistic_Regression with Full Model")
```



[75]:
```
                                Model  AUC Score  Precision Score  \
0             Logistic Regression (SGD)   0.555556         0.396226
1                     decision_tree_grid   0.947917         0.928571
2                      Naive_Bayes_Model   0.969577         0.902439
3                             svm_linear   0.982143         0.948718
4                               svm_poly   0.890542         1.000000
5                                 rf_cls   0.890542         0.973684
6  Logistic_Regression with Full Model   0.996032         0.975610

   Recall Score  Accuracy Score  Kappa Score  f1-Score
0      0.438596        0.438596     0.084337  0.567568
1      0.947368        0.947368     0.886905  0.928571
2      0.921053        0.921053     0.829511  0.891566
3      0.938596        0.938596     0.866062  0.913580
4      0.885965        0.885965     0.738070  0.816901
5      0.947368        0.947368     0.884615  0.925000
6      0.973684        0.973684     0.943170  0.963855
```



[76]:
```
# Backward elimination function
def backward_elimination(data, target):
    features = list(data.columns)
    features.remove(target)

    while len(features) > 0:
        model = sm.Logit(data[target], sm.add_constant(data[features]))
        result = model.fit(disp=False)
        max_pvalue = result.pvalues.idxmax()

        # If the highest p-value is greater than a threshold (e.g., 0.05),␣
↪remove the corresponding feature
        if result.pvalues[max_pvalue] > 0.05:
            features.remove(max_pvalue)
        else:
            break  # If all p-values are below the threshold, stop

    return features

# Example usage
target_variable = 'diagnosis'
selected_features_backward = backward_elimination(data_dummy, target_variable)

print("Selected Features (Backward):", selected_features_backward)
```

```
Selected Features (Backward): ['texture_mean', 'area_mean', 'concavity_mean',
'area_se', 'smoothness_se', 'symmetry_se', 'fractal_dimension_se',
'symmetry_worst', 'fractal_dimension_worst']
```

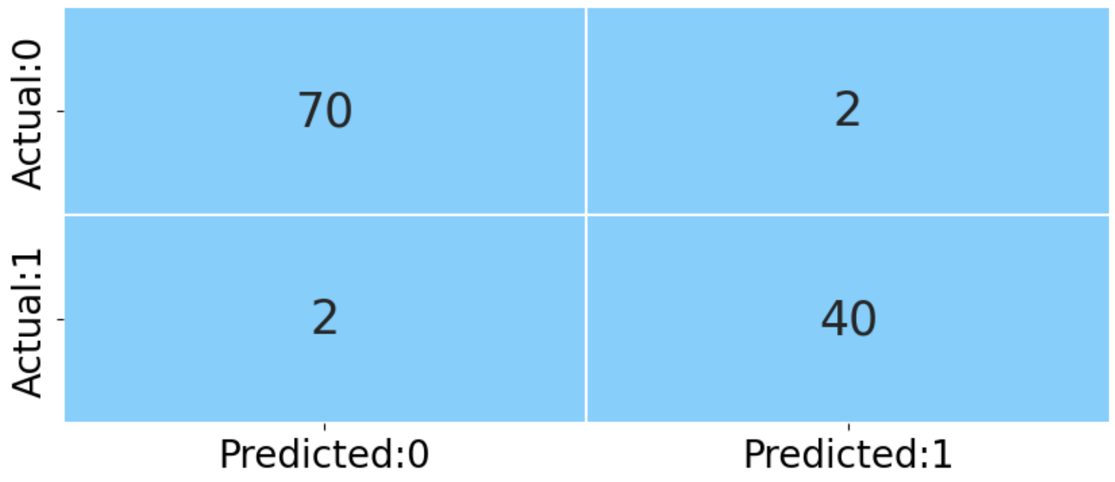
[77]:
```
X = data_dummy[ ['texture_mean', 'area_mean', 'concavity_mean', 'area_se',␣
↪'smoothness_se', 'symmetry_se', 'fractal_dimension_se', 'symmetry_worst',␣
↪'fractal_dimension_worst']]
X=sm.add_constant(X)
y = pd.DataFrame(data_dummy['diagnosis'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
↪random_state = 1)
```

[78]:
```
Log_Reg_Backward_Model_Selection=sm.Logit(y_train,X_train).fit()
print(Log_Reg_Backward_Model_Selection.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.054031
         Iterations 15
                           Logit Regression Results
==============================================================================
Dep. Variable:              diagnosis   No. Observations:                  455
Model:                          Logit   Df Residuals:                      445
Method:                           MLE   Df Model:                            9
Date:                Sun, 16 Jun 2024   Pseudo R-squ.:                  0.9182
Time:                        18:55:03   Log-Likelihood:                -24.584
converged:                       True   LL-Null:                       -300.69
Covariance Type:            nonrobust   LLR p-value:                3.735e-113
================================================================================
===========
                              coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-----------
const                     -43.4490      9.998     -4.346      0.000     -63.045
-23.853
texture_mean                0.3607      0.109      3.319      0.001       0.148
0.574
area_mean                   0.0079      0.004      1.930      0.054      -0.000
0.016
concavity_mean             65.8154     19.362      3.399      0.001      27.867
103.764
area_se                     0.2680      0.078      3.452      0.001       0.116
0.420
smoothness_se             523.6865    223.121      2.347      0.019      86.378
960.995
symmetry_se              -257.4583    111.192     -2.315      0.021    -475.391
-39.526
fractal_dimension_se    -2174.5554    692.011     -3.142      0.002   -3530.872
-818.239
symmetry_worst             33.7927     13.054      2.589      0.010       8.207
59.379
```

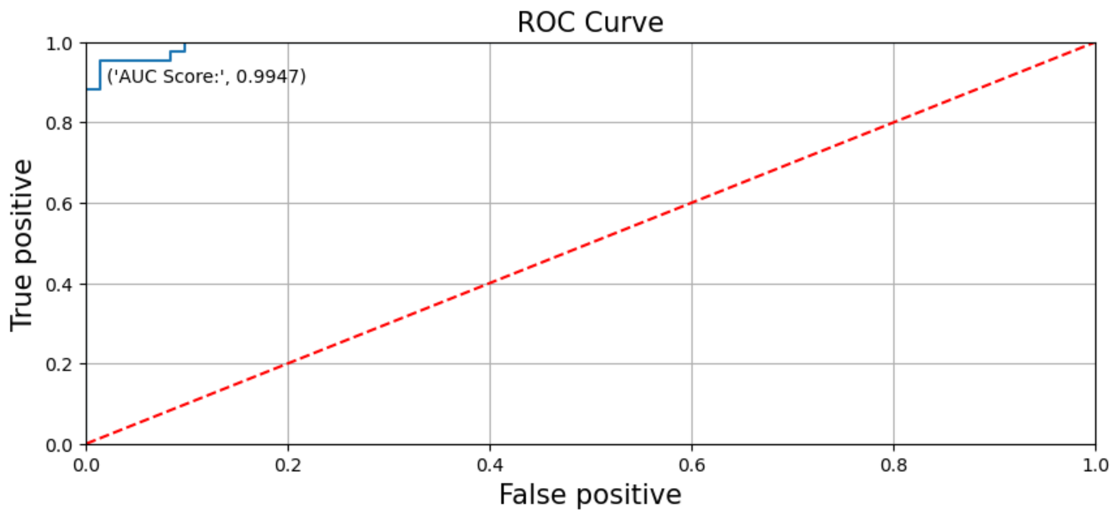
```
fractal_dimension_worst    192.6693     77.463      2.487      0.013      40.844
344.494
================================================================================
===========

Possibly complete quasi-separation: A fraction 0.61 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

[79]:
```
y_pred_prob=Log_Reg_Backward_Model_Selection.predict(X_test)
y_pred=["0" if x<0.5 else "1" for x in y_pred_prob]
y_pred=np.array(y_pred,dtype=np.float32)
y_pred[0:5]
plot_confusion_matrix(Log_Reg_Backward_Model_Selection)
plot_roc(Log_Reg_Backward_Model_Selection)
update_score_card(model_name="Log_Reg_Backward_Model_Selection")
```



[79]:
```
                                Model  AUC Score  Precision Score  \
0             Logistic Regression (SGD)   0.555556         0.396226
1                     decision_tree_grid   0.947917         0.928571
2                       Naive_Bayes_Model   0.969577         0.902439
3                              svm_linear   0.982143         0.948718
4                                svm_poly   0.890542         1.000000
5                                  rf_cls   0.890542         0.973684
6  Logistic_Regression with Full Model   0.996032         0.975610
7     Log_Reg_Backward_Model_Selection   0.994709         0.952381

   Recall Score  Accuracy Score  Kappa Score  f1-Score
0      0.438596        0.438596     0.084337  0.567568
1      0.947368        0.947368     0.886905  0.928571
2      0.921053        0.921053     0.829511  0.891566
```

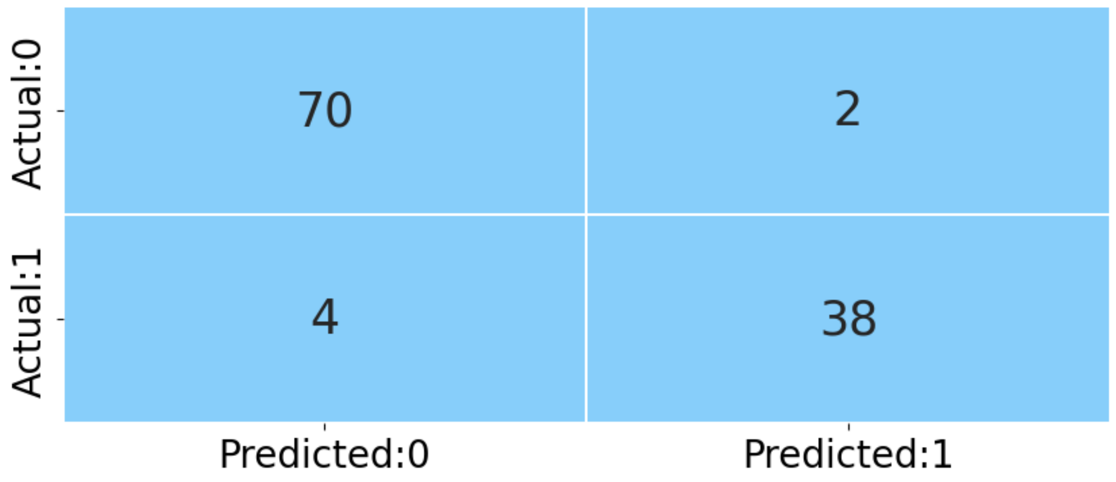
```
3       0.938596          0.938596      0.866062  0.913580
4       0.885965          0.885965      0.738070  0.816901
5       0.947368          0.947368      0.884615  0.925000
6       0.973684          0.973684      0.943170  0.963855
7       0.964912          0.964912      0.924603  0.952381
```



[80]:
```python
X = data_dummy.drop(['diagnosis'], axis = 1)
y = pd.DataFrame(data_dummy['diagnosis'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 1)
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# Initialize the GridSearchCV with RandomForestClassifier
grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42),
                           param_grid=param_grid, cv=5)

# Fit the GridSearchCV to the training data
```

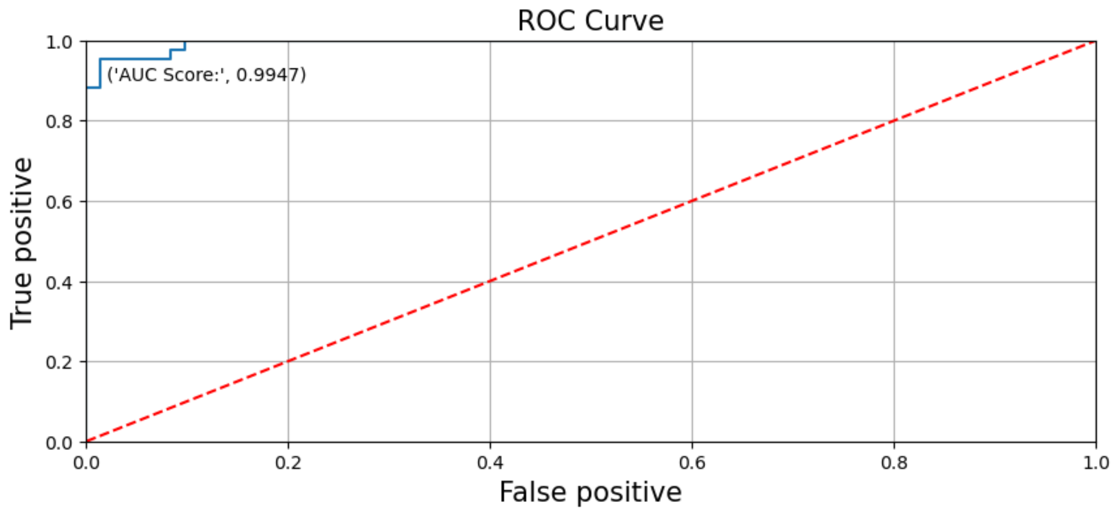
```
grid_search.fit(X_train, y_train)

# Retrieve the best parameters and the best estimator
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
print("Best Parameters: ", best_params)

# Predict the test set using the best model
y_pred = best_model.predict(X_test)

# Evaluate the model
print(classification_report(y_test, y_pred))
plot_confusion_matrix(best_model)
plot_roc(best_model)
update_score_card(model_name="Hyper_Parameter_RF")
```

```
Best Parameters:  {'bootstrap': True, 'max_depth': 10, 'max_features': 'auto',
'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
              precision    recall  f1-score   support

           0       0.95      0.97      0.96        72
           1       0.95      0.90      0.93        42

    accuracy                           0.95       114
   macro avg       0.95      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114
```



```
[80]:                              Model  AUC Score  Precision Score  \
     0          Logistic Regression (SGD)   0.555556         0.396226
```

```
1                  decision_tree_grid   0.947917         0.928571
2                   Naive_Bayes_Model   0.969577         0.902439
3                          svm_linear   0.982143         0.948718
4                            svm_poly   0.890542         1.000000
5                              rf_cls   0.890542         0.973684
6 Logistic_Regression with Full Model   0.996032         0.975610
7     Log_Reg_Backward_Model_Selection   0.994709         0.952381
8                  Hyper_Parameter_RF   0.994709         0.950000

   Recall Score  Accuracy Score  Kappa Score  f1-Score
0      0.438596        0.438596     0.084337  0.567568
1      0.947368        0.947368     0.886905  0.928571
2      0.921053        0.921053     0.829511  0.891566
3      0.938596        0.938596     0.866062  0.913580
4      0.885965        0.885965     0.738070  0.816901
5      0.947368        0.947368     0.884615  0.925000
6      0.973684        0.973684     0.943170  0.963855
7      0.964912        0.964912     0.924603  0.952381
8      0.947368        0.947368     0.885772  0.926829
```

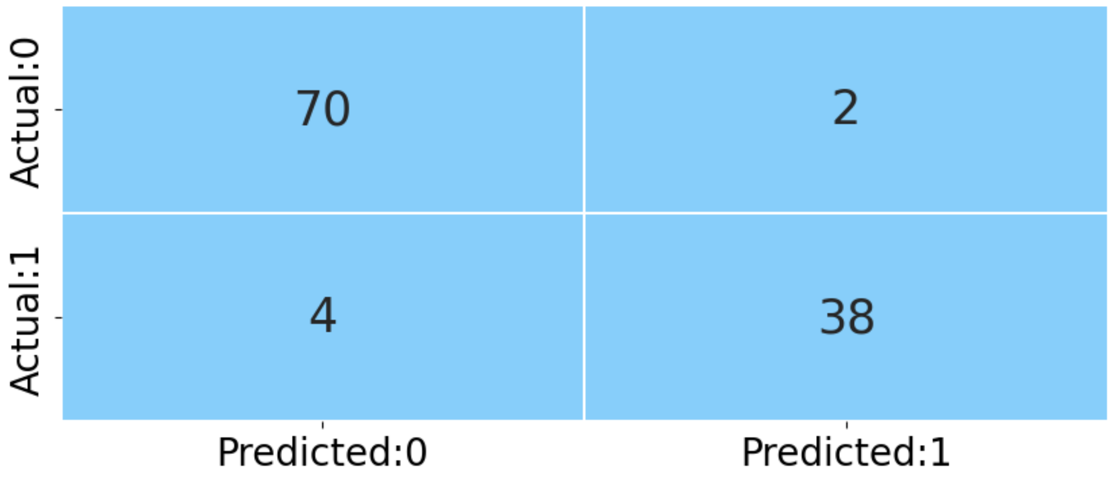


[81]:
```python
from sklearn.ensemble import BaggingClassifier
from sklearn import tree
meta_estimator=BaggingClassifier(tree.DecisionTreeClassifier(random_state=10))
meta_estimator.fit(X_train,y_train)
```

[81]:
```
BaggingClassifier(estimator=DecisionTreeClassifier(random_state=10))
```

[82]:
```python
y_pred=meta_estimator.predict(X_test)
```

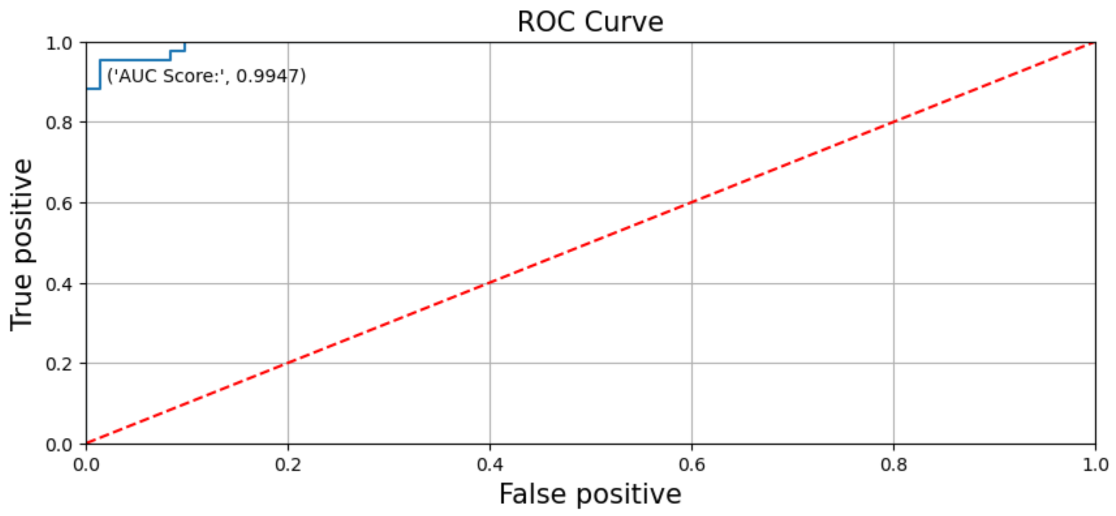
[83]:
```
plot_confusion_matrix(meta_estimator)
test_report = get_test_report(meta_estimator)
print(test_report)
plot_roc(meta_estimator)
update_score_card(model_name = 'Bagging_meta_estimator')
```



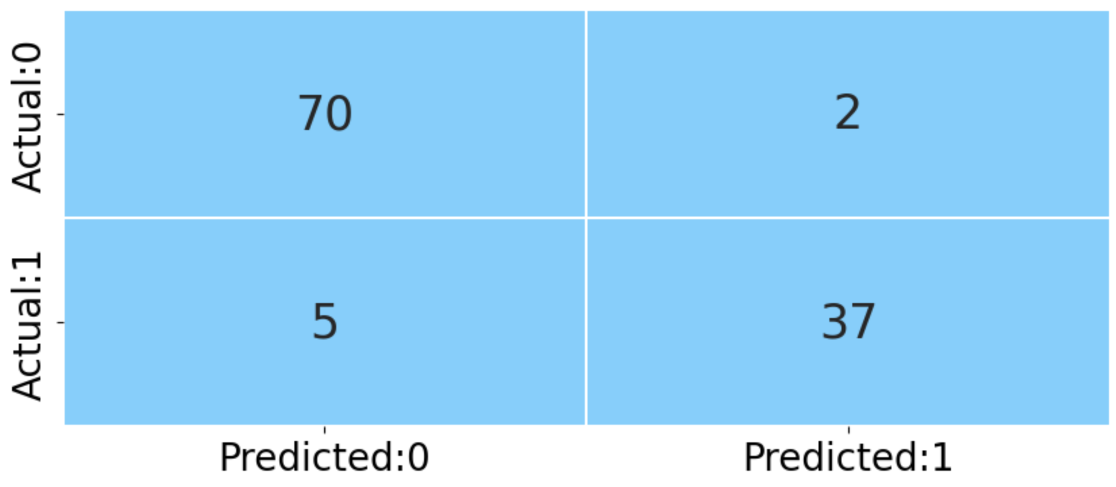
```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96        72
           1       0.95      0.90      0.93        42

    accuracy                           0.95       114
   macro avg       0.95      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114
```

[83]:
```
                                Model  AUC Score  Precision Score  \
0            Logistic Regression (SGD)   0.555556         0.396226
1                   decision_tree_grid   0.947917         0.928571
2                    Naive_Bayes_Model   0.969577         0.902439
3                           svm_linear   0.982143         0.948718
4                             svm_poly   0.890542         1.000000
5                               rf_cls   0.890542         0.973684
6  Logistic_Regression with Full Model   0.996032         0.975610
7      Log_Reg_Backward_Model_Selection   0.994709         0.952381
8                   Hyper_Parameter_RF   0.994709         0.950000
9                       meta_estimator   0.994709         0.950000

   Recall Score  Accuracy Score  Kappa Score  f1-Score
0      0.438596        0.438596     0.084337  0.567568
```

```
1       0.947368         0.947368      0.886905  0.928571
2       0.921053         0.921053      0.829511  0.891566
3       0.938596         0.938596      0.866062  0.913580
4       0.885965         0.885965      0.738070  0.816901
5       0.947368         0.947368      0.884615  0.925000
6       0.973684         0.973684      0.943170  0.963855
7       0.964912         0.964912      0.924603  0.952381
8       0.947368         0.947368      0.885772  0.926829
9       0.947368         0.947368      0.885772  0.926829
```



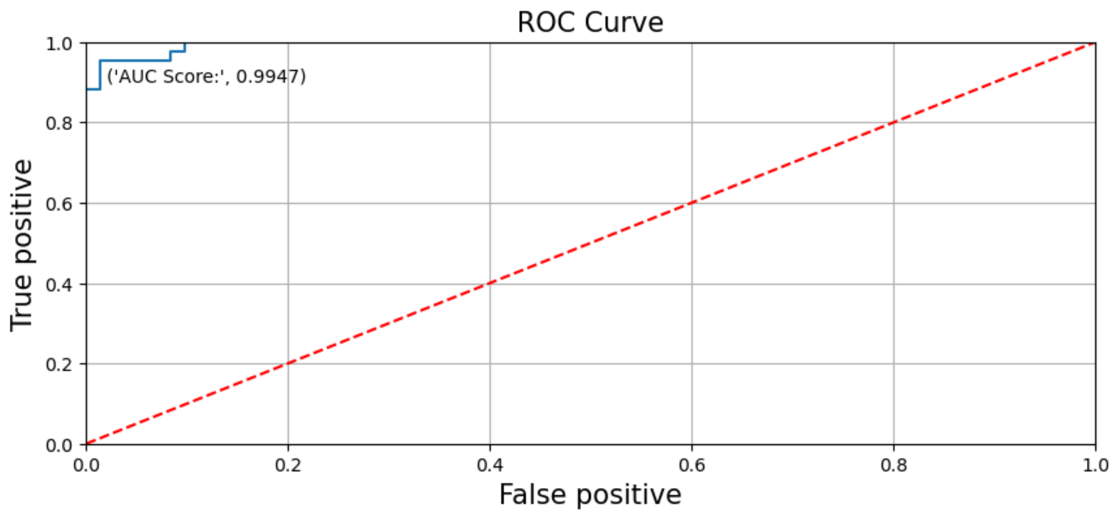
```
[84]: from sklearn.ensemble import AdaBoostClassifier
      Adaboost=AdaBoostClassifier(random_state=10)
      Adaboost.fit(X_train,y_train)
      y_pred=Adaboost.predict(X_test)
      plot_confusion_matrix(Adaboost)
      test_report = get_test_report(Adaboost)
      print(test_report)
      plot_roc(Adaboost)
      update_score_card(model_name = 'Adaboost_Estimator')
```



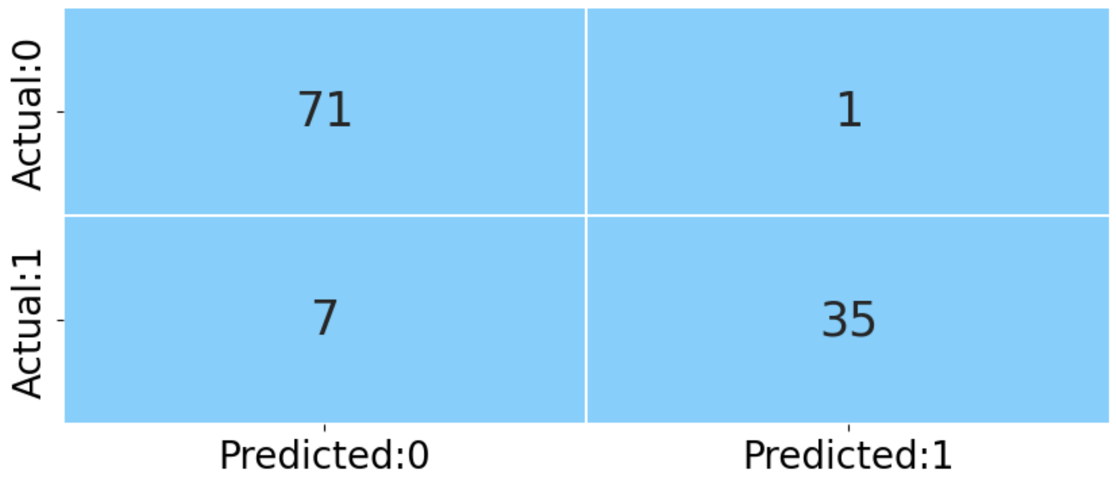
```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95        72
           1       0.95      0.88      0.91        42

    accuracy                           0.94       114
   macro avg       0.94      0.93      0.93       114
weighted avg       0.94      0.94      0.94       114
```

```
[84]:                                  Model  AUC Score  Precision Score  \
      0             Logistic Regression (SGD)   0.555556         0.396226
      1                    decision_tree_grid   0.947917         0.928571
      2                     Naive_Bayes_Model   0.969577         0.902439
      3                            svm_linear   0.982143         0.948718
      4                              svm_poly   0.890542         1.000000
      5                                rf_cls   0.890542         0.973684
      6   Logistic_Regression with Full Model   0.996032         0.975610
      7        Log_Reg_Backward_Model_Selection   0.994709         0.952381
      8                    Hyper_Parameter_RF   0.994709         0.950000
      9                        meta_estimator   0.994709         0.950000
      10                             Adaboost   0.994709         0.948718

          Recall Score  Accuracy Score  Kappa Score  f1-Score
      0       0.438596        0.438596     0.084337  0.567568
      1       0.947368        0.947368     0.886905  0.928571
      2       0.921053        0.921053     0.829511  0.891566
      3       0.938596        0.938596     0.866062  0.913580
      4       0.885965        0.885965     0.738070  0.816901
      5       0.947368        0.947368     0.884615  0.925000
```

```
6         0.973684          0.973684       0.943170  0.963855
7         0.964912          0.964912       0.924603  0.952381
8         0.947368          0.947368       0.885772  0.926829
9         0.947368          0.947368       0.885772  0.926829
10        0.938596          0.938596       0.866062  0.913580
```



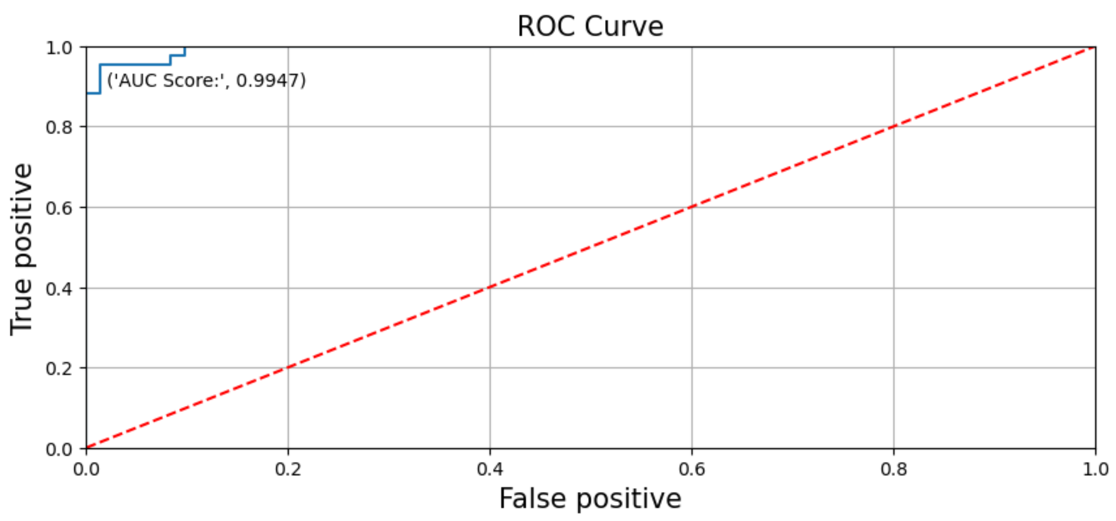
[86]:
```
from xgboost.sklearn import XGBClassifier
XGbm=XGBClassifier(random_state=1,learning_rate=0.01)
XGbm.fit(X_train,y_train)
y_pred=XGbm.predict(X_test)
plot_confusion_matrix(XGbm)
test_report = get_test_report(XGbm)
print(test_report)
plot_roc(XGbm)
update_score_card(model_name = 'XGBoost_Esimator')
```



```
              precision    recall  f1-score   support

           0       0.91      0.99      0.95        72
           1       0.97      0.83      0.90        42

    accuracy                           0.93       114
   macro avg       0.94      0.91      0.92       114
weighted avg       0.93      0.93      0.93       114
```

[86]:

```
                                Model  AUC Score  Precision Score  \
0            Logistic Regression (SGD)   0.555556         0.396226
1                   decision_tree_grid   0.947917         0.928571
2                    Naive_Bayes_Model   0.969577         0.902439
3                           svm_linear   0.982143         0.948718
4                             svm_poly   0.890542         1.000000
5                               rf_cls   0.890542         0.973684
6   Logistic_Regression with Full Model   0.996032         0.975610
7       Log_Reg_Backward_Model_Selection   0.994709         0.952381
8                   Hyper_Parameter_RF   0.994709         0.950000
9                       meta_estimator   0.994709         0.950000
10                            Adaboost   0.994709         0.948718
11                   XGBoost_Esimator   0.994709         0.972222

    Recall Score  Accuracy Score  Kappa Score  f1-Score
0       0.438596        0.438596     0.084337  0.567568
1       0.947368        0.947368     0.886905  0.928571
2       0.921053        0.921053     0.829511  0.891566
3       0.938596        0.938596     0.866062  0.913580
4       0.885965        0.885965     0.738070  0.816901
5       0.947368        0.947368     0.884615  0.925000
6       0.973684        0.973684     0.943170  0.963855
7       0.964912        0.964912     0.924603  0.952381
8       0.947368        0.947368     0.885772  0.926829
9       0.947368        0.947368     0.885772  0.926829
10      0.938596        0.938596     0.866062  0.913580
11      0.929825        0.929825     0.844581  0.897436
```



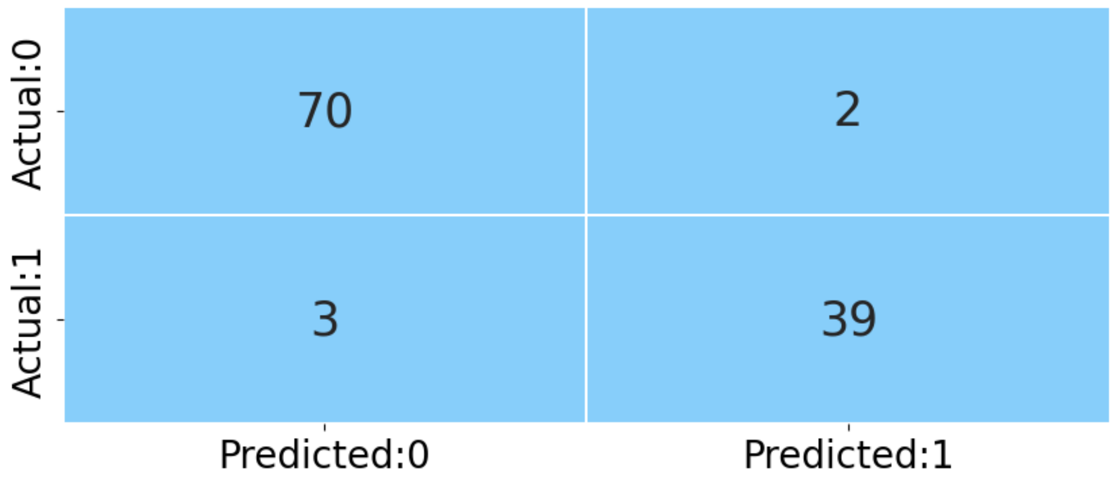
12 Random Undersampling randomly removes samples from the majority class to balance the dataset. This can be easily implemented using the RandomUnderSampler from imbalanced-learn.

[87]:
```
from imblearn.under_sampling import RandomUnderSampler

# Define the undersampling method
undersample = RandomUnderSampler(sampling_strategy='auto', random_state=42)

# Fit and transform the training data
X_train_res, y_train_res = undersample.fit_resample(X_train, y_train)

# Train the model
model_random_forest_undersample = RandomForestClassifier(random_state=42)
model_random_forest_undersample.fit(X_train_res, y_train_res)

# Predict the test set
y_pred =model_random_forest_undersample.predict(X_test)

# Evaluate the model
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.97        72
           1       0.95      0.93      0.94        42
```

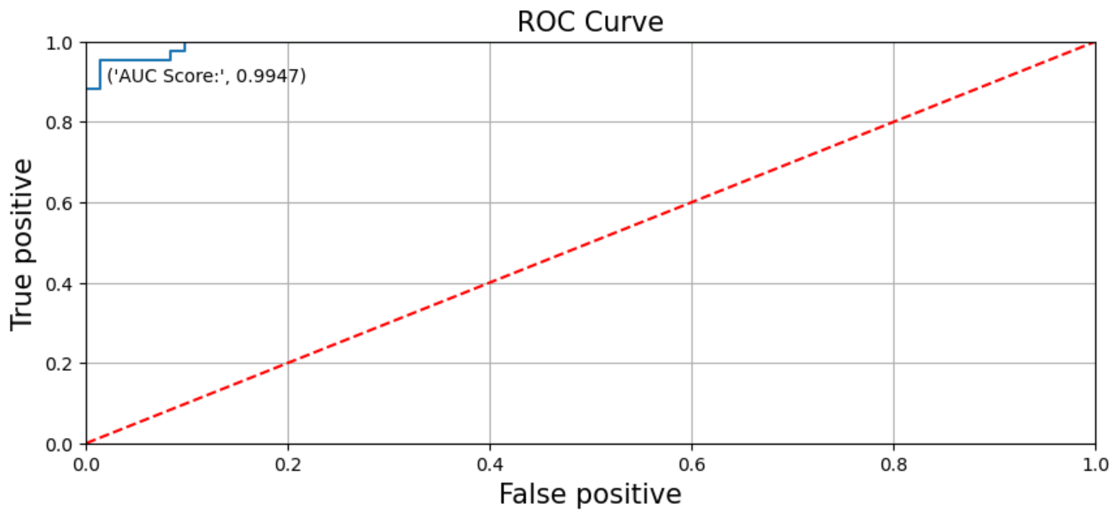
```
    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

[88]:
```
plot_confusion_matrix(model_random_forest_undersample)
plot_roc(model_random_forest_undersample)
update_score_card(model_name="Random_forest_undersample")
```



[88]:
```
                              Model  AUC Score  Precision Score  \
0         Logistic Regression (SGD)   0.555556         0.396226
1                 decision_tree_grid   0.947917         0.928571
2                  Naive_Bayes_Model   0.969577         0.902439
3                         svm_linear   0.982143         0.948718
4                           svm_poly   0.890542         1.000000
5                             rf_cls   0.890542         0.973684
6   Logistic_Regression with Full Model   0.996032         0.975610
7      Log_Reg_Backward_Model_Selection   0.994709         0.952381
8                    Hyper_Parameter_RF   0.994709         0.950000
9                       meta_estimator   0.994709         0.950000
10                            Adaboost   0.994709         0.948718
11                    XGBoost_Esimator   0.994709         0.972222
12           Random_forest_undersample   0.994709         0.951220

    Recall Score  Accuracy Score  Kappa Score  f1-Score
0       0.438596        0.438596     0.084337  0.567568
1       0.947368        0.947368     0.886905  0.928571
2       0.921053        0.921053     0.829511  0.891566
3       0.938596        0.938596     0.866062  0.913580
4       0.885965        0.885965     0.738070  0.816901
5       0.947368        0.947368     0.884615  0.925000
```

```
6        0.973684        0.973684      0.943170   0.963855
7        0.964912        0.964912      0.924603   0.952381
8        0.947368        0.947368      0.885772   0.926829
9        0.947368        0.947368      0.885772   0.926829
10       0.938596        0.938596      0.866062   0.913580
11       0.929825        0.929825      0.844581   0.897436
12       0.956140        0.956140      0.905284   0.939759
```



13 Feature Selection Using Random Forest Technique

[94]:
```
X = data.drop(['diagnosis'], axis = 1)
y = pd.DataFrame(data_dummy['diagnosis'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 1)

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
importances = rf_model.feature_importances_
importances
```

[94]:
```
array([0.03622661, 0.01036493, 0.07247401, 0.06825188, 0.00590276,
       0.01109463, 0.07436603, 0.08758439, 0.00247502, 0.00205805,
       0.01205829, 0.0038451 , 0.00878858, 0.02936742, 0.00254418,
       0.00513772, 0.00619549, 0.00321809, 0.00428113, 0.00630913,
       0.09071398, 0.01907397, 0.09181907, 0.13444054, 0.00999055,
       0.01959695, 0.04021379, 0.12764984, 0.00878272, 0.00517515])
```

[91]:
```
X.head()
```

```
[91]:    radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
      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

         compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
      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

         fractal_dimension_mean  radius_se  texture_se  perimeter_se  area_se  \
      0                 0.07871     1.0950      0.9053         8.589   153.40
      1                 0.05667     0.5435      0.7339         3.398    74.08
      2                 0.05999     0.7456      0.7869         4.585    94.03
      3                 0.09744     0.4956      1.1560         3.445    27.23
      4                 0.05883     0.7572      0.7813         5.438    94.44

         smoothness_se  compactness_se  concavity_se  concave points_se  \
      0       0.006399         0.04904       0.05373            0.01587
      1       0.005225         0.01308       0.01860            0.01340
      2       0.006150         0.04006       0.03832            0.02058
      3       0.009110         0.07458       0.05661            0.01867
      4       0.011490         0.02461       0.05688            0.01885

         symmetry_se  fractal_dimension_se  radius_worst  texture_worst  \
      0      0.03003              0.006193         25.38          17.33
      1      0.01389              0.003532         24.99          23.41
      2      0.02250              0.004571         23.57          25.53
      3      0.05963              0.009208         14.91          26.50
      4      0.01756              0.005115         22.54          16.67

         perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
      0           184.60      2019.0            0.1622             0.6656
      1           158.80      1956.0            0.1238             0.1866
      2           152.50      1709.0            0.1444             0.4245
      3            98.87       567.7            0.2098             0.8663
      4           152.20      1575.0            0.1374             0.2050

         concavity_worst  concave points_worst  symmetry_worst  \
      0           0.7119                0.2654          0.4601
      1           0.2416                0.1860          0.2750
      2           0.4504                0.2430          0.3613
      3           0.6869                0.2575          0.6638
```

```
4           0.4000                0.1625          0.2364

   fractal_dimension_worst
0                  0.11890
1                  0.08902
2                  0.08758
3                  0.17300
4                  0.07678
```

[100]:
```
feature_names = X.columns.tolist()
print(feature_names)
```

```
['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']
```

[102]:
```
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance':␣
 ↪importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance',␣
 ↪ascending=False)
feature_importance_df.head(10)
```

[102]:
```
                 Feature  Importance
23            area_worst    0.134441
27  concave points_worst    0.127650
22       perimeter_worst    0.091819
20          radius_worst    0.090714
7    concave points_mean    0.087584
6         concavity_mean    0.074366
2         perimeter_mean    0.072474
3              area_mean    0.068252
26       concavity_worst    0.040214
0            radius_mean    0.036227
```

[110]:
```
# Select top 'n' features or based on a threshold
selected_features = feature_importance_df[feature_importance_df['Importance']␣
 ↪>= 0.04]['Feature'].tolist()
selected_features =list(selected_features)
selected_features
```

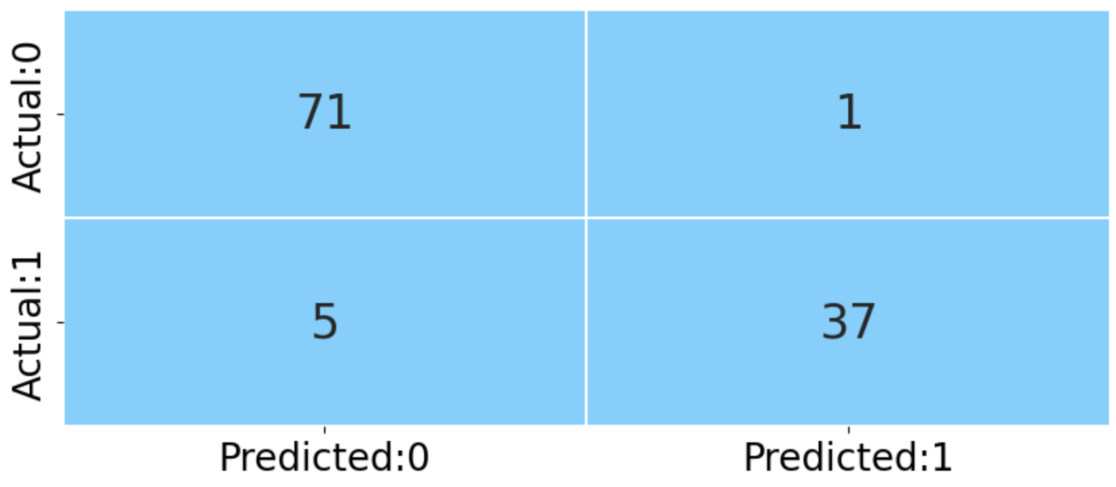
[110]:
```
['area_worst',
 'concave points_worst',
 'perimeter_worst',
 'radius_worst',
 'concave points_mean',
 'concavity_mean',
 'perimeter_mean',
 'area_mean',
 'concavity_worst']
```

[115]:
```
# Drop the 'diagnosis' column and the selected feature columns
#columns_to_drop = ['diagnosis'] + selected_features
#X = data.drop(columns_to_drop, axis=1)
X=data[selected_features]

# Assuming 'data_dummy' is another DataFrame containing the 'diagnosis' column
y = pd.DataFrame(data['diagnosis'])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=1)
```

[119]:
```
#intantiate the regressor
Random_Forest_Features_Selection = RandomForestClassifier(n_estimators=100,␣
 ↪random_state=10)

# fit the regressor with training dataset
Random_Forest_Features_Selection.fit(X_train, y_train)
# Predict the test set
y_pred =Random_Forest_Features_Selection.predict(X_test)
```

[120]:
```
test_report = get_test_report(Random_Forest_Features_Selection)
print(Random_Forest_Features_Selection)
plot_confusion_matrix(model_random_forest_undersample)
plot_roc(Random_Forest_Features_Selection)
update_score_card(model_name = 'Random_Forest_Features_Selection')
```

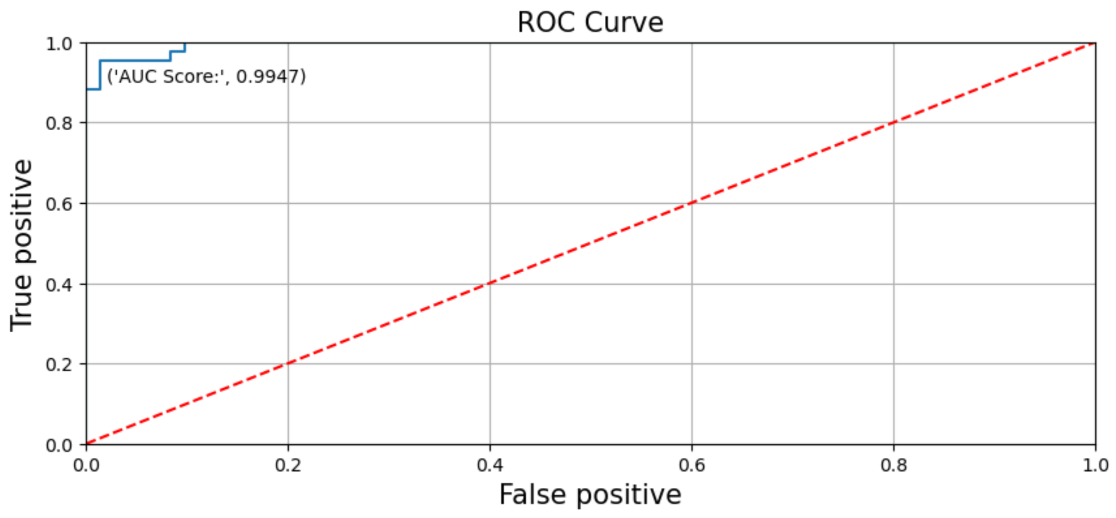
```
RandomForestClassifier(random_state=10)
```



[120]:

```
                                 Model  AUC Score  Precision Score  \
0             Logistic Regression (SGD)   0.555556         0.396226
1                    decision_tree_grid   0.947917         0.928571
2                     Naive_Bayes_Model   0.969577         0.902439
3                            svm_linear   0.982143         0.948718
4                              svm_poly   0.890542         1.000000
5                                rf_cls   0.890542         0.973684
6   Logistic_Regression with Full Model   0.996032         0.975610
7         Log_Reg_Backward_Model_Selection   0.994709         0.952381
8                     Hyper_Parameter_RF   0.994709         0.950000
9                        meta_estimator   0.994709         0.950000
10                             Adaboost   0.994709         0.948718
11                    XGBoost_Esimator   0.994709         0.972222
12             Random_forest_undersample   0.994709         0.951220
13       Random_Forest_Features_Selection   0.994709         0.951220
14       Random_Forest_Features_Selection   0.994709         0.973684

    Recall Score  Accuracy Score  Kappa Score  f1-Score
0       0.438596        0.438596     0.084337  0.567568
1       0.947368        0.947368     0.886905  0.928571
2       0.921053        0.921053     0.829511  0.891566
3       0.938596        0.938596     0.866062  0.913580
4       0.885965        0.885965     0.738070  0.816901
5       0.947368        0.947368     0.884615  0.925000
6       0.973684        0.973684     0.943170  0.963855
7       0.964912        0.964912     0.924603  0.952381
8       0.947368        0.947368     0.885772  0.926829
9       0.947368        0.947368     0.885772  0.926829
10      0.938596        0.938596     0.866062  0.913580
11      0.929825        0.929825     0.844581  0.897436
```

```
12       0.956140          0.956140      0.905284   0.939759
13       0.956140          0.956140      0.905284   0.939759
14       0.947368          0.947368      0.884615   0.925000
```



14 Cluster Analysis

[54]:
```
#The os.chdir function is used to change the current working directory to the␣
 ↪specified path.
import os
os.chdir(r"C:\DKS\Machine_Learning\Random_Forest")

##Load the Dataset
data= pd.read_csv('cancer.csv')
#The sample(15) method is used to display a random sample of 15 rows from the␣
 ↪loaded DataFrame
data.sample(15)
```

[54]:
```
             id diagnosis  radius_mean  texture_mean  perimeter_mean  \
345     898677         B       10.260         14.71           66.20
146     869691         M       11.800         16.58           78.99
213  881094802         M       17.420         25.56          114.50
78     8610862         M       20.180         23.97          143.70
74     8610175         B       12.310         16.52           79.19
222    8812844         B       10.180         17.53           65.12
27      852781         M       18.610         20.25          122.10
21     8510824         B        9.504         12.44           60.34
50      857343         B       11.760         21.60           74.72
460  911296201         M       17.080         27.15          111.20
```

```
89     861598          B      14.640         15.24           95.77
482    912519          B      13.470         14.06           87.32
64   85922302          M      12.680         23.84           82.69
542    921644          B      14.740         25.42           94.70
101    862722          B       6.981         13.43           43.79

     area_mean  smoothness_mean  compactness_mean  concavity_mean  \
345      321.6          0.09882           0.09159         0.03581
146      432.0          0.10910           0.17000         0.16590
213      948.0          0.10060           0.11460         0.16820
78      1245.0          0.12860           0.34540         0.37540
74       470.9          0.09172           0.06829         0.03372
222      313.1          0.10610           0.08502         0.01768
27      1094.0          0.09440           0.10660         0.14900
21       273.9          0.10240           0.06492         0.02956
50       427.9          0.08637           0.04966         0.01657
460      930.9          0.09898           0.11100         0.10070
89       651.9          0.11320           0.13390         0.09966
482      546.3          0.10710           0.11550         0.05786
64       499.0          0.11220           0.12620         0.11280
542      668.6          0.08275           0.07214         0.04105
101      143.5          0.11700           0.07568         0.00000

     concave points_mean  symmetry_mean  fractal_dimension_mean  radius_se  \
345              0.02037         0.1633                 0.07005     0.3380
146              0.07415         0.2678                 0.07371     0.3197
213              0.06597         0.1308                 0.05866     0.5296
78               0.16040         0.2906                 0.08142     0.9317
74               0.02272         0.1720                 0.05914     0.2505
222              0.01915         0.1910                 0.06908     0.2467
27               0.07731         0.1697                 0.05699     0.8529
21               0.02076         0.1815                 0.06905     0.2773
50               0.01115         0.1495                 0.05888     0.4062
460              0.06431         0.1793                 0.06281     0.9291
89               0.07064         0.2116                 0.06346     0.5115
482              0.05266         0.1779                 0.06639     0.1588
64               0.06873         0.1905                 0.06590     0.4255
542              0.03027         0.1840                 0.05680     0.3031
101              0.00000         0.1930                 0.07818     0.2241

     texture_se  perimeter_se  area_se  smoothness_se  compactness_se  \
345      2.5090         2.394   19.330       0.017360        0.046710
146      1.4260         2.281   24.720       0.005427        0.036330
213      1.6670         3.767   58.530       0.031130        0.085550
78       1.8850         8.649  116.400       0.010380        0.068350
74       1.0250         1.740   19.680       0.004854        0.018190
222      1.2170         1.641   15.050       0.007899        0.014000
```

```
27       1.8490          5.632   93.540       0.010750        0.027220
21       0.9768          1.909   15.700       0.009606        0.014320
50       1.2100          2.635   28.470       0.005857        0.009758
460      1.1520          6.051  115.200       0.008740        0.022190
89       0.7372          3.814   42.760       0.005508        0.044120
482      0.5733          1.102   12.840       0.004450        0.014520
64       1.1780          2.927   36.460       0.007781        0.026480
542      1.3850          2.177   27.410       0.004775        0.011720
101      1.5080          1.553    9.833       0.010190        0.010840

     concavity_se  concave points_se  symmetry_se  fractal_dimension_se  \
345      0.026110           0.012960      0.03675              0.006758
146      0.046490           0.018430      0.05628              0.004635
213      0.143800           0.039270      0.02175              0.012560
78       0.109100           0.025930      0.07895              0.005987
74       0.018260           0.007965      0.01386              0.002304
222      0.008534           0.007624      0.02637              0.003761
27       0.050810           0.019110      0.02293              0.004217
21       0.019850           0.014210      0.02027              0.002968
50       0.011680           0.007445      0.02406              0.001769
460      0.027210           0.014580      0.02045              0.004417
89       0.044360           0.016230      0.02427              0.004841
482      0.013340           0.008791      0.01698              0.002787
64       0.029730           0.012900      0.01635              0.003601
542      0.019470           0.012690      0.01870              0.002626
101      0.000000           0.000000      0.02659              0.004100

     radius_worst  texture_worst  perimeter_worst  area_worst  \
345         10.88          19.48            70.89       357.1
146         13.74          26.38            91.93       591.7
213         18.07          28.07           120.40      1021.0
78          23.37          31.72           170.30      1623.0
74          14.11          23.21            89.71       611.1
222         11.17          22.84            71.94       375.6
27          21.31          27.26           139.90      1403.0
21          10.23          15.66            65.13       314.9
50          12.98          25.72            82.98       516.5
460         22.96          34.49           152.10      1648.0
89          16.34          18.24           109.40       803.6
482         14.83          18.32            94.94       660.2
64          17.09          33.47           111.80       888.3
542         16.51          32.29           107.40       826.4
101          7.93          19.54            50.41       185.2

     smoothness_worst  compactness_worst  concavity_worst  \
345            0.1360            0.16360          0.07162
146            0.1385            0.40920          0.45040
```

```
213           0.1243            0.17930          0.28030
78            0.1639            0.61640          0.76810
74            0.1176            0.18430          0.17030
222           0.1406            0.14400          0.06572
27            0.1338            0.21170          0.34460
21            0.1324            0.11480          0.08867
50            0.1085            0.08615          0.05523
460           0.1600            0.24440          0.26390
89            0.1277            0.30890          0.26040
482           0.1393            0.24990          0.18480
64            0.1851            0.40610          0.40240
542           0.1060            0.13760          0.16110
101           0.1584            0.12020          0.00000

     concave points_worst  symmetry_worst  fractal_dimension_worst  \
345               0.04074          0.2434                  0.08488
146               0.18650          0.5774                  0.10300
213               0.10990          0.1603                  0.06818
78                0.25080          0.5440                  0.09964
74                0.08660          0.2618                  0.07609
222               0.05575          0.3055                  0.08797
27                0.14900          0.2341                  0.07421
21                0.06227          0.2450                  0.07773
50                0.03715          0.2433                  0.06563
460               0.15550          0.3010                  0.09060
89                0.13970          0.3151                  0.08473
482               0.13350          0.3227                  0.09326
64                0.17160          0.3383                  0.10310
542               0.10950          0.2722                  0.06956
101               0.00000          0.2932                  0.09382

     Unnamed: 32
345          NaN
146          NaN
213          NaN
78           NaN
74           NaN
222          NaN
27           NaN
21           NaN
50           NaN
460          NaN
89           NaN
482          NaN
64           NaN
542          NaN
101          NaN
```

[55]:
```
# Dropping the 'id' and 'Unnamed: 32' columns from the DataFrame
# The 'id' column is typically an identifier that is not useful for modeling
# 'Unnamed: 32' might be an empty or irrelevant column that can be safely␣
 ↪removed
data = data.drop(['id', 'Unnamed: 32'], axis=1)

# Display the first few rows of the cleaned dataset to verify the changes
data.head()
```

[55]:
```
  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  fractal_dimension_mean  radius_se  texture_se  perimeter_se  \
0         0.2419                 0.07871     1.0950      0.9053         8.589
1         0.1812                 0.05667     0.5435      0.7339         3.398
2         0.2069                 0.05999     0.7456      0.7869         4.585
3         0.2597                 0.09744     0.4956      1.1560         3.445
4         0.1809                 0.05883     0.7572      0.7813         5.438

   area_se  smoothness_se  compactness_se  concavity_se  concave points_se  \
0   153.40       0.006399         0.04904       0.05373            0.01587
1    74.08       0.005225         0.01308       0.01860            0.01340
2    94.03       0.006150         0.04006       0.03832            0.02058
3    27.23       0.009110         0.07458       0.05661            0.01867
4    94.44       0.011490         0.02461       0.05688            0.01885

   symmetry_se  fractal_dimension_se  radius_worst  texture_worst  \
0      0.03003              0.006193         25.38          17.33
1      0.01389              0.003532         24.99          23.41
2      0.02250              0.004571         23.57          25.53
3      0.05963              0.009208         14.91          26.50
4      0.01756              0.005115         22.54          16.67

   perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           184.60      2019.0            0.1622             0.6656
1           158.80      1956.0            0.1238             0.1866
```

```
2          152.50      1709.0             0.1444              0.4245
3           98.87       567.7             0.2098              0.8663
4          152.20      1575.0             0.1374              0.2050

   concavity_worst  concave points_worst  symmetry_worst  \
0           0.7119                0.2654          0.4601
1           0.2416                0.1860          0.2750
2           0.4504                0.2430          0.3613
3           0.6869                0.2575          0.6638
4           0.4000                0.1625          0.2364

   fractal_dimension_worst
0                  0.11890
1                  0.08902
2                  0.08758
3                  0.17300
4                  0.07678
```

[56]:
```
features=data.drop(["diagnosis"],axis=1)
features.head()
```

[56]:
```
   radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
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

   compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
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

   fractal_dimension_mean  radius_se  texture_se  perimeter_se  area_se  \
0                 0.07871     1.0950      0.9053         8.589   153.40
1                 0.05667     0.5435      0.7339         3.398    74.08
2                 0.05999     0.7456      0.7869         4.585    94.03
3                 0.09744     0.4956      1.1560         3.445    27.23
4                 0.05883     0.7572      0.7813         5.438    94.44

   smoothness_se  compactness_se  concavity_se  concave points_se  \
0       0.006399         0.04904       0.05373            0.01587
1       0.005225         0.01308       0.01860            0.01340
2       0.006150         0.04006       0.03832            0.02058
3       0.009110         0.07458       0.05661            0.01867
```

```
4       0.011490         0.02461      0.05688            0.01885

   symmetry_se  fractal_dimension_se  radius_worst  texture_worst  \
0      0.03003              0.006193         25.38          17.33
1      0.01389              0.003532         24.99          23.41
2      0.02250              0.004571         23.57          25.53
3      0.05963              0.009208         14.91          26.50
4      0.01756              0.005115         22.54          16.67

   perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           184.60      2019.0            0.1622             0.6656
1           158.80      1956.0            0.1238             0.1866
2           152.50      1709.0            0.1444             0.4245
3            98.87       567.7            0.2098             0.8663
4           152.20      1575.0            0.1374             0.2050

   concavity_worst  concave points_worst  symmetry_worst  \
0           0.7119                0.2654          0.4601
1           0.2416                0.1860          0.2750
2           0.4504                0.2430          0.3613
3           0.6869                0.2575          0.6638
4           0.4000                0.1625          0.2364

   fractal_dimension_worst
0                  0.11890
1                  0.08902
2                  0.08758
3                  0.17300
4                  0.07678
```

[57]:
```
scale=StandardScaler().fit(features)
features_s=scale.transform(features)
```

[58]:
```
features_scaled=pd.DataFrame(features_s,columns=data.columns[1:])
features_scaled.head()
```

[58]:
```
   radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0     1.097064     -2.073335        1.269934   0.984375         1.568466
1     1.829821     -0.353632        1.685955   1.908708        -0.826962
2     1.579888      0.456187        1.566503   1.558884         0.942210
3    -0.768909      0.253732       -0.592687  -0.764464         3.283553
4     1.750297     -1.151816        1.776573   1.826229         0.280372

   compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0          3.283515        2.652874             2.532475       2.217515
1         -0.487072       -0.023846             0.548144       0.001392
2          1.052926        1.363478             2.037231       0.939685
```

```
3         3.402909        1.915897            1.451707       2.867383
4         0.539340        1.371011            1.428493      -0.009560

   fractal_dimension_mean  radius_se  texture_se  perimeter_se   area_se  \
0                2.255747   2.489734   -0.565265      2.833031  2.487578
1               -0.868652   0.499255   -0.876244      0.263327  0.742402
2               -0.398008   1.228676   -0.780083      0.850928  1.181336
3                4.910919   0.326373   -0.110409      0.286593 -0.288378
4               -0.562450   1.270543   -0.790244      1.273189  1.190357

   smoothness_se  compactness_se  concavity_se  concave points_se  \
0      -0.214002        1.316862      0.724026           0.660820
1      -0.605351       -0.692926     -0.440780           0.260162
2      -0.297005        0.814974      0.213076           1.424827
3       0.689702        2.744280      0.819518           1.115007
4       1.483067       -0.048520      0.828471           1.144205

   symmetry_se  fractal_dimension_se  radius_worst  texture_worst  \
0     1.148757              0.907083      1.886690      -1.359293
1    -0.805450             -0.099444      1.805927      -0.369203
2     0.237036              0.293559      1.511870      -0.023974
3     4.732680              2.047511     -0.281464       0.133984
4    -0.361092              0.499328      1.298575      -1.466770

   perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0         2.303601    2.001237          1.307686           2.616665
1         1.535126    1.890489         -0.375612          -0.430444
2         1.347475    1.456285          0.527407           1.082932
3        -0.249939   -0.550021          3.394275           3.893397
4         1.338539    1.220724          0.220556          -0.313395

   concavity_worst  concave points_worst  symmetry_worst  \
0         2.109526              2.296076        2.750622
1        -0.146749              1.087084       -0.243890
2         0.854974              1.955000        1.152255
3         1.989588              2.175786        6.046041
4         0.613179              0.729259       -0.868353

   fractal_dimension_worst
0                 1.937015
1                 0.281190
2                 0.201391
3                 4.935010
4                -0.397100
```

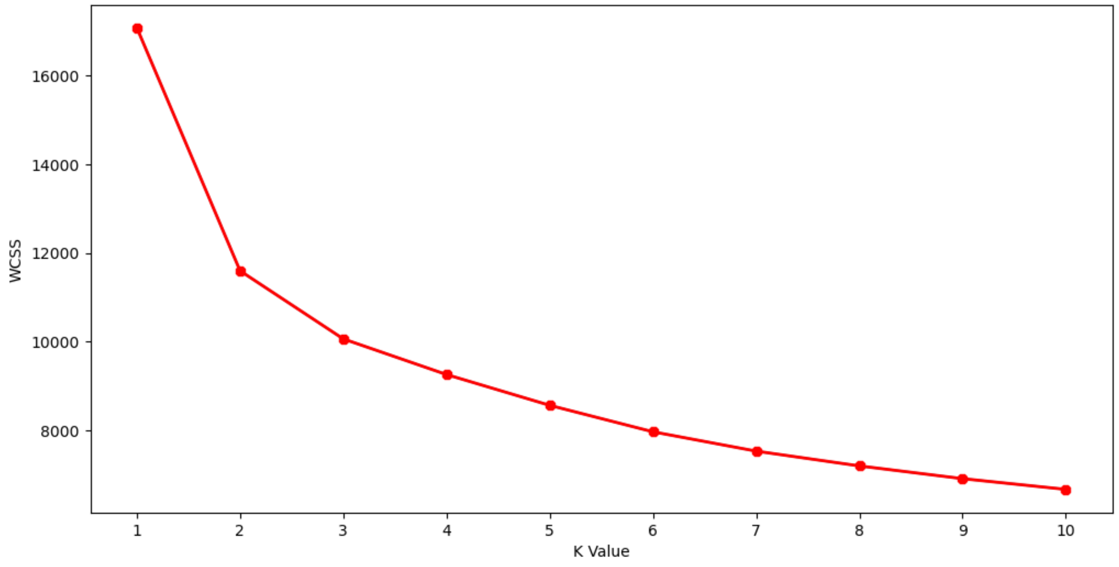
15 Build a Model with Multiple K

We constructed our models using the silhouette score method. Silhouette is a technique for interpreting and validating the consistency within clusters of data. We do not know the optimal number of clusters that would yield the most useful results. Therefore, we create clusters by varying K from 2 to 8 and subsequently determine the optimum number of clusters (K) with the assistance of the silhouette score.

[59]:
```
import warnings
warnings.filterwarnings("ignore")
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
n_clusters=[2,3,4,5,6,7,8]
for K in n_clusters:
    cluster=KMeans(n_clusters=K,random_state=10)
    predict=cluster.fit_predict(features_scaled)
    score=silhouette_score(features_scaled,predict,random_state=10)
    print("For n_clusters={}, silhoutte score is {}".format(K,score))
```

```
For n_clusters=2, silhoutte score is 0.3449740051034408
For n_clusters=3, silhoutte score is 0.3143840098608098
For n_clusters=4, silhoutte score is 0.27998963703382607
For n_clusters=5, silhoutte score is 0.15972213282998096
For n_clusters=6, silhoutte score is 0.16253401800989778
For n_clusters=7, silhoutte score is 0.1531205740823681
For n_clusters=8, silhoutte score is 0.157000597501773
```

[60]:
```
#Importing KMeans from sklearn

from sklearn.cluster import KMeans
#Now we calculate the Within Cluster Sum of Squared Errors (WSS) for different␣
 ↪values of k. Next, we
#choose the k for which WSS first starts to diminish. This value of K gives us␣
 ↪the best number of
#clusters to make from the raw data.
wcss=[]
for i in range(1,11):
    km=KMeans(n_clusters=i)
#n_clusters - The number of clusters to form as well as the number of centroids␣
 ↪to generate
    km.fit(features_scaled)
    wcss.append(km.inertia_)
#inertia_ -Sum of squared distances of samples to their closest cluster center ␣
 ↪
#The elbow curve
plt.figure(figsize=(12,6))
plt.plot(range(1,11),wcss)
plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")
plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS")
plt.show()
```



The optimal value for K is identified by the highest silhouette score. From the above output, it is evident that, for K = 2, the silhouette score is the highest. Consequently, we construct the clusters with K = 2."

[61]:
```
# building a K-Means model for K = 2
model = KMeans(n_clusters= 2, random_state= 10)

# fit the model
model.fit(features_scaled)
```

[61]:
```
KMeans(n_clusters=2, random_state=10)
```

[62]:
```
print(f"Length of features DataFrame: {len(features)}")
print(f"Length of model.labels_: {len(model.labels_)}")
```

```
Length of features DataFrame: 569
Length of model.labels_: 569
```

[63]:
```
features.head()
```

[63]:
```
   radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
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

   compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
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

   fractal_dimension_mean  radius_se  texture_se  perimeter_se  area_se  \
0                 0.07871     1.0950      0.9053         8.589   153.40
1                 0.05667     0.5435      0.7339         3.398    74.08
2                 0.05999     0.7456      0.7869         4.585    94.03
3                 0.09744     0.4956      1.1560         3.445    27.23
4                 0.05883     0.7572      0.7813         5.438    94.44

   smoothness_se  compactness_se  concavity_se  concave points_se  \
0       0.006399         0.04904       0.05373            0.01587
1       0.005225         0.01308       0.01860            0.01340
2       0.006150         0.04006       0.03832            0.02058
3       0.009110         0.07458       0.05661            0.01867
4       0.011490         0.02461       0.05688            0.01885

   symmetry_se  fractal_dimension_se  radius_worst  texture_worst  \
0      0.03003              0.006193         25.38          17.33
1      0.01389              0.003532         24.99          23.41
2      0.02250              0.004571         23.57          25.53
3      0.05963              0.009208         14.91          26.50
4      0.01756              0.005115         22.54          16.67

   perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           184.60      2019.0            0.1622             0.6656
1           158.80      1956.0            0.1238             0.1866
2           152.50      1709.0            0.1444             0.4245
3            98.87       567.7            0.2098             0.8663
4           152.20      1575.0            0.1374             0.2050

   concavity_worst  concave points_worst  symmetry_worst  \
0           0.7119                0.2654          0.4601
1           0.2416                0.1860          0.2750
2           0.4504                0.2430          0.3613
3           0.6869                0.2575          0.6638
4           0.4000                0.1625          0.2364

   fractal_dimension_worst
0                  0.11890
```

```
1                     0.08902
2                     0.08758
3                     0.17300
4                     0.07678
```

Now, let's explore these two clusters to gain insights about them.

16 Retrieve the Clusters

[64]:
```
data_output =features.copy()
# add a column 'Cluster' in the data giving cluster number corresponding to␣
↪each observation
data_output['Cluster'] = model.labels_
# Reset the index, starting from 1
data_output.index = range(1, len(data_output) + 1)

# head() to display top five rows
data_output.head()
```

[64]:
```
   radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
1        17.99         10.38          122.80     1001.0          0.11840
2        20.57         17.77          132.90     1326.0          0.08474
3        19.69         21.25          130.00     1203.0          0.10960
4        11.42         20.38           77.58      386.1          0.14250
5        20.29         14.34          135.10     1297.0          0.10030

   compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
1           0.27760          0.3001              0.14710         0.2419
2           0.07864          0.0869              0.07017         0.1812
3           0.15990          0.1974              0.12790         0.2069
4           0.28390          0.2414              0.10520         0.2597
5           0.13280          0.1980              0.10430         0.1809

   fractal_dimension_mean  radius_se  texture_se  perimeter_se  area_se  \
1                 0.07871     1.0950      0.9053         8.589   153.40
2                 0.05667     0.5435      0.7339         3.398    74.08
3                 0.05999     0.7456      0.7869         4.585    94.03
4                 0.09744     0.4956      1.1560         3.445    27.23
5                 0.05883     0.7572      0.7813         5.438    94.44

   smoothness_se  compactness_se  concavity_se  concave points_se  \
1       0.006399         0.04904       0.05373            0.01587
2       0.005225         0.01308       0.01860            0.01340
3       0.006150         0.04006       0.03832            0.02058
4       0.009110         0.07458       0.05661            0.01867
5       0.011490         0.02461       0.05688            0.01885
```

```
   symmetry_se  fractal_dimension_se  radius_worst  texture_worst  \
1      0.03003              0.006193         25.38          17.33   
2      0.01389              0.003532         24.99          23.41   
3      0.02250              0.004571         23.57          25.53   
4      0.05963              0.009208         14.91          26.50   
5      0.01756              0.005115         22.54          16.67   

   perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
1           184.60      2019.0            0.1622             0.6656   
2           158.80      1956.0            0.1238             0.1866   
3           152.50      1709.0            0.1444             0.4245   
4            98.87       567.7            0.2098             0.8663   
5           152.20      1575.0            0.1374             0.2050   

   concavity_worst  concave points_worst  symmetry_worst  \
1           0.7119                0.2654          0.4601   
2           0.2416                0.1860          0.2750   
3           0.4504                0.2430          0.3613   
4           0.6869                0.2575          0.6638   
5           0.4000                0.1625          0.2364   

   fractal_dimension_worst  Cluster  
1                  0.11890        1  
2                  0.08902        1  
3                  0.08758        1  
4                  0.17300        1  
5                  0.07678        1  
```

We have added a column named ‘cluster’ to the dataframe, indicating the cluster number for each observation.

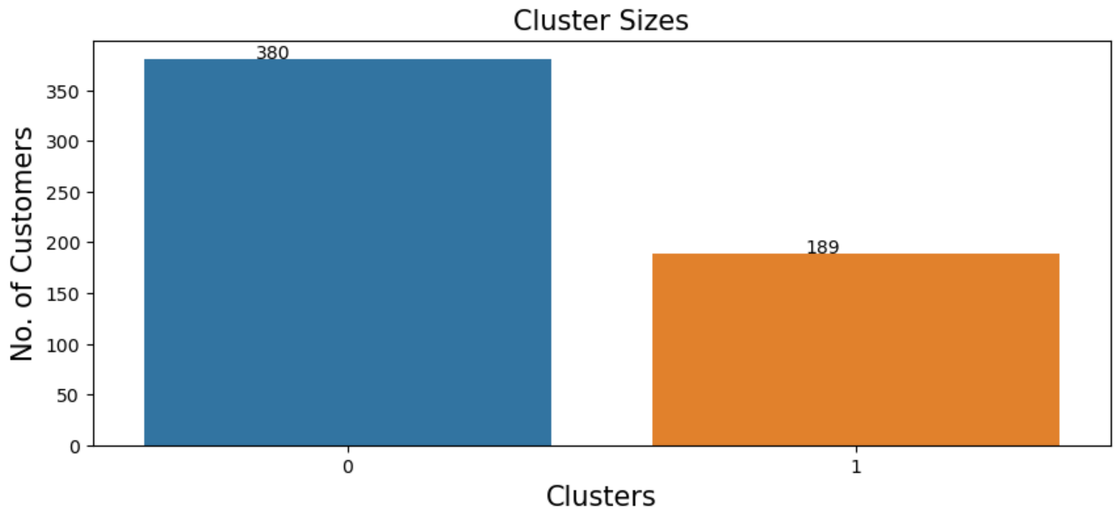
```
[65]: # 'return_counts = True' gives the number observation in each cluster
np.unique(model.labels_, return_counts=True)
```

```
[65]: (array([0, 1]), array([380, 189], dtype=int64))
```

17 Plot a barplot to visualize the cluster sizes

```
[68]: # use 'seaborn' library to plot a barplot for cluster size
sns.countplot(data= data_output, x = 'Cluster')

# set the axes and plot labels
# set the font size using 'fontsize'
plt.title('Cluster Sizes', fontsize = 15)
plt.xlabel('Clusters', fontsize = 15)
plt.ylabel('No. of Customers', fontsize = 15)
```

```
# add values in the graph
# 'x' and 'y' assigns the position to the text
# 's' represents the text on the plot
plt.text(x = -0.18, y =381, s = np.unique(model.labels_,␣
 ↪return_counts=True)[1][0])
plt.text(x = 0.9, y =190, s = np.unique(model.labels_,␣
 ↪return_counts=True)[1][1])

plt.show()
```



18 Cluster Centers

The cluster centers can give information about the variables belonging to the clusters

[73]:
```
# form a dataframe containing cluster centers
# 'cluster_centers_' returns the co-ordinates of a cluster center
centers = pd.DataFrame(model.cluster_centers_, columns=  data_output.columns[0:
 ↪30])
# head() to display top five rows
centers.head()
```

[73]:
```
   radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0    -0.484425     -0.239490       -0.500668  -0.479228        -0.303024
1     0.973976      0.481514        1.006635   0.963527         0.609254

   compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0         -0.507662       -0.566716            -0.579226      -0.303961
1          1.020696        1.139429             1.164582       0.611139
```

```
   fractal_dimension_mean  radius_se  texture_se  perimeter_se   area_se  \
0               -0.125451  -0.427039   -0.021258     -0.427876 -0.401430
1                0.252230   0.858596    0.042741      0.860279  0.807108

   smoothness_se  compactness_se  concavity_se  concave points_se  \
0      -0.008485       -0.345696     -0.316772          -0.386077
1       0.017061        0.695051      0.636895           0.776239

   symmetry_se  fractal_dimension_se  radius_worst  texture_worst  \
0    -0.069822             -0.206424     -0.517305      -0.251823
1     0.140382              0.415032      1.040084       0.506310

   perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0        -0.530180   -0.498937         -0.302546          -0.472916
1         1.065971    1.003154          0.608293           0.950837

   concavity_worst  concave points_worst  symmetry_worst  \
0        -0.519401             -0.570089       -0.297136
1         1.044298              1.146211        0.597416

   fractal_dimension_worst
0                -0.309597
1                 0.622469
```

Now, extract the variables in each of the clusters and attempt to assign a name to each cluster based on the variables

19 Clusters Analysis

6.1 Analysis of Cluster_1 Here, we analyze the first cluster by: Checking the size of the cluster. Sorting the variables belonging to the cluster. Computing the statistical summary for observations in the cluster.

[77]:
```
# sort the variables based on cluster centers
cluster_1 = sorted(zip(list(centers.iloc[0,:]), list(centers.columns)), reverse␣
 ↪= True)[:9]
```

[78]:
```
# size of a cluster_1
np.unique(model.labels_, return_counts=True)[1][0]
```

[78]:
```
380
```

[79]:
```
# retrieve the top 3 variables present in the cluster
cluster1_var = pd.DataFrame(cluster_1)[1]
cluster1_var
```

[79]:
```
0             smoothness_se
1                texture_se
2               symmetry_se
3     fractal_dimension_mean
4       fractal_dimension_se
5              texture_mean
6             texture_worst
7            symmetry_worst
8          smoothness_worst
Name: 1, dtype: object
```

Here, we conduct an analysis of the first cluster, initially examining its size, followed by sorting the variables that belong to the cluster. Subsequently, we compute a statistical summary for the observations within the cluster.

Upon inspection, the first cluster comprises 380 observations. The top three variables in this cluster, ranked by importance, are texture_se, symmetry_se, fractal_dimension_mean, fractal_dimension_se, texture_mean, texture_worst. This suggests that these factors play a significant role within the cluster and may warrant further investigation or attention in the context of the overall dataset.

[81]:
```
# get summary for observations in the cluster
# consider the number of orders and customer gender for cluster analysis
data_output[["texture_se","symmetry_se","fractal_dimension_mean","fractal_dimension_se","textu
 ↪"smoothness_worst"]][data_output.Cluster == 0].describe()
```

[81]:
```
       texture_se  symmetry_se  fractal_dimension_mean  fractal_dimension_se  \
count  380.000000   380.000000              380.000000            380.000000
mean     1.205137     0.019966                0.061913              0.003249
std      0.582977     0.006957                0.005938              0.002111
min      0.360200     0.007882                0.049960              0.000895
25%      0.791675     0.014985                0.057688              0.001986
50%      1.095000     0.018695                0.061075              0.002724
75%      1.478250     0.022925                0.065015              0.003757
max      4.885000     0.061460                0.095750              0.021930

       texture_mean  texture_worst  symmetry_worst  smoothness_worst
count    380.000000     380.000000      380.000000        380.000000
mean      18.260500      24.130816        0.271709          0.125467
std        4.054345       5.695397        0.044129          0.019890
min        9.710000      12.020000        0.156500          0.071170
25%       15.457500      19.837500        0.243375          0.110800
50%       17.780000      23.265000        0.269100          0.125600
75%       20.330000      27.822500        0.299175          0.138825
max       33.810000      41.780000        0.488200          0.200600
```

20 Analysis of Cluster_2

Here, we analyze the second cluster by: Checking the size of the cluster. Sorting the variables belonging to the cluster. Computing the statistical summary for observations in the cluster.

[82]:
```
# sort the variables based on cluster centers
cluster_2 = sorted(zip(list(centers.iloc[1,:]), list(centers.columns)), reverse␣
 ↪= True)[:9]

# size of a cluster_2
np.unique(model.labels_, return_counts=True)[1][1]

# retrieve the top 10 variables present in the cluster
cluster2_var = pd.DataFrame(cluster_2)[1]
cluster2_var
```

[82]:
```
0      concave points_mean
1     concave points_worst
2           concavity_mean
3          perimeter_worst
4          concavity_worst
5             radius_worst
6         compactness_mean
7           perimeter_mean
8               area_worst
Name: 1, dtype: object
```

[83]:
```
# get summary for observations in the cluster
# consider the number of orders and customer gender for cluster analysis
data_output[["texture_se","symmetry_se","fractal_dimension_mean","fractal_dimension_se","textu
 ↪"smoothness_worst"]][data_output.Cluster == 1].describe()
```

[83]:
```
       texture_se  symmetry_se  fractal_dimension_mean  fractal_dimension_se  \
count  189.000000   189.000000              189.000000            189.000000
mean     1.240411     0.021702                0.064577              0.004892
std      0.483156     0.010337                0.008646              0.003219
min      0.550300     0.009947                0.050240              0.001575
25%      0.920900     0.015350                0.057960              0.003224
50%      1.152000     0.018840                0.062810              0.004168
75%      1.466000     0.023830                0.069370              0.005617
max      3.568000     0.078950                0.097440              0.029840

       texture_mean  texture_worst  symmetry_worst  smoothness_worst
count    189.000000     189.000000      189.000000        189.000000
mean      21.358836      28.786402        0.327004          0.146245
std        4.038248       5.847089        0.074737          0.022083
min       10.380000      16.380000        0.160300          0.088220
25%       18.820000      25.090000        0.281200          0.132200
```

```
50%      21.240000     28.140000       0.313800          0.144600
75%      23.750000     32.070000       0.361300          0.157400
max      39.280000     49.540000       0.663800          0.222600
```

[84]:
```
# get summary for observations in the cluster
# consider the number of orders and customer gender for cluster analysis
data_output[["concave␣
↪points_worst","concavity_mean","perimeter_worst","concavity_worst","radius_worst","compactn
↪Cluster == 1].describe()
```

[84]:
```
       concave points_worst  concavity_mean  perimeter_worst  concavity_worst  \
count            189.000000      189.000000       189.000000       189.000000
mean               0.189883        0.179555       143.049048         0.489863
std                0.040901        0.070475        31.590984         0.184672
min                0.091810        0.084220        65.500000         0.196000
25%                0.161300        0.126700       122.100000         0.359700
50%                0.184800        0.165500       142.200000         0.460900
75%                0.213400        0.213300       161.100000         0.591100
max                0.291000        0.426800       251.200000         1.252000

       radius_worst  compactness_mean  perimeter_mean   area_worst
count    189.000000        189.000000      189.000000   189.000000
mean      21.291746          0.158199      116.407725  1451.233862
std        4.672595          0.049057       23.416863   636.079636
min       10.060000          0.078640       58.790000   297.100000
25%       17.790000          0.123100      101.700000   975.200000
50%       21.310000          0.151100      117.300000  1403.000000
75%       24.220000          0.183800      130.700000  1750.000000
max       36.040000          0.345400      188.500000  4254.000000
```

[85]:
```
# get summary for observations in the cluster
# consider the number of orders and customer gender for cluster analysis
data_output[["concave␣
↪points_worst","concavity_mean","perimeter_worst","concavity_worst","radius_worst","compactn
↪Cluster == 0].describe()
```

[85]:
```
       concave points_worst  concavity_mean  perimeter_worst  concavity_worst  \
count            380.000000      380.000000       380.000000       380.000000
mean               0.077166        0.043661        89.461474         0.163924
std                0.037607        0.030174        15.517725         0.113715
min                0.000000        0.000000        50.410000         0.000000
25%                0.053635        0.021562        79.657500         0.079737
50%                0.078715        0.038045        88.110000         0.145750
75%                0.100325        0.061542        99.040000         0.230400
max                0.225800        0.146300       139.200000         0.772700

       radius_worst  compactness_mean  perimeter_mean   area_worst
```

```
count    380.000000        380.000000        380.0000   380.000000
mean      13.771129          0.077554         79.8140   596.759474
std        2.311456          0.028641         12.9192   204.856376
min        7.930000          0.019380         43.7900   185.200000
25%       12.355000          0.056355         71.7825   465.525000
50%       13.585000          0.073760         79.0450   563.350000
75%       15.110000          0.095820         87.8875   702.825000
max       21.310000          0.220400        120.9000  1410.000000
```

21 It can be observed that in the second cluster, most data points exhibit higher mean values for features such as "concave points_worst," "concavity_mean," "perimeter_worst," "concavity_worst," "radius_worst," "compactness_mean," "perimeter_mean," and "area_worst" compared to the first cluster. Higher values in these features are often associated with malignant cancer. Therefore, we may categorize the second cluster as the 'malignant group' and the first cluster as the 'benign group,' suggesting significant differences in health characteristics between the two clusters.

These findings highlight the importance of the identified features in differentiating between benign and malignant cases. For instance, features related to the worst case scenarios of concavity and perimeter indicate the severity of the malignancy, as higher values in these features typically correlate with more aggressive cancer forms. Additionally, the radius and area measurements, both mean and worst-case, are critical indicators of tumor size and spread, further supporting the malignancy classification.

The clear separation of clusters based on these significant features can aid in early detection and more accurate diagnosis, potentially leading to better treatment outcomes. The ability to distinguish between benign and malignant cases through clustering can also enhance the decision-making process for healthcare providers, enabling them to prioritize patients who may require more immediate and intensive care.

By leveraging these insights, healthcare professionals can develop targeted intervention strategies and improve patient management protocols. Furthermore, this clustering approach can be integrated into automated diagnostic systems, offering a robust tool for real-time analysis and classification of breast cancer cases.

[87]:
```python
from sklearn.metrics import accuracy_score
import pandas as pd

data["diagnosis"]=data.diagnosis.replace({"M":1,"B":0})

# Now you can calculate accuracy
accuracy = accuracy_score(data_output['Cluster'], data["diagnosis"])
```

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

```
Accuracy: 0.9103690685413005
```

[88]:
```python
data_output.head()
```

[88]:
```
   radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
1        17.99         10.38          122.80     1001.0          0.11840
2        20.57         17.77          132.90     1326.0          0.08474
3        19.69         21.25          130.00     1203.0          0.10960
4        11.42         20.38           77.58      386.1          0.14250
5        20.29         14.34          135.10     1297.0          0.10030

   compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
1           0.27760          0.3001              0.14710         0.2419
2           0.07864          0.0869              0.07017         0.1812
3           0.15990          0.1974              0.12790         0.2069
4           0.28390          0.2414              0.10520         0.2597
5           0.13280          0.1980              0.10430         0.1809

   fractal_dimension_mean  radius_se  texture_se  perimeter_se  area_se  \
1                 0.07871     1.0950      0.9053         8.589   153.40
2                 0.05667     0.5435      0.7339         3.398    74.08
3                 0.05999     0.7456      0.7869         4.585    94.03
4                 0.09744     0.4956      1.1560         3.445    27.23
5                 0.05883     0.7572      0.7813         5.438    94.44

   smoothness_se  compactness_se  concavity_se  concave points_se  \
1       0.006399         0.04904       0.05373            0.01587
2       0.005225         0.01308       0.01860            0.01340
3       0.006150         0.04006       0.03832            0.02058
4       0.009110         0.07458       0.05661            0.01867
5       0.011490         0.02461       0.05688            0.01885

   symmetry_se  fractal_dimension_se  radius_worst  texture_worst  \
1      0.03003              0.006193         25.38          17.33
2      0.01389              0.003532         24.99          23.41
3      0.02250              0.004571         23.57          25.53
4      0.05963              0.009208         14.91          26.50
5      0.01756              0.005115         22.54          16.67

   perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
1           184.60      2019.0            0.1622             0.6656
2           158.80      1956.0            0.1238             0.1866
3           152.50      1709.0            0.1444             0.4245
4            98.87       567.7            0.2098             0.8663
5           152.20      1575.0            0.1374             0.2050
```

```
   concavity_worst  concave points_worst  symmetry_worst  \
1           0.7119                0.2654          0.4601   
2           0.2416                0.1860          0.2750   
3           0.4504                0.2430          0.3613   
4           0.6869                0.2575          0.6638   
5           0.4000                0.1625          0.2364   

   fractal_dimension_worst  Cluster  
1                  0.11890        1  
2                  0.08902        1  
3                  0.08758        1  
4                  0.17300        1  
5                  0.07678        1  
```

22 In this data frame, '1' represents malignant cancer, and '0' represents benign cancer. These labels were assigned through cluster analysis. However, we have the actual labels available, allowing us to compare them with the cluster-assigned labels and calculate the accuracy score.

The availability of actual labels provides an opportunity to evaluate the performance of our clustering algorithm. By comparing the cluster-assigned labels with the actual labels, we can determine how accurately our model is classifying the data points. This comparison can be quantified using an accuracy score, which measures the proportion of correctly classified instances out of the total instances.

Calculating the accuracy score is essential for validating the effectiveness of the clustering approach. It helps identify any discrepancies between the predicted and actual classifications, highlighting areas for potential improvement. A high accuracy score would indicate that the clustering algorithm is effectively distinguishing between malignant and benign cases, while a lower score might suggest the need for further refinement of the model or feature selection process.

Additionally, analyzing the misclassified instances can provide insights into the limitations of the clustering approach. Understanding why certain data points were incorrectly labeled can reveal important characteristics that the current model may be overlooking. This analysis can guide the development of more sophisticated models or the incorporation of additional features to improve classification accuracy.

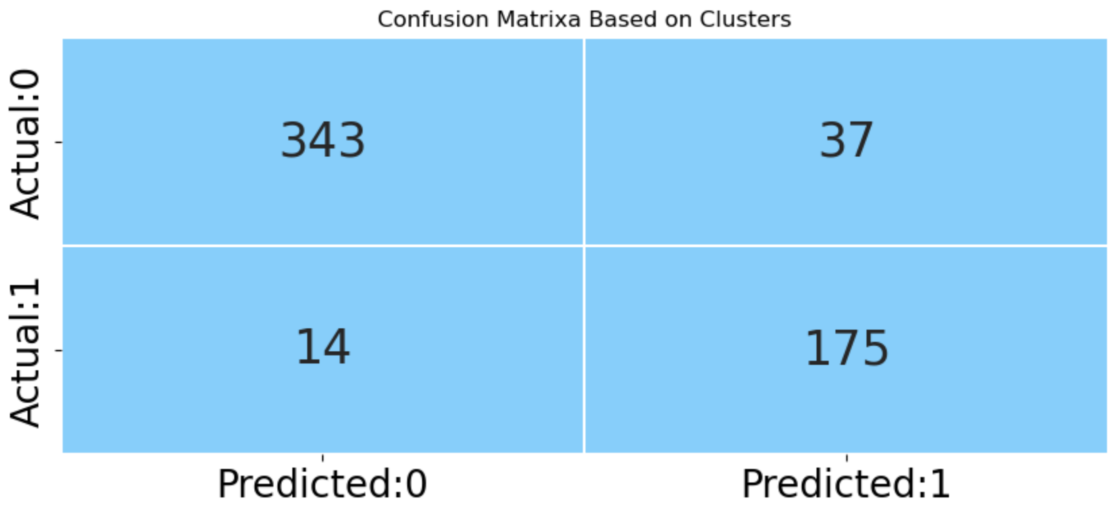
Moreover, assessing the accuracy of cluster-assigned labels against actual labels can help in fine-tuning the clustering algorithm parameters, such as the number of clusters or the choice of distance metrics. This iterative process of evaluation and adjustment is crucial for achieving optimal performance in unsupervised learning tasks.

Overall, the comparison between cluster-assigned and actual labels not only validates the current model but also offers a pathway for continuous improvement, ultimately enhancing the reliability of cancer classification and supporting better clinical decision-making.

```
[89]: from sklearn.metrics import roc_auc_score
      accuracy =roc_auc_score(data_output['Cluster'], data["diagnosis"])
      print("roc_auc_score:", accuracy)
```

```
roc_auc_score: 0.9142787524366472
```

```
[90]: from sklearn.metrics import confusion_matrix
      confusion_matrix =confusion_matrix(data_output['Cluster'], data["diagnosis"])
      cm = confusion_matrix
      conf_matrix = pd.DataFrame(data = cm,columns = ['Predicted:0','Predicted:1'],␣
       ↪index = ['Actual:0','Actual:1'])
      sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap =␣
       ↪ListedColormap(['lightskyblue']),cbar = False, linewidths = 0.1, annot_kws =␣
       ↪{'size':25})
      plt.xticks(fontsize = 20)
      plt.yticks(fontsize = 20)
      plt.title("Confusion Matrixa Based on Clusters")
      plt.show()
```



23 The accuracy score of 91.04% suggests that the cluster labeling method is correct in approximately 91 out of 100 instances, indicating a strong performance. This high accuracy instills confidence in the clustering algorithm's ability to differentiate between malignant and benign cases based on the identified features. The clusters formed are well-separated and distinct, capturing meaningful variations in cancer characteristics. Despite the high accuracy, it's important to acknowledge that no clustering algorithm is perfect, and there may still be instances of misclassification or overlap between clusters.¶.

Examining misclassified instances can provide insights into nuances not fully captured by current features. Other metrics like precision, recall, and F1-score can offer a more nuanced evaluation, especially in imbalanced datasets. Continued validation, feedback from domain experts, and feature refinement can enhance accuracy and effectiveness over time. The robustness of the clustering algorithm is crucial in medical applications, impacting patient outcomes and treatment strategies.

Monitoring and refining clustering results contribute to improved cancer diagnosis and patient care.

24 Conclusion: We applied 15 different machine learning algorithms to the cancer dataset, including logistic regression, SGD classifier, random forest with hyperparameter tuning, XGBoost, Adaboost, meta-estimator bagging technique, SVM classifier, Naive Bayes, and others. These models aimed to predict whether a person has malignant or benign cancer.

Among all the models, the logistic regression model with backward model selection stood out as the top performer. It achieved an impressive accuracy score of 97%, with all performance metrics surpassing 94%. This indicates the model's high precision, recall, F1-score, and AUC score, showcasing its robustness in correctly classifying cancer cases.

The success of the logistic regression model with backward model selection highlights the importance of feature selection and optimization in enhancing predictive accuracy. By identifying and incorporating the most relevant features, the model can effectively differentiate between malignant and benign cases, contributing significantly to accurate cancer diagnosis.

Furthermore, the model's high accuracy score of 97% signifies its potential for practical deployment in real-world scenarios. Its ability to consistently achieve high performance across various evaluation metrics makes it a reliable choice for cancer prediction tasks.

It's crucial to note that while the logistic regression model with backward model selection performed exceptionally well in this study, ongoing monitoring, validation, and further experimentation may lead to continued improvements and refinement of the predictive model.

Overall, the success of the logistic regression model underscores the value of advanced machine learning techniques and optimization strategies in the field of medical diagnostics, particularly in cancer diagnosis, where accuracy and reliability are paramount.

```
[ ]:
```