breast cancer dataset unit2_python_assignment

May 5, 2025

[ ]:

[ ]:

[2]:
```
#basic exploratory data analysis(EDA) assignment:
#Breast cancer dataset

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#loading dataset
from sklearn.datasets import load_breast_cancer
breast_cancer = load_breast_cancer()
df_cancer = pd.DataFrame( data=breast_cancer.data, columns = breast_cancer.
 ↪feature_names,)
# add the target variables
df_cancer['target'] = breast_cancer.target

#display first 5 rows

df_cancer.head()

#to see the features(columns)
print(df_cancer.columns.tolist())

#shape of the dataset(no of rows and columns)

print(" shape :", df_cancer.shape)

#find out the data types

df_cancer.info()

#descriptive statistics
df_cancer.describe()
```

```
#to find mean , median , std of mean radius

print(df_cancer['mean radius'].mean())
print(df_cancer['mean radius'].median())
print(df_cancer['mean radius'].std())

# missing values check

print(df_cancer.isnull().sum())
#if df_cancer.isnull().sum().sum() == 0:
   # print('there are no missing values')
#else:
   # print('missing values present')

#data cleaning
#handling missing values if any
#check target variables
print(df_cancer['target'].value_counts())
# to check duplicate values
df_cancer.duplicated().sum()


#descriptive statistics - mean meadian std for mean area mean compactness
for i in ['mean area','mean compactness']:
    print(i)
    print(df_cancer[i].mean())
    print(df_cancer[i].median())
    print(df_cancer[i].std())

#check extra spaces and also to see what unique values are present in a column
df_cancer.columns = df_cancer.columns.str.strip()
print(df_cancer['target'].unique())

#detecting OUtliers manually and treatment of outliers
q1 = df_cancer.quantile(0.25)
q3 = df_cancer.quantile(0.75)
iqr = q3 - q1

outliers = ((df_cancer < (q1 - 1.5*iqr))| (df_cancer > (q3 + 1.5*iqr)))
print(outliers.sum())
#using boxplot to show the outliers of mean area column just for learning␣
 ↪purpose(for one column)
plt.figure(figsize = (12 ,10))
sns.boxplot(x=df_cancer['mean area'] , color= 'lightgreen')
plt.title('boxplot')
plt.ylabel('mean area')
plt.xlabel('value')
```

```
plt.show()

#box plot of mean area grouped by target variables(distribution of␣
 ↪categories)(for two columns comparison purpose)
sns.boxplot(data = df_cancer, x= 'target', y = 'mean area')
plt.title('mean area vs target')
plt.xlabel('target')
plt.ylabel('mean area')
plt.show
#target 0 (malignant tumors) mean area is significantly higher than benign␣
 ↪tumors. the spread
#iqr is wider means more variability in tumor size(very diff sizes of tumors).
 ↪There are more outliers are on the high end with  malignant
#tumors having very large areas where as benign have few outliers with not so␣
 ↪extreme areas

#(histogram of mean radius to get the shape of the distribution curve so that
#we can use the proper method for statistical analysis or machine learning␣
 ↪methods)
plt.figure(figsize = (8,5))
plt.hist(df_cancer['mean radius'],bins = 30 , edgecolor = 'black' )
plt.title('histogram of mean radius')
plt.xlabel('mean')
plt.ylabel('frequency')
plt.show()
#so we can see that the histo shows right skewed distribution thatswhy we need␣
 ↪to treat it using LOG TRANSFORMATION OR SQUARE ROOT TRANS OR BOX-COX TRANS)
df_cancer['mean radius log']= np.log(df_cancer['mean radius'])
#now showing the updated histogram
plt.figure(figsize = (8,5))
plt.hist(df_cancer['mean radius log'], bins=30,edgecolor= 'green')
plt.title('updated histo')
plt.xlabel('log mean radius')
plt.ylabel('frequency')
plt.show()
#boxplot for mean texture
plt.figure(figsize=(6,4))
sns.boxplot(x = df_cancer['mean texture'],color = 'skyblue')
plt.title('boxplt for mean texture')
plt.xlabel('values')
plt.ylabel('mean texture')
plt.show()

#group by target
group = df_cancer.groupby('target')['mean perimeter'].mean()
print(group)
```

```
#bar chart of benign vs malignant
sns.countplot(x='target',data = df_cancer)
plt.xticks([0,1],['malignant','bengin'])
plt.title('counts')
plt.show()
#pairplot
sns.pairplot( df_cancer[['mean radius','mean texture','mean perimeter','mean␣
 ↪area','target']],hue = 'target')
plt.show()
#correlation heatmap
plt.figure(figsize=(12,10))
sns.heatmap(df_cancer.corr(),cmap ='coolwarm', annot = True, linewidths = 0.5,␣
 ↪annot_kws = {'size': 8} )
plt.title('corr map')
plt.show()

#feature engineering
# adding new features: area_to_perimeter_ratio feature
df_cancer['area_to_perimeter_ratio'] = df_cancer['mean area']/df_cancer['mean␣
 ↪perimeter']
#compare by target variables

sns.boxplot(x = 'target' , y = 'area_to_perimeter_ratio', data = df_cancer)
plt.title('area perimeter ratio by target')
plt.show()

#feature selection
#create corr matrix
corr_matrix = df_cancer.drop('target' , axis = 1).corr().abs()
#select the upper triangle of corr matrix
upper = corr_matrix.where(np.triu (np.ones(corr_matrix.shape) , k =1).
 ↪astype(bool))
#finding the features with corr greater than 0.95
todrop = [ i for i in upper.columns if any(upper[i]> 0.95)]
df_new = df_cancer.drop( columns = todrop)
print(df_new)
print(todrop)
print(df_new.head())
print(df_new.shape)

#next is FEATURE SCALING - ML models perform better when data is scaled

from sklearn.preprocessing import StandardScaler
#separate features and target
x = df_new.drop('target' , axis = 1) #it contains all features excluding target␣
 ↪variables
```

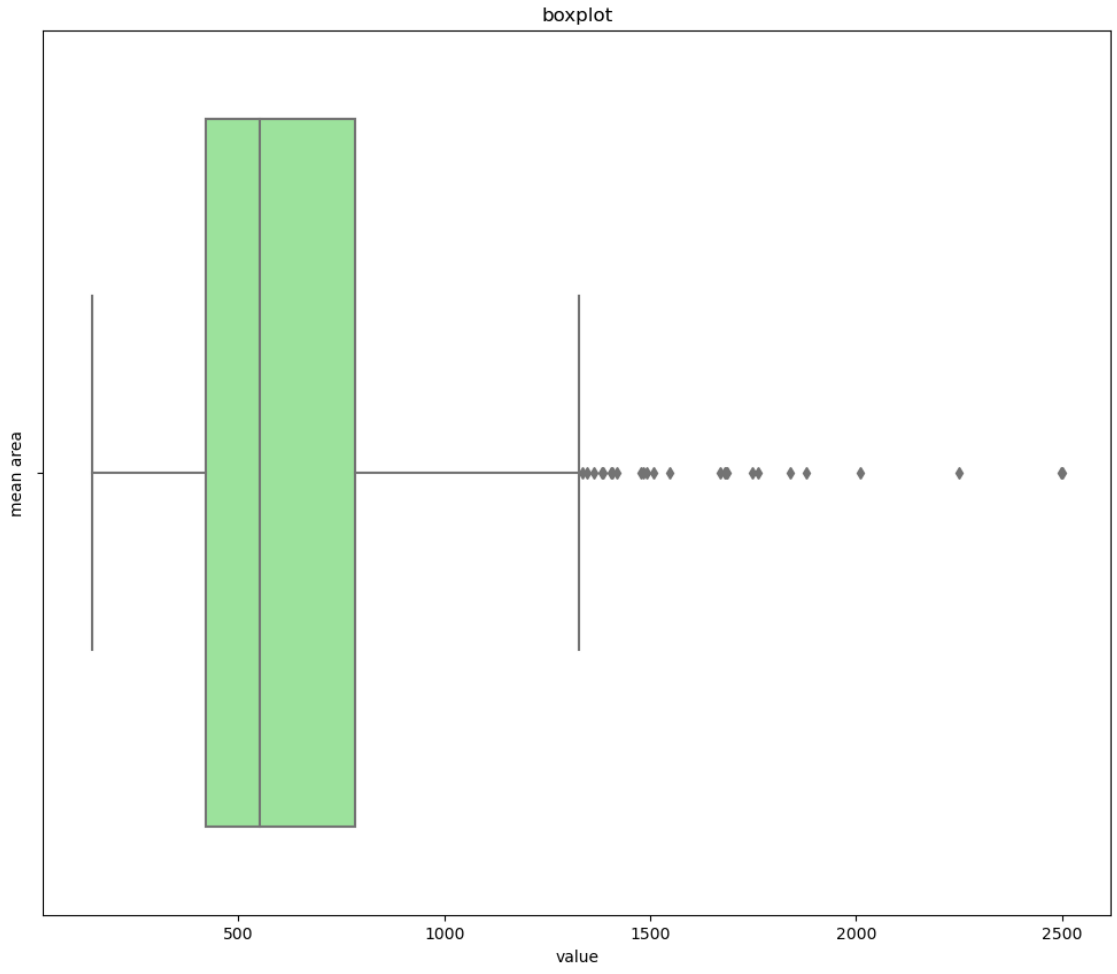
```
y = df_new['target'] # contains target variables

# standardize the features
scaler =  StandardScaler()
xscale = scaler.fit_transform(x)

#train - test split - split your data into training and test sets
from sklearn.model_selection import train_test_split
x_train , x_test , y_train, y_test = train_test_split(xscale , y , test_size =␣
 ↪0.2, random_state = 42)

#model building (logistic regression model to classify the tumors)
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score␣
 ↪,classification_report,confusion_matrix

#train the model
model = LogisticRegression()
model.fit(x_train , y_train)

#predict(x_test contains the input features radius ,texture,area etc these␣
 ↪value are needed for prediction)
#(y_test actual values 0 and 1 it is the  actual output that u will comparr␣
 ↪with x_test to see accuracy)

y_pred = model.predict(x_test)

print('accuracy', accuracy_score(y_pred,y_test))
print('classification report', classification_report(y_pred,y_test))
print('confusion matrix', confusion_matrix(y_pred,y_test))
```

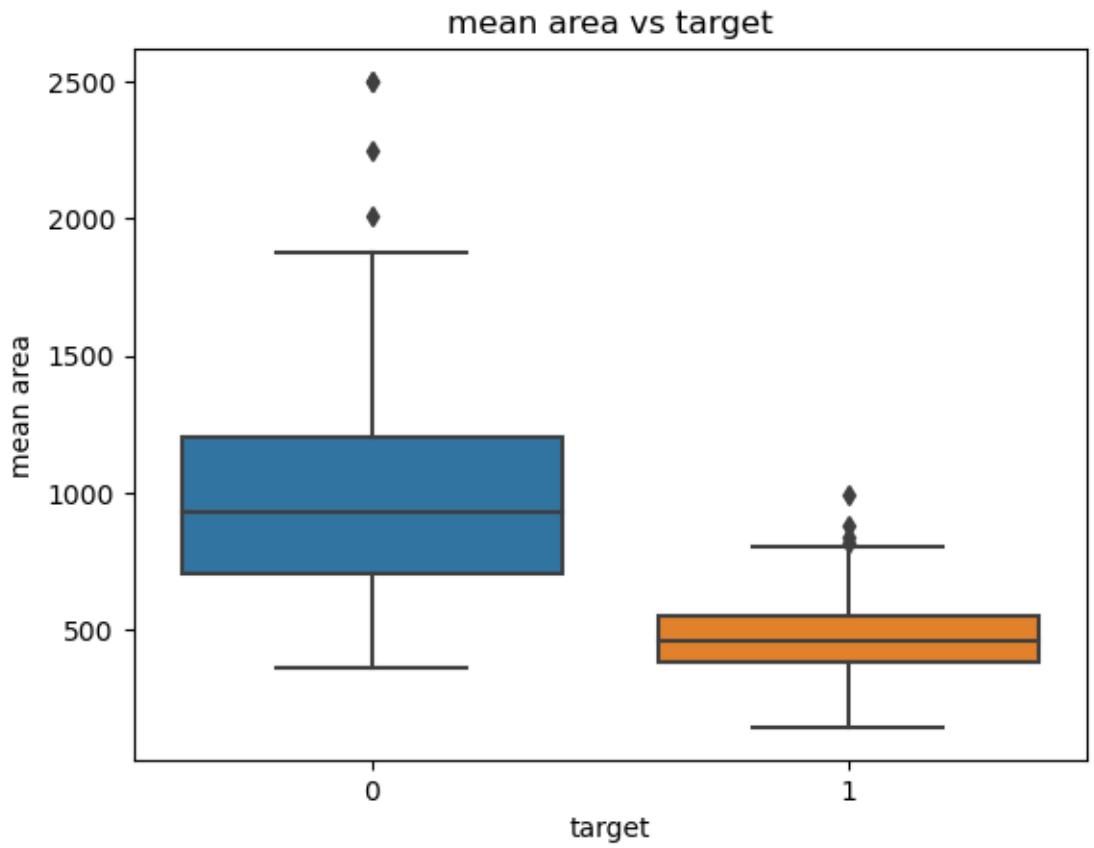
```
['mean radius', 'mean texture', 'mean perimeter', 'mean area', 'mean
smoothness', 'mean compactness', 'mean concavity', 'mean concave points', 'mean
symmetry', 'mean fractal dimension', 'radius error', 'texture error', 'perimeter
error', 'area error', 'smoothness error', 'compactness error', 'concavity
error', 'concave points error', 'symmetry error', 'fractal dimension error',
'worst radius', 'worst texture', 'worst perimeter', 'worst area', 'worst
smoothness', 'worst compactness', 'worst concavity', 'worst concave points',
'worst symmetry', 'worst fractal dimension', 'target']
 shape : (569, 31)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
```

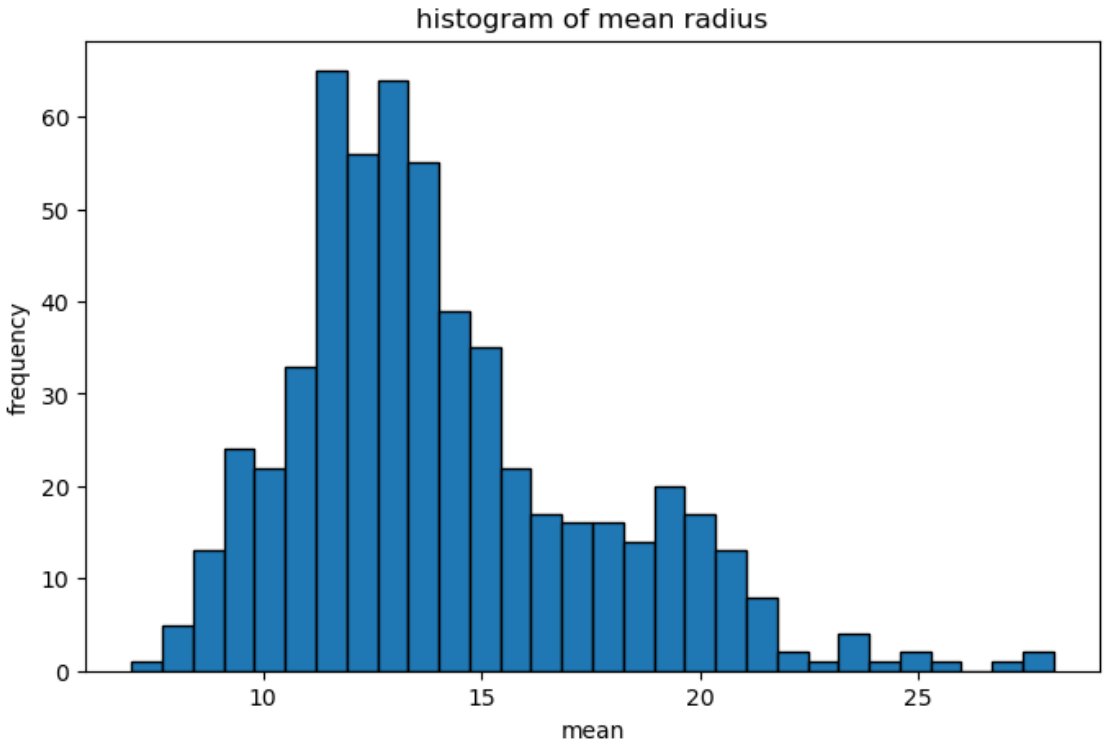
```
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
14.127291739894552
13.37
3.524048826212078
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
```

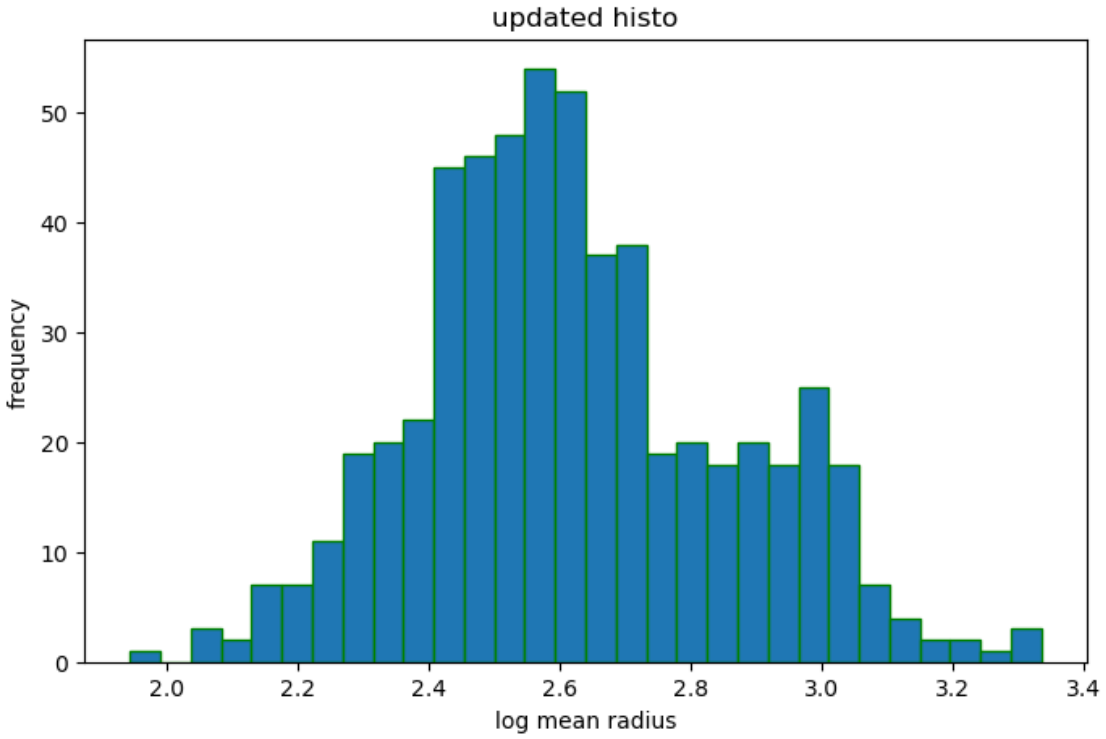
```
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
target
1    357
0    212
Name: count, dtype: int64
mean area
654.8891036906855
551.1
351.9141291816527
mean compactness
0.10434098418277679
0.09263
0.0528127579325122
[0 1]
mean radius                14
mean texture                7
mean perimeter             13
mean area                  25
mean smoothness             6
mean compactness           16
mean concavity             18
mean concave points        10
mean symmetry              15
mean fractal dimension     15
radius error               38
texture error              20
perimeter error            38
area error                 65
smoothness error           30
compactness error          28
concavity error            22
```

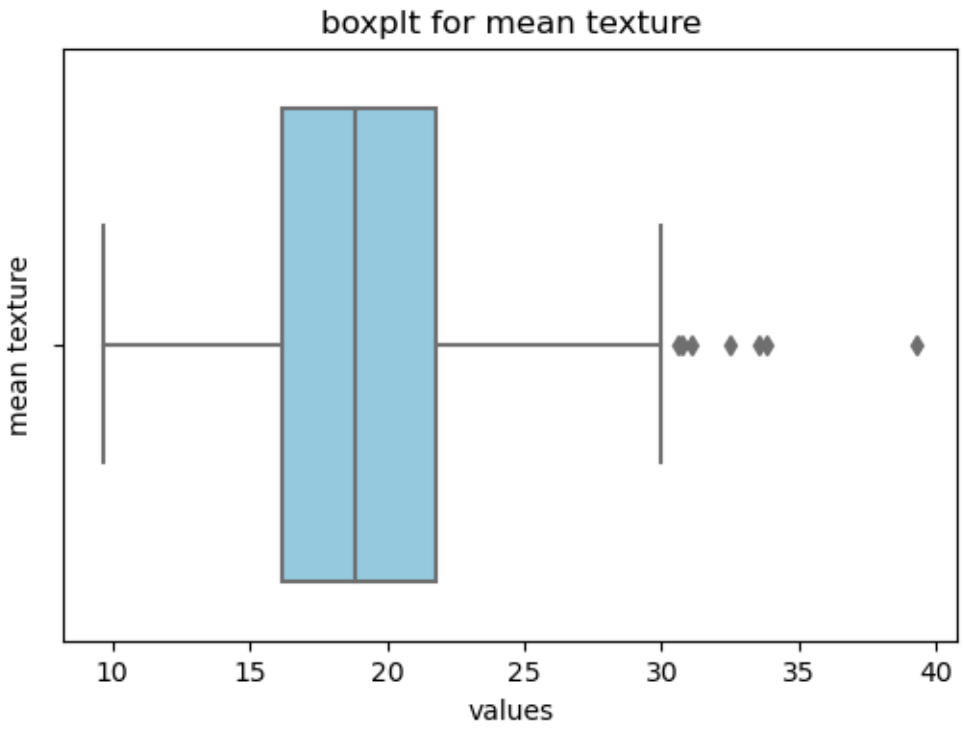
```
concave points error       19
symmetry error             27
fractal dimension error    28
worst radius               17
worst texture               5
worst perimeter            15
worst area                 35
worst smoothness            7
worst compactness          16
worst concavity            12
worst concave points        0
worst symmetry             23
worst fractal dimension    24
target                      0
dtype: int64
```



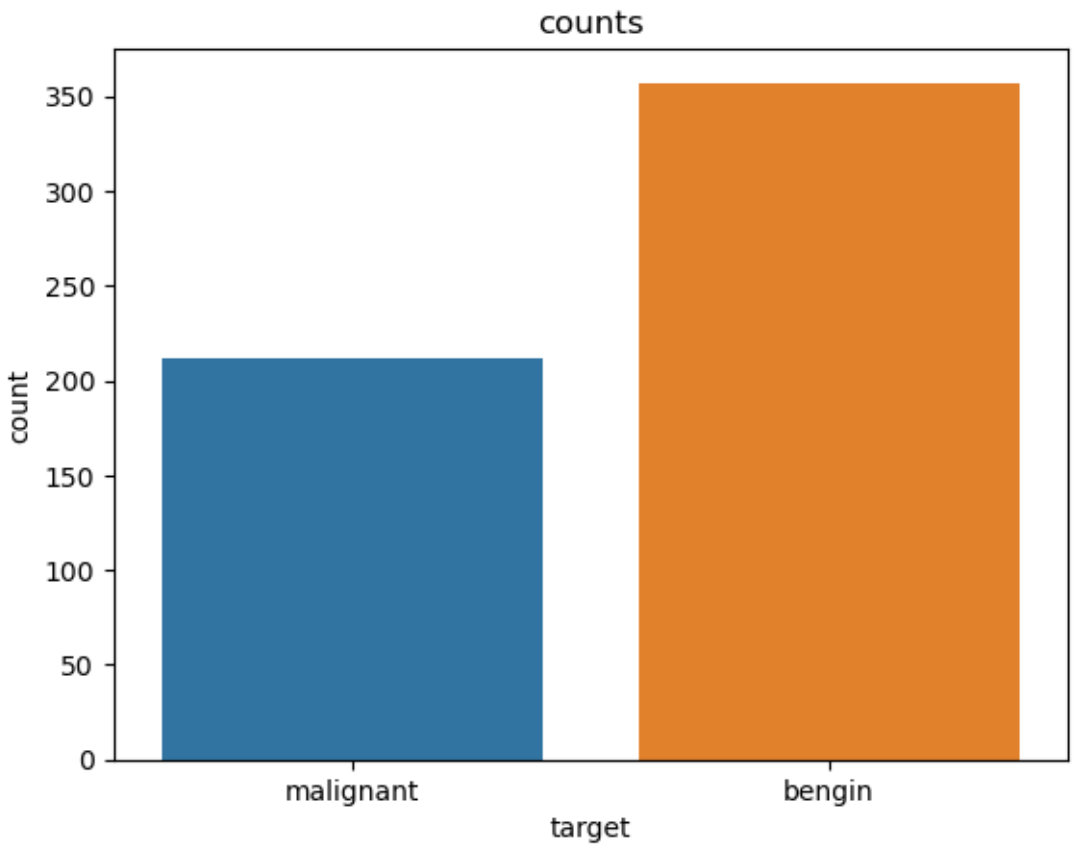

mean area vs target
2500
2000
1500
1000
500
mean area
0
1
target


histogram of mean radius
frequency
mean
0
10
20
30
40
50
60
10
15
20
25

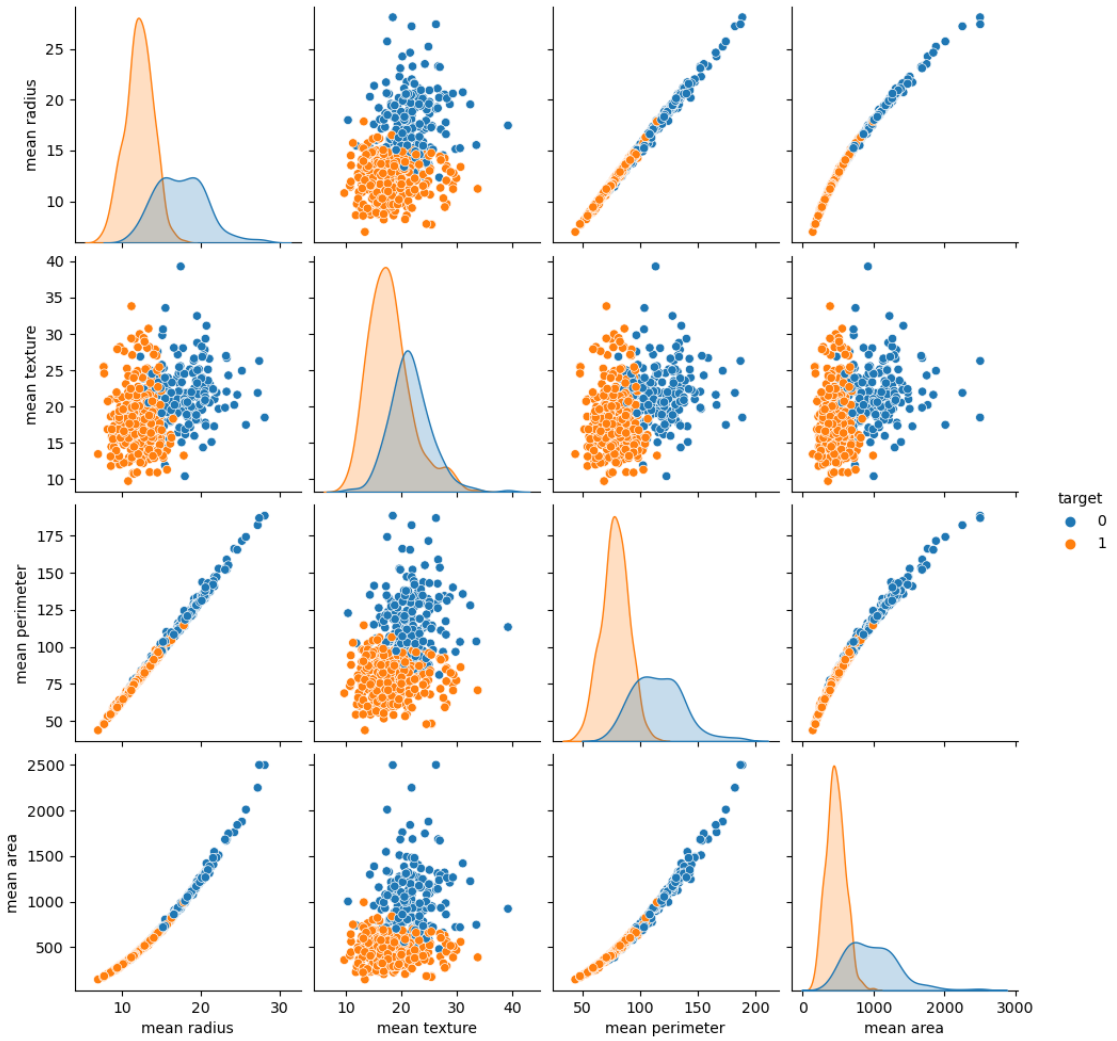

updated histo
frequency
log mean radius
0
10
20
30
40
50
2.0
2.2
2.4
2.6
2.8
3.0
3.2
3.4

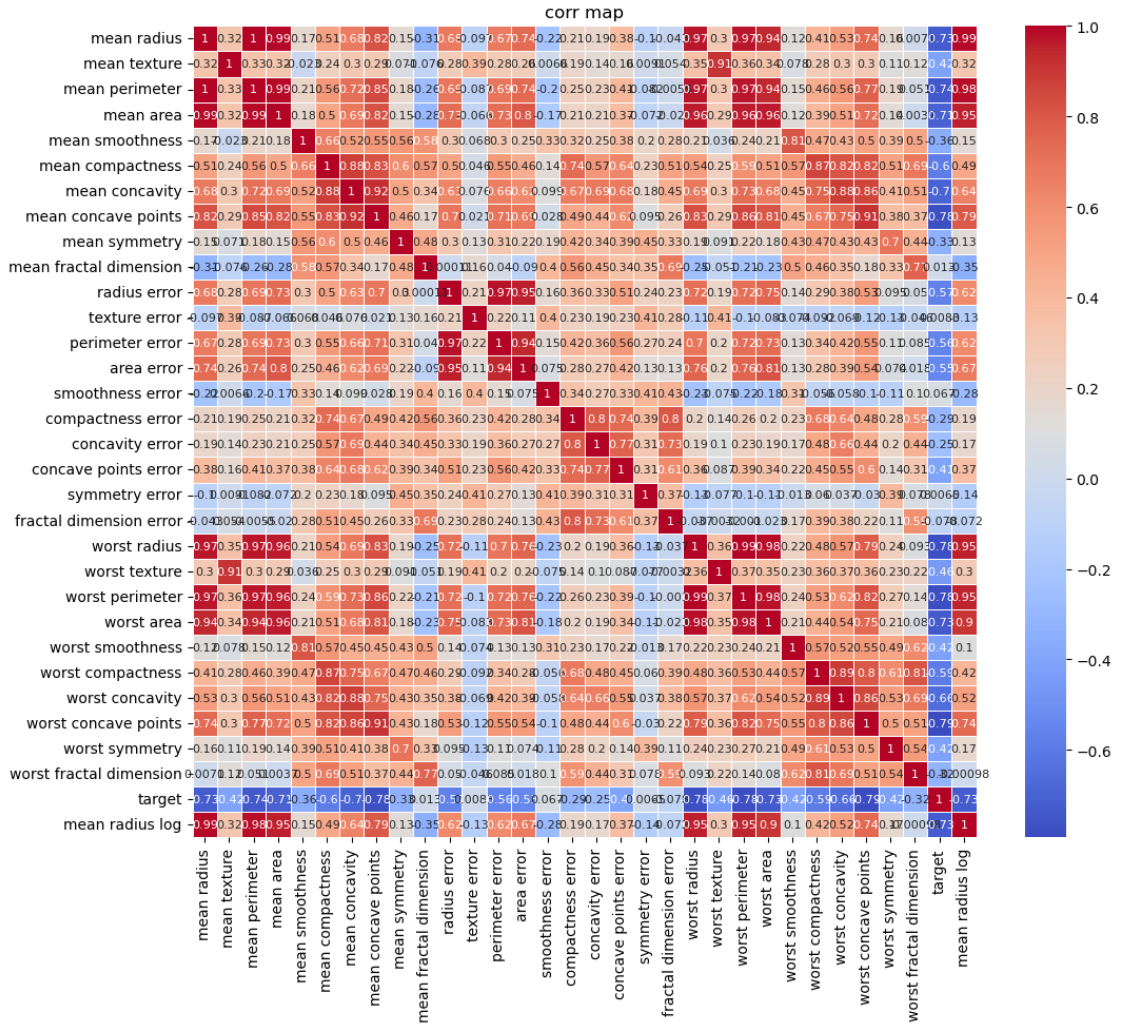


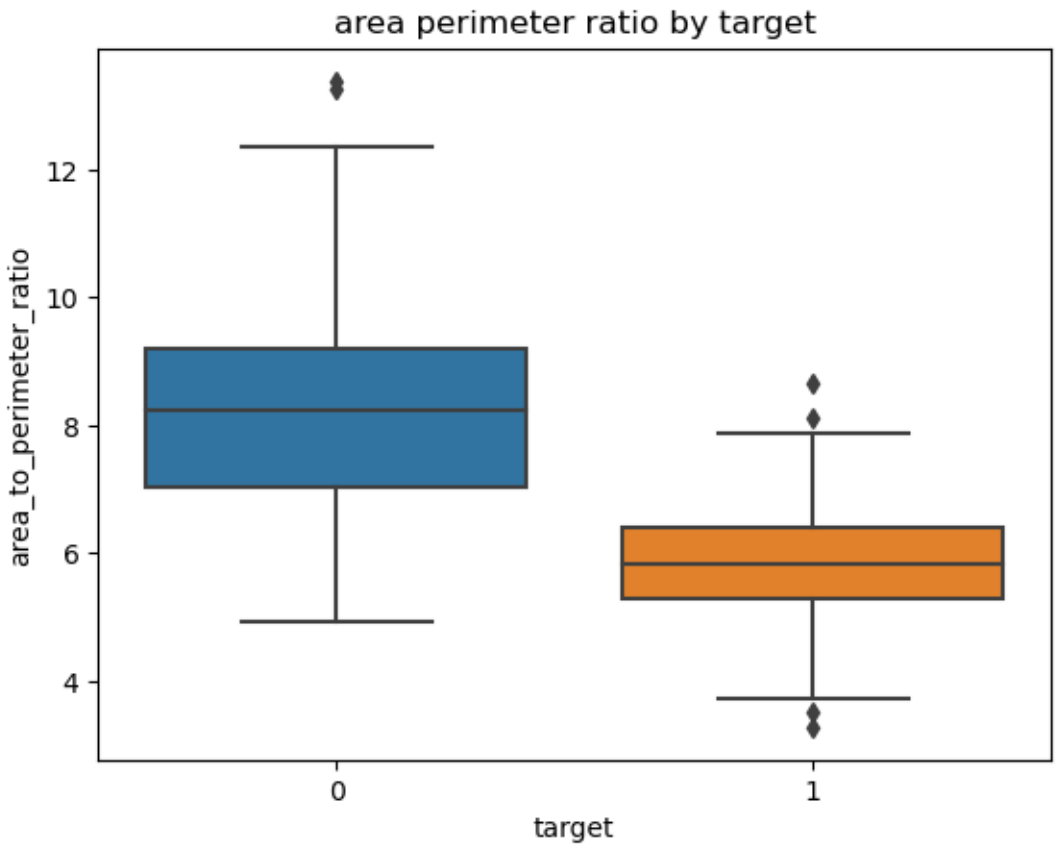

```
target
0    115.365377
1     78.075406
Name: mean perimeter, dtype: float64
```




```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```


mean radius
mean texture
mean perimeter
mean area
target
0
1


corr map
mean radius
mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
texture error
perimeter error
area error
smoothness error
compactness error
concavity error
concave points error
symmetry error
fractal dimension error
worst radius
worst texture
worst perimeter
worst area
worst smoothness
worst compactness
worst concavity
worst concave points
worst symmetry
worst fractal dimension
target
mean radius log
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6



	mean radius	mean texture	mean smoothness	mean compactness	\
0	17.99	10.38	0.11840	0.27760	
1	20.57	17.77	0.08474	0.07864	
2	19.69	21.25	0.10960	0.15990	
3	11.42	20.38	0.14250	0.28390	
4	20.29	14.34	0.10030	0.13280	
..	…	…	…	…	
564	21.56	22.39	0.11100	0.11590	
565	20.13	28.25	0.09780	0.10340	
566	16.60	28.08	0.08455	0.10230	
567	20.60	29.33	0.11780	0.27700	
568	7.76	24.54	0.05263	0.04362	

	mean concavity	mean concave points	mean symmetry	\
0	0.30010	0.14710	0.2419	
1	0.08690	0.07017	0.1812	
2	0.19740	0.12790	0.2069	
3	0.24140	0.10520	0.2597	
4	0.19800	0.10430	0.1809	
..	…	…	…	

```
564         0.24390              0.13890         0.1726
565         0.14400              0.09791         0.1752
566         0.09251              0.05302         0.1590
567         0.35140              0.15200         0.2397
568         0.00000              0.00000         0.1587

     mean fractal dimension  radius error  texture error  …  symmetry error  \
0                   0.07871        1.0950         0.9053  …         0.03003
1                   0.05667        0.5435         0.7339  …         0.01389
2                   0.05999        0.7456         0.7869  …         0.02250
3                   0.09744        0.4956         1.1560  …         0.05963
4                   0.05883        0.7572         0.7813  …         0.01756
..                      …             …              …  …             …
564                 0.05623        1.1760         1.2560  …         0.01114
565                 0.05533        0.7655         2.4630  …         0.01898
566                 0.05648        0.4564         1.0750  …         0.01318
567                 0.07016        0.7260         1.5950  …         0.02324
568                 0.05884        0.3857         1.4280  …         0.02676

     fractal dimension error  worst texture  worst smoothness  \
0                   0.006193          17.33           0.16220
1                   0.003532          23.41           0.12380
2                   0.004571          25.53           0.14440
3                   0.009208          26.50           0.20980
4                   0.005115          16.67           0.13740
..                       …              …                …
564                 0.004239          26.40           0.14100
565                 0.002498          38.25           0.11660
566                 0.003892          34.12           0.11390
567                 0.006185          39.42           0.16500
568                 0.002783          30.37           0.08996

     worst compactness  worst concavity  worst concave points  worst symmetry  \
0              0.66560           0.7119                0.2654          0.4601
1              0.18660           0.2416                0.1860          0.2750
2              0.42450           0.4504                0.2430          0.3613
3              0.86630           0.6869                0.2575          0.6638
4              0.20500           0.4000                0.1625          0.2364
..                 …                …                     …               …
564            0.21130           0.4107                0.2216          0.2060
565            0.19220           0.3215                0.1628          0.2572
566            0.30940           0.3403                0.1418          0.2218
567            0.86810           0.9387                0.2650          0.4087
568            0.06444           0.0000                0.0000          0.2871

     worst fractal dimension  target
0                    0.11890       0
1                    0.08902       0
```

```
2                    0.08758       0
3                    0.17300       0
4                    0.07678       0
..                       …     …
564                  0.07115       0
565                  0.06637       0
566                  0.07820       0
567                  0.12400       0
568                  0.07039       1

[569 rows x 24 columns]
['mean perimeter', 'mean area', 'perimeter error', 'area error', 'worst radius',
'worst perimeter', 'worst area', 'mean radius log', 'area_to_perimeter_ratio']
   mean radius  mean texture  mean smoothness  mean compactness  \
0        17.99         10.38          0.11840           0.27760
1        20.57         17.77          0.08474           0.07864
2        19.69         21.25          0.10960           0.15990
3        11.42         20.38          0.14250           0.28390
4        20.29         14.34          0.10030           0.13280

   mean concavity  mean concave points  mean symmetry  mean fractal dimension  \
0          0.3001              0.14710         0.2419                 0.07871
1          0.0869              0.07017         0.1812                 0.05667
2          0.1974              0.12790         0.2069                 0.05999
3          0.2414              0.10520         0.2597                 0.09744
4          0.1980              0.10430         0.1809                 0.05883

   radius error  texture error  …  symmetry error  fractal dimension error  \
0        1.0950         0.9053  …         0.03003                 0.006193
1        0.5435         0.7339  …         0.01389                 0.003532
2        0.7456         0.7869  …         0.02250                 0.004571
3        0.4956         1.1560  …         0.05963                 0.009208
4        0.7572         0.7813  …         0.01756                 0.005115

   worst texture  worst smoothness  worst compactness  worst concavity  \
0          17.33            0.1622             0.6656           0.7119
1          23.41            0.1238             0.1866           0.2416
2          25.53            0.1444             0.4245           0.4504
3          26.50            0.2098             0.8663           0.6869
4          16.67            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension  target
0                0.2654          0.4601                  0.11890       0
1                0.1860          0.2750                  0.08902       0
2                0.2430          0.3613                  0.08758       0
3                0.2575          0.6638                  0.17300       0
4                0.1625          0.2364                  0.07678       0
```

```
[5 rows x 24 columns]
(569, 24)
accuracy 0.9649122807017544
classification report               precision    recall  f1-score   support

           0       0.95      0.95      0.95        43
           1       0.97      0.97      0.97        71

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

confusion matrix [[41  2]
 [ 2 69]]
```