arthikeyn-r-task-13-batch-5-team-4

December 25, 2023

[4]:

```
import numpy as np
import pandas as pd
df = pd.read_csv("C:/Users/KARTHIK/Downloads/breast-cancer.csv",␣
 ↪index_col=False)
```

[5]:

```
df.head()
```

[5]:

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

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

   …  radius_worst  texture_worst  perimeter_worst  area_worst  \
0  …         25.38          17.33           184.60      2019.0
1  …         24.99          23.41           158.80      1956.0
2  …         23.57          25.53           152.50      1709.0
3  …         14.91          26.50            98.87       567.7
4  …         22.54          16.67           152.20      1575.0

   smoothness_worst  compactness_worst  concavity_worst  concave points_worst  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   symmetry_worst  fractal_dimension_worst
0          0.4601                  0.11890
```

```
1         0.2750                   0.08902
2         0.3613                   0.08758
3         0.6638                   0.17300
4         0.2364                   0.07678

[5 rows x 32 columns]
```

[7]:
```
df.shape
```

[7]:
```
(569, 32)
```

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

[9]:
```
df.isnull().any()
```

[9]:
```
id                         False
diagnosis                  False
radius_mean                False
texture_mean               False
perimeter_mean             False
area_mean                  False
smoothness_mean            False
compactness_mean           False
concavity_mean             False
concave points_mean        False
symmetry_mean              False
fractal_dimension_mean     False
radius_se                  False
texture_se                 False
perimeter_se               False
area_se                    False
smoothness_se              False
compactness_se             False
concavity_se               False
concave points_se          False
symmetry_se                False
fractal_dimension_se       False
radius_worst               False
texture_worst              False
perimeter_worst            False
area_worst                 False
smoothness_worst           False
compactness_worst          False
concavity_worst            False
concave points_worst       False
symmetry_worst             False
fractal_dimension_worst    False
dtype: bool
```

[10]:
```
df.diagnosis.unique()
```

[10]:
```
array(['M', 'B'], dtype=object)
```

[11]:
```
%matplotlib inline
import matplotlib.pyplot as plt

#Load libraries for data processing
import pandas as pd #data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np
from scipy.stats import norm
import seaborn as sns # data visualization


plt.rcParams['figure.figsize'] = (15,8)
plt.rcParams['axes.titlesize'] = 'large'
```

[14]:
```
df.describe()
```

[14]:
```
                 id  radius_mean  texture_mean  perimeter_mean    area_mean  \
count  5.690000e+02   569.000000    569.000000      569.000000   569.000000
mean   3.037183e+07    14.127292     19.289649       91.969033   654.889104
std    1.250206e+08     3.524049      4.301036       24.298981   351.914129
min    8.670000e+03     6.981000      9.710000       43.790000   143.500000
25%    8.692180e+05    11.700000     16.170000       75.170000   420.300000
50%    9.060240e+05    13.370000     18.840000       86.240000   551.100000
75%    8.813129e+06    15.780000     21.800000      104.100000   782.700000
max    9.113205e+08    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  …  radius_worst  texture_worst  perimeter_worst  \
count     569.000000  …    569.000000     569.000000       569.000000
mean        0.181162  …     16.269190      25.677223       107.261213
std         0.027414  …      4.833242       6.146258        33.602542
min         0.106000  …      7.930000      12.020000        50.410000
25%         0.161900  …     13.010000      21.080000        84.110000
50%         0.179200  …     14.970000      25.410000        97.660000
75%         0.195700  …     18.790000      29.720000       125.400000
max         0.304000  …     36.040000      49.540000       251.200000

        area_worst  smoothness_worst  compactness_worst  concavity_worst  \
count   569.000000        569.000000         569.000000       569.000000
```

```
mean    880.583128          0.132369           0.254265          0.272188
std     569.356993          0.022832           0.157336          0.208624
min     185.200000          0.071170           0.027290          0.000000
25%     515.300000          0.116600           0.147200          0.114500
50%     686.500000          0.131300           0.211900          0.226700
75%    1084.000000          0.146000           0.339100          0.382900
max    4254.000000          0.222600           1.058000          1.252000

       concave points_worst  symmetry_worst  fractal_dimension_worst
count            569.000000      569.000000               569.000000
mean               0.114606        0.290076                 0.083946
std                0.065732        0.061867                 0.018061
min                0.000000        0.156500                 0.055040
25%                0.064930        0.250400                 0.071460
50%                0.099930        0.282200                 0.080040
75%                0.161400        0.317900                 0.092080
max                0.291000        0.663800                 0.207500

[8 rows x 31 columns]
```

[15]:

```
df.diagnosis.unique()
```

[15]:

```
array(['M', 'B'], dtype=object)
```

[16]:

```
# Group by diagnosis and review the output.
diag_gr = df.groupby('diagnosis', axis=0)
pd.DataFrame(diag_gr.size(), columns=[' of observations'])
```

```
C:\Users\KARTHIK\AppData\Local\Temp\ipykernel_9936\3498095263.py:2:
FutureWarning: The 'axis' keyword in DataFrame.groupby is deprecated and will be
removed in a future version.
  diag_gr = df.groupby('diagnosis', axis=0)
```

[16]:

```
            of observations
diagnosis
B                       357
M                       212
```

[17]:

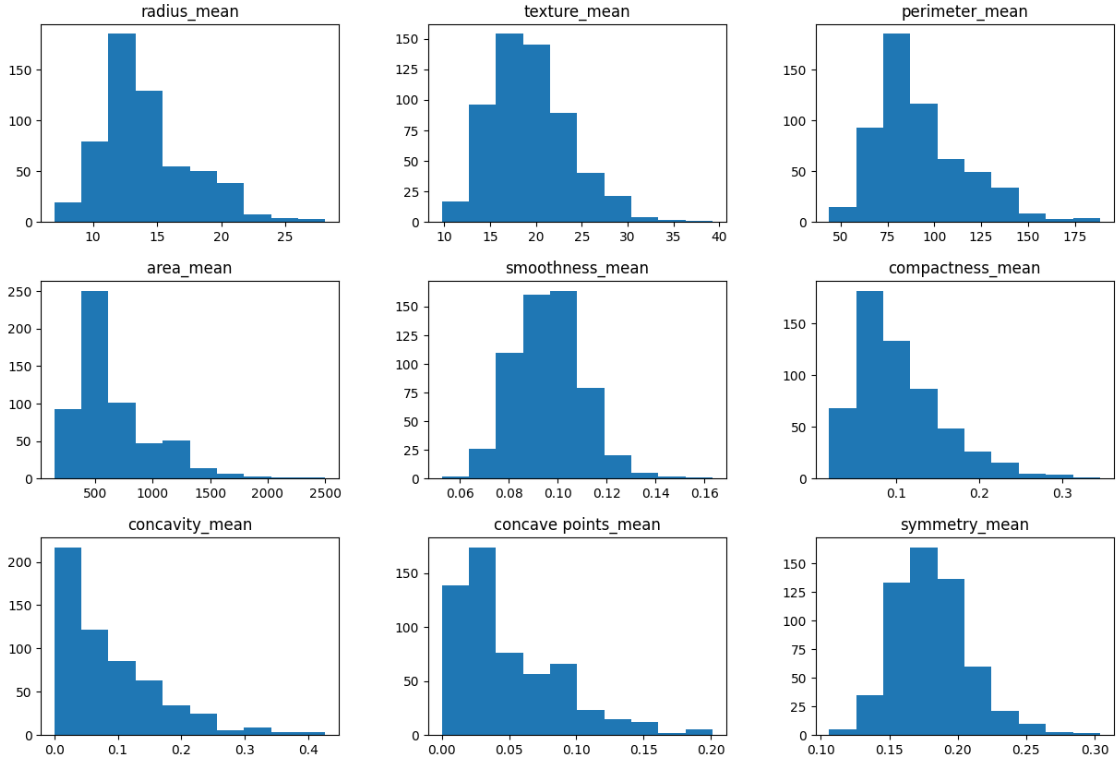
```
#Break up columns into groups, according to their suffix designation
#(_mean, _se,and __worst) to perform visualisation plots off.
#Join the 'ID' and 'Diagnosis' back on
df_id_diag=df.loc[:,["id","diagnosis"]]
df_diag=df.loc[:,["diagnosis"]]

#For a merge + slice:
df_mean=df.iloc[:,1:11]
df_se=df.iloc[:,11:22]
```

```
df_worst=df.iloc[:,23:]

print(df_id_diag.columns)
#print(data_mean.columns)
#print(data_se.columns)
#print(data_worst.columns)
```

```
Index(['id', 'diagnosis'], dtype='object')
```

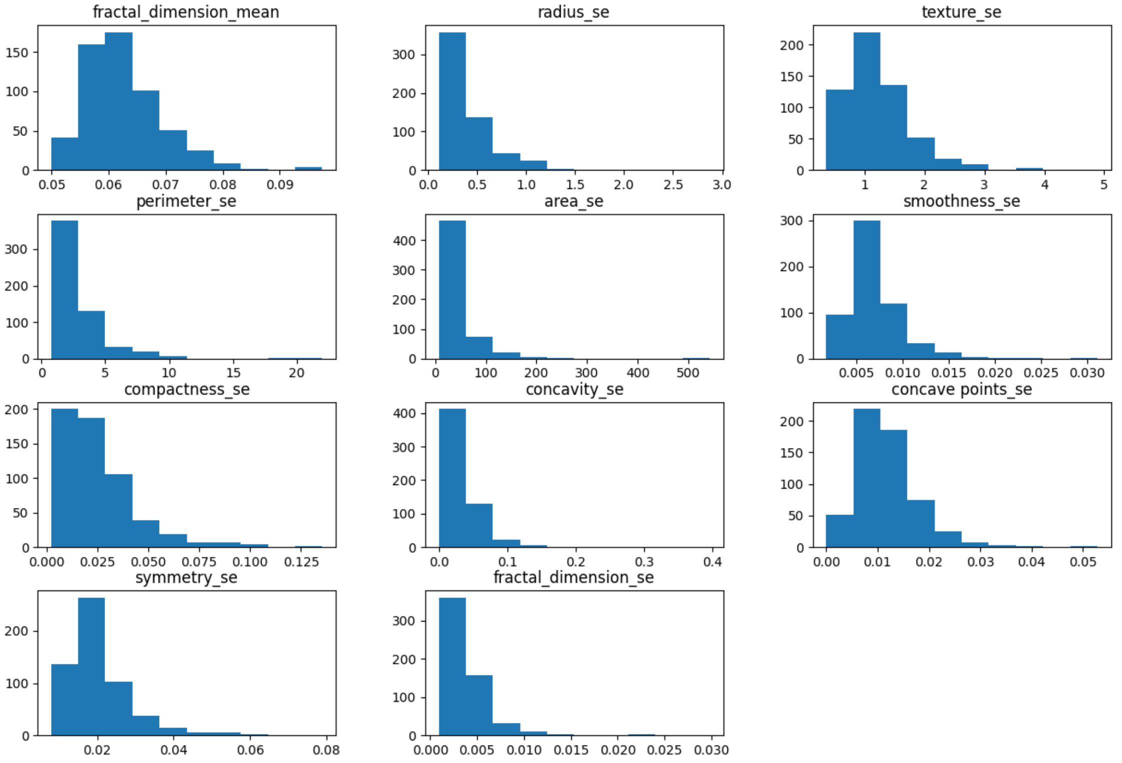
[18]:

```
#Plot histograms of CUT1 variables
hist_mean=df_mean.hist(bins=10, figsize=(15, 10),grid=False,)

#Any individual histograms, use this:
#df_cut['radius_worst'].hist(bins=100)
```

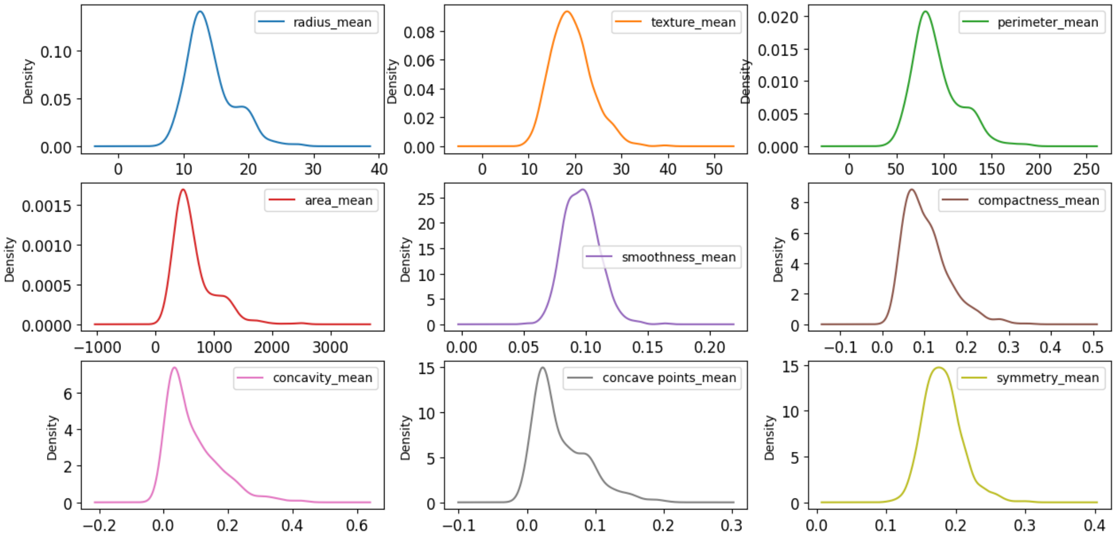


[19]:

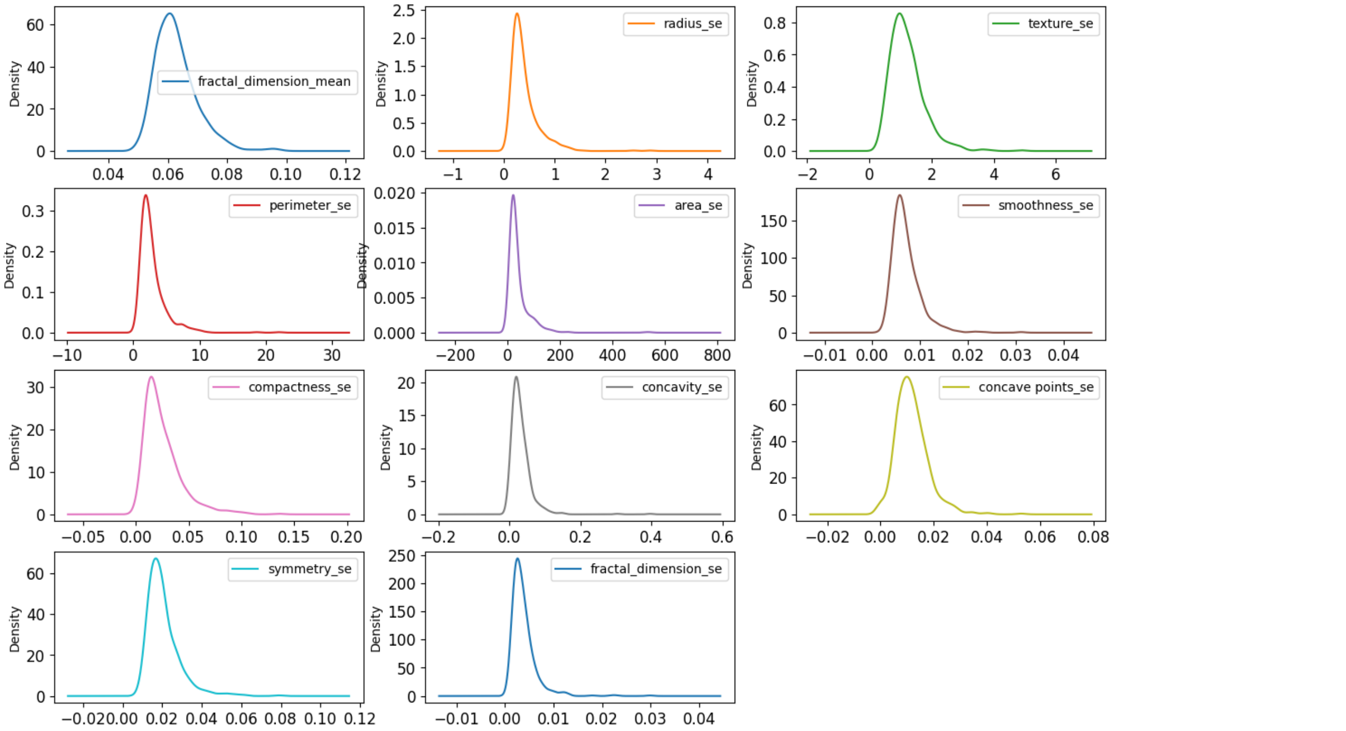
```
#Plot histograms of _se variables
hist_se=df_se.hist(bins=10, figsize=(15, 10),grid=False,)
```



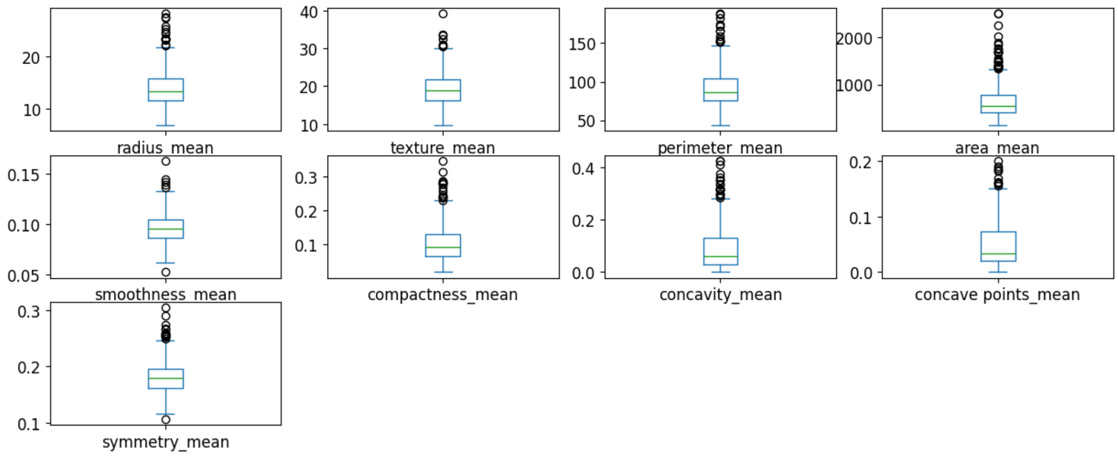
```
[20]: #Density Plots
      plt = df_mean.plot(kind= 'density', subplots=True, layout=(4,3), sharex=False,
                         sharey=False, fontsize=12, figsize=(15,10))
```



[21]:
```
#Density Plots
plt = df_se.plot(kind= 'density', subplots=True, layout=(4,3), sharex=False,
                    sharey=False, fontsize=12, figsize=(15,10))
```

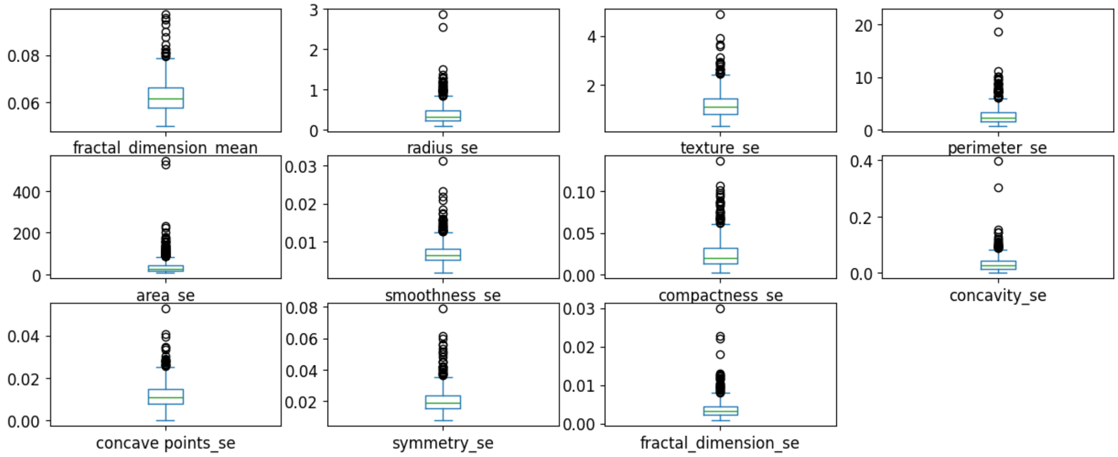


[22]:
```
# box and whisker plots
plt=df_mean.plot(kind= 'box' , subplots=True, layout=(4,4), sharex=False,␣
 ↪sharey=False,
                 fontsize=12)
```



[23]:

```
# box and whisker plots
plt=df_se.plot(kind= 'box' , subplots=True, layout=(4,4), sharex=False,␣
 ↪sharey=False,
                fontsize=12)
```



[26]:

```
%matplotlib inline
import matplotlib.pyplot as plt

#Load libraries for data processing
import pandas as pd #data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np
from scipy.stats import norm

# visualization
import seaborn as sns
plt.style.use('fivethirtyeight')
sns.set_style("white")


plt.rcParams['figure.figsize'] = (8,4)
#plt.rcParams['axes.titlesize'] = 'large'

df = pd.read_csv('C:/Users/KARTHIK/Downloads/breast-cancer.csv',␣
 ↪index_col=False)
df.head(3)
```

[26]:

```
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38           122.8     1001.0
1    842517         M        20.57         17.77           132.9     1326.0
```

```
2  84300903         M        19.69         21.25           130.0     1203.0

   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

   …  radius_worst  texture_worst  perimeter_worst  area_worst  \
0  …         25.38          17.33            184.6      2019.0
1  …         24.99          23.41            158.8      1956.0
2  …         23.57          25.53            152.5      1709.0

   smoothness_worst  compactness_worst  concavity_worst  concave points_worst  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430

   symmetry_worst  fractal_dimension_worst
0          0.4601                  0.11890
1          0.2750                  0.08902
2          0.3613                  0.08758

[3 rows x 32 columns]
```

[27]:

```python
#Assign predictors to a variable of ndarray (matrix) type
array = df.values
X = array[:,1:31]
y = array[:,0]
X
```

[27]:

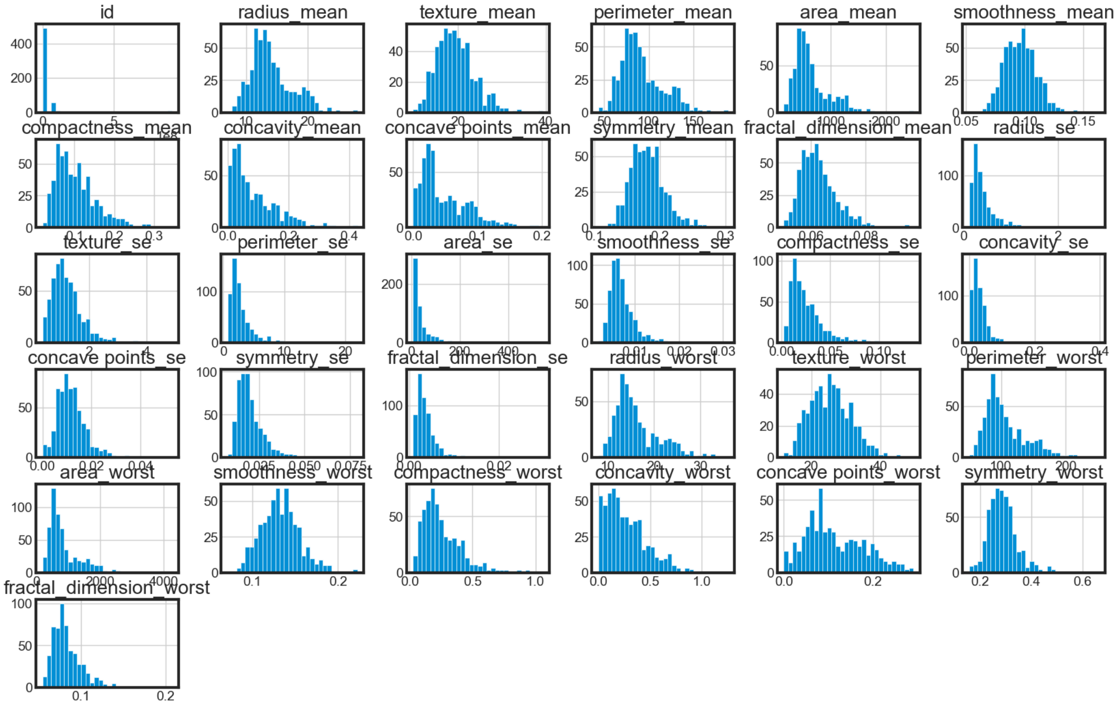
```
array([['M', 17.99, 10.38, …, 0.7119, 0.2654, 0.4601],
       ['M', 20.57, 17.77, …, 0.2416, 0.186, 0.275],
       ['M', 19.69, 21.25, …, 0.4504, 0.243, 0.3613],
       …,
       ['M', 16.6, 28.08, …, 0.3403, 0.1418, 0.2218],
       ['M', 20.6, 29.33, …, 0.9387, 0.265, 0.4087],
       ['B', 7.76, 24.54, …, 0.0, 0.0, 0.2871]], dtype=object)
```

[28]:

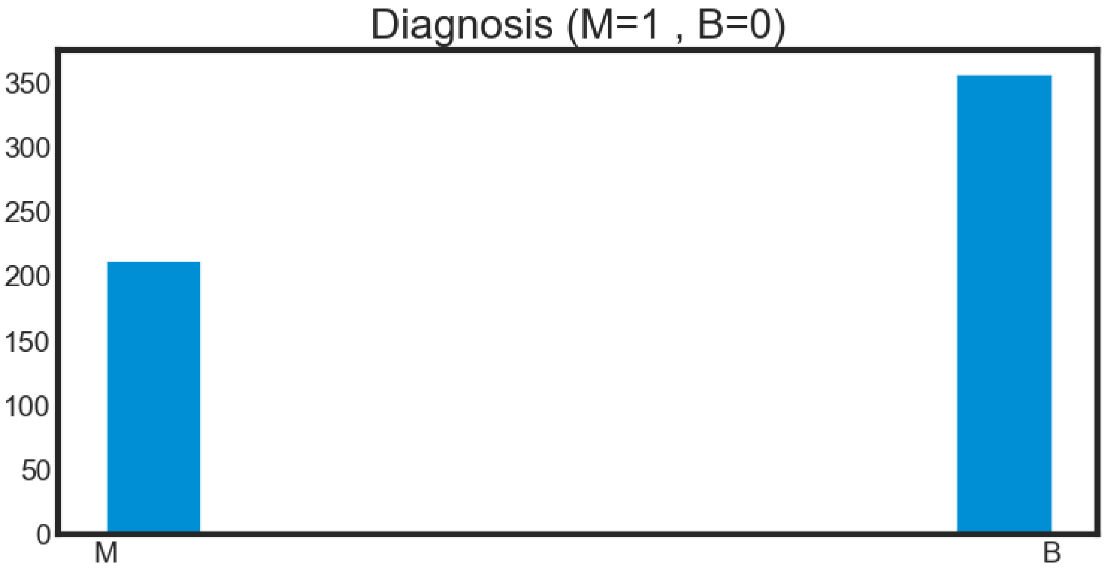
```python
#transform the class labels from their original string representation (M and B)␣
 ↪into integers
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
y

# Call the transform method of LabelEncorder on two dummy variables
# le.transform (['M', 'B'])
```

```
[28]: array([ 41,  42, 488, 489, 490,  43,  44, 491,  45, 492,  46, 493,  47,
              48, 494, 495,  49, 496,  50, 385, 386, 387, 388,  51,  52,  53,
              54,  55,  56,  57,  58,  59, 497,  60,  61,  62,  63,  64,  65,
              66,  67,  68,  69,  70, 498,  71, 499,   4,  72,  73,  74,  75,
              76,  77,  78, 500,  79,  80,  81,  82,  83,  84,  85,  86, 501,
              87,  88,  89,  90,  91,  92,  93,  94,  95, 389, 390, 391, 392,
             393, 394,  96, 395, 396, 397, 398, 399, 502, 503,  97,  98,  99,
             100, 101, 102, 103,   5,   6, 104, 105, 106, 107, 108, 109, 110,
             111, 112, 113, 114,   7, 115, 116,   8,   9, 117, 118, 119, 120,
             121, 122, 123, 124,  10, 125, 126, 127,  11, 128, 129, 130, 131,
             132,   0, 504, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
             143, 144, 145, 146, 505, 506, 147, 558, 559, 400,  12, 401, 402,
             403, 404, 148, 149, 405, 406, 150, 407, 408, 409,  13, 410, 411,
             412, 507,  14,  15, 151, 152, 153, 154, 508, 155, 156, 157, 158,
             159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 509,
             171, 172, 173, 174, 175, 176, 177, 178,  16,  17, 179, 180, 181,
             413, 414, 560, 415, 416, 561, 417, 418, 419, 420, 421, 510, 422,
             423, 424, 425, 426, 511, 512, 513, 514, 182, 183, 515, 516, 517,
             184, 518, 519, 185, 186, 520, 521, 187, 188, 522, 189, 190, 191,
             192, 523, 193, 194, 524, 195, 427, 196, 197, 525, 198, 199, 526,
             200, 201, 202, 203, 204, 527, 428, 429, 430, 431, 432, 433, 434,
             435, 436, 437, 438, 439, 440, 441, 442, 443,  18, 444, 445, 446,
             447,   1, 448, 528, 529,   2, 205, 206, 207, 208, 209, 210, 211,
             212, 213, 214, 530, 215,  19, 216,  20,  21, 217, 218, 219, 531,
             532, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 533,
             534,  22, 231, 449, 232, 233, 234, 235, 236, 237, 535, 238, 239,
             240,  23,  24, 241,  25, 242, 536, 243, 244,  26, 245, 246, 247,
             248, 249, 450, 250, 451, 452, 251, 453, 562, 563, 252, 454, 455,
             456, 253, 457, 458, 459, 460, 461, 462, 463, 254, 464, 255, 256,
             465, 466, 467, 257, 258,  27,  28, 259,  29, 260, 261, 262,  30,
             537, 263, 264, 265, 266, 267, 538, 539, 268, 269, 540, 270, 271,
               3, 272, 273, 274, 275, 541, 276, 277, 278, 279, 280, 281, 282,
             283, 542, 284, 285, 286, 287, 288, 289, 290, 291, 292,  31, 543,
             544, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304,
             545, 305, 468, 469, 470, 471, 306, 564, 472, 473, 474, 307, 308,
             475, 476, 477, 478, 479, 565, 566, 480, 567, 568, 481, 482, 483,
             484, 309, 485, 486, 310, 487, 311, 312, 313, 314, 315, 316, 317,
              32, 318, 319, 320, 321, 322, 323, 324, 325, 546, 547, 326, 327,
             328, 329, 330, 331, 332,  33, 333,  34,  35, 334, 335, 336, 548,
             549, 337, 338,  36, 339, 340, 341, 550, 342, 343, 344, 345, 346,
             347, 551,  37, 348, 349,  38, 552, 553, 350, 351,  39, 554, 555,
             556, 352, 353, 557, 354, 355, 356, 357, 358, 359, 360, 361, 362,
             363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375,
             376, 377, 378, 379, 380, 381, 382, 383, 384,  40])
```

```
[29]: df.hist(bins=30, figsize=(18,12))
      plt.show()
```

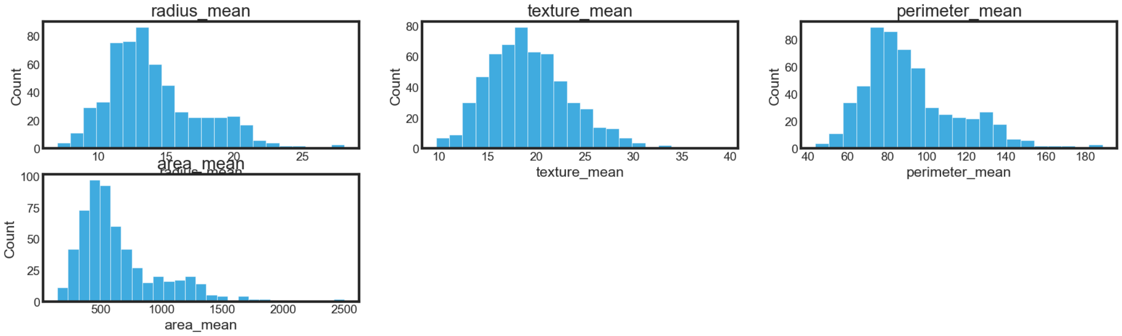


[30]:
```
df.describe()
plt.hist(df['diagnosis'])
plt.title('Diagnosis (M=1 , B=0)')
plt.show()
```



[35]:

```
num_list=['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean']
fig = plt.figure(figsize=(20,30))

for i in range(len(num_list)):
    plt.subplot(10,3,i+1)
    plt.title(num_list[i])
    #Target for Dataset 1 and 2
    #sns.histplot(data=df,x=df[num_list[i]],hue='diagnosis')

    #Target for Dataset 3
    sns.histplot(data=df,x=df[num_list[i]])

plt.show()
```



[ ]: