ASumbaraju_Project_Milestone1

July 16, 2021

1 DSC 550 Project Milestone1

2 Aditya Sumbaraju

3 07/15/2021

[143]:
```
import pandas as pd
import numpy as np
import yellowbrick
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from yellowbrick.features import Rank2D
from yellowbrick.features import ParallelCoordinates
from yellowbrick.style import set_palette
```

[129]:
```
#Step 1: Load data into a dataframe
bc_df = "C:\BU\DSC550\project\data/data.csv"
data = pd.read_csv(bc_df)
```

4 Data cleansing Steps

[130]:
```
#deleting the "id" column
data.drop("id",axis=1,inplace=True)

# print the summary of the dataset
print (data.info())

#count total rows in each column which contain null values
print ("\n \n Check for null values \n", data.isna().sum())


#'duplicated()' function in pandas return the duplicate row as True and othter␣
↪as False
```

```python
print(" \n \n  Dupe Check \n" , sum(data.duplicated()))
```

```
<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
None


 Check for null values
 diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
```

```
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64


  Dupe Check
 0
```

[131]:
```
# check the dimension of the table
print("The dimension of the table is: ", data.shape)
```

```
The dimension of the table is:  (569, 31)
```

[132]:
```
#Summarizing the numerical data and categorical data
print("BC Numerical data summary:")
print(data.describe())
print("BC Categorical data summary:")
print(data.describe(include=['O']))
```

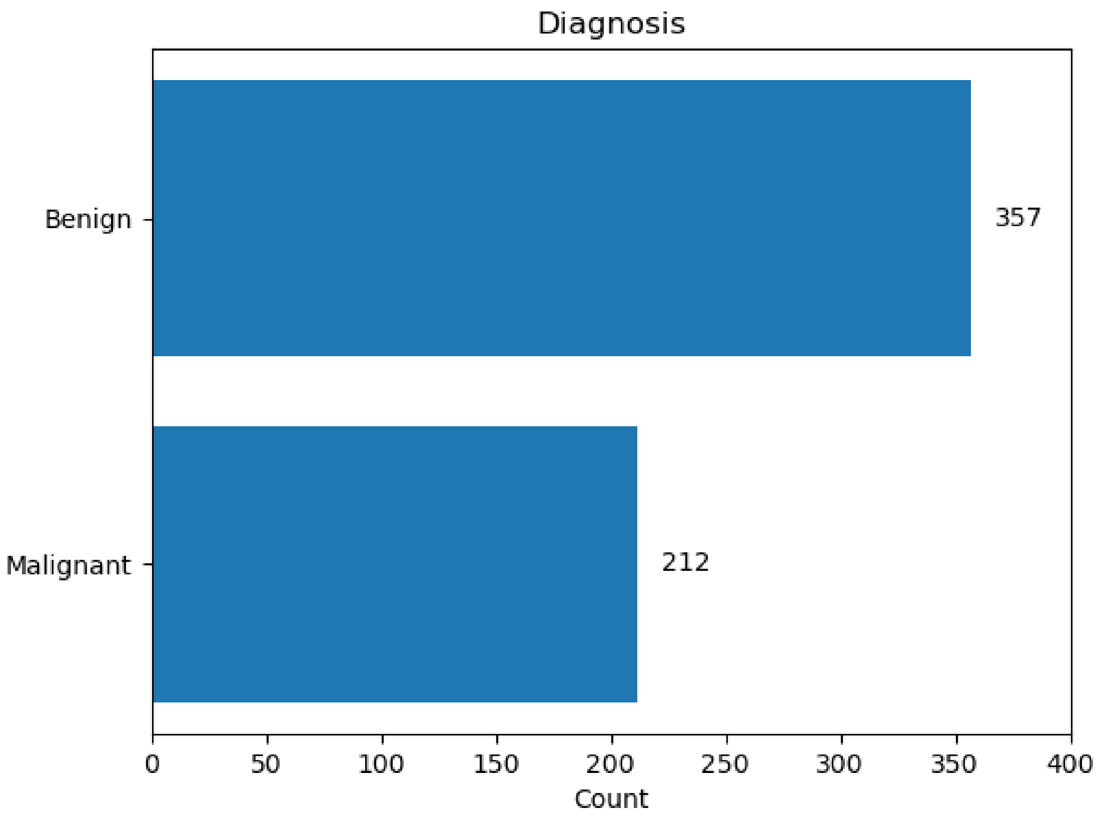
```
BC Numerical data summary:
       radius_mean  texture_mean  perimeter_mean   area_mean  \
count   569.000000    569.000000      569.000000  569.000000
mean     14.127292     19.289649       91.969033  654.889104
std       3.524049      4.301036       24.298981  351.914129
```

```
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_worst  \
count     569.000000              569.000000  …    569.000000
mean        0.181162                0.062798  …     16.269190
std         0.027414                0.007060  …      4.833242
min         0.106000                0.049960  …      7.930000
25%         0.161900                0.057700  …     13.010000
50%         0.179200                0.061540  …     14.970000
75%         0.195700                0.066120  …     18.790000
max         0.304000                0.097440  …     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
```

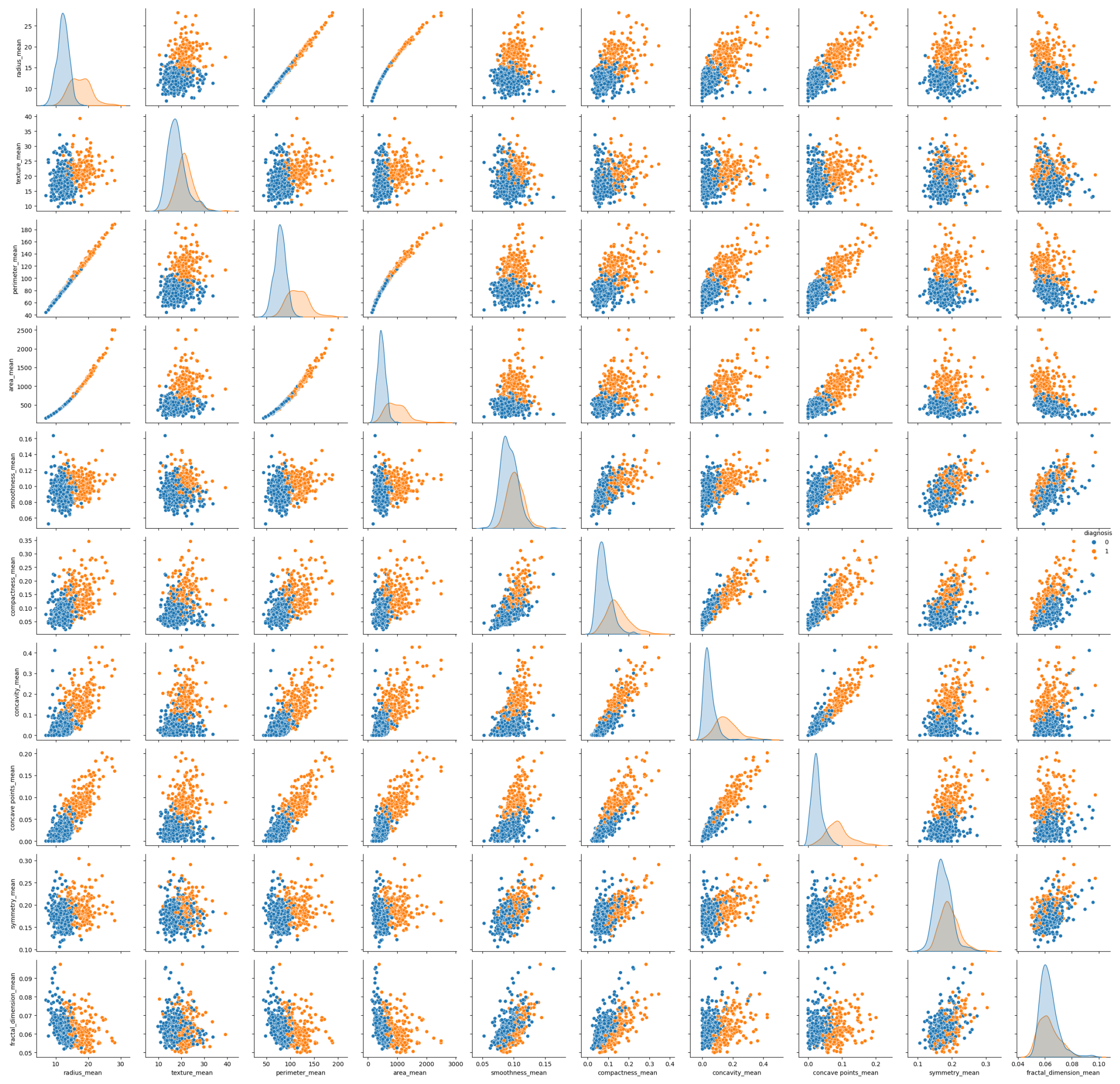
```
mean         0.290076                 0.083946
std          0.061867                 0.018061
min          0.156500                 0.055040
25%          0.250400                 0.071460
50%          0.282200                 0.080040
75%          0.317900                 0.092080
max          0.663800                 0.207500

[8 rows x 30 columns]
BC Categorical data summary:
       diagnosis
count        569
unique         2
top            B
freq         357
```

[133]:
```python
# Bar chart
plt.rcdefaults()
plt.figure()
benign = len(data[data['diagnosis'] == 'B'])
malignant = len(data[data['diagnosis'] == 'M'])
fig, ax = plt.subplots()
y = ('Benign', 'Malignant')
y_pos = np.arange(len(y))
x = (benign, malignant)
ax.barh(y_pos, x, align='center')
ax.set_xticks(np.arange(0,401,50))
ax.set_yticks(y_pos)
ax.set_yticklabels(y)
ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('Count')
ax.set_title('Diagnosis')
for i, v in enumerate(x):
    ax.text(v + 10, i, str(v), color='black', va='center', fontweight='normal')
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

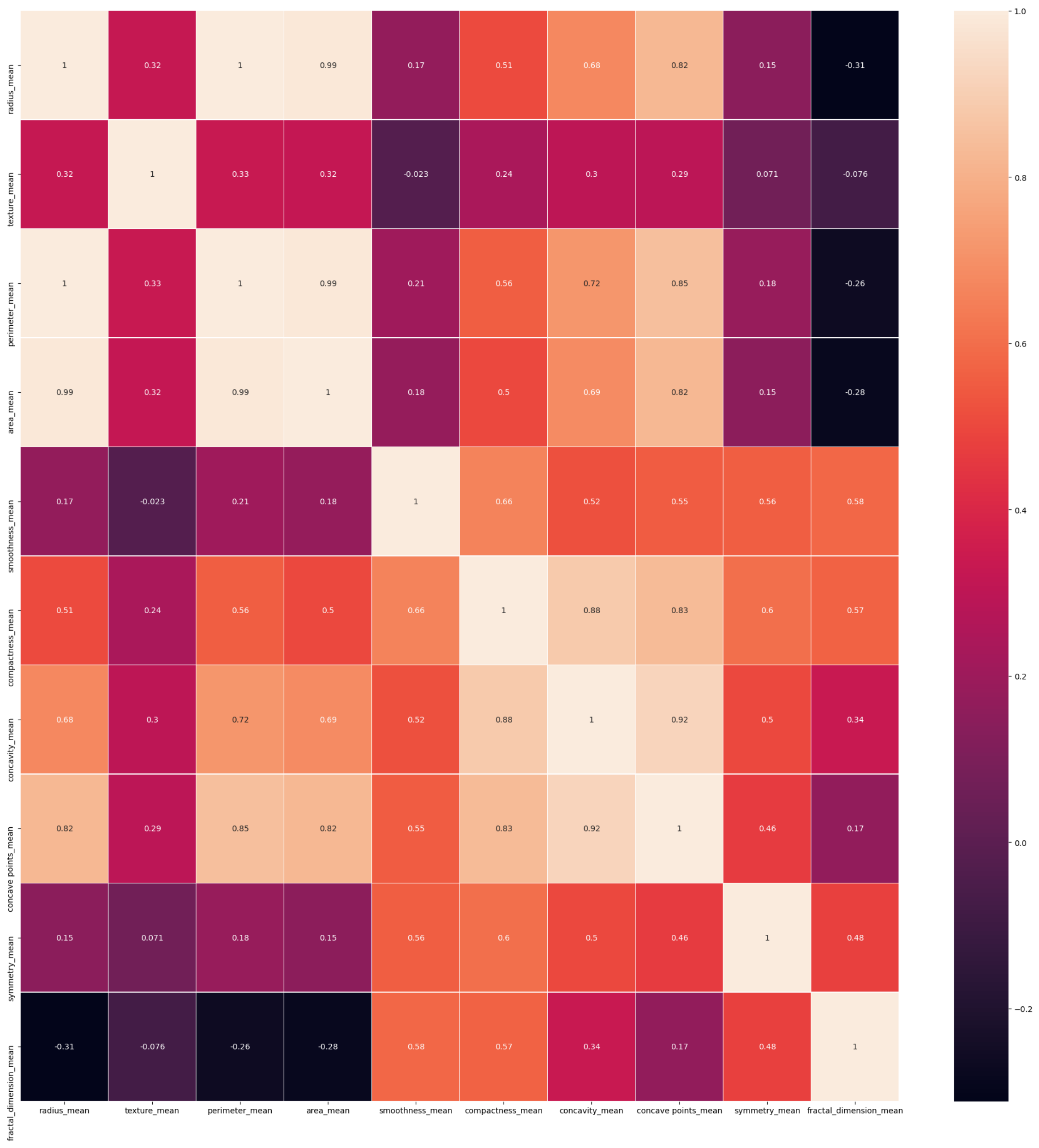


[134]:
```
# Vizualize Correlations Between "mean" Features
cols = ['radius_mean',␣
↪'texture_mean','perimeter_mean','area_mean','smoothness_mean','compactness_mean','concavity
↪points_mean','symmetry_mean','fractal_dimension_mean']

sns.pairplot(df,vars = cols, size=2.5,hue = 'diagnosis')
plt.tight_layout()
plt.show()
```



```
[135]: # Since there are 33 features, let check for multicollinearity for "Mean"␣
       ↪features
       # Let us use heatmap function
       plt.figure(figsize = (25, 25))
       sns.heatmap(df[cols].corr(), annot = True, linewidths = 0.20)
```

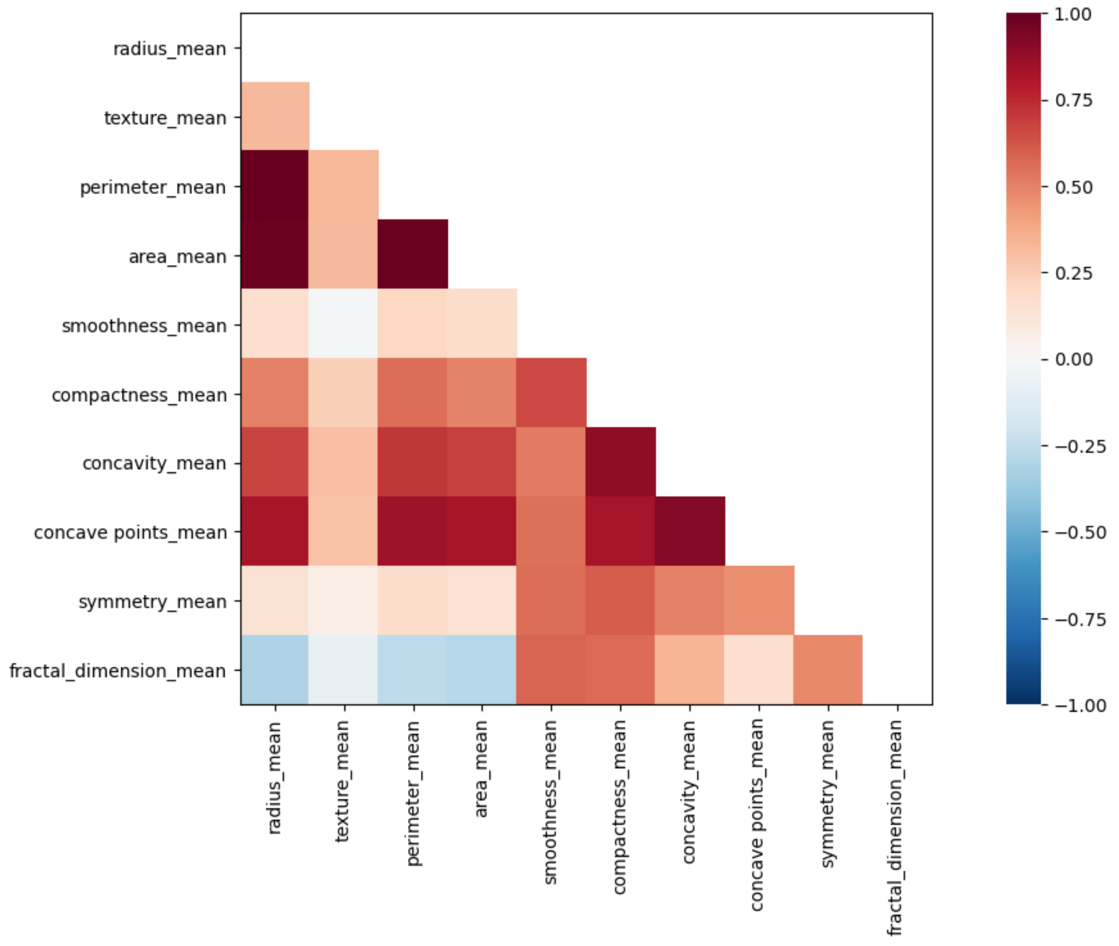
```
[135]: <AxesSubplot:>
```



	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
radius_mean	1	0.32	1	0.99	0.17	0.51	0.68	0.82	0.15	-0.31
texture_mean	0.32	1	0.33	0.32	-0.023	0.24	0.3	0.29	0.071	-0.076
perimeter_mean	1	0.33	1	0.99	0.21	0.56	0.72	0.85	0.18	-0.26
area_mean	0.99	0.32	0.99	1	0.18	0.5	0.69	0.82	0.15	-0.28
smoothness_mean	0.17	-0.023	0.21	0.18	1	0.66	0.52	0.55	0.56	0.58
compactness_mean	0.51	0.24	0.56	0.5	0.66	1	0.88	0.83	0.6	0.57
concavity_mean	0.68	0.3	0.72	0.69	0.52	0.88	1	0.92	0.5	0.34
concave points_mean	0.82	0.29	0.85	0.82	0.55	0.83	0.92	1	0.46	0.17
symmetry_mean	0.15	0.071	0.18	0.15	0.56	0.6	0.5	0.46	1	0.48
fractal_dimension_mean	-0.31	-0.076	-0.26	-0.28	0.58	0.57	0.34	0.17	0.48	1

[136]:
```
# Pearson Ranking
#%matplotlib inline
plt.rcParams['figure.figsize'] = (15, 7)
X = data[cols].values
fig = Rank2D(features=cols, algorithm='pearson')
fig.fit(X)
fig.transform(X)
```

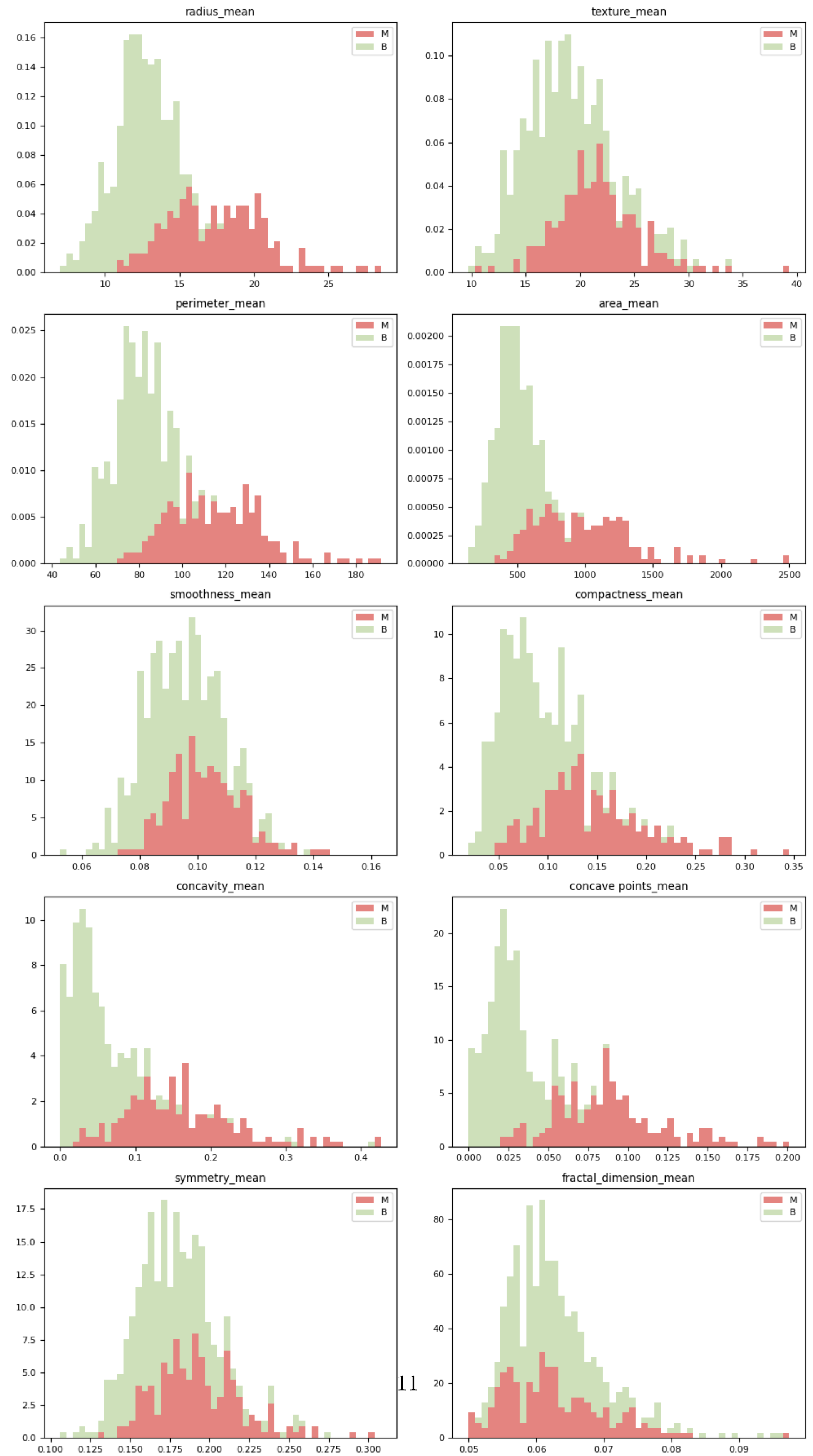
```
[136]: array([[1.799e+01, 1.038e+01, 1.228e+02, …, 1.471e-01, 2.419e-01,
                7.871e-02],
               [2.057e+01, 1.777e+01, 1.329e+02, …, 7.017e-02, 1.812e-01,
                5.667e-02],
               [1.969e+01, 2.125e+01, 1.300e+02, …, 1.279e-01, 2.069e-01,
                5.999e-02],
               …,
               [1.660e+01, 2.808e+01, 1.083e+02, …, 5.302e-02, 1.590e-01,
                5.648e-02],
               [2.060e+01, 2.933e+01, 1.401e+02, …, 1.520e-01, 2.397e-01,
                7.016e-02],
               [7.760e+00, 2.454e+01, 4.792e+01, …, 0.000e+00, 1.587e-01,
                5.884e-02]])
```



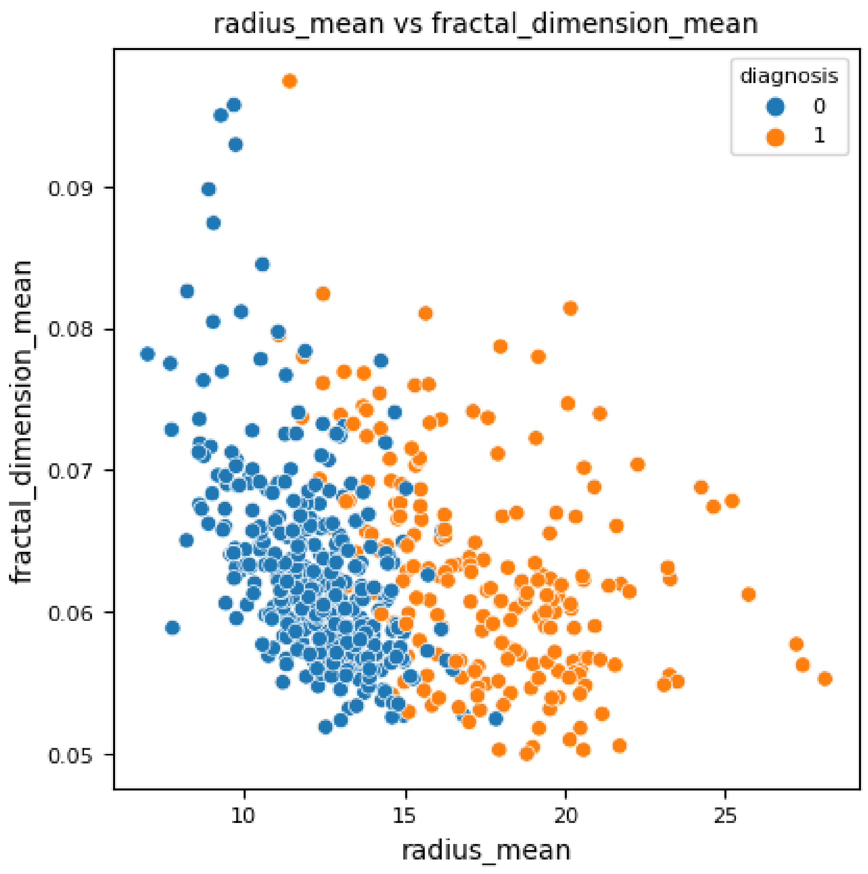
```
[137]: # Barcharts: set up the figure size
       data['diagnosis'] = data['diagnosis'].map({'M':1,'B':0})
       features_mean=list(data.columns[1:29])
```

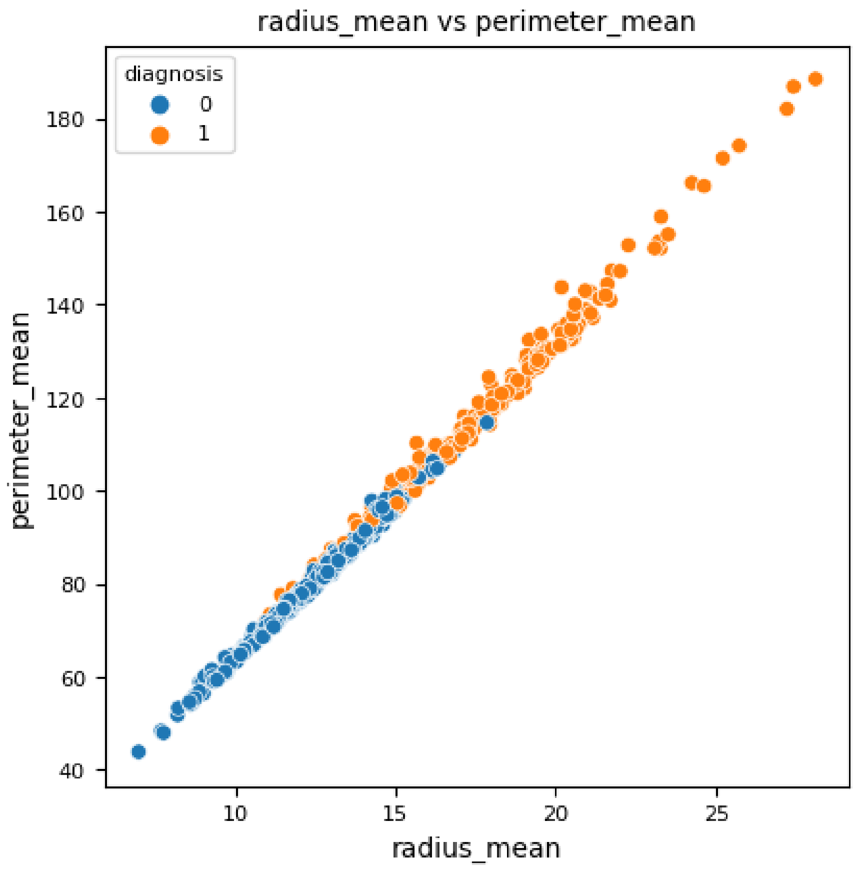
```
# split dataframe into two based on diagnosis
dfM=data[data['diagnosis'] ==1]
dfB=data[data['diagnosis'] ==0]
```

```
[138]: #Stack the data
plt.rcParams.update({'font.size': 8})
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(10,18))
axes = axes.ravel()
for idx,ax in enumerate(axes):
    ax.figure
    binwidth= (max(data[features_mean[idx]]) - min(data[features_mean[idx]]))/50
    ax.hist([dfM[features_mean[idx]],dfB[features_mean[idx]]], bins=np.
 ↪arange(min(data[features_mean[idx]]), max(data[features_mean[idx]]) +␣
 ↪binwidth, binwidth) , alpha=0.5,stacked=True, density = True,␣
 ↪label=['M','B'],color=['r','g'])
    ax.legend(loc='upper right')
    ax.set_title(features_mean[idx])
plt.tight_layout()
plt.show()
```

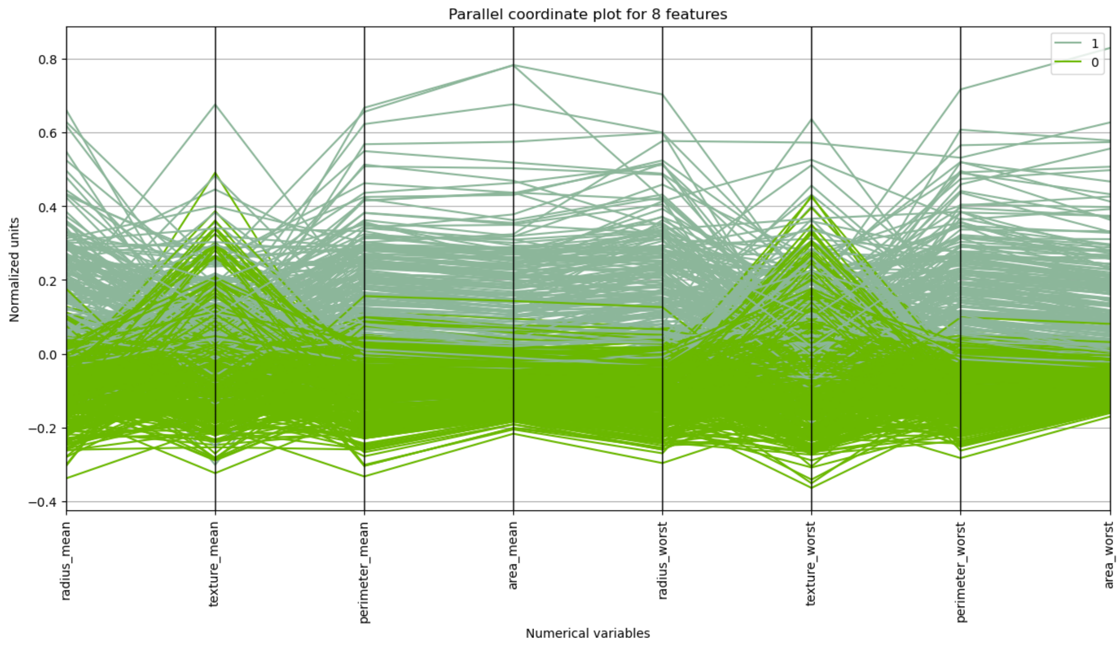

radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
M
B

[139]:
```
# The highly corelated pairs are:
# radius_mean vs fractal_dimension_mean
# radius_mean vs perimeter_mean


plt.figure(figsize = (5,5))
sns.scatterplot(x = 'radius_mean', y ='fractal_dimension_mean', hue =␣
 ↪'diagnosis', data = df)
plt.xlabel('radius_mean', fontsize = 10)
plt.ylabel('fractal_dimension_mean', fontsize = 10)
plt.title('radius_mean vs fractal_dimension_mean', fontsize = 10)
plt.show()


plt.figure(figsize = (5,5))
sns.scatterplot(x = 'radius_mean', y = 'perimeter_mean', hue = 'diagnosis',␣
 ↪data = df)
plt.xlabel('radius_mean', fontsize = 10)
plt.ylabel('perimeter_mean', fontsize = 10)
plt.title('radius_mean vs perimeter_mean', fontsize = 10)
plt.show()
```


radius_mean vs fractal_dimension_mean
diagnosis
0
1
0.09
0.08
0.07
0.06
0.05
fractal_dimension_mean
10
15
20
25
radius_mean



[149]:
```
# Parallel coordinate plot for the numerical variables to compare the features␣
↪mean vs worst

#%matplotlib inline
cols = ['radius_mean', 'texture_mean', 'perimeter_mean',␣
↪'area_mean','radius_worst',  'texture_worst',  'perimeter_worst',␣
↪'area_worst']

# copy data to a new dataframe
data_norm = data.copy()

for feature in cols:
    data_norm[feature] = (data[feature] - data[feature].mean(skipna=True)) /␣
↪(data[feature].max(skipna=True) - data[feature].min(skipna=True))

pd.plotting.parallel_coordinates(data_norm, class_column = 'diagnosis', cols =␣
↪cols)
plt.xlabel('Numerical variables')
plt.ylabel('Normalized units')
plt.title('Parallel coordinate plot for 8 features')
plt.xticks(rotation = 90)
plt.show()
```



[ ]:

```
```