breast-cancer-prediction

February 3, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df = pd.read_csv('B_C_data.csv')
df.head()
```

[2]:

	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	

	…	texture_worst	perimeter_worst	area_worst	smoothness_worst	\
0	…	17.33	184.60	2019.0	0.1622	
1	…	23.41	158.80	1956.0	0.1238	
2	…	25.53	152.50	1709.0	0.1444	
3	…	26.50	98.87	567.7	0.2098	
4	…	16.67	152.20	1575.0	0.1374	

	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	\
0	0.6656	0.7119	0.2654	0.4601	
1	0.1866	0.2416	0.1860	0.2750	
2	0.4245	0.4504	0.2430	0.3613	
3	0.8663	0.6869	0.2575	0.6638	
4	0.2050	0.4000	0.1625	0.2364	

	fractal_dimension_worst	Unnamed: 32

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

[5 rows x 33 columns]
```

[3]:
```
df.shape
```

[3]:
```
(569, 33)
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 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
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

[5]:
```
df.isnull().sum()
```

[5]:
```
id                         0
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
Unnamed: 32                569
dtype: int64
```

[6]:
```
df.drop('Unnamed: 32', axis=1, inplace=True)
```

[7]:
```
df.drop('id', axis=1, inplace=True)
```

[8]:
```
df.describe().T
```

[8]:

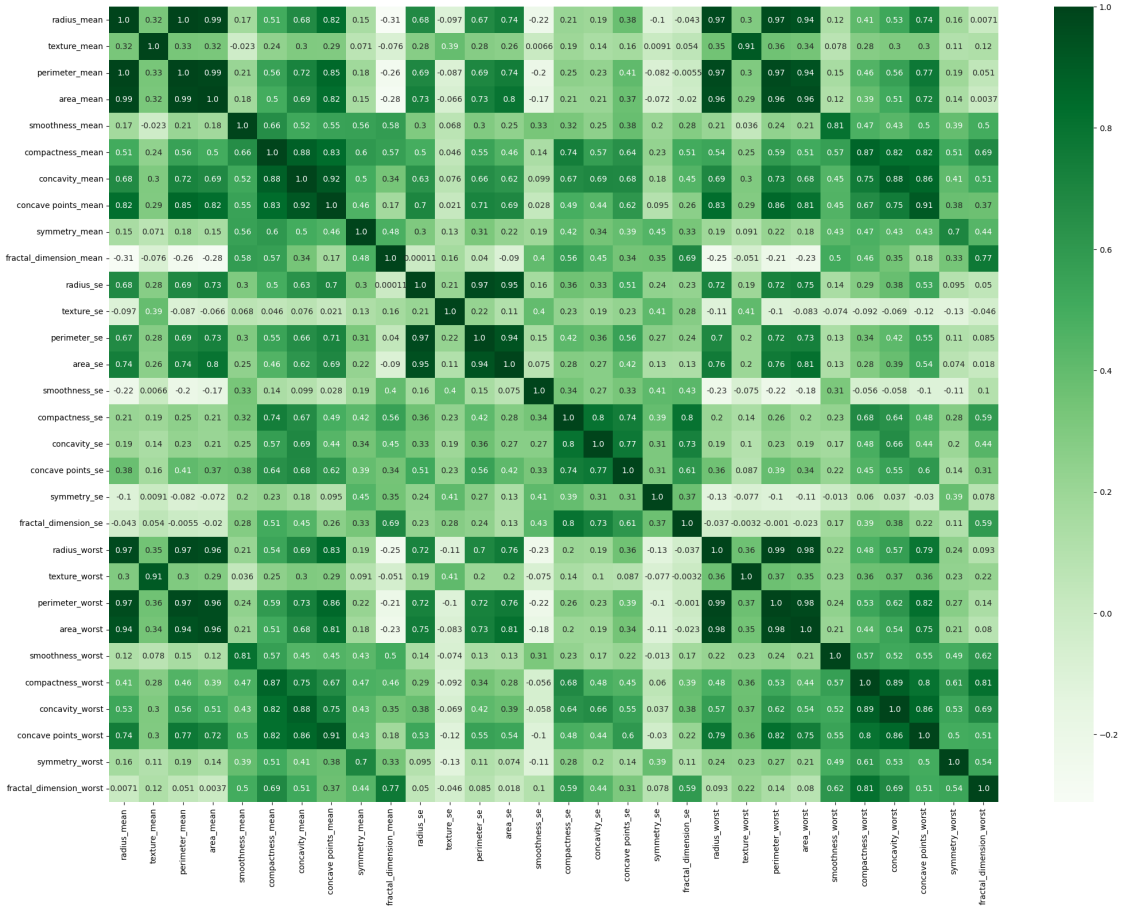
	count	mean	std	min	\
radius_mean	569.0	14.127292	3.524049	6.981000	
texture_mean	569.0	19.289649	4.301036	9.710000	
perimeter_mean	569.0	91.969033	24.298981	43.790000	
area_mean	569.0	654.889104	351.914129	143.500000	
smoothness_mean	569.0	0.096360	0.014064	0.052630	
compactness_mean	569.0	0.104341	0.052813	0.019380	
concavity_mean	569.0	0.088799	0.079720	0.000000	
concave points_mean	569.0	0.048919	0.038803	0.000000	
symmetry_mean	569.0	0.181162	0.027414	0.106000	
fractal_dimension_mean	569.0	0.062798	0.007060	0.049960	
radius_se	569.0	0.405172	0.277313	0.111500	
texture_se	569.0	1.216853	0.551648	0.360200	
perimeter_se	569.0	2.866059	2.021855	0.757000	
area_se	569.0	40.337079	45.491006	6.802000	
smoothness_se	569.0	0.007041	0.003003	0.001713	
compactness_se	569.0	0.025478	0.017908	0.002252	
concavity_se	569.0	0.031894	0.030186	0.000000	
concave points_se	569.0	0.011796	0.006170	0.000000	
symmetry_se	569.0	0.020542	0.008266	0.007882	
fractal_dimension_se	569.0	0.003795	0.002646	0.000895	
radius_worst	569.0	16.269190	4.833242	7.930000	
texture_worst	569.0	25.677223	6.146258	12.020000	
perimeter_worst	569.0	107.261213	33.602542	50.410000	
area_worst	569.0	880.583128	569.356993	185.200000	
smoothness_worst	569.0	0.132369	0.022832	0.071170	
compactness_worst	569.0	0.254265	0.157336	0.027290	
concavity_worst	569.0	0.272188	0.208624	0.000000	
concave points_worst	569.0	0.114606	0.065732	0.000000	
symmetry_worst	569.0	0.290076	0.061867	0.156500	
fractal_dimension_worst	569.0	0.083946	0.018061	0.055040	

	25%	50%	75%	max
radius_mean	11.700000	13.370000	15.780000	28.11000
texture_mean	16.170000	18.840000	21.800000	39.28000
perimeter_mean	75.170000	86.240000	104.100000	188.50000
area_mean	420.300000	551.100000	782.700000	2501.00000
smoothness_mean	0.086370	0.095870	0.105300	0.16340
compactness_mean	0.064920	0.092630	0.130400	0.34540
concavity_mean	0.029560	0.061540	0.130700	0.42680
concave points_mean	0.020310	0.033500	0.074000	0.20120
symmetry_mean	0.161900	0.179200	0.195700	0.30400
fractal_dimension_mean	0.057700	0.061540	0.066120	0.09744

radius_se	0.232400	0.324200	0.478900	2.87300
texture_se	0.833900	1.108000	1.474000	4.88500
perimeter_se	1.606000	2.287000	3.357000	21.98000
area_se	17.850000	24.530000	45.190000	542.20000
smoothness_se	0.005169	0.006380	0.008146	0.03113
compactness_se	0.013080	0.020450	0.032450	0.13540
concavity_se	0.015090	0.025890	0.042050	0.39600
concave points_se	0.007638	0.010930	0.014710	0.05279
symmetry_se	0.015160	0.018730	0.023480	0.07895
fractal_dimension_se	0.002248	0.003187	0.004558	0.02984
radius_worst	13.010000	14.970000	18.790000	36.04000
texture_worst	21.080000	25.410000	29.720000	49.54000
perimeter_worst	84.110000	97.660000	125.400000	251.20000
area_worst	515.300000	686.500000	1084.000000	4254.00000
smoothness_worst	0.116600	0.131300	0.146000	0.22260
compactness_worst	0.147200	0.211900	0.339100	1.05800
concavity_worst	0.114500	0.226700	0.382900	1.25200
concave points_worst	0.064930	0.099930	0.161400	0.29100
symmetry_worst	0.250400	0.282200	0.317900	0.66380
fractal_dimension_worst	0.071460	0.080040	0.092080	0.20750

[9]:
```
plt.figure(figsize=(25,18))
sns.heatmap(df.corr(), cmap='Greens', fmt='.2', annot=True)
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_1648\456566374.py:2: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), cmap='Greens', fmt='.2', annot=True)
```



[10]:
```
df.head()
```

[10]:

	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	…	radius_worst	texture_worst	perimeter_worst	\
0	0.2419	…	25.38	17.33	184.60	
1	0.1812	…	24.99	23.41	158.80	

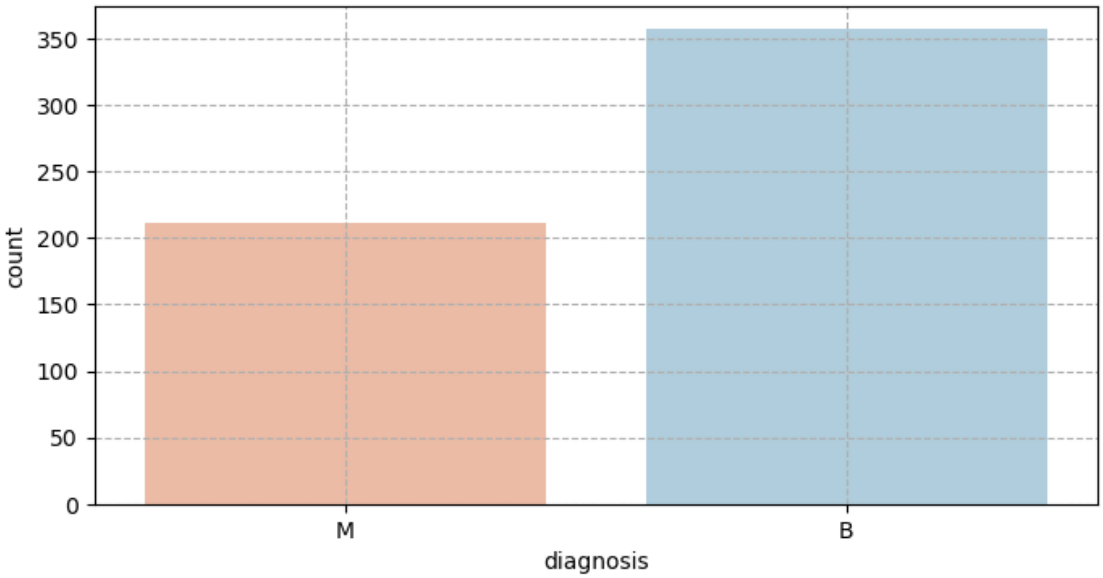
```
2         0.2069  …         23.57          25.53           152.50
3         0.2597  …         14.91          26.50            98.87
4         0.1809  …         22.54          16.67           152.20

   area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   concave points_worst  symmetry_worst  fractal_dimension_worst
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 31 columns]
```

[11]:
```
df['diagnosis'].value_counts()
```

[11]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

[12]:
```
plt.figure(figsize=(8,4))
sns.countplot(x = 'diagnosis', data=df, palette='RdBu')
plt.grid(True, linestyle='--')

B,M = df['diagnosis'].value_counts()
print('Number of Cells labeled B: ',B)
print('Number of Cells labeled M: ',M)
print('')
print('% of cells labeled B :' , round(B/len(df)*100, 2), '%')
print('% of cells labeled M :' , round(M/len(df)*100, 2), '%')
```

```
Number of Cells labeled B:  357
Number of Cells labeled M:  212

% of cells labeled B : 62.74 %
% of cells labeled M : 37.26 %
```



[13]:
```
df.columns.tolist()
```

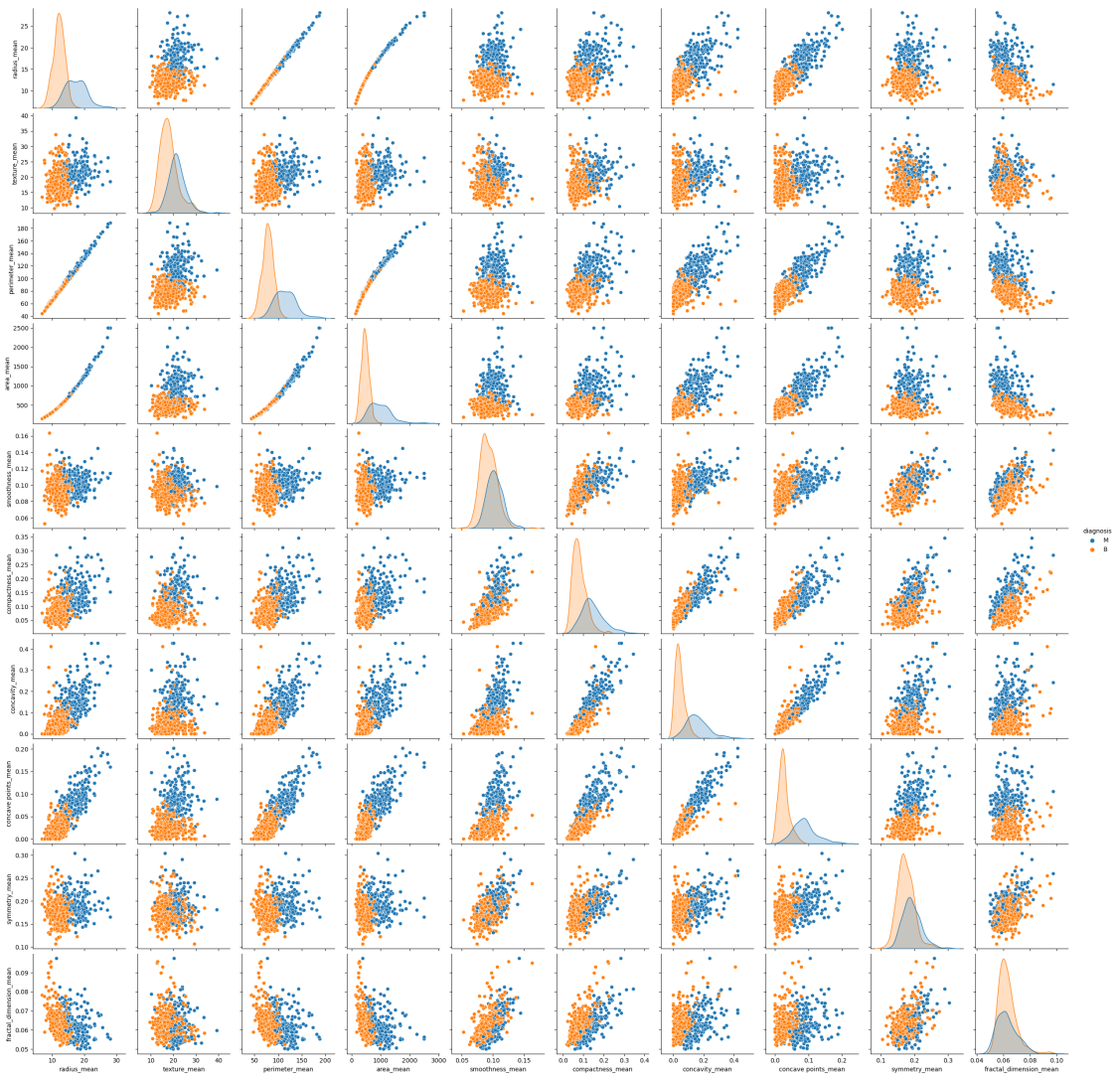
[13]:
```
['diagnosis',
 'radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'concave points_mean',
 'symmetry_mean',
 'fractal_dimension_mean',
 'radius_se',
 'texture_se',
 'perimeter_se',
 'area_se',
 'smoothness_se',
 'compactness_se',
 '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']
```

[14]:
```
cols = ['diagnosis',
 'radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
 'concave points_mean',
 'symmetry_mean',
 'fractal_dimension_mean']
```

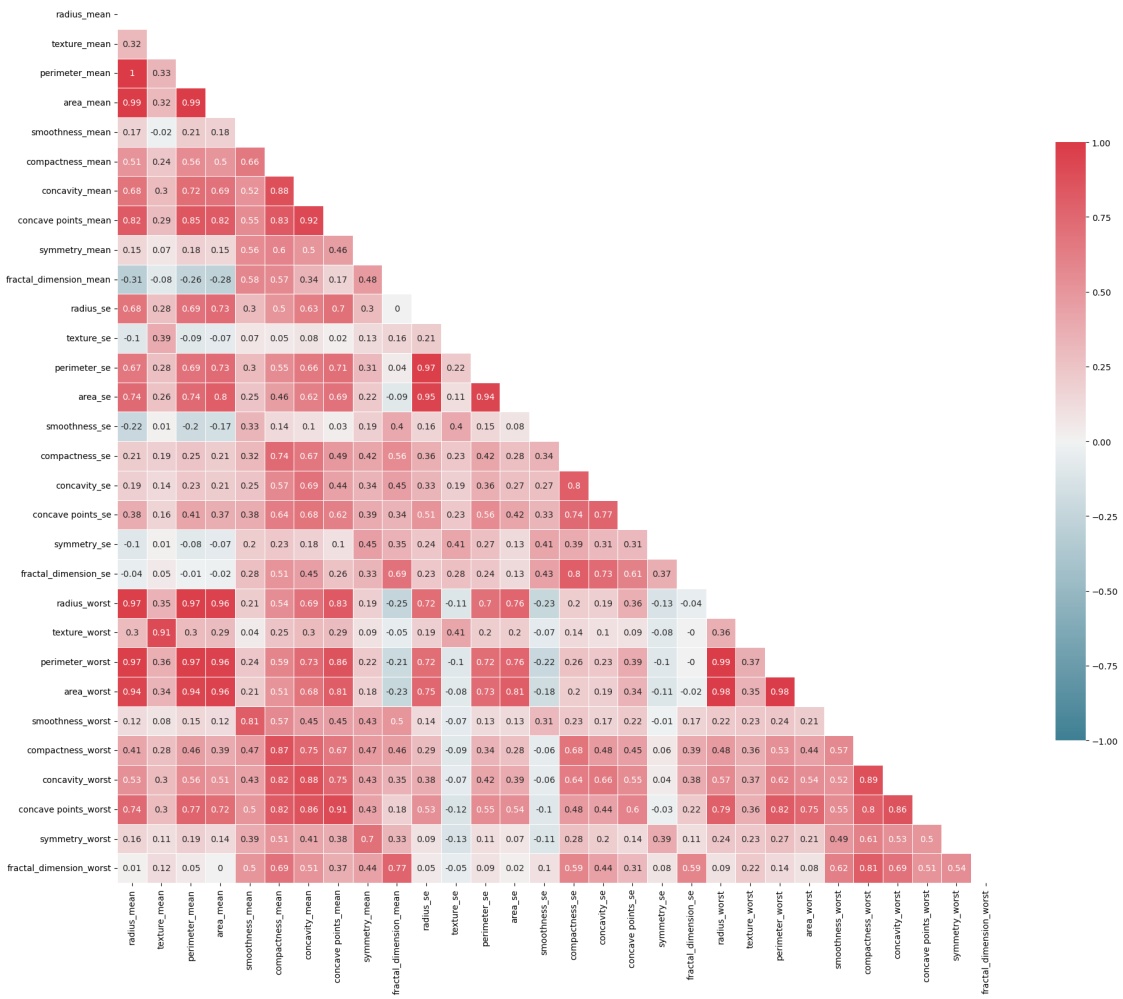
[15]:
```
sns.pairplot(data=df[cols], hue = 'diagnosis')
```

[15]:
```
<seaborn.axisgrid.PairGrid at 0x28e6e458cd0>
```



[16]:
```
corr = df.corr().round(2)

mask = np.zeros_like(corr, dtype=np.bool_)
mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize=(20, 20))

cmap = sns.diverging_palette(220, 10, as_cmap=True)

sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0,
            square=True, linewidths=.5, cbar_kws={'shrink': .5}, annot=True)
plt.tight_layout()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_1648\3898929710.py:1: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  corr = df.corr().round(2)
```



[17]:
```
df.columns.tolist()
```

[17]:
```
['diagnosis',
 'radius_mean',
 'texture_mean',
 'perimeter_mean',
 'area_mean',
 'smoothness_mean',
 'compactness_mean',
 'concavity_mean',
```

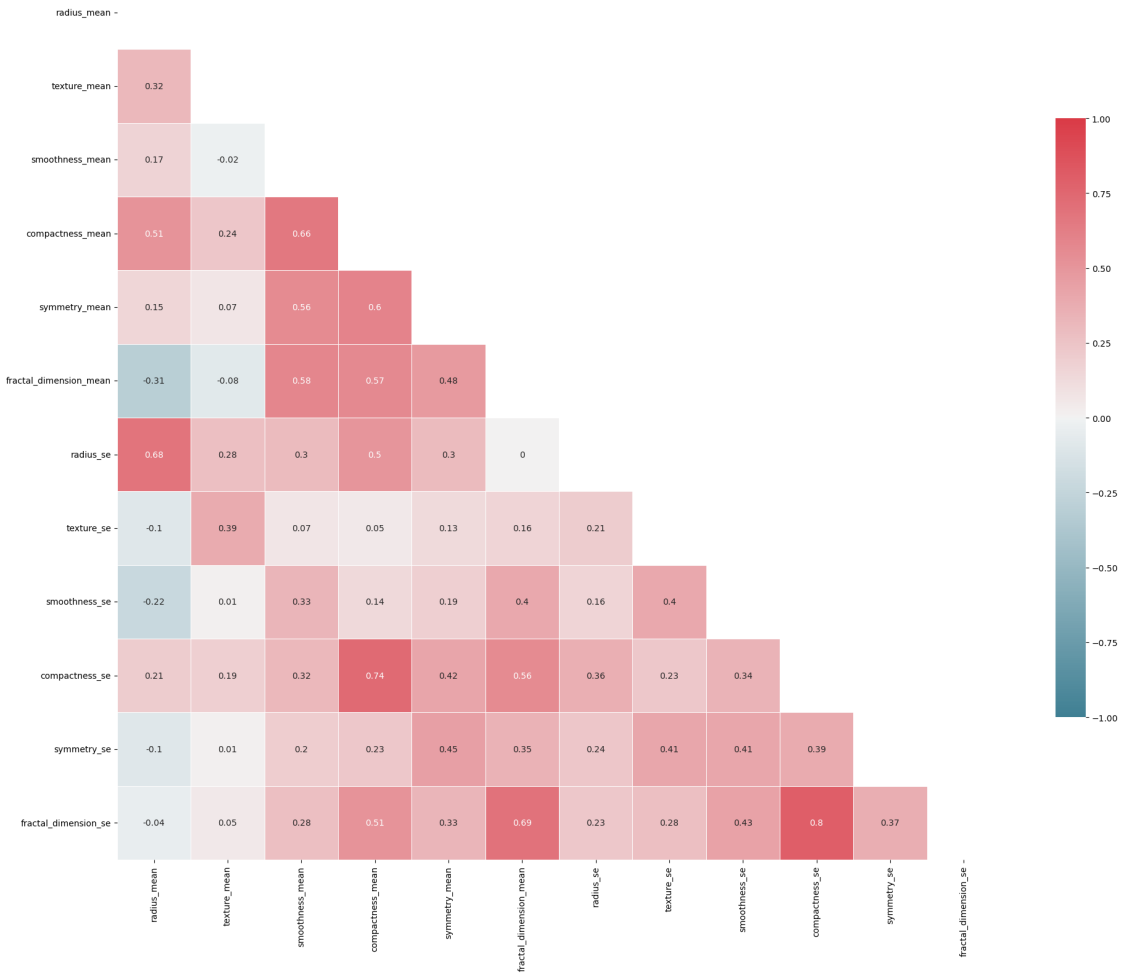
```
 'concave points_mean',
 'symmetry_mean',
 'fractal_dimension_mean',
 'radius_se',
 'texture_se',
 'perimeter_se',
 'area_se',
 'smoothness_se',
 'compactness_se',
 'concavity_se',
 'concave points_se',
 'symmetry_se',
 'fractal_dimension_se',
 'radius_worst',
 'texture_worst',
 'perimeter_worst',
 'area_worst',
 'smoothness_worst',
 'compactness_worst',
 'concavity_worst',
 'concave points_worst',
 'symmetry_worst',
 'fractal_dimension_worst']
```

[18]:
```
cols = ['radius_worst',
 'texture_worst',
 'perimeter_worst',
 'area_worst',
 'smoothness_worst',
 'compactness_worst',
 'concavity_worst',
 'concave points_worst',
 'symmetry_worst',
 'fractal_dimension_worst']
df = df.drop(cols, axis=1)

cols = ['perimeter_se',
 'area_se','perimeter_mean',
 'area_mean',]
df = df.drop(cols, axis=1)

cols = ['concavity_mean',
 'concave points_mean','concavity_se',
 'concave points_se']
df = df.drop(cols, axis=1)
df.columns
```

[18]:
```
Index(['diagnosis', 'radius_mean', 'texture_mean', 'smoothness_mean',
       'compactness_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'smoothness_se', 'compactness_se',
       'symmetry_se', 'fractal_dimension_se'],
      dtype='object')
```

[19]:
```
corr = df.corr().round(2)
mask = np.zeros_like(corr, dtype=np.bool_)
mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize = (20,20))
sns.heatmap(corr, mask=mask, cmap=cmap,vmax=1,vmin=-1, center=0,
           square=True, linewidths=.5, cbar_kws={'shrink':.5}, annot=True)
plt.tight_layout()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_1648\2220098136.py:1: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  corr = df.corr().round(2)
```



[20]:
```
from sklearn.preprocessing import StandardScaler
import sklearn.linear_model as LinearRegression
from sklearn import preprocessing
from sklearn import   neighbors
from sklearn.metrics import␣
 ↪confusion_matrix,classification_report,precision_score, accuracy_score
```

[21]:
```
x = df
y = df['diagnosis']
```

[22]:
```
x.head()
```

[22]:

	diagnosis	radius_mean	texture_mean	smoothness_mean	compactness_mean	\
0	M	17.99	10.38	0.11840	0.27760	
1	M	20.57	17.77	0.08474	0.07864	
2	M	19.69	21.25	0.10960	0.15990	

```
3         M        11.42         20.38           0.14250           0.28390
4         M        20.29         14.34           0.10030           0.13280

   symmetry_mean  fractal_dimension_mean  radius_se  texture_se  \
0         0.2419                 0.07871     1.0950      0.9053
1         0.1812                 0.05667     0.5435      0.7339
2         0.2069                 0.05999     0.7456      0.7869
3         0.2597                 0.09744     0.4956      1.1560
4         0.1809                 0.05883     0.7572      0.7813

   smoothness_se  compactness_se  symmetry_se  fractal_dimension_se
0       0.006399         0.04904      0.03003              0.006193
1       0.005225         0.01308      0.01389              0.003532
2       0.006150         0.04006      0.02250              0.004571
3       0.009110         0.07458      0.05963              0.009208
4       0.011490         0.02461      0.01756              0.005115
```

[23]:
```
y.head()
```

[23]:
```
0    M
1    M
2    M
3    M
4    M
Name: diagnosis, dtype: object
```

[24]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.3,␣
 ↪random_state=42)
```

[25]:
```
cols = df.columns.drop('diagnosis')
formula = 'diagnosis ~ '+ '+'.join(cols)
print(formula,'\n')
```

```
diagnosis ~ radius_mean+texture_mean+smoothness_mean+compactness_mean+symmetry_m
ean+fractal_dimension_mean+radius_se+texture_se+smoothness_se+compactness_se+sym
metry_se+fractal_dimension_se
```

[26]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

[27]:
```
model = smf.glm(formula=formula, data=x_train, family = sm.families.Binomial())
logistic_fit = model.fit()
print(logistic_fit.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
```

```
============
Dep. Variable:     ['diagnosis[B]', 'diagnosis[M]']   No. Observations:
398
Model:                                         GLM   Df Residuals:
385
Model Family:                             Binomial   Df Model:
12
Link Function:                               Logit   Scale:
1.0000
Method:                                       IRLS   Log-Likelihood:
-61.139
Date:                             Sat, 27 Jan 2024   Deviance:
122.28
Time:                                     22:01:34   Pearson chi2:
175.
No. Iterations:                                8   Pseudo R-squ. (CS):
0.6377
Covariance Type:                          nonrobust
==============================================================================
==========
                             coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----------
Intercept                 41.7089     10.191      4.093      0.000      21.736
61.682
radius_mean               -1.0805      0.268     -4.027      0.000      -1.606
-0.555
texture_mean              -0.4381      0.084     -5.207      0.000      -0.603
-0.273
smoothness_mean          -64.2673     38.115     -1.686      0.092    -138.971
10.437
compactness_mean         -12.8072     19.607     -0.653      0.514     -51.236
25.621
symmetry_mean            -26.1639     14.675     -1.783      0.075     -54.926
2.599
fractal_dimension_mean  -101.0262    117.577     -0.859      0.390    -331.472
129.420
radius_se                 -7.4129      2.747     -2.699      0.007     -12.797
-2.029
texture_se                 1.3190      0.772      1.708      0.088      -0.195
2.833
smoothness_se             41.2550    105.608      0.391      0.696    -165.734
248.244
compactness_se           -19.7195     39.170     -0.503      0.615     -96.491
57.052
symmetry_se               55.9431     46.811      1.195      0.232     -35.805
147.691
```

```
fractal_dimension_se     468.8827    304.065      1.542      0.123    -127.073
1064.839
================================================================================
==========
```

[28]:
```
prediction = logistic_fit.predict(x_test)
prediction[1:6]
```

[28]:
```
70     2.684854e-04
131    2.655733e-02
431    9.901461e-01
540    9.991130e-01
567    1.682200e-08
dtype: float64
```

[29]:
```
prediction_nominal = ['M' if x < 0.5 else "B" for x in prediction]
prediction_nominal[1:6]
```

[29]:
```
['M', 'M', 'B', 'B', 'M']
```

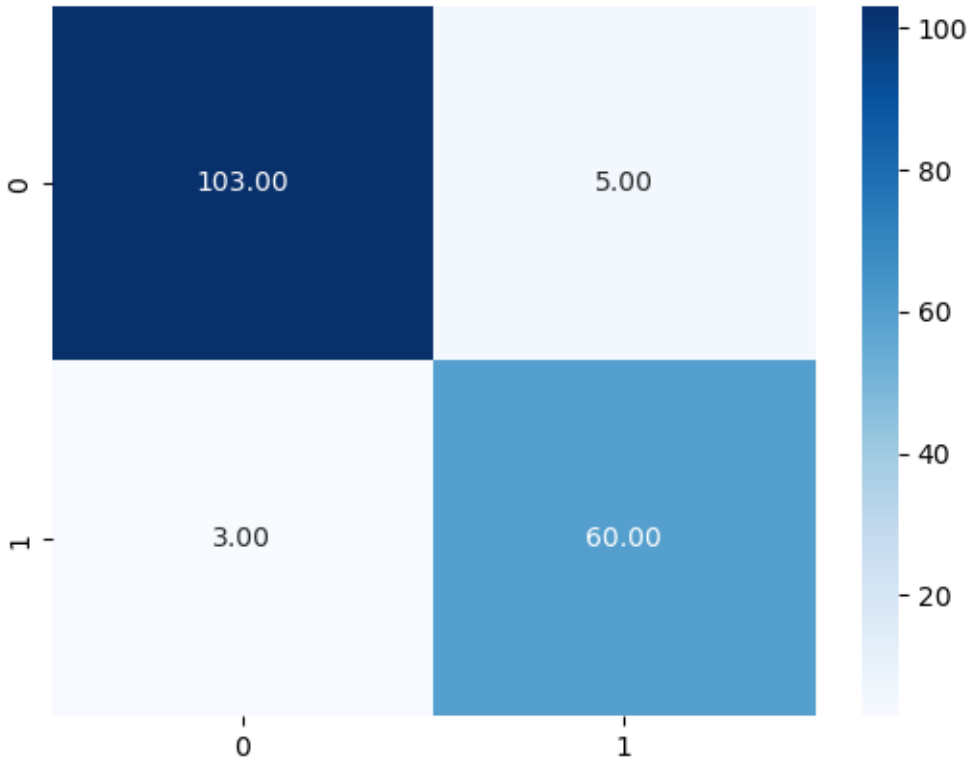
[30]:
```
print(classification_report(y_test, prediction_nominal, digits=3))
cfm = confusion_matrix(y_test, prediction_nominal)
acs = accuracy_score(y_test, prediction_nominal)*100
true_negative = cfm[0][0]
false_positive = cfm[0][1]
false_negative = cfm[1][0]
true_positive = cfm[1][1]

print('Confusion Matrix: \n ', cfm, '\n')
print('Accuracy Score: \n ', acs, '\n')
print('True Negative',true_negative)
print('False Positive',false_positive)
print('False Negative',false_negative)
print('True Positive',true_positive)
print('Correct Prediction',
     round((true_negative + true_positive)/ len(prediction_nominal)*100,1),'%')
```

```
              precision    recall  f1-score   support

           B      0.972     0.954     0.963       108
           M      0.923     0.952     0.938        63

    accuracy                          0.953       171
   macro avg      0.947     0.953     0.950       171
weighted avg      0.954     0.953     0.953       171

Confusion Matrix:
  [[103   5]
```

```
 [  3  60]]

Accuracy Score:
  95.32163742690058

True Negative 103
False Positive 5
False Negative 3
True Positive 60
Correct Prediction 95.3 %
```

[31]:
```
sns.heatmap(cfm, annot=True, fmt='.2f', cmap='Blues')
```

[31]: <Axes: >



[ ]: