2.2.Feature-Selection-Log-Reg

May 15, 2023

[91]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
%matplotlib inline
from sklearn.model_selection import train_test_split
from scipy.stats import zscore
```

[92]:
```
data=pd.read_csv('wdbc.data',header=None)
```

data.head()

[93]:
```
headers=['id','diagnosis','mean_radius','mean_texture','mean_perimeter','mean_area','mean_smoo
 ↪points','mean_symmetry','mean_fractal␣
 ↪dimension','SE_radius','SE_texture','SE_perimeter','SE_area','SE_smoothness','SE_compactnes
 ↪points','SE_symmetry','SE_fractal␣
 ↪dimension','worst_radius','worst_texture','worst_perimeter','worst_area','worst_smoothness'
 ↪points','worst_symmetry','worst_fractal dimension']
```

[94]:
```
data.to_csv('labeledData.csv',header=headers,index=False)
```

[95]:
```
data=pd.read_csv('labeledData.csv')
data.head()
```

[95]:

	id	diagnosis	mean_radius	mean_texture	mean_perimeter	mean_area	\
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	

	mean_smoothness	mean_compactness	mean_concavity	mean_concave points	
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

   …  worst_radius  worst_texture  worst_perimeter  worst_area
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

   worst_smoothness  worst_compactness  worst_concavity  worst_concave points
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

   worst_symmetry  worst_fractal dimension
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]
```

[96]:
```
data.shape
```

[96]:
```
(569, 32)
```

[97]:
```
data.isna().sum()
```

[97]:
```
id                         0
diagnosis                  0
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
SE_radius                  0
SE_texture                 0
SE_perimeter               0
SE_area                    0
```

```
SE_smoothness              0
SE_compactness             0
SE_concavity               0
SE_concave points          0
SE_symmetry                0
SE_fractal dimension       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
dtype: int64
```

[98]:
```
data['diagnosis'].value_counts()
```

[98]:
```
diagnosis
B    357
M    212
Name: count, dtype: int64
```

[99]:
```
data.dtypes
```

[99]:
```
id                         int64
diagnosis                 object
mean_radius              float64
mean_texture             float64
mean_perimeter           float64
mean_area                float64
mean_smoothness          float64
mean_compactness         float64
mean_concavity           float64
mean_concave points      float64
mean_symmetry            float64
mean_fractal dimension   float64
SE_radius                float64
SE_texture               float64
SE_perimeter             float64
SE_area                  float64
SE_smoothness            float64
SE_compactness           float64
SE_concavity             float64
SE_concave points        float64
```

```
SE_symmetry                float64
SE_fractal dimension       float64
worst_radius               float64
worst_texture              float64
worst_perimeter            float64
worst_area                 float64
worst_smoothness           float64
worst_compactness          float64
worst_concavity            float64
worst_concave points       float64
worst_symmetry             float64
worst_fractal dimension    float64
dtype: object
```

[100]:
```
list=['id','diagnosis']
y=data.diagnosis
x=data.drop(list,axis=1)
x.head()
```

[100]:

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	\
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	

	mean_compactness	mean_concavity	mean_concave points	mean_symmetry	\
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	

	mean_fractal dimension	…	worst_radius	worst_texture	worst_perimeter	\
0	0.07871	…	25.38	17.33	184.60	\
1	0.05667	…	24.99	23.41	158.80	
2	0.05999	…	23.57	25.53	152.50	
3	0.09744	…	14.91	26.50	98.87	
4	0.05883	…	22.54	16.67	152.20	

	worst_area	worst_smoothness	worst_compactness	worst_concavity	\
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	

```
   worst_concave points  worst_symmetry  worst_fractal dimension
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 30 columns]
```

[101]:
```
x.describe()
```

[101]:
```
       mean_radius  mean_texture  mean_perimeter    mean_area
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

       mean_smoothness  mean_compactness  mean_concavity  mean_concave points
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

       mean_symmetry  mean_fractal dimension  …  worst_radius
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

       worst_texture  worst_perimeter   worst_area  worst_smoothness
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

       worst_compactness  worst_concavity  worst_concave points
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

       worst_symmetry  worst_fractal dimension
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]
```

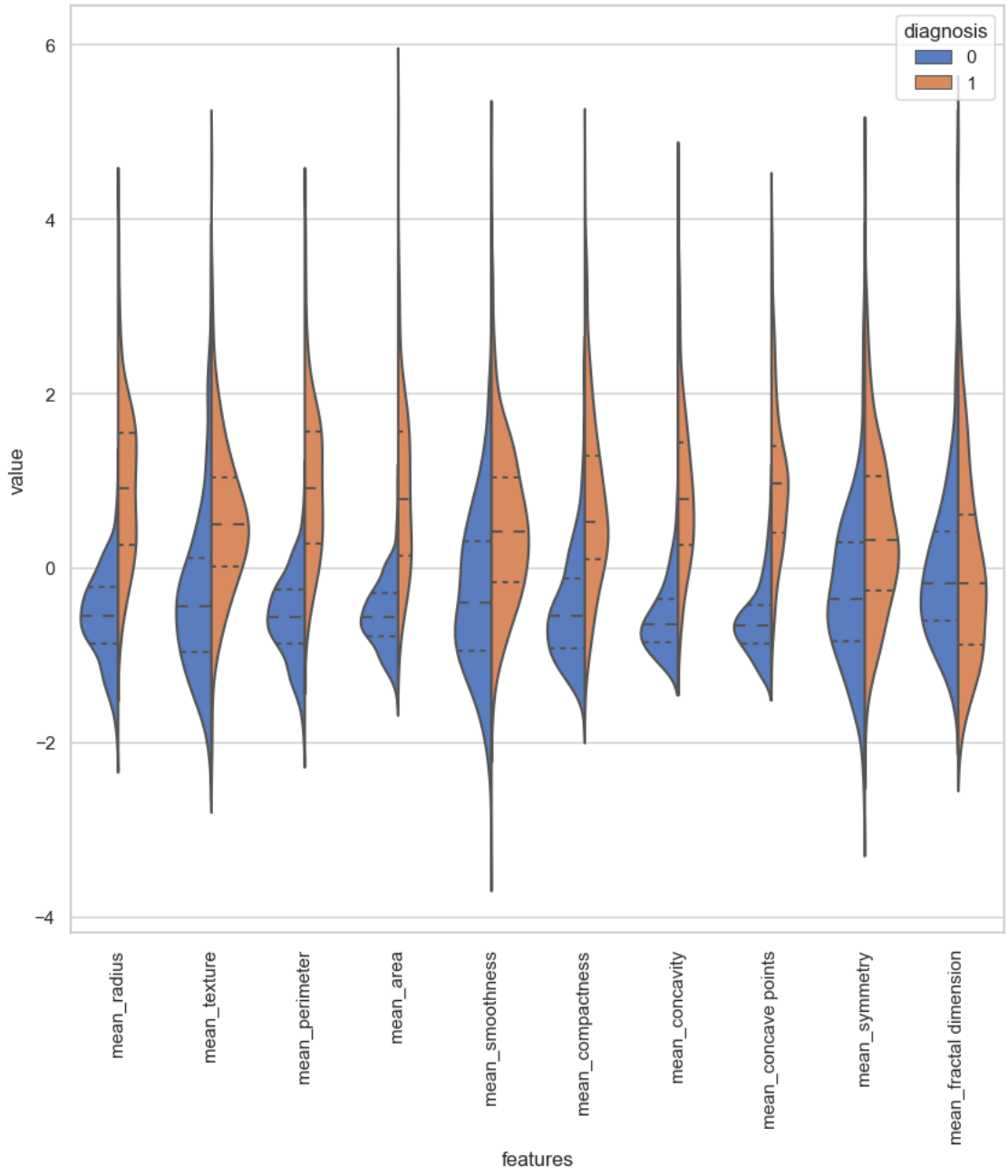
[102]:
```
def diag(z):
    if z== 'M':
        return 1
    else:
        return 0

z=data['diagnosis'].apply(diag)
data.diagnosis=z
```

[103]:
```
y=data.diagnosis
data=x
data_std=(data-data.mean())/(data.std())
```

[104]:
```
data=pd.concat([y,data_std.iloc[:,0:10]],axis=1)
data=pd.melt(data,id_vars='diagnosis',var_name='features',value_name='value')
plt.figure(figsize=(10,10))
sns.
 ↪violinplot(x='features',y='value',hue='diagnosis',data=data,split=True,inner='quart')
plt.xticks(rotation=90)
```

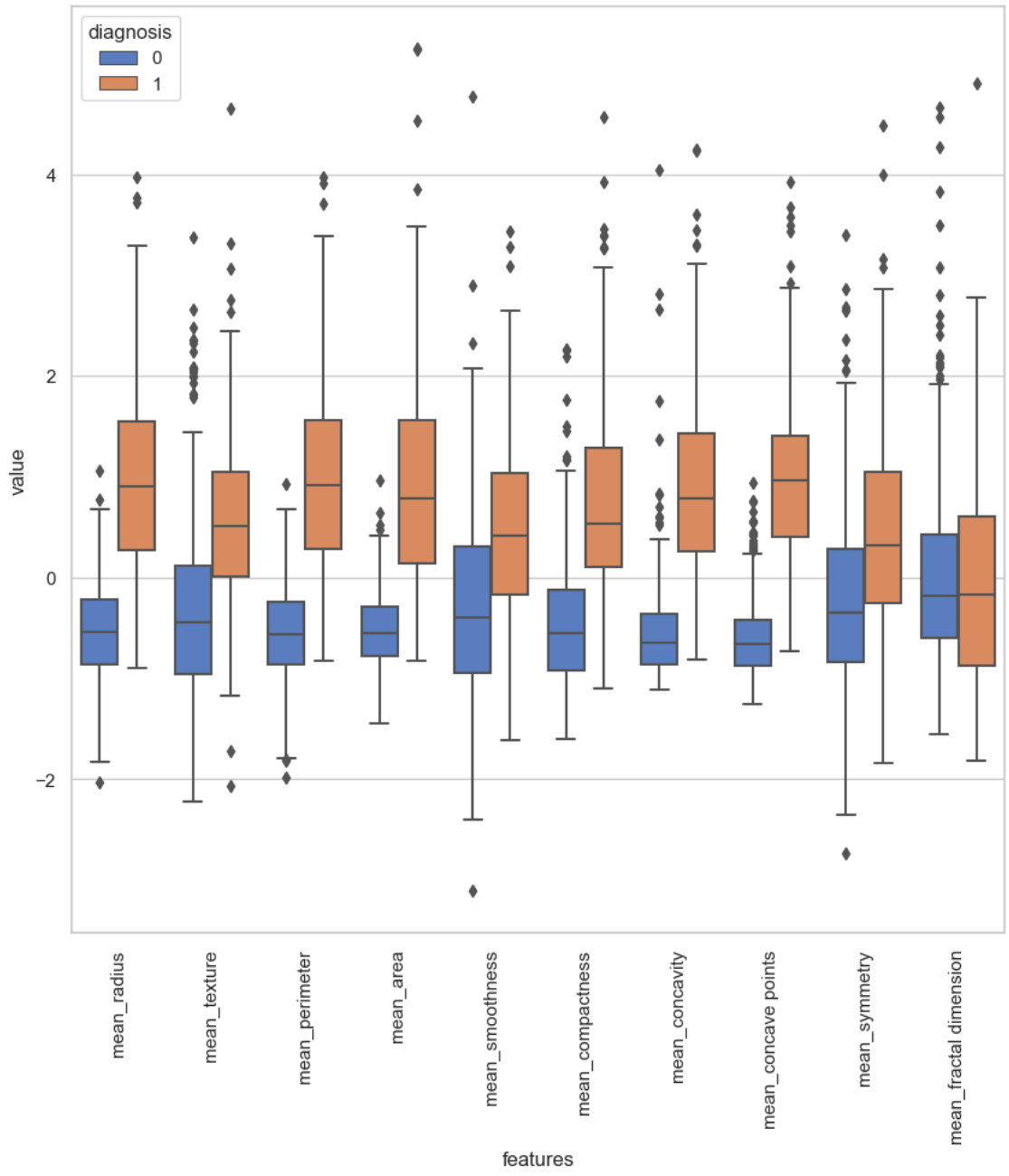
```
[104]: (array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
        [Text(0, 0, 'mean_radius'),
         Text(1, 0, 'mean_texture'),
         Text(2, 0, 'mean_perimeter'),
         Text(3, 0, 'mean_area'),
         Text(4, 0, 'mean_smoothness'),
         Text(5, 0, 'mean_compactness'),
         Text(6, 0, 'mean_concavity'),
         Text(7, 0, 'mean_concave points'),
         Text(8, 0, 'mean_symmetry'),
         Text(9, 0, 'mean_fractal dimension')])
```



[105]:
```
plt.figure(figsize=(10,10))
sns.boxplot(x="features",y="value",hue='diagnosis',data=data)
plt.xticks(rotation=90)
```

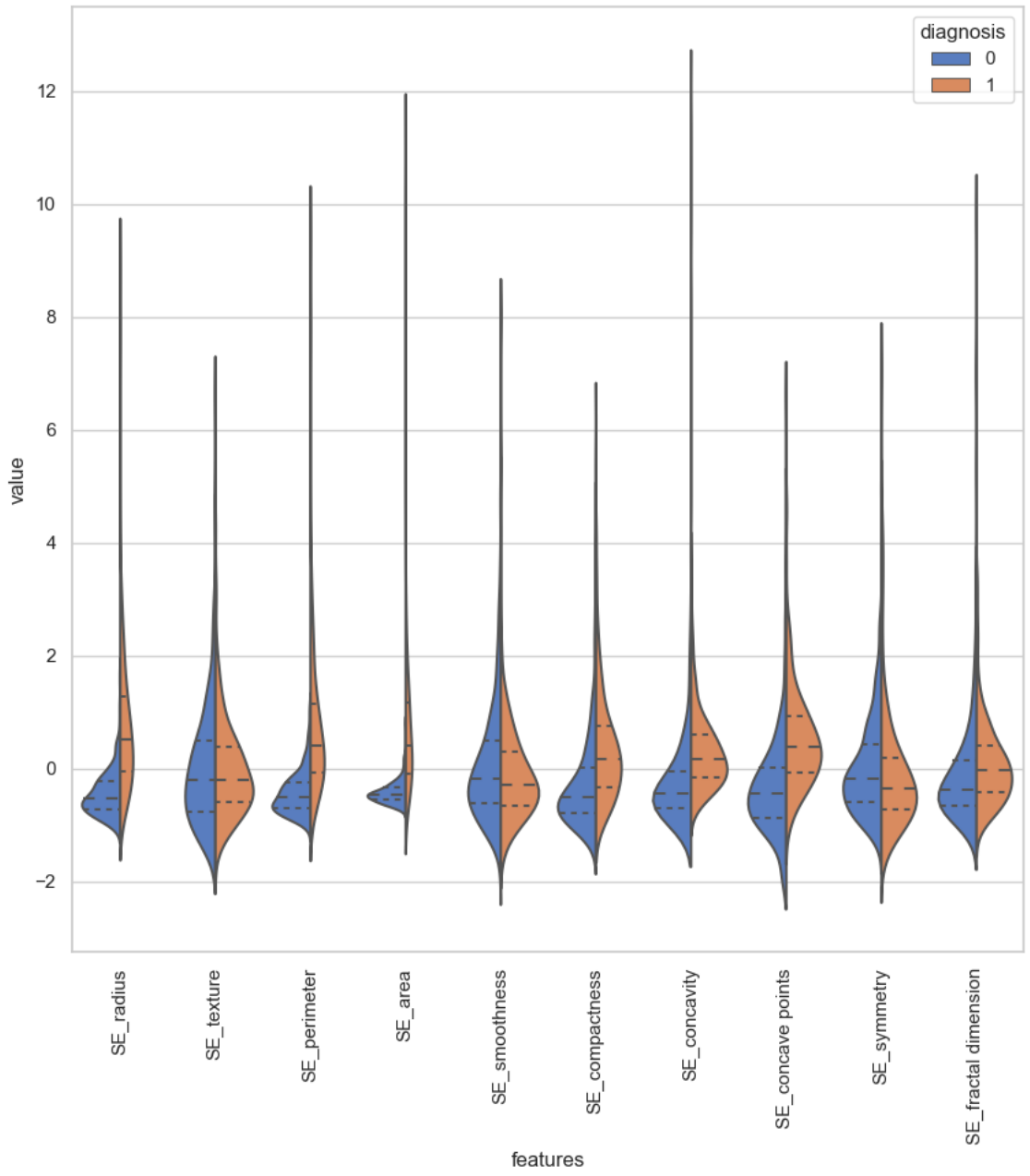
[105]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'mean_radius'),
  Text(1, 0, 'mean_texture'),
  Text(2, 0, 'mean_perimeter'),
```

```
Text(3, 0, 'mean_area'),
Text(4, 0, 'mean_smoothness'),
Text(5, 0, 'mean_compactness'),
Text(6, 0, 'mean_concavity'),
Text(7, 0, 'mean_concave points'),
Text(8, 0, 'mean_symmetry'),
Text(9, 0, 'mean_fractal dimension')])
```



[106]:
```
data = pd.concat([y,data_std.iloc[:,10:20]],axis=1)
data = pd.melt(data,id_vars="diagnosis",
                    var_name="features",
                    value_name='value')
plt.figure(figsize=(10,10))
sns.violinplot(x="features", y="value", hue="diagnosis", data=data,split=True,␣
 ↪inner="quart")
plt.xticks(rotation=90)
```

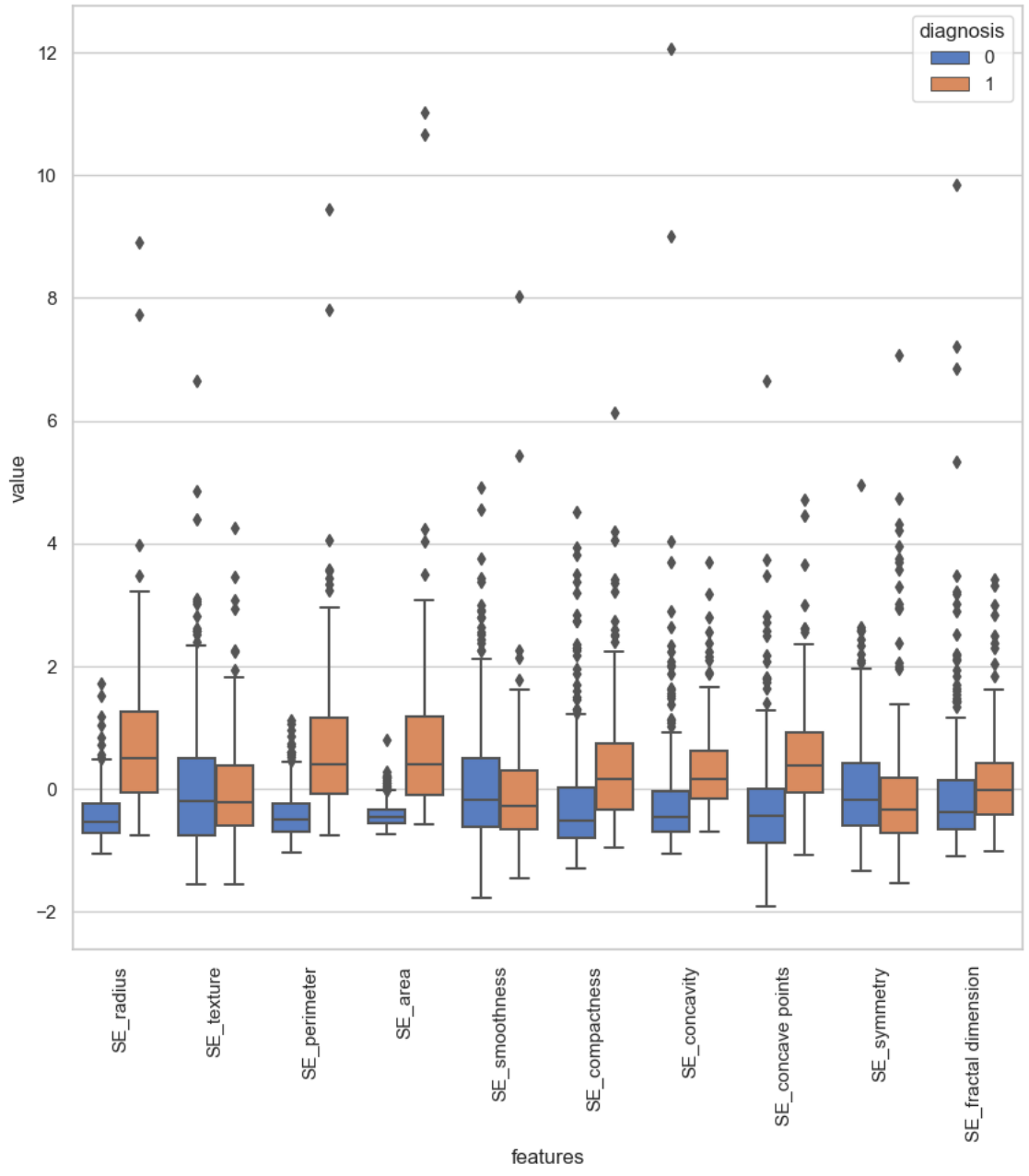
[106]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'SE_radius'),
  Text(1, 0, 'SE_texture'),
  Text(2, 0, 'SE_perimeter'),
  Text(3, 0, 'SE_area'),
  Text(4, 0, 'SE_smoothness'),
  Text(5, 0, 'SE_compactness'),
  Text(6, 0, 'SE_concavity'),
  Text(7, 0, 'SE_concave points'),
  Text(8, 0, 'SE_symmetry'),
  Text(9, 0, 'SE_fractal dimension')])
```



[107]:
```
plt.figure(figsize=(10,10))
sns.boxplot(x="features",y="value",hue='diagnosis',data=data)
plt.xticks(rotation=90)
```

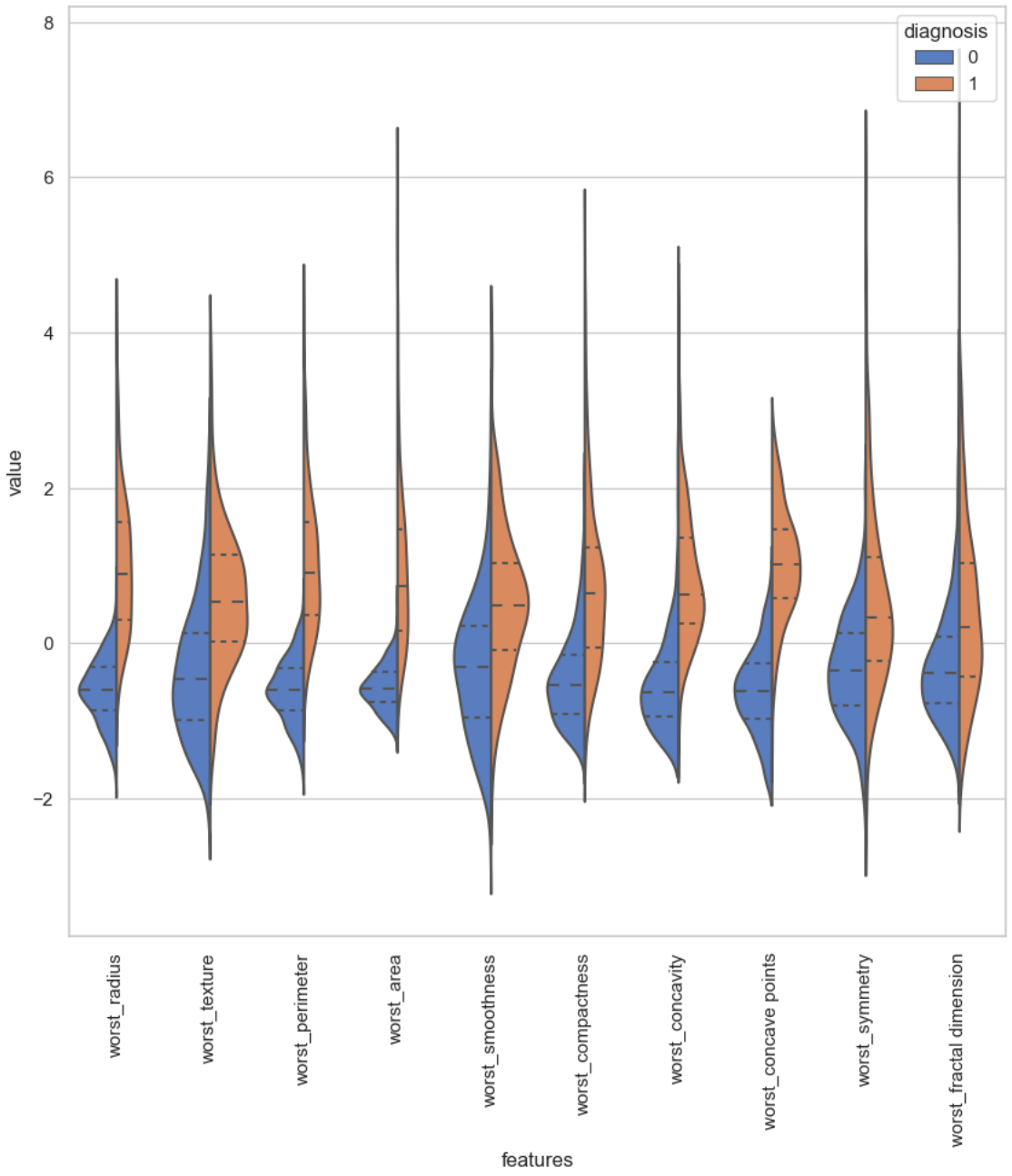
[107]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'SE_radius'),
  Text(1, 0, 'SE_texture'),
  Text(2, 0, 'SE_perimeter'),
```

```
Text(3, 0, 'SE_area'),
Text(4, 0, 'SE_smoothness'),
Text(5, 0, 'SE_compactness'),
Text(6, 0, 'SE_concavity'),
Text(7, 0, 'SE_concave points'),
Text(8, 0, 'SE_symmetry'),
Text(9, 0, 'SE_fractal dimension')])
```



[108]:
```
data = pd.concat([y,data_std.iloc[:,20:31]],axis=1)
data = pd.melt(data,id_vars="diagnosis",
                    var_name="features",
                    value_name='value')
plt.figure(figsize=(10,10))
sns.violinplot(x="features", y="value", hue="diagnosis", data=data,split=True,␣
 ↪inner="quart")
plt.xticks(rotation=90)
```

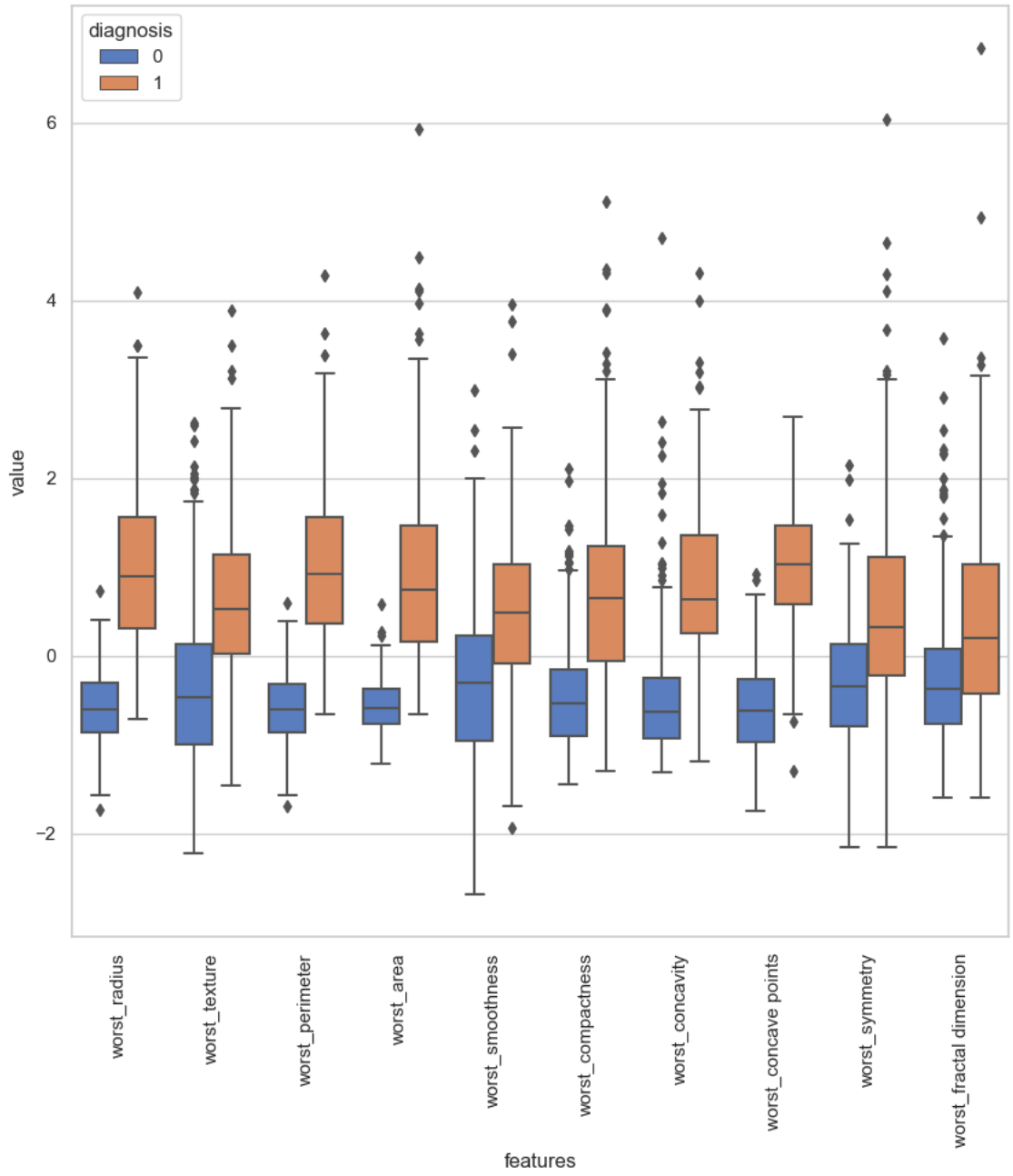
[108]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'worst_radius'),
  Text(1, 0, 'worst_texture'),
  Text(2, 0, 'worst_perimeter'),
  Text(3, 0, 'worst_area'),
  Text(4, 0, 'worst_smoothness'),
  Text(5, 0, 'worst_compactness'),
  Text(6, 0, 'worst_concavity'),
  Text(7, 0, 'worst_concave points'),
  Text(8, 0, 'worst_symmetry'),
  Text(9, 0, 'worst_fractal dimension')])
```



[109]:
```
plt.figure(figsize=(10,10))
sns.boxplot(x="features",y="value",hue='diagnosis',data=data)
plt.xticks(rotation=90)
```

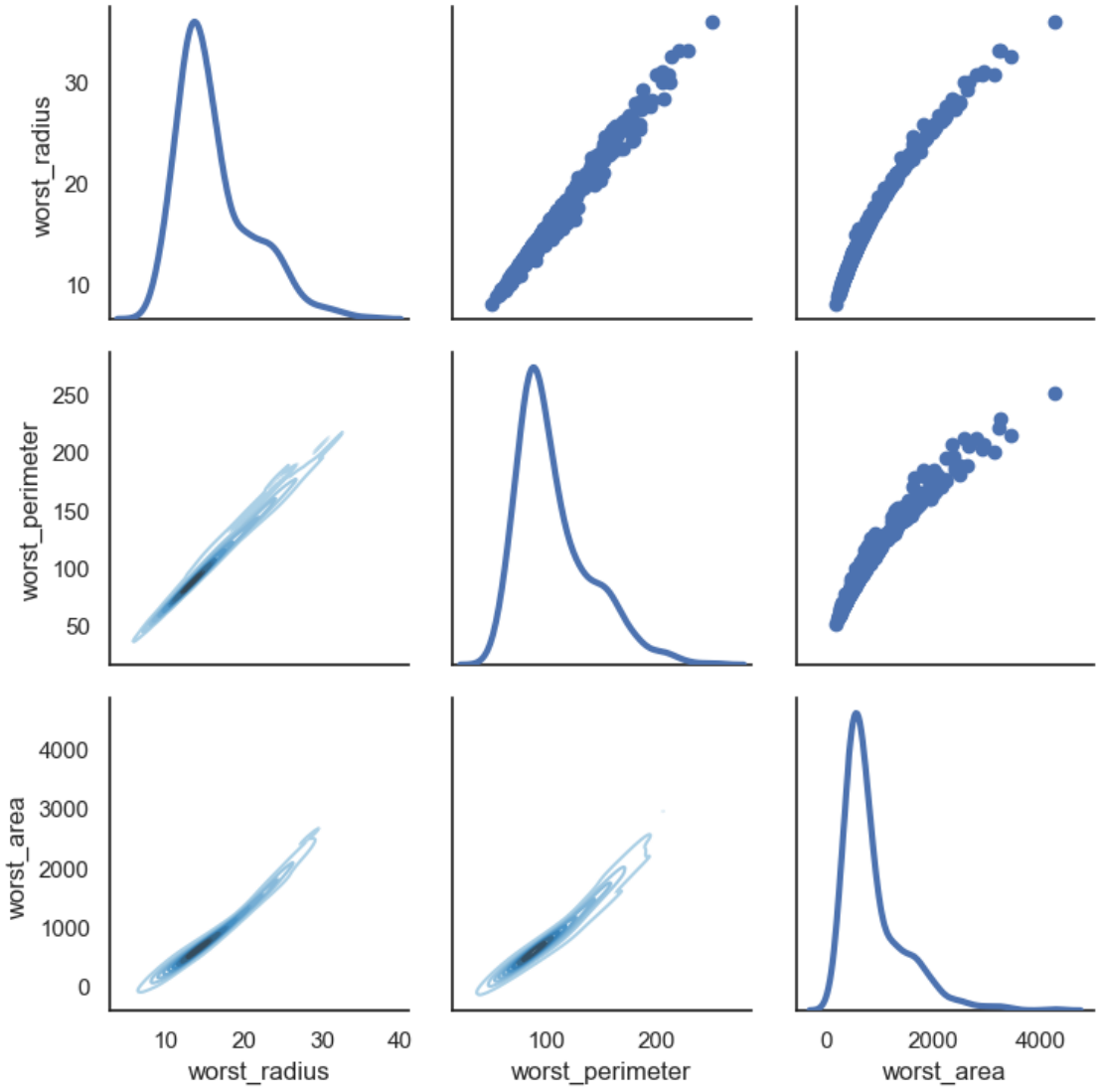
[109]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'worst_radius'),
  Text(1, 0, 'worst_texture'),
  Text(2, 0, 'worst_perimeter'),
```

```
Text(3, 0, 'worst_area'),
Text(4, 0, 'worst_smoothness'),
Text(5, 0, 'worst_compactness'),
Text(6, 0, 'worst_concavity'),
Text(7, 0, 'worst_concave points'),
Text(8, 0, 'worst_symmetry'),
Text(9, 0, 'worst_fractal dimension')])
```



[110]:
```
sns.set(style='white')
df=x.loc[:,['worst_radius','worst_perimeter','worst_area']]
g=sns.PairGrid(df,diag_sharey=False)
g.map_lower(sns.kdeplot,cmap='Blues_d')
g.map_upper(plt.scatter)
g.map_diag(sns.kdeplot,lw=3)
```

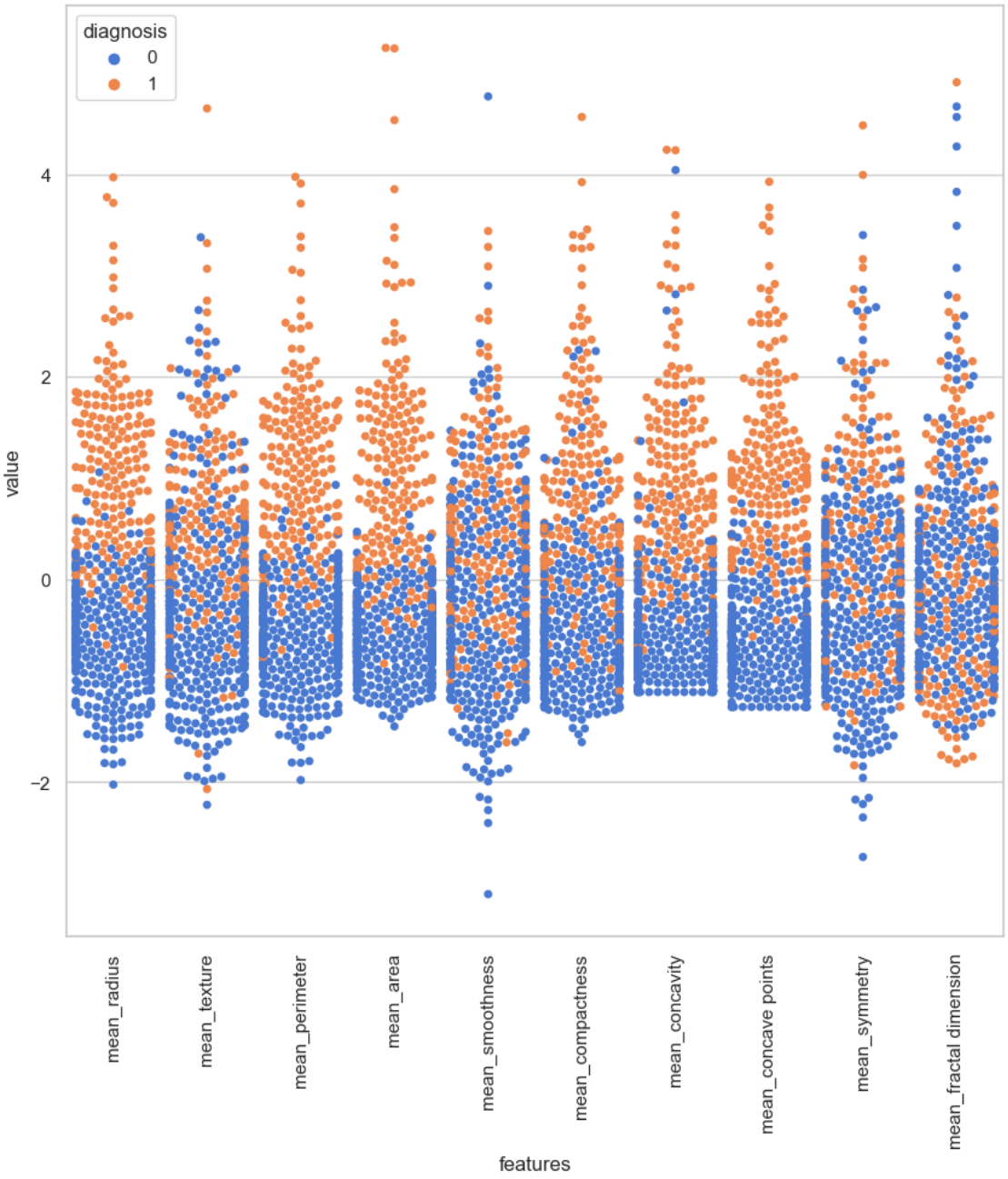
[110]: <seaborn.axisgrid.PairGrid at 0x2b34b244850>



[111]:
```
sns.set(style='whitegrid',palette='muted')
diag=y
data=x
data_n=(data-data.mean())/(data.std())
```

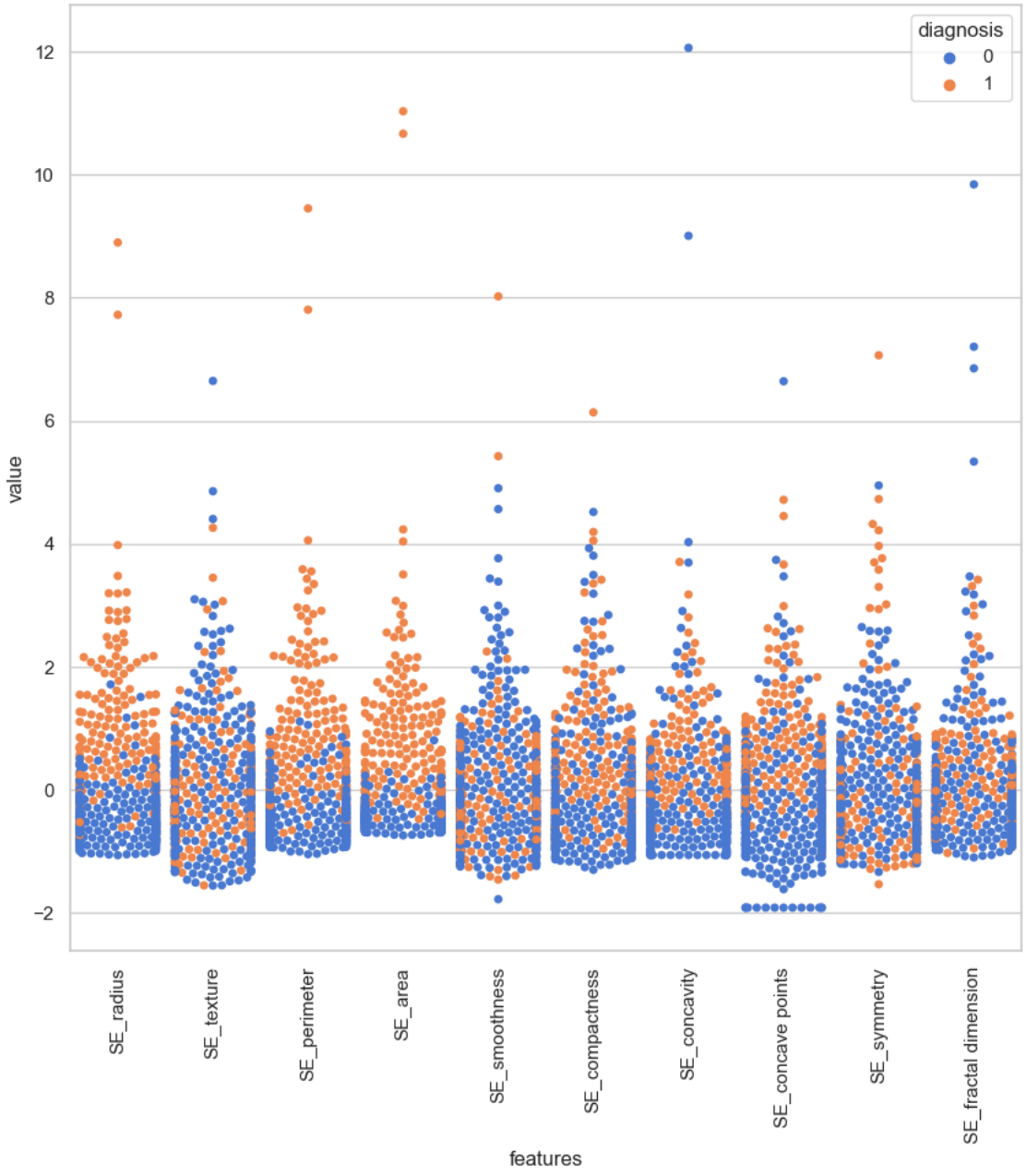
```
data=pd.concat([y,data_n.iloc[:,0:10]],axis=1)
data=pd.melt(data,id_vars='diagnosis',var_name='features',value_name='value')
plt.figure(figsize=(10,10))
tic=time.time()
sns.swarmplot(x='features',y='value',hue='diagnosis',data=data)
plt.xticks(rotation=90)
```

[111]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
 [Text(0, 0, 'mean_radius'),
  Text(1, 0, 'mean_texture'),
  Text(2, 0, 'mean_perimeter'),
  Text(3, 0, 'mean_area'),
  Text(4, 0, 'mean_smoothness'),
  Text(5, 0, 'mean_compactness'),
  Text(6, 0, 'mean_concavity'),
  Text(7, 0, 'mean_concave points'),
  Text(8, 0, 'mean_symmetry'),
  Text(9, 0, 'mean_fractal dimension')])
```



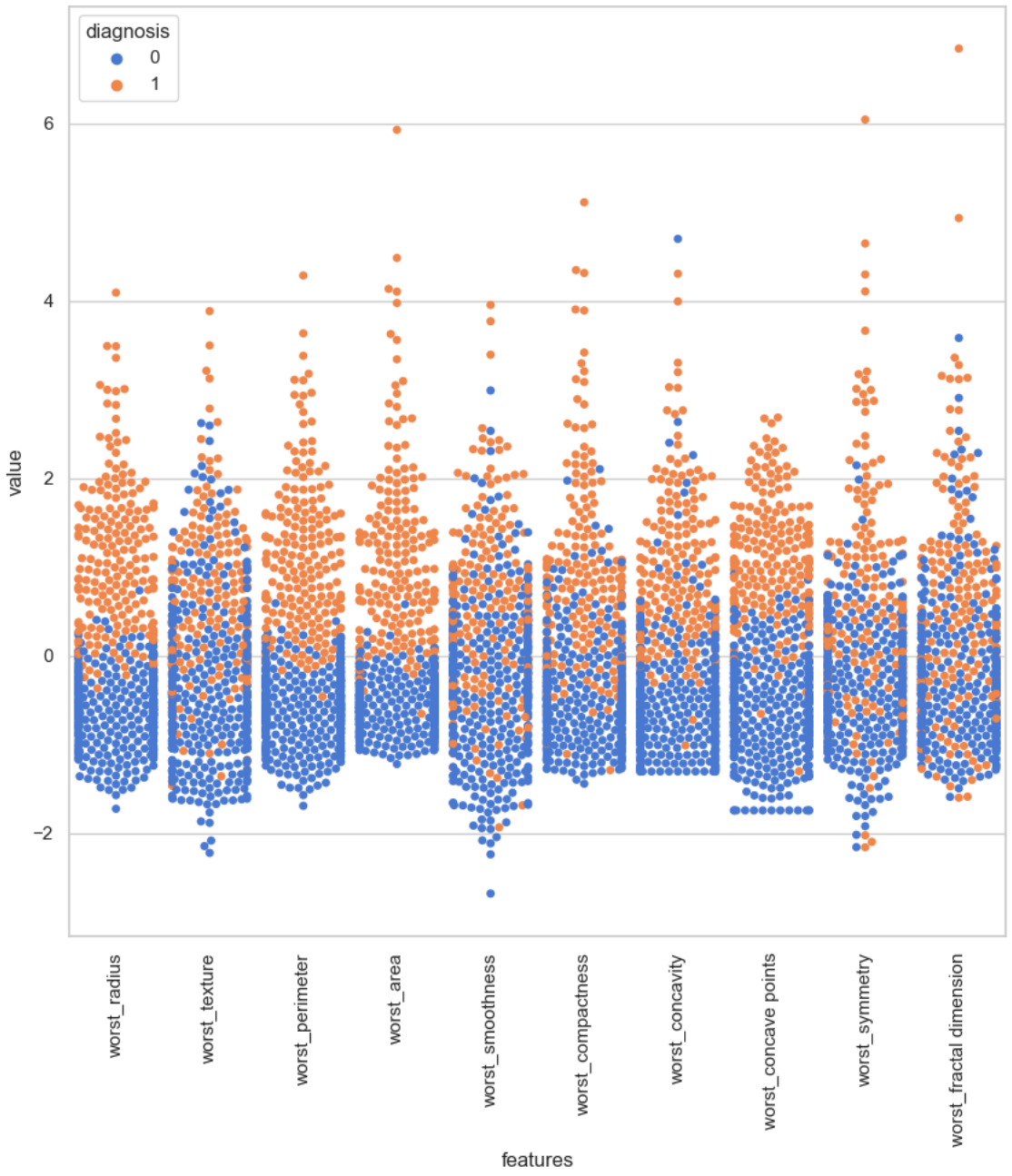
[112]:
```
data=pd.concat([y,data_n.iloc[:,10:20]],axis=1)
data=pd.melt(data,id_vars='diagnosis',var_name='features',value_name='value')
plt.figure(figsize=(10,10))
tic=time.time()
sns.swarmplot(x='features',y='value',hue='diagnosis',data=data)
plt.xticks(rotation=90)
```

[112]: ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
 [Text(0, 0, 'SE_radius'),
 Text(1, 0, 'SE_texture'),
 Text(2, 0, 'SE_perimeter'),
 Text(3, 0, 'SE_area'),
 Text(4, 0, 'SE_smoothness'),
 Text(5, 0, 'SE_compactness'),
 Text(6, 0, 'SE_concavity'),
 Text(7, 0, 'SE_concave points'),
 Text(8, 0, 'SE_symmetry'),
 Text(9, 0, 'SE_fractal dimension')])



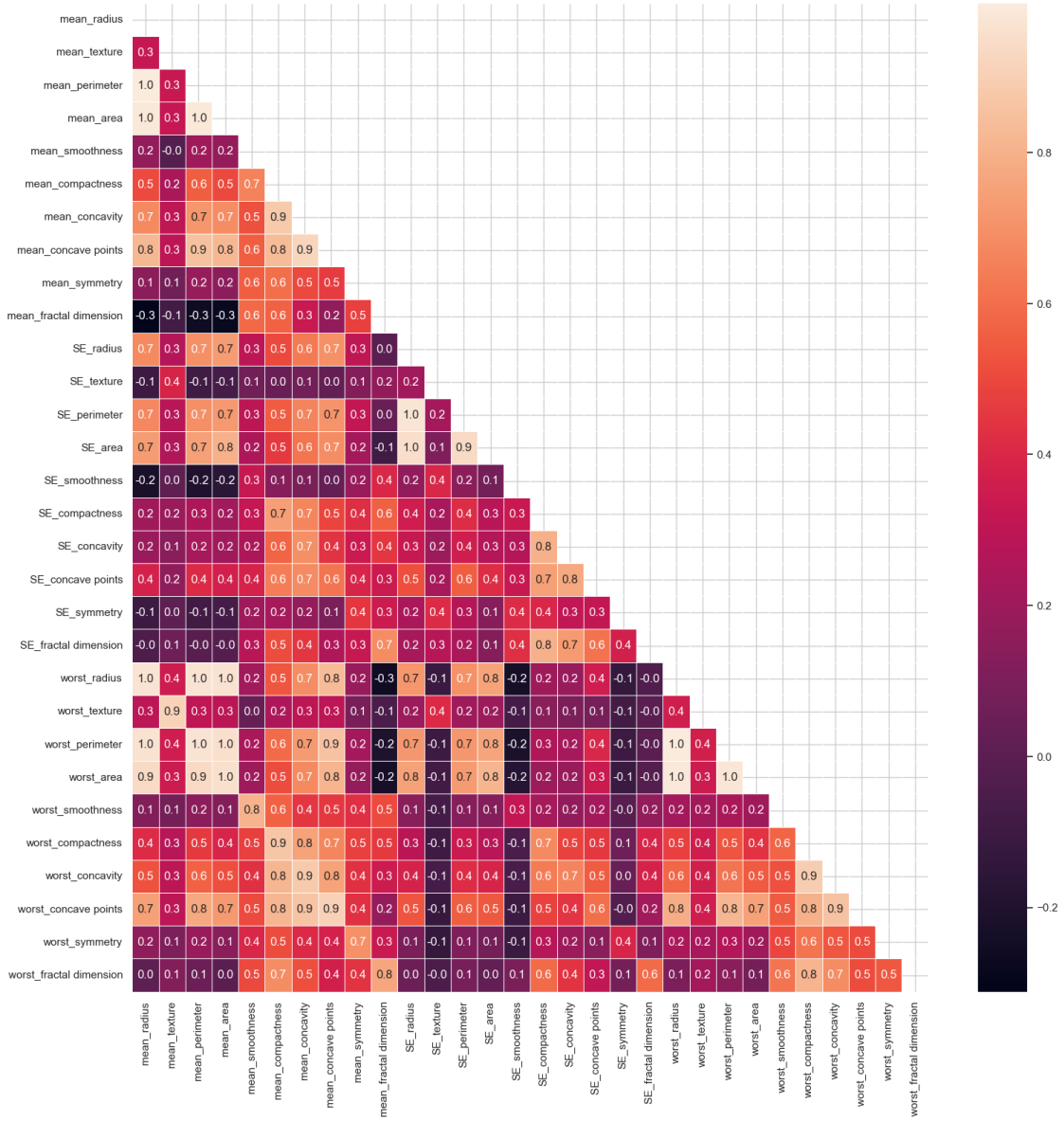
[128]:
```
data=pd.concat([y,data_n.iloc[:,20:31]],axis=1)
data=pd.melt(data,id_vars='diagnosis',var_name='features',value_name='value')
plt.figure(figsize=(10,10))
tic=time.time()
sns.swarmplot(x='features',y='value',hue='diagnosis',data=data)
plt.xticks(rotation=90)
```

[128]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
 [Text(0, 0, 'worst_radius'),
  Text(1, 0, 'worst_texture'),
  Text(2, 0, 'worst_perimeter'),
  Text(3, 0, 'worst_area'),
  Text(4, 0, 'worst_smoothness'),
  Text(5, 0, 'worst_compactness'),
  Text(6, 0, 'worst_concavity'),
  Text(7, 0, 'worst_concave points'),
  Text(8, 0, 'worst_symmetry'),
  Text(9, 0, 'worst_fractal dimension')])
```



[130]:
```
f,ax = plt.subplots(figsize=(18, 18))
matrix = np.triu(x.corr())
sns.heatmap(x.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax, mask=matrix)
```

[130]: <Axes: >



[131]:
```
# Create correlation matrix
corr_matrix = x.corr().abs()# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))


# Find index of feature columns with correlation greater than 0.8
to_drop = [column for column in upper.columns if any(upper[column] > 0.8)]
```

[132]:
```
to_drop
```

[132]:
```
['mean_perimeter',
 'mean_area',
 'mean_concavity',
 'mean_concave points',
 'SE_perimeter',
 'SE_area',
 'SE_concavity',
 'SE_fractal dimension',
 'worst_radius',
 'worst_texture',
 'worst_perimeter',
 'worst_area',
 'worst_smoothness',
 'worst_compactness',
 'worst_concavity',
 'worst_concave points',
 'worst_fractal dimension']
```

[133]:
```
# Drop features
x1 = x.drop(x[to_drop], axis=1)
x1.columns
```

[133]:
```
Index(['mean_radius', 'mean_texture', 'mean_smoothness', 'mean_compactness',
       'mean_symmetry', 'mean_fractal dimension', 'SE_radius', 'SE_texture',
       'SE_smoothness', 'SE_compactness', 'SE_concave points', 'SE_symmetry',
       'worst_symmetry'],
      dtype='object')
```

[134]:
```
x1.head()
```

[134]:

	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_symmetry	mean_fractal dimension	SE_radius	SE_texture	\
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	

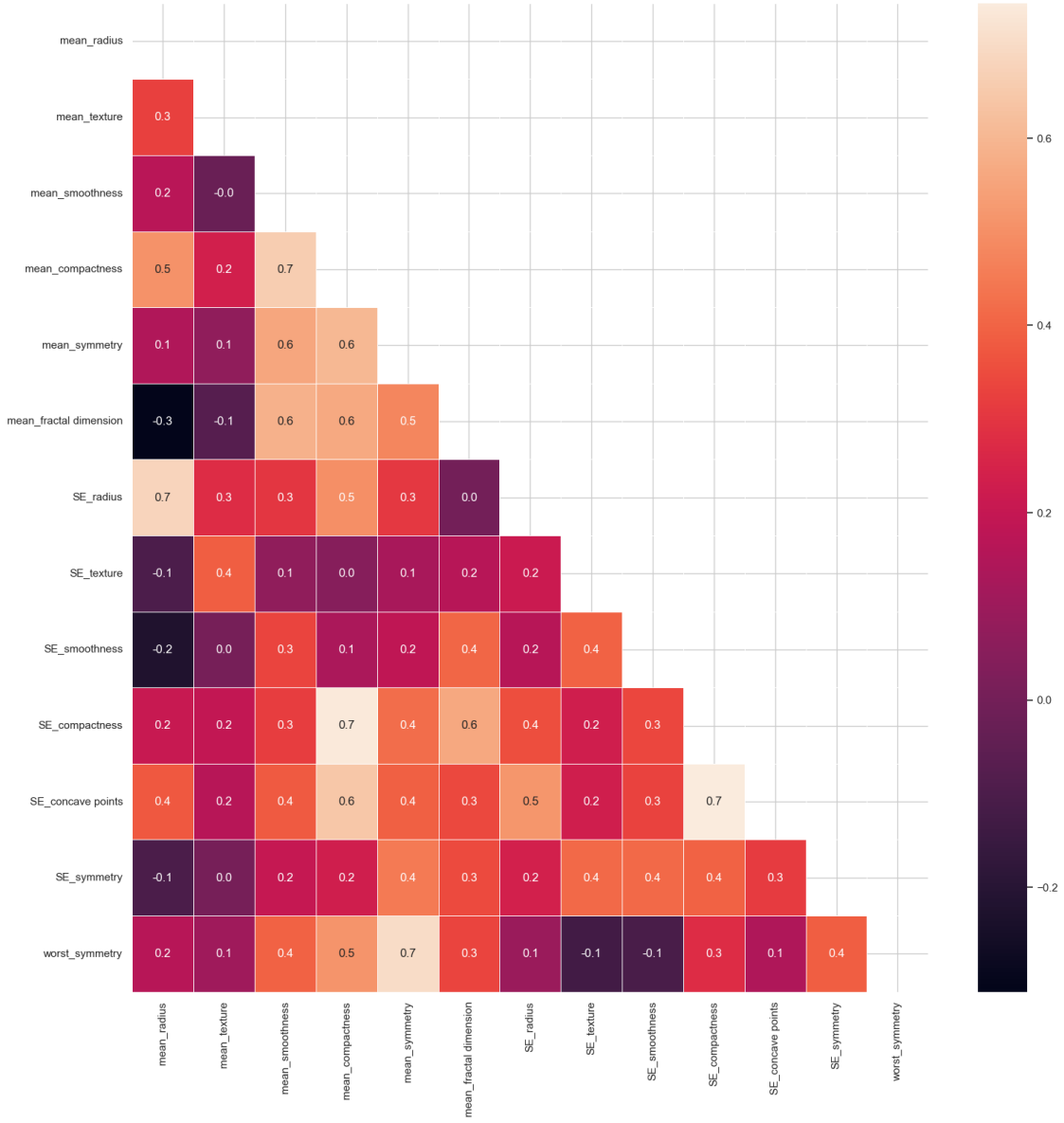
	SE_smoothness	SE_compactness	SE_concave points	SE_symmetry	\
0	0.006399	0.04904	0.01587	0.03003	\
1	0.005225	0.01308	0.01340	0.01389	

```
2       0.006150         0.04006           0.02058      0.02250
3       0.009110         0.07458           0.01867      0.05963
4       0.011490         0.02461           0.01885      0.01756

   worst_symmetry
0          0.4601
1          0.2750
2          0.3613
3          0.6638
4          0.2364
```

[135]:
```
f,ax = plt.subplots(figsize=(18, 18))
matrix = np.triu(x1.corr())
sns.heatmap(x1.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax, mask=matrix)
```

[135]: <Axes: >



[136]:
```
from sklearn.model_selection import train_test_split
```

[137]:
```
x_train,x_test,y_train,y_test=train_test_split(x1,y,test_size=0.
↪3,random_state=42)
```

[138]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
log_reg=LogisticRegression()
log_reg=log_reg.fit(x_train,y_train)
y_pred=log_reg.predict(x_test)
```

[139]:
```
log_reg.score(x_test,y_test)
```

[139]: 0.9239766081871345

[140]:
```
from sklearn.metrics import␣
↪roc_auc_score,roc_curve,classification_report,confusion_matrix,ConfusionMatrixDisplay
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94       108
           1       0.92      0.87      0.89        63

    accuracy                           0.92       171
   macro avg       0.92      0.91      0.92       171
weighted avg       0.92      0.92      0.92       171
```

[141]:
```
print('Confusion Matrix')
cm=metrics.confusion_matrix(y_test,y_pred,labels=[0,1])
df_cm=pd.DataFrame(cm,index=[i for i in [0,1]],
                   columns=[i for i in ['Predict 0','Predict 1']])
plt.figure(figsize=(7,5))
sns.heatmap(df_cm,annot=True,fmt='.5g',cmap='YlGn')
```

```
Confusion Matrix
```

[141]: <Axes: >

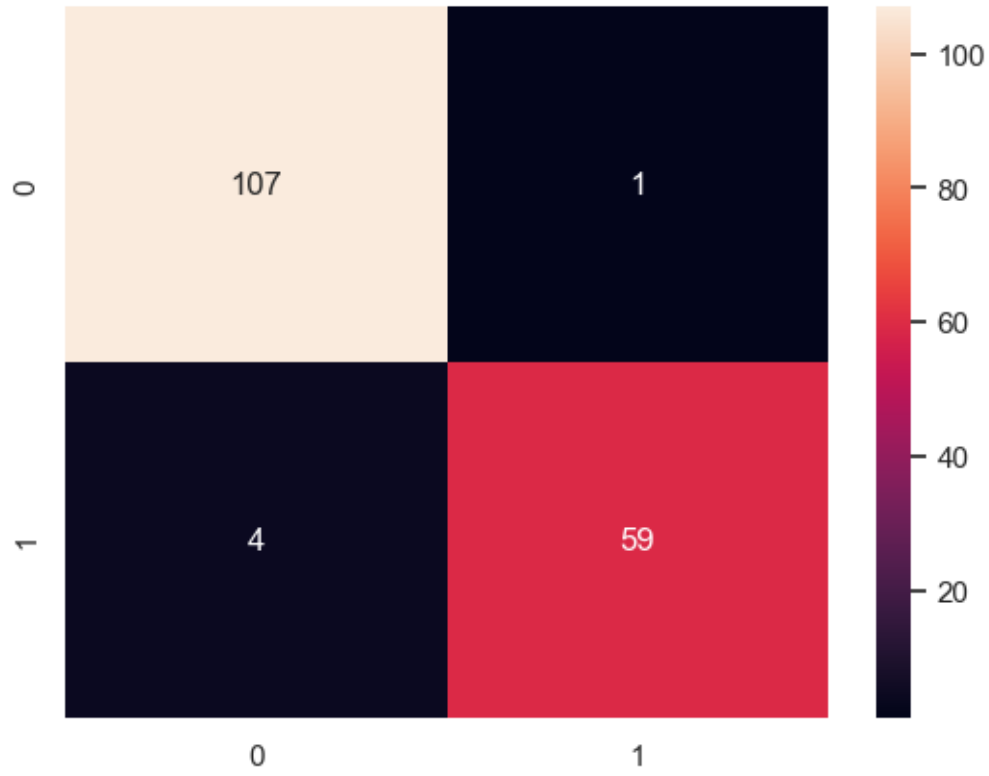
[148]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.metrics import f1_score,confusion_matrix
from sklearn.metrics import accuracy_score
```

[152]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪3,random_state=42)
```

[153]:
```
log_reg1=LogisticRegression()
log_reg1=log_reg1.fit(x_train,y_train)
acc1=accuracy_score(y_test,log_reg1.predict(x_test))
print('Accuracy is:',acc1)
cm2=confusion_matrix(y_test,log_reg1.predict(x_test))
sns.heatmap(cm2,annot=True,fmt='d')
```

```
Accuracy is: 0.9707602339181286
```

[153]: <Axes: >



[154]:
```
feature_select=SelectKBest(chi2,k=5).fit(x_train,y_train)
print('Score list:',feature_select.scores_)
print('Feature list:',x_train.columns)
```

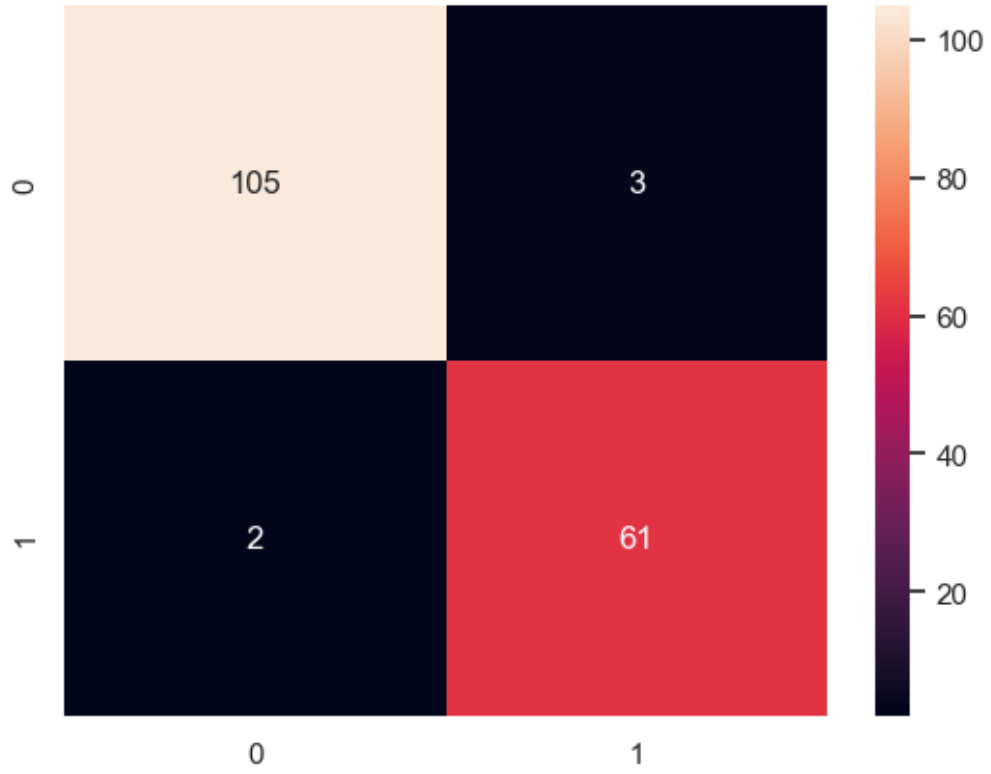
```
Score list: [1.77946492e+02 6.06916433e+01 1.34061092e+03 3.66899557e+04
 1.00015175e-01 3.41839493e+00 1.30547650e+01 7.09766457e+00
 1.95982847e-01 3.42575072e-04 2.45882967e+01 4.07131026e-02
 1.72696840e+02 6.12741067e+03 1.32470372e-03 3.74071521e-01
 6.92896719e-01 2.01587194e-01 1.39557806e-03 2.65927071e-03
 3.25782599e+02 1.16958562e+02 2.40512835e+03 7.50217341e+04
 2.63226314e-01 1.19077581e+01 2.58858117e+01 8.90751003e+00
 1.00635138e+00 1.23087347e-01]
Feature list: Index(['mean_radius', 'mean_texture', 'mean_perimeter',
'mean_area',
       'mean_smoothness', 'mean_compactness', 'mean_concavity',
       'mean_concave points', 'mean_symmetry', 'mean_fractal dimension',
       'SE_radius', 'SE_texture', 'SE_perimeter', 'SE_area', 'SE_smoothness',
       'SE_compactness', 'SE_concavity', 'SE_concave points', 'SE_symmetry',
       'SE_fractal dimension', 'worst_radius', 'worst_texture',
       'worst_perimeter', 'worst_area', 'worst_smoothness',
       'worst_compactness', 'worst_concavity', 'worst_concave points',
       'worst_symmetry', 'worst_fractal dimension'],
```

```
      dtype='object')
```

[156]:
```
x_train_2=feature_select.transform(x_train)
x_test_2=feature_select.transform(x_test)
log_reg2=LogisticRegression()
log_reg2=log_reg2.fit(x_train_2,y_train)
acc2=accuracy_score(y_test,log_reg2.predict(x_test_2))
print('Accuracy is:',acc2)
cm2=confusion_matrix(y_test,log_reg2.predict(x_test_2))
sns.heatmap(cm2,annot=True,fmt='d')
```

```
Accuracy is: 0.9707602339181286
```

[156]: <Axes: >

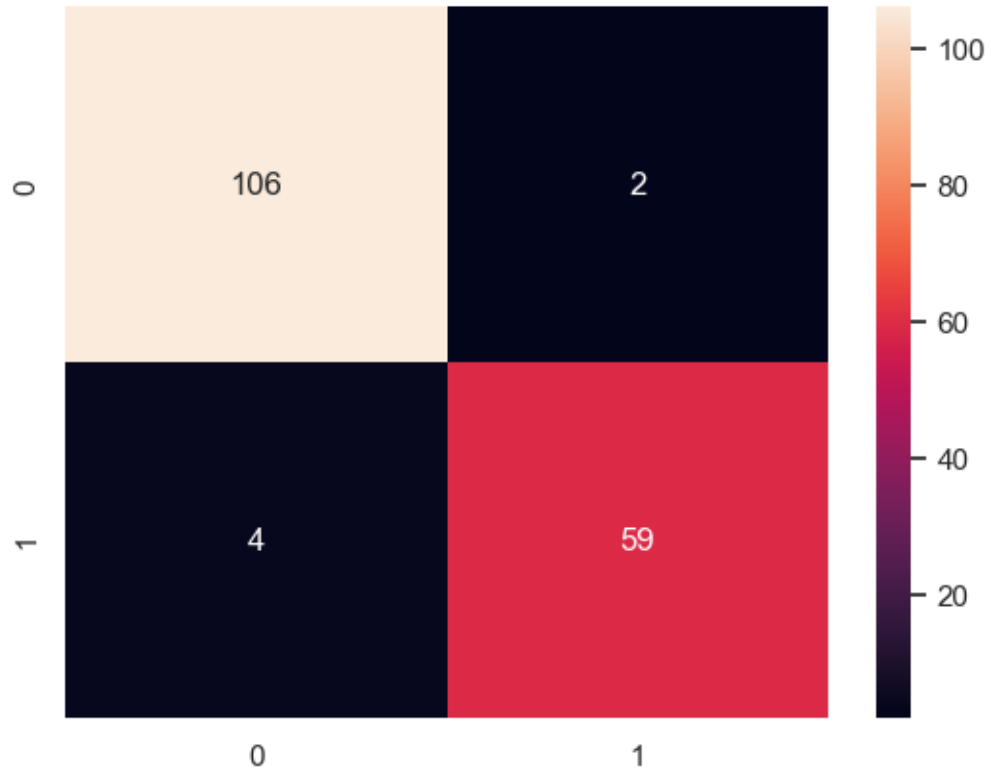


[157]:
```
from sklearn.feature_selection import RFE
log_reg3=LogisticRegression()
rfe=RFE(estimator=log_reg3,n_features_to_select=5,step=1)
rfe=rfe.fit(x_train,y_train)
x_train_3=rfe.transform(x_train)
x_test_3=rfe.transform(x_test)
```

```
log_reg3=log_reg3.fit(x_train_3,y_train)
acc3=accuracy_score(y_test,log_reg3.predict(x_test_3))
print('Accuracy is:',acc2)
cm2=confusion_matrix(y_test,log_reg3.predict(x_test_3))
sns.heatmap(cm2,annot=True,fmt='d')
```

```
Accuracy is: 0.9649122807017544
```

[157]: <Axes: >



[158]:
```
from sklearn.feature_selection import RFECV
log_reg4=LogisticRegression()
rfecv=RFECV(estimator=log_reg4,step=1,cv=5,scoring='accuracy')
rfecv=rfecv.fit(x_train,y_train)
print('Optimal number of features:',rfecv.n_features_)
print('Best features:',x_train.columns[rfecv.support_])
```

```
Optimal number of features: 18
Best features: Index(['mean_radius', 'mean_texture', 'mean_compactness',
'mean_concavity',
       'mean_concave points', 'SE_texture', 'SE_perimeter', 'SE_area',
       'SE_compactness', 'SE_symmetry', 'worst_radius', 'worst_texture',
```

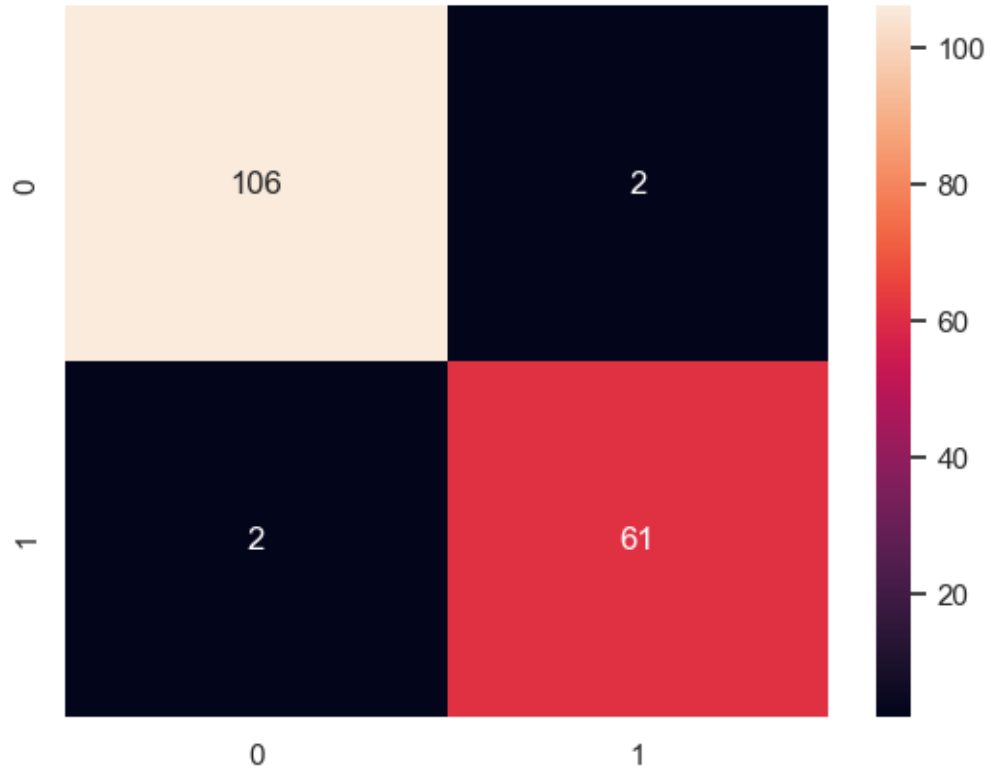
```
       'worst_perimeter', 'worst_smoothness', 'worst_compactness',
       'worst_concavity', 'worst_concave points', 'worst_symmetry'],
      dtype='object')
```

[160]:
```
x_train_4=rfecv.transform(x_train)
x_test_4=rfecv.transform(x_test)

log_reg4=log_reg4.fit(x_train_4,y_train)
acc4=accuracy_score(y_test,log_reg4.predict(x_test_4))
print('Accuracy is:',acc2)
cm2=confusion_matrix(y_test,log_reg4.predict(x_test_4))
sns.heatmap(cm2,annot=True,fmt='d')
```

```
Accuracy is: 0.9649122807017544
```

[160]:
```
<Axes: >
```



[161]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪3,random_state=42)
```

[163]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import fbeta_score, make_scorer
```

```
ftwo_scorer = make_scorer(fbeta_score, beta=2)
# Create logistic regression
log_reg_tuned = LogisticRegression()

# Create regularization penalty space
penalty = ['l1', 'l2']

# Create regularization hyperparameter space
C = np.arange(0, 1, 0.001)

# Create hyperparameter options
hyperparameters = dict(C=C, penalty=penalty)

# Create grid search using 5-fold cross validation
log_reg_tuned = GridSearchCV(log_reg_tuned, hyperparameters, cv=5,␣
 ↪scoring=ftwo_scorer, verbose=0)
```

[164]:
```
log_reg_tuned=log_reg_tuned.fit(x_train,y_train)
```

[165]:
```
print('Best Penalty:', log_reg_tuned.best_estimator_.get_params()['penalty'])
print('Best C:', log_reg_tuned.best_estimator_.get_params()['C'])
```

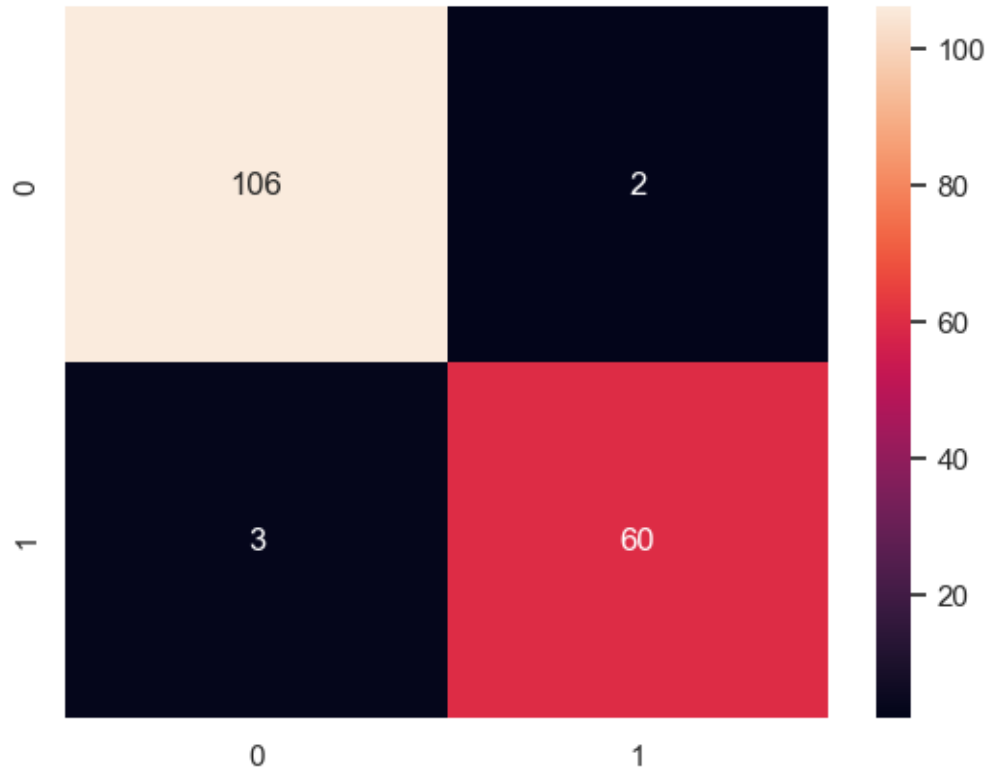
```
Best Penalty: l2
Best C: 0.8160000000000001
```

[166]:
```
y_pred=log_reg_tuned.predict(x_test)
```

[169]:
```
log_reg_tuned=log_reg_tuned.fit(x_train,y_train)
acc5=accuracy_score(y_test,log_reg_tuned.predict(x_test))
print('Accuracy is:',acc5)
cm2=confusion_matrix(y_test,log_reg_tuned.predict(x_test))
sns.heatmap(cm2,annot=True,fmt='d')
```

```
Accuracy is: 0.9707602339181286
```

[169]:
```
<Axes: >
```


106
2
3
60
0
1
0
1
100
80
60
40
20

[ ]: