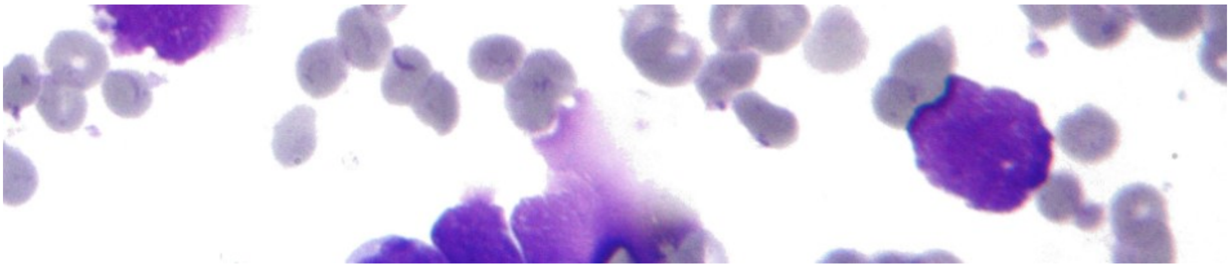
Tumor Diagnosis (Part 1): Exploratory Data Analysis



About the Dataset:

The Breast Cancer Diagnostic data is available on the UCI Machine Learning Repository. This database is also available through the UW CS ftp server.

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. n the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

Attribute Information:

- ID number
- Diagnosis (M = malignant, B = benign) 3-32)

Ten real-valued features are computed for each cell nucleus:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness (perimeter^2 / area - 1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recoded with four significant digits.

Missing attribute values: none

Class distribution: 357 benign, 212 malignant

Task 1: Loading Libraries and Data

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

In [ ]:
```
data = pd.read_csv('data.csv')
```

Exploratory Data Analysis

Task 2: Separate Target from Features

Note: If you are starting the notebook from this task, you can run cells from all the previous tasks in the kernel by going to the top menu and Kernel > Restart and Run All

---

In [ ]:
```
data.head()
```

Out[ ]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11
1	842517	M	20.57	17.77	132.90	1326.0	0.08
2	84300903	M	19.69	21.25	130.00	1203.0	0.10
3	84348301	M	11.42	20.38	77.58	386.1	0.14
4	84358402	M	20.29	14.34	135.10	1297.0	0.10

5 rows × 33 columns

ID has to be either dropped or put as index

Diagnosis column: is my target that i want to predict

Unnamed: 32 column has to be dropped

```
In [ ]: col = data.columns
        print(col)
```

```
Index(['id', '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', 'Unnamed: 32'],
      dtype='object')
```

```
In [ ]: y = data.diagnosis

        # drop_cols = ['id', 'diagnosis', 'Unnamed: 32']
        drop_cols = ['id', 'Unnamed: 32']

        data_new = data.drop(drop_cols, axis=1)      #axis=1 because dropping columns, if droppi
        data_new
```

Out[ ]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	con
0	M	17.99	10.38	122.80	1001.0	0.11840	
1	M	20.57	17.77	132.90	1326.0	0.08474	
2	M	19.69	21.25	130.00	1203.0	0.10960	
3	M	11.42	20.38	77.58	386.1	0.14250	
4	M	20.29	14.34	135.10	1297.0	0.10030	
...	...	...	...	...	...	...	
564	M	21.56	22.39	142.00	1479.0	0.11100	
565	M	20.13	28.25	131.20	1261.0	0.09780	
566	M	16.60	28.08	108.30	858.1	0.08455	
567	M	20.60	29.33	140.10	1265.0	0.11780	
568	B	7.76	24.54	47.92	181.0	0.05263	

569 rows × 31 columns

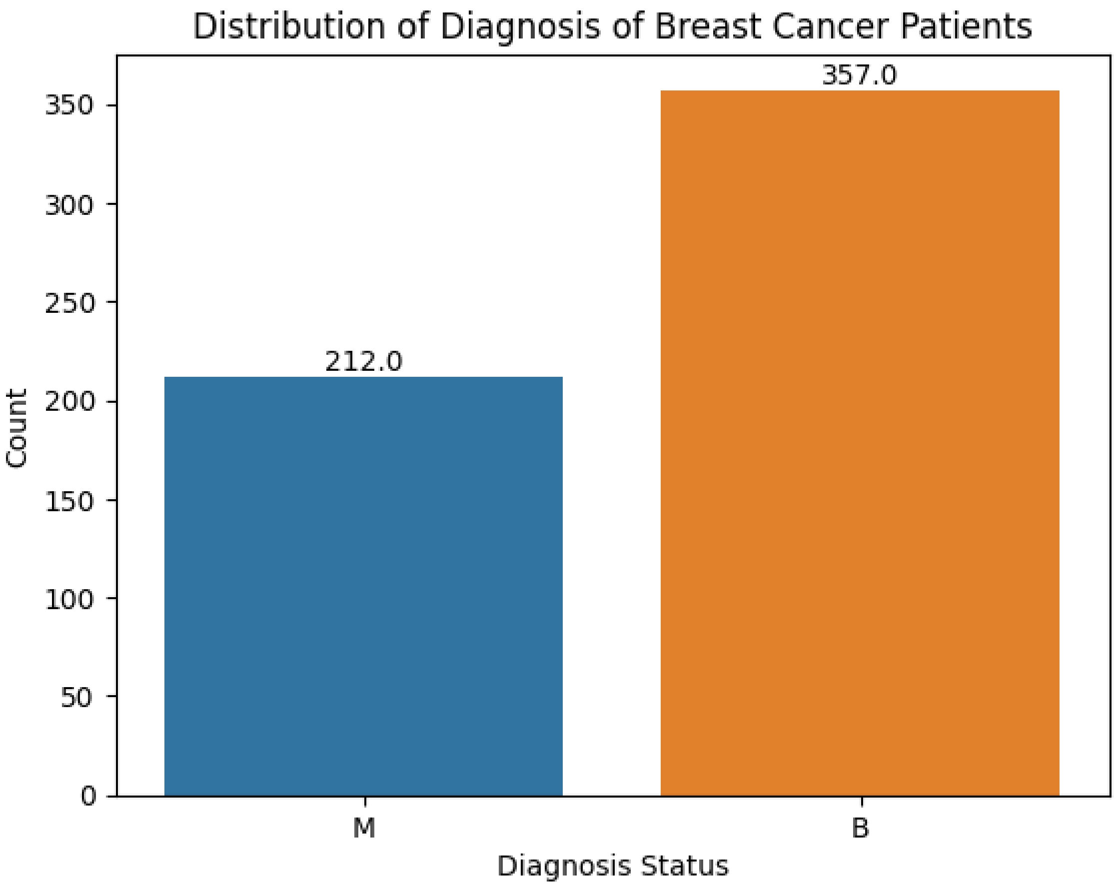
Task 3: Plot Diagnosis Distributions

Note: If you are starting the notebook from this task, you can run cells from all the previous tasks in the kernel by going to the top menu and Kernel > Restart and Run All

---

In [ ]:

```python
# Create a DataFrame from the sample data
import pandas as pd
df = pd.DataFrame(data_new)

# Create the count plot
ax = sns.countplot(x='diagnosis', data=df)

# Add labels to the bars
for p in ax.patches:
   ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()), ha

# Add a title and labels
plt.title('Distribution of Diagnosis of Breast Cancer Patients')
plt.xlabel('Diagnosis Status')
plt.ylabel('Count')

# Display the plot
plt.show()
```



In [ ]:

```python
x= df.drop('diagnosis', axis=1)      #axis=1 because dropping columns, if dropping rows 
x.describe()
```

Out[ ]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactnes
count	569.000000	569.000000	569.000000	569.000000	569.000000	569
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0
std	3.524049	4.301036	24.298981	351.914129	0.014064	0
min	6.981000	9.710000	43.790000	143.500000	0.052630	0
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0

8 rows × 30 columns

Data Visualization

Task 4: Visualizing Standardized Data with Seaborn

Note: If you are starting the notebook from this task, you can run cells from all the previous tasks in the kernel by going to the top menu and Kernel > Restart and Run All

In [ ]:

```
data = x

data_std = (data - data.mean()) / data.std()
data_std
```

Out[ ]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_m
0	1.096100	-2.071512	1.268817	0.983510	1.567087	3.28(
1	1.828212	-0.353322	1.684473	1.907030	-0.826235	-0.48(
2	1.578499	0.455786	1.565126	1.557513	0.941382	1.05.
3	-0.768233	0.253509	-0.592166	-0.763792	3.280667	3.39
4	1.748758	-1.150804	1.775011	1.824624	0.280125	0.53
...	...	...	...	...	...	
564	2.109139	0.720838	2.058974	2.341795	1.040926	0.21
565	1.703356	2.083301	1.614511	1.722326	0.102368	-0.01
566	0.701667	2.043775	0.672084	0.577445	-0.839745	-0.03
567	1.836725	2.334403	1.980781	1.733693	1.524426	3.26
568	-1.806811	1.220718	-1.812793	-1.346604	-3.109349	-1.14

569 rows × 30 columns

In [ ]:

```
data = pd.concat([y, data_std.iloc[:, 0:10]], axis=1) #all rows, but columns 0 to 9 so i
data.head()
```

Out[ ]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
0	M	1.096100	-2.071512	1.268817	0.983510	1.567087	
1	M	1.828212	-0.353322	1.684473	1.907030	-0.826235	
2	M	1.578499	0.455786	1.565126	1.557513	0.941382	
3	M	-0.768233	0.253509	-0.592166	-0.763792	3.280667	
4	M	1.748758	-1.150804	1.775011	1.824624	0.280125	

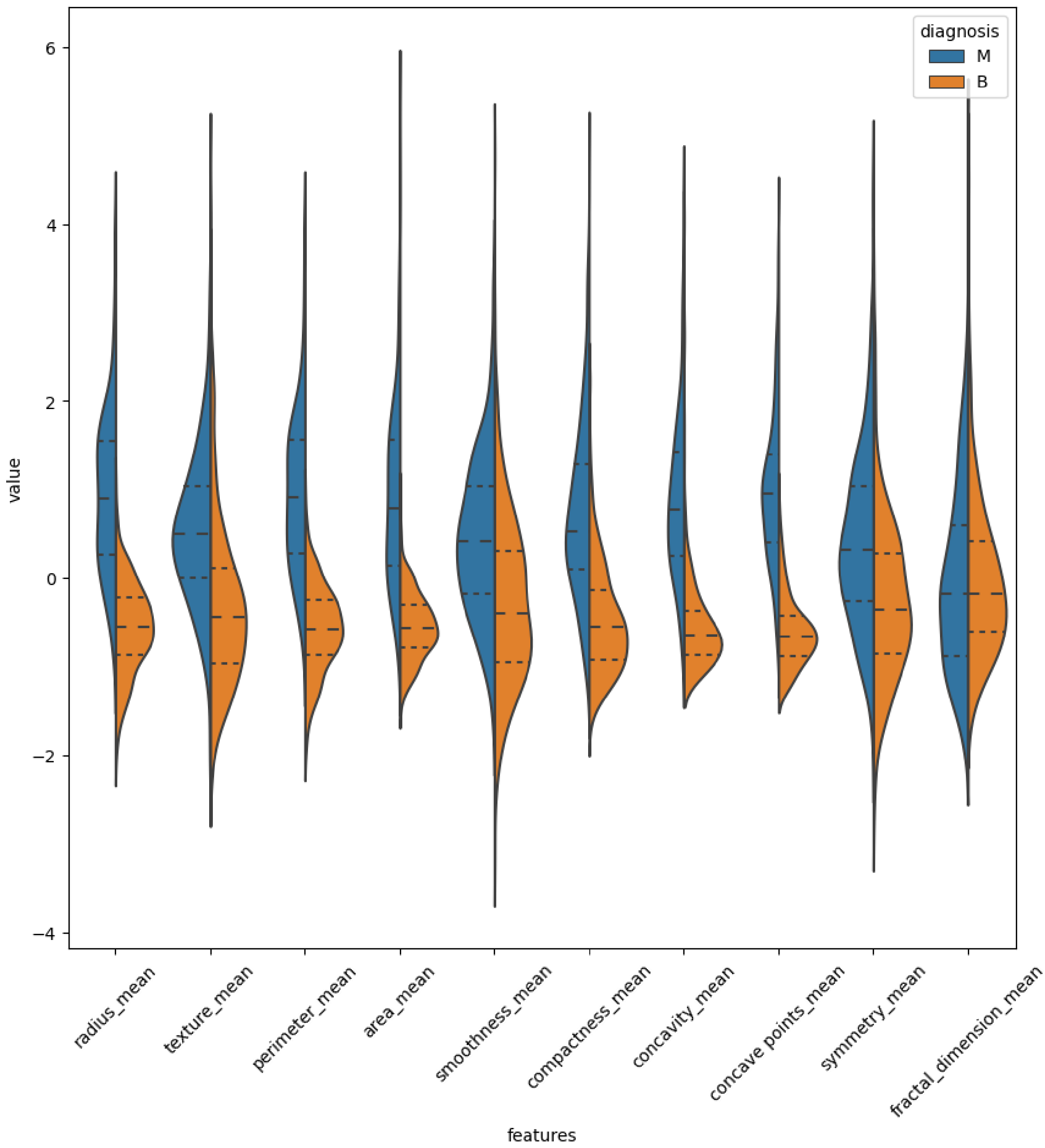
In [ ]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 11 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
dtypes: float64(10), object(1)
memory usage: 49.0+ KB
```

In [ ]:

```
#If we try to get plot now, we'll get an error  cuz data is in long format
# we need to unpivot it to wide format, we can use melt method
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',     #hue will color according to diagnosis B or M
               data=data,
               split=True,
               inner='quart',
              )

plt.xticks(rotation=45);

#Beacuse 10 features, so alot of names, so we want feature names to be rotated to be eas
```



When we check the above diagram, for example:

Texture_Mean, it's median is far for B and M, so would indicate it as good feature for differentiation. However, if you look at the last one, the median doesn't look like it is well separated, so might mean it will not give good prediction.

Task 5: Violin Plots and Box Plots

Note: If you are starting the notebook from this task, you can run cells from all the previous tasks in the kernel by going to the top menu and Kernel > Restart and Run All

---

In [ ]:
```python
#Now we will try to make comparison between features 10th to 19th
data = pd.concat([y, data_std.iloc[:, 10:20]], axis=1)
data
```

Out[ ]:

	diagnosis	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se
0	M	2.487545	-0.564768	2.830540	2.485391	-0.213814	1.315704
1	M	0.498816	-0.875473	0.263095	0.741749	-0.604819	-0.692317
2	M	1.227596	-0.779398	0.850180	1.180298	-0.296744	0.814257
3	M	0.326087	-0.110312	0.286341	-0.288125	0.689095	2.741868
4	M	1.269426	-0.789549	1.272070	1.189310	1.481763	-0.048477
...	...	...	...	...	...	...	...
564	M	2.779634	0.070963	2.377491	2.601897	1.085429	0.191637
565	M	1.299356	2.258951	1.155840	1.290429	-0.423637	-0.069697
566	M	0.184730	-0.257145	0.276450	0.180539	-0.379008	0.660696
567	M	1.156917	0.685485	1.437265	1.008615	-0.172848	2.015943
568	B	-0.070217	0.382756	-0.157311	-0.465742	0.049299	-1.162493

569 rows × 11 columns

In [ ]:
```python
data = pd.melt(data, id_vars = 'diagnosis',
               var_name = 'features',
               value_name = 'value')

data.head()
```

Out[ ]:

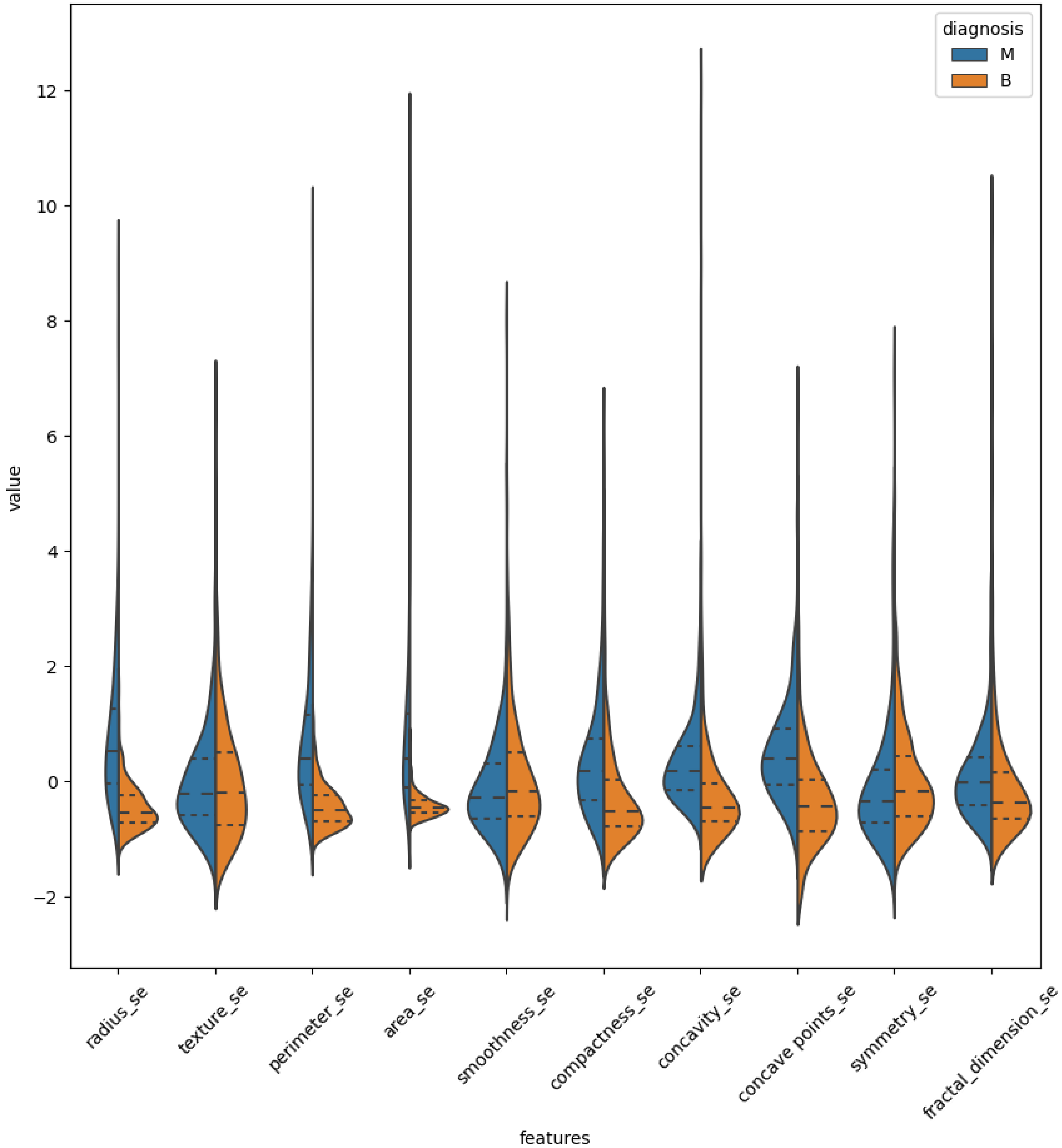
	diagnosis	features	value
0	M	radius_se	2.487545
1	M	radius_se	0.498816
2	M	radius_se	1.227596
3	M	radius_se	0.326087
4	M	radius_se	1.269426

In [ ]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5690 entries, 0 to 5689
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   diagnosis  5690 non-null   object 
 1   features   5690 non-null   object 
 2   value      5690 non-null   float64
dtypes: float64(1), object(2)
memory usage: 133.5+ KB
```

In [ ]:

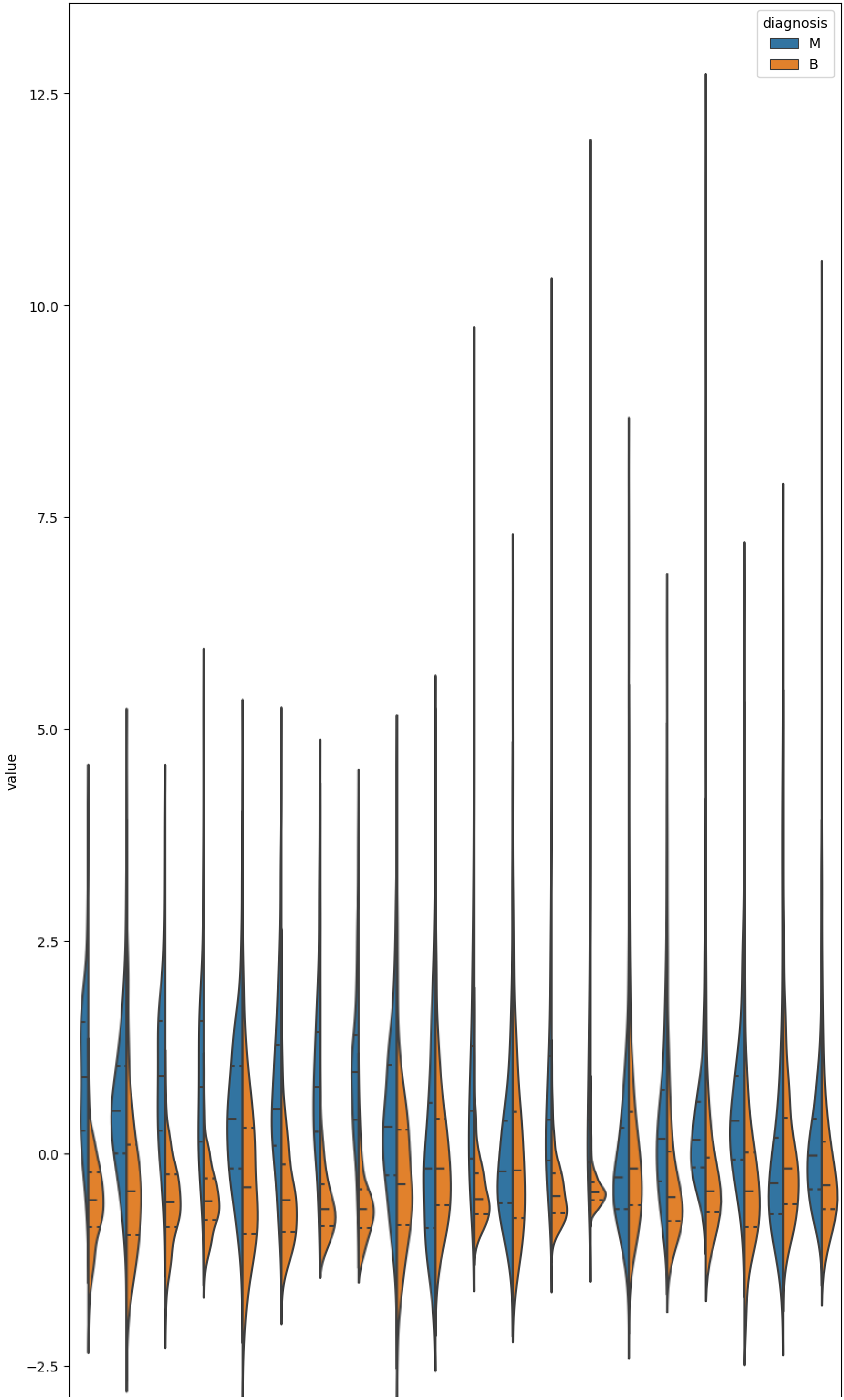
```
plt.figure(figsize=(10,10))

sns.violinplot(x='features',
               y='value',
               hue='diagnosis',    #hue will color according to diagnosis B or M
               data=data,
               split=True,
               inner='quart',
              )

plt.xticks(rotation=45);
```

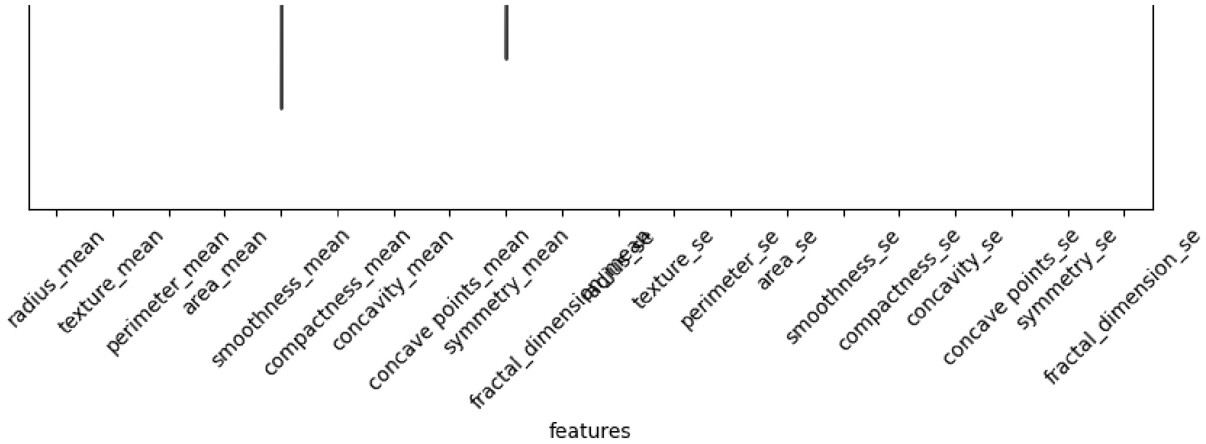


Clearly we have a stark difference in features here

In [ ]:
```
#Now we will try to make comparison between more feature 0 to 19th
data = pd.concat([y, data_std.iloc[:, 0:20]], axis=1)

data = pd.melt(data, id_vars = 'diagnosis',
               var_name = 'features',
               value_name = 'value')

#We need to decrease the figure size
plt.figure(figsize=(10,20))

sns.violinplot(x='features',
               y='value',
               hue='diagnosis',
               data=data,
               split=True,
```

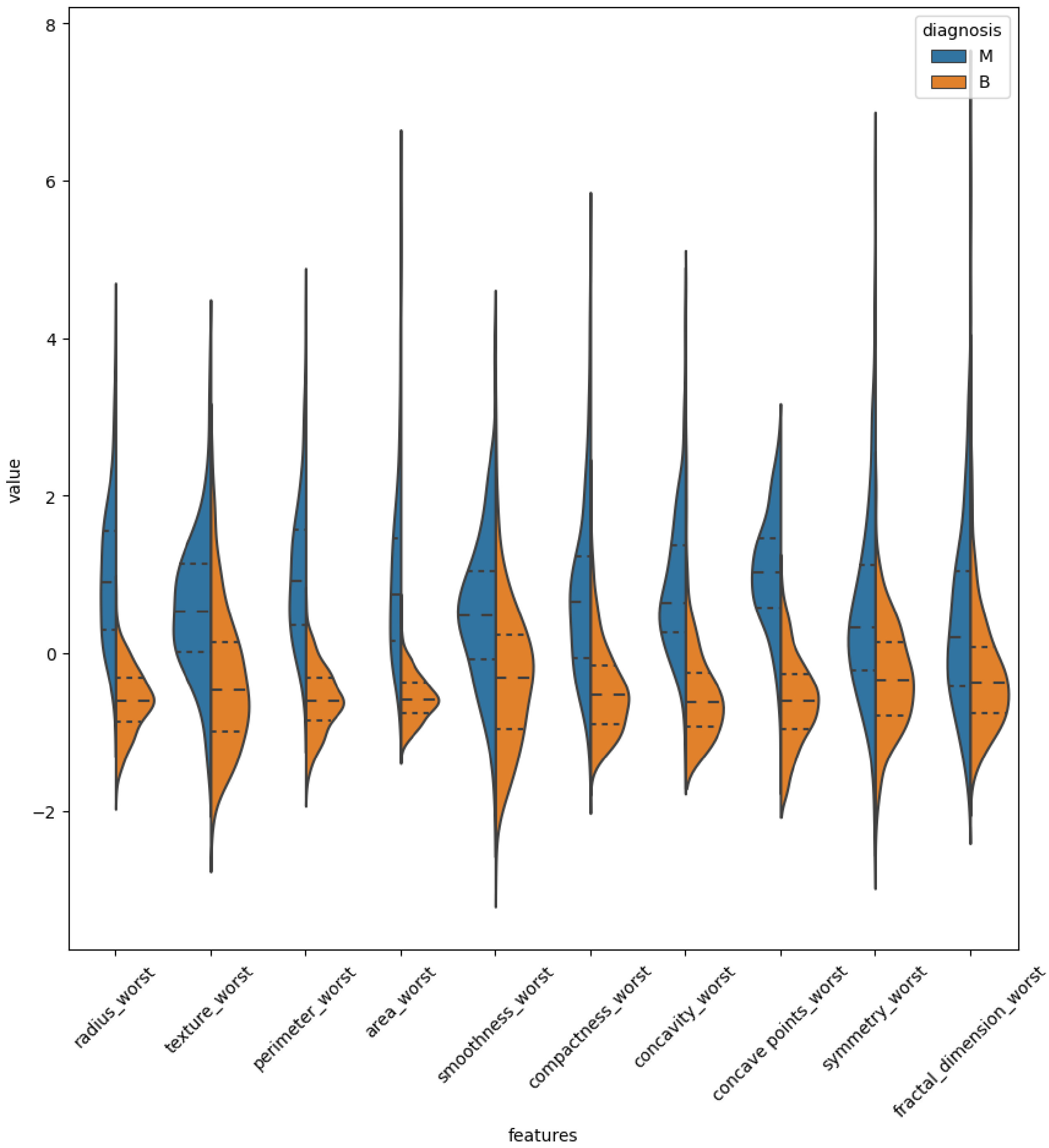
```
                 inner='quart',
                )

plt.xticks(rotation=45);
```


value
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
diagnosis
M
B



As seen, this is all cluttered, so this is **NOT** best way to visualize your data and make interpretations

In [ ]:
```
#Now we will return to normal 10 features and compare between features 20th to 30th
data = pd.concat([y, data_std.iloc[:, 20:30]], 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',     #hue will color according to diagnosis B or M
               data=data,
               split=True,
               inner='quart',
              )

plt.xticks(rotation=45);
```

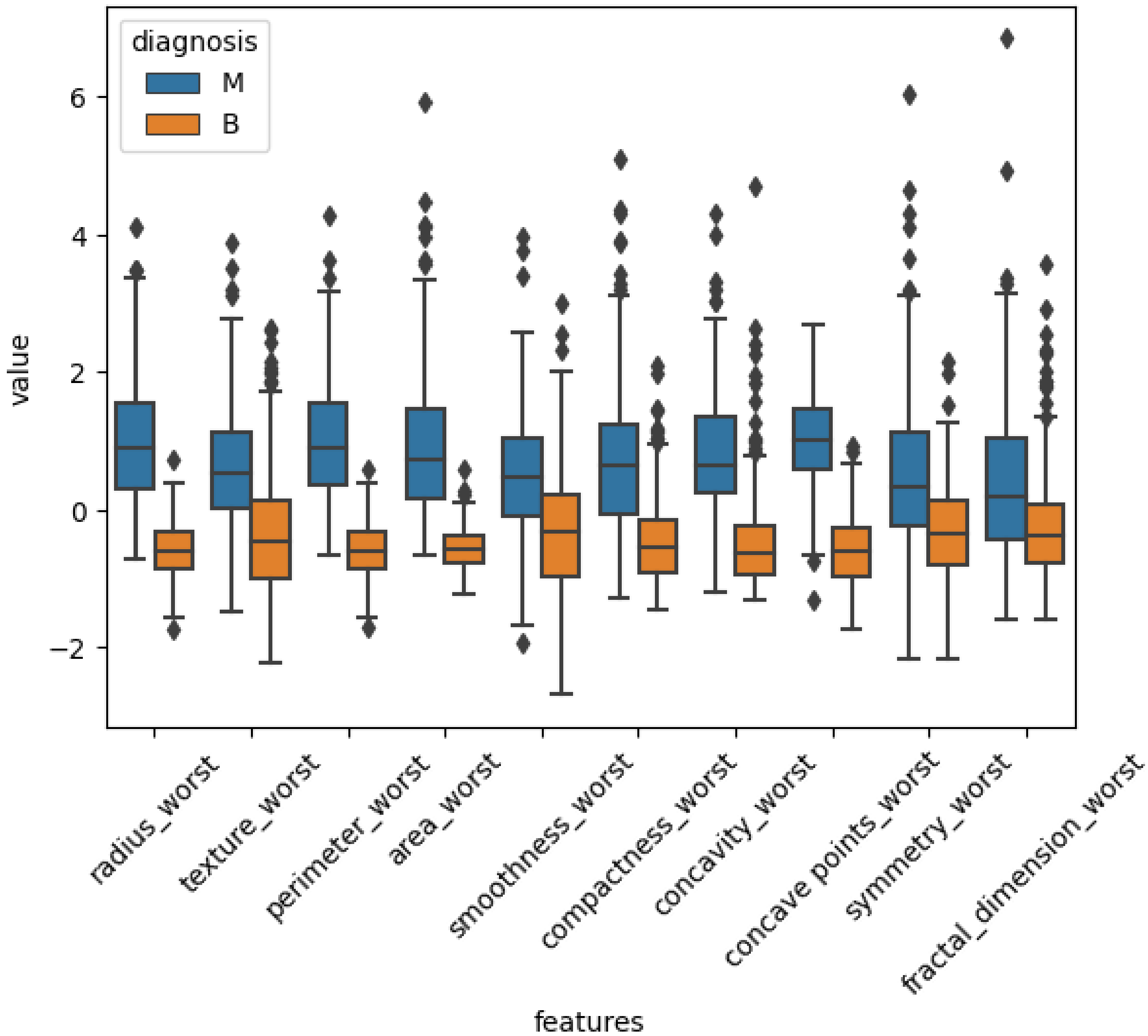


Features: *compactness worst*, *concativity worst* and *concave points worst*

look similar so we might need to explore if they are related to each other and remove one of them. Because they might negatively impact our predictive classifier.

```
In [ ]: sns.boxplot(x='features',
                    y='value',
                    hue = 'diagnosis',
                    data=data
                   )

        plt.xticks(rotation=45)
```

```
Out[ ]: (array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
         [Text(0, 0, 'radius_worst'),
          Text(1, 0, 'texture_worst'),
          Text(2, 0, 'perimeter_worst'),
          Text(3, 0, 'area_worst'),
          Text(4, 0, 'smoothness_worst'),
          Text(5, 0, 'compactness_worst'),
          Text(6, 0, 'concavity_worst'),
          Text(7, 0, 'concave points_worst'),
          Text(8, 0, 'symmetry_worst'),
          Text(9, 0, 'fractal_dimension_worst')])
```



This boxplot clearly shows the outliers in each feature in my data.

Task 6: Using Joint Plots for Feature Comparison

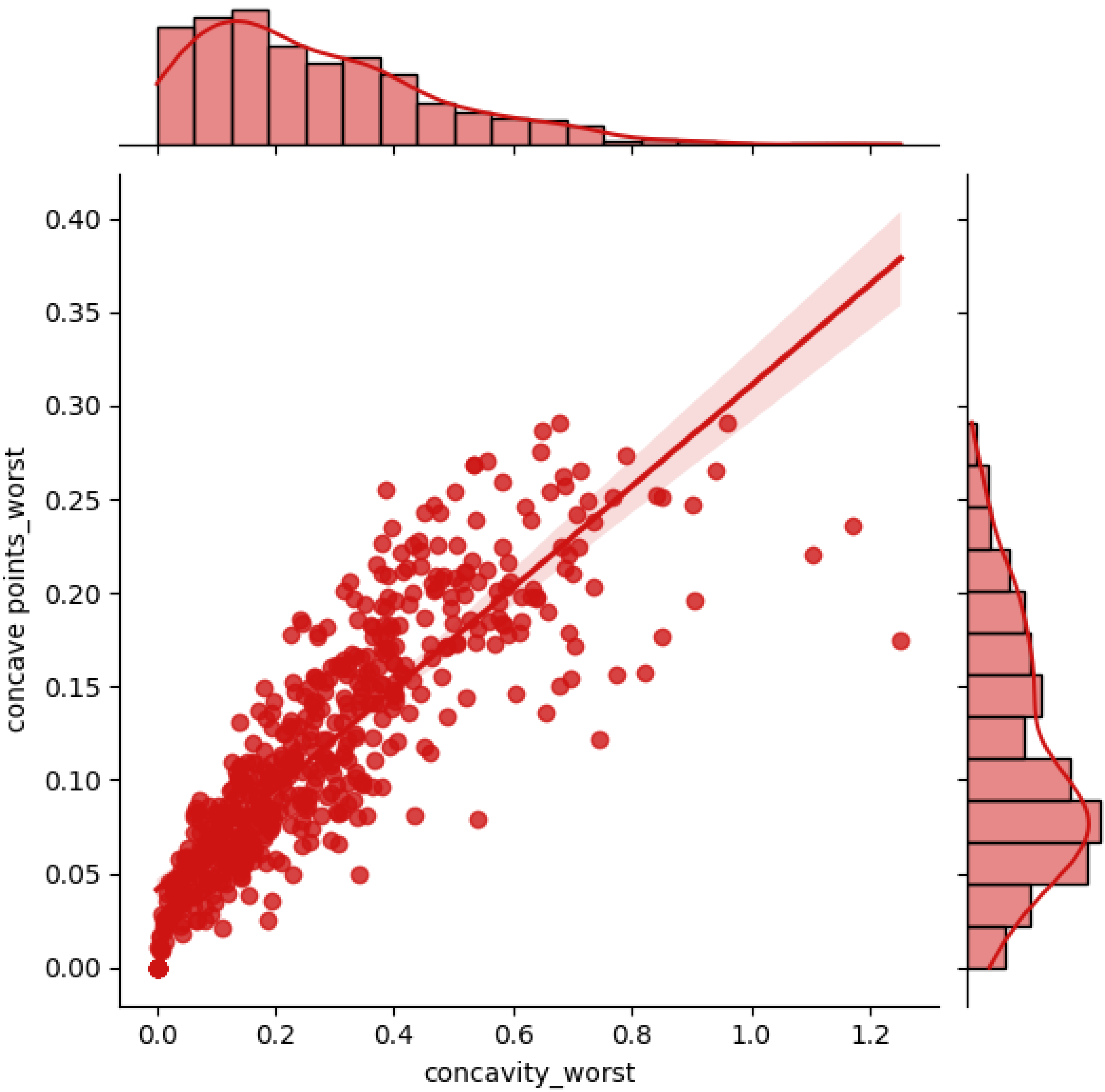
Note: If you are starting the notebook from this task, you can run cells from all the previous tasks in the kernel by going to the top menu and Kernel > Restart and Run All

---

```
In [ ]: sns.jointplot(
            x=x.loc[:, 'concavity_worst'],        # Data for the x-axis (all rows of the 'concav
            y=x.loc[:, 'concave points_worst'],   # Data for the y-axis (all rows of the 'concav
```

```
    kind='reg',                                      # Regression plot with scatter points and a re
    color='#ce1413'                                  # Color of the plot (you can specify any valid
)
```

Out[ ]: <seaborn.axisgrid.JointGrid at 0x1d192670f40>



We can tell that these 2 features are highly correlated. The pearson correlation is not given here, but you can calculate that also using more quantitative methods.

Task 7: Observing the Distribution of Values and their Variance with Swarm Plots

---

Note: If you are starting the notebook from this task, you can run cells from all the previous tasks in the kernel by going to the top menu and Kernel > Restart and Run All

---

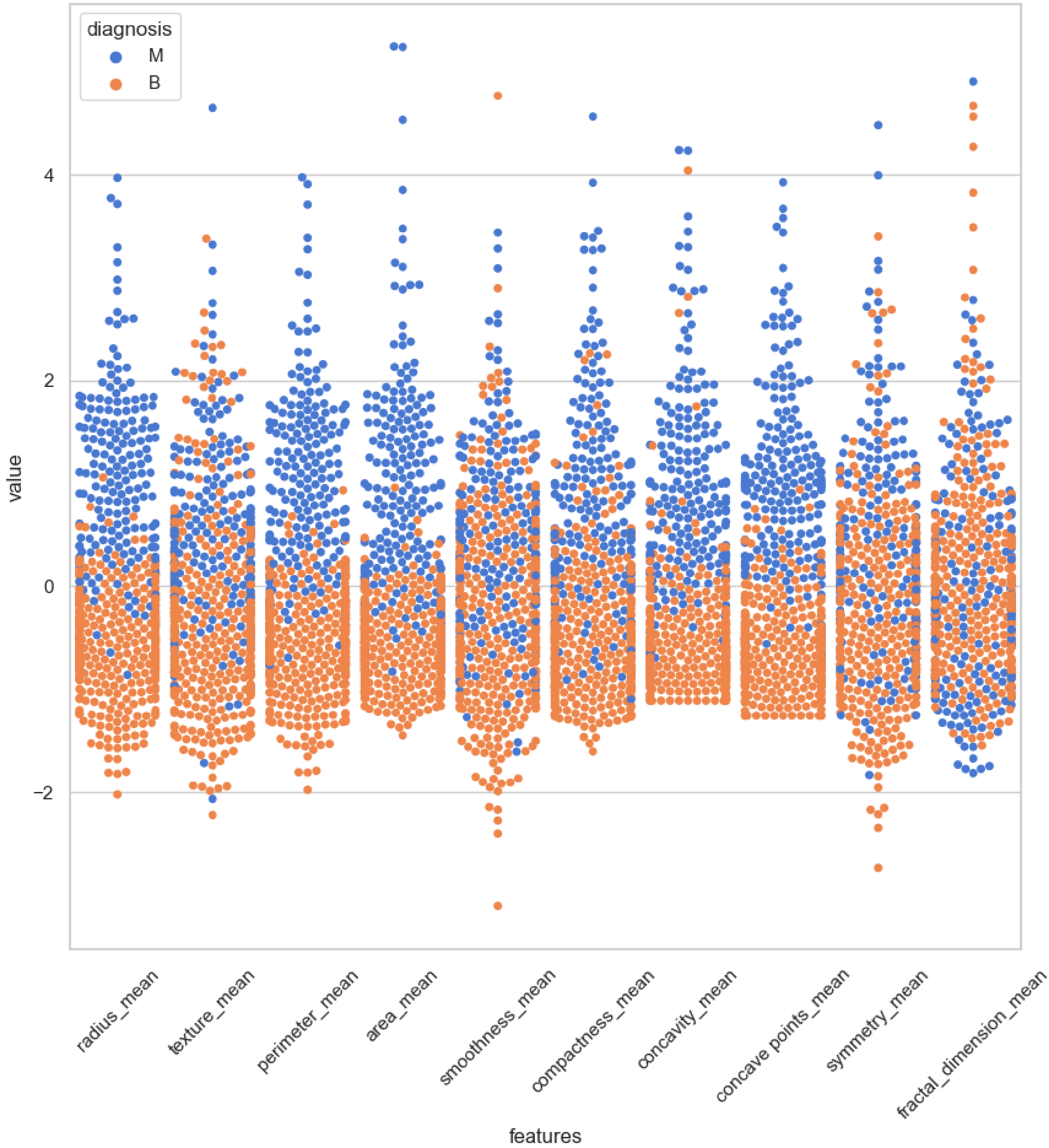
The cool thing in swarm plots is that you clearly see the variance between the features of your target.

In [ ]:

```
sns.set(style='whitegrid',
       palette='muted',    #dull the colors abit to not hurt your eyes
       )

data = x
data_std = (data - data.mean()) / data.std()
data = pd.concat([y, data_std.iloc[:, 0:10]], axis=1)
#all rows, but columns 0 to 9 so it doesn't become cluttered
#axis = 1 because columns

#If we try to get plot now, we'll get an error bec. data is in long format
# we need to unpivot it to wide format, we can use melt method

data = pd.melt(data, id_vars = 'diagnosis',
              var_name = 'features',
              value_name = 'value')

plt.figure(figsize=(10,10))

sns.swarmplot(x='features',
               y='value',
               hue='diagnosis',     #hue will color according to diagnosis B or M
               data=data
              #we removed the split and hue which only work with violin plots
              )

plt.xticks(rotation=45);

#Beacuse 10 features, so alot of names, so we want feature names to be rotated to be eas
```

```
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 40.6% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 40.2% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 45.5% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 38.3% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 43.9% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 48.5% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 45.9% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 40.4% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 42.2% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 41.3% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 41.1% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 46.0% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 38.7% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 44.1% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 42.9% of the points cannot be placed; you may want to decrease
```

```
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

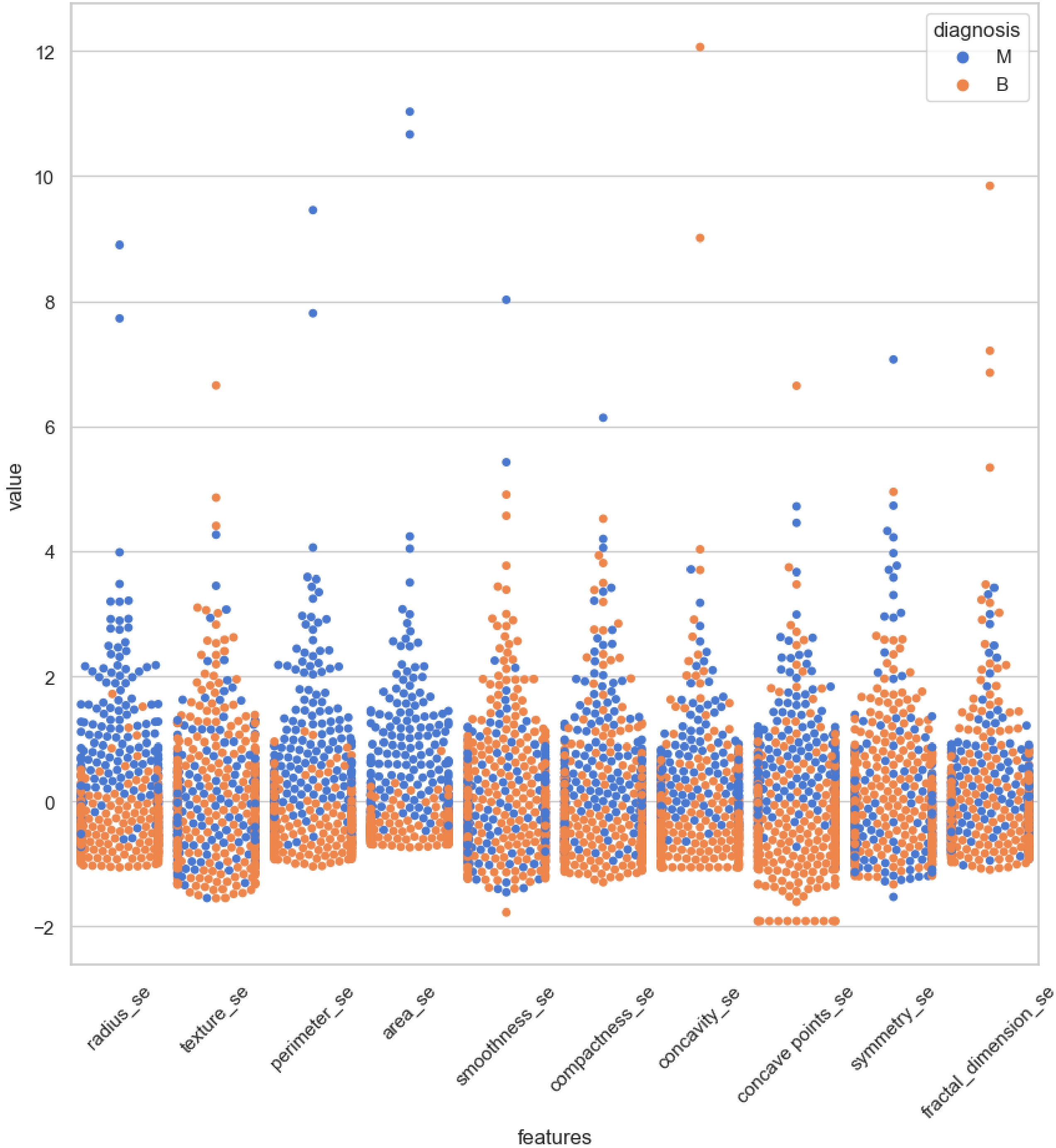


In [ ]:
```python
sns.set(style='whitegrid',
        palette='muted',   #dull the colors abit to not hurt your eyes
        )

data = x
data_std = (data - data.mean()) / data.std()
data = pd.concat([y, data_std.iloc[:, 10:20]], axis=1)
#all rows, but columns 0 to 9 so it doesn't become cluttered
#axis = 1 because columns

#If we try to get plot now, we'll get an error bec. data is in long format
# we need to unpivot it to wide format, we can use melt method

data = pd.melt(data, id_vars = 'diagnosis',
               var_name = 'features',
```

```
               value_name = 'value')

plt.figure(figsize=(10,10))

sns.swarmplot(x='features',
               y='value',
               hue='diagnosis',    #hue will color according to diagnosis B or M
               data=data
              #we removed the split and hue which only work with violin plots
              )

plt.xticks(rotation=45);

#Beacuse 10 features, so alot of names, so we want feature names to be rotated to be eas
```

```
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 64.7% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 61.2% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 66.4% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 71.2% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 61.7% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 63.4% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 67.7% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 60.1% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 63.1% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 67.1% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 64.9% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 61.3% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 66.6% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 71.4% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 62.0% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
```

```
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 67.8% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 60.5% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 67.5% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



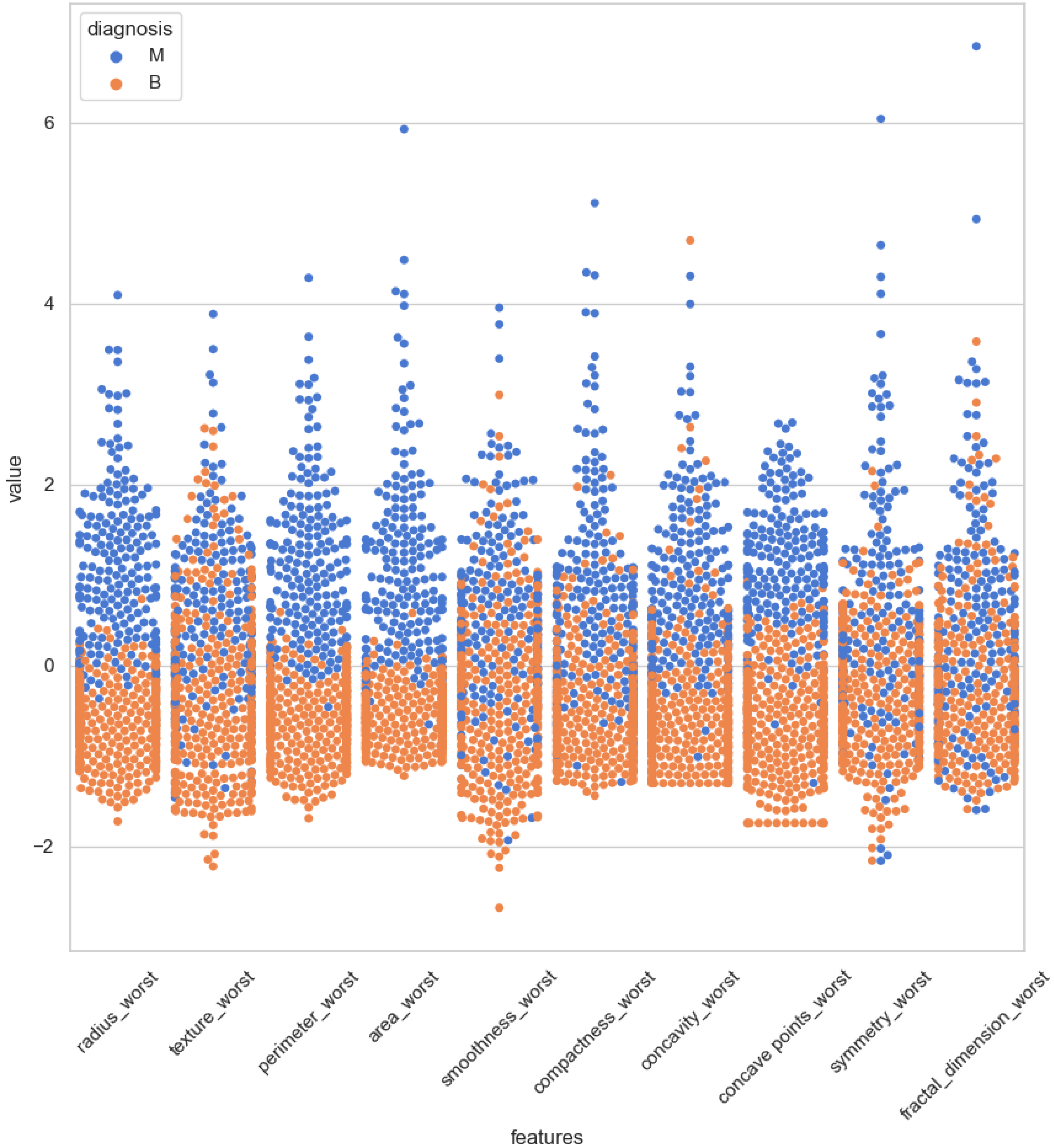
In [ ]:

```
sns.set(style='whitegrid',
        palette='muted',    #dull the colors abit to not hurt your eyes
        )
```

```
data = x
data_std = (data - data.mean()) / data.std()
data = pd.concat([y, data_std.iloc[:, 20:30]], axis=1)
#all rows, but columns 0 to 9 so it doesn't become cluttered
#axis = 1 because columns

#If we try to get plot now, we'll get an error bec. data is in long format
# we need to unpivot it to wide format, we can use melt method

data = pd.melt(data, id_vars = 'diagnosis',
              var_name = 'features',
              value_name = 'value')

plt.figure(figsize=(10,10))

sns.swarmplot(x='features',
               y='value',
               hue='diagnosis',    #hue will color according to diagnosis B or M
               data=data
              #we removed the split and hue which only work with violin plots
              )

plt.xticks(rotation=45);

#Beacuse 10 features, so alot of names, so we want feature names to be rotated to be eas
```

```
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 46.4% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 43.8% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 46.6% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 53.4% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 42.9% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 49.7% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 48.7% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 43.9% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 49.0% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 48.0% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 46.7% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 44.3% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 53.8% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 43.4% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categori
cal.py:3544: UserWarning: 50.1% of the points cannot be placed; you may want to decrease
the size of the markers or use stripplot.
```

```
  warnings.warn(msg, UserWarning)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\categorical.py:3544: UserWarning: 49.9% of the points cannot be placed; you want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



As observed in smoothness worst feature, this is bad feature to use because there is a mixture and not well separated.

On the contrary, *perimeter worst* or *area worst* have good predictive power.

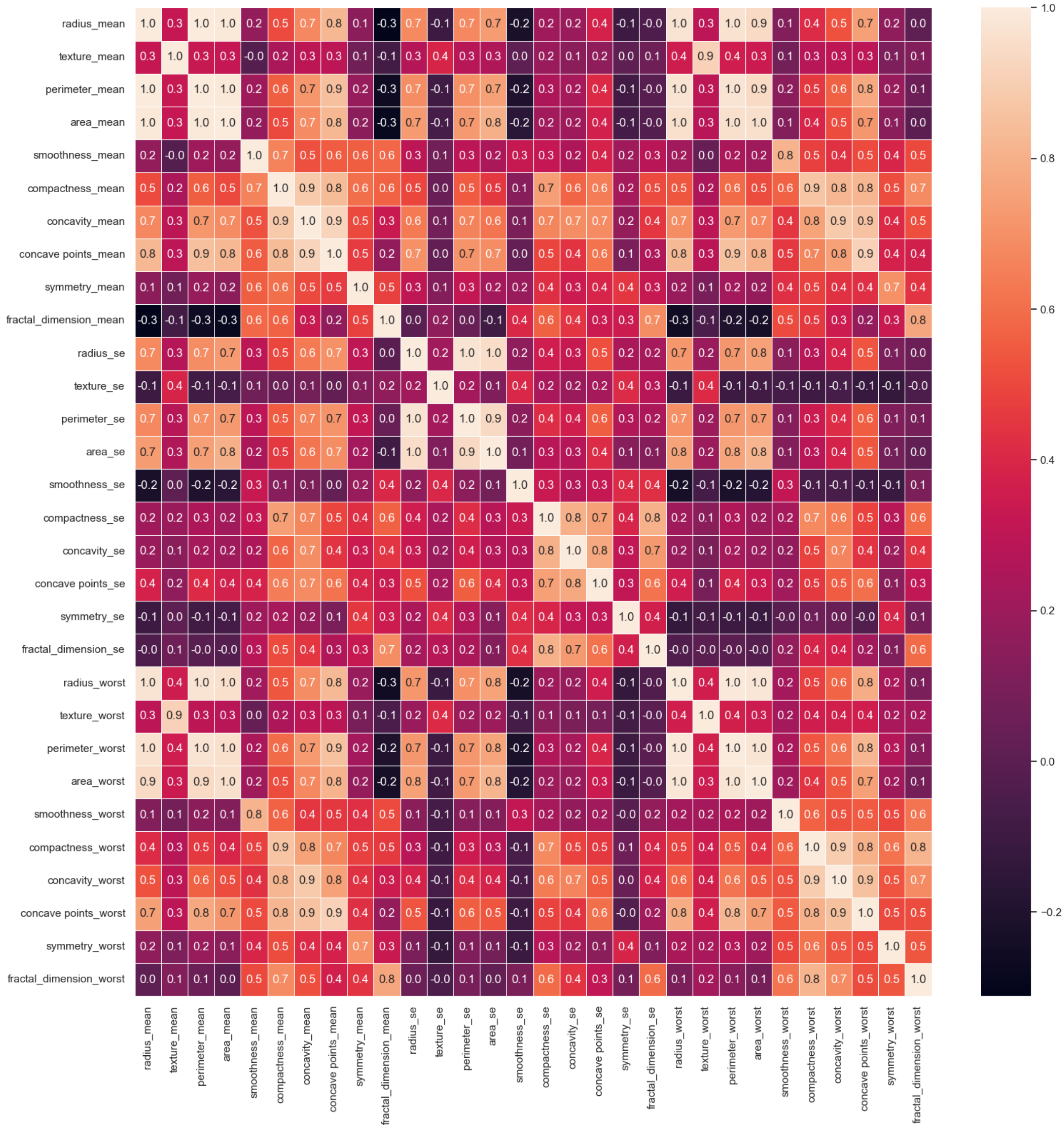
Ok, all this time we were seeing the correlations in batches, but now we want to see the correlation between all the features so we will make a correlation matrix with heat map built on top of it.

Task 8: Observing all Pair-wise Correlations

Note: If you are starting the notebook from this task, you can run cells from all the previous tasks in the kernel by going to the top menu and Kernel > Restart and Run All

In [ ]:

```
f, ax = plt.subplots(figsize=(18,18))    #30 features
sns.heatmap(x.corr(),
            annot=True,    #To see pearson-correlation annotation
            linewidth=.5,
            #small value to be able to see value instead of being shadowed by the lines
            fmt='.1f',   #1 decimal place
            ax = ax     #axis variable
           );
```



Many thanks to MarwaEshra and Instructor (Snehan Kekre)