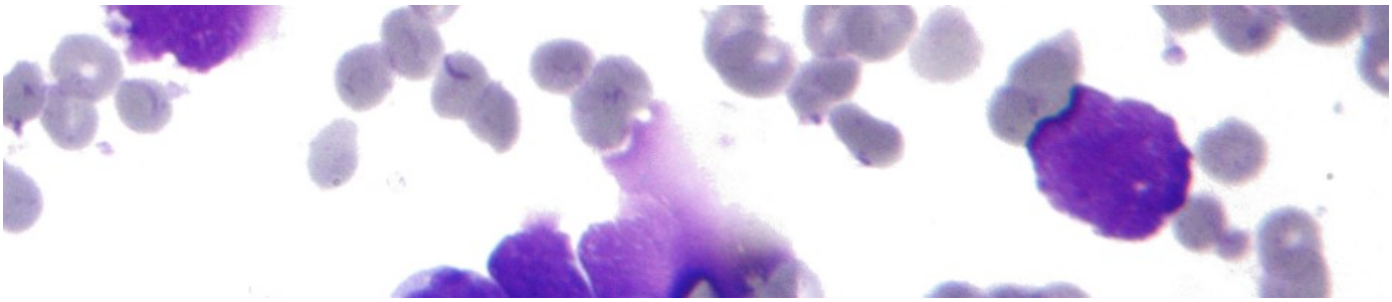
Tumor Diagnosis: Exploratory Data Analysis



About the Dataset:

The Breast Cancer Diagnostic data (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29) is available on the UCI Machine Learning Repository. This database is also available through the UW CS ftp server (http://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/cancer/WDBC/).

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.

Loading Libraries and Data

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns  # data visualization library
import matplotlib.pyplot as plt
import time
```

In [2]:

```
data = pd.read_csv('data.csv')
```

Exploratory Data Analysis

Separate Target from Features

In [3]:

```
data.head()
```

Out[3]:

an	concave points_mean	...	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness
01	0.14710	...	17.33	184.60	2019.0	0.1622	
69	0.07017	...	23.41	158.80	1956.0	0.1238	
74	0.12790	...	25.53	152.50	1709.0	0.1444	
14	0.10520	...	26.50	98.87	567.7	0.2098	
80	0.10430	...	16.67	152.20	1575.0	0.1374	

In [4]:

```
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_s
e',
       '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 [5]:

```
y = data.diagnosis
drop_cols = ['Unnamed: 32','id','diagnosis']
x = data.drop(drop_cols, axis=1)
x.head()
```

Out[5]:

l_dimension_mean	...	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_wor
0.07871	...	25.38	17.33	184.60	2019.0	0.162
0.05667	...	24.99	23.41	158.80	1956.0	0.123
0.05999	...	23.57	25.53	152.50	1709.0	0.144
0.09744	...	14.91	26.50	98.87	567.7	0.209
0.05883	...	22.54	16.67	152.20	1575.0	0.137

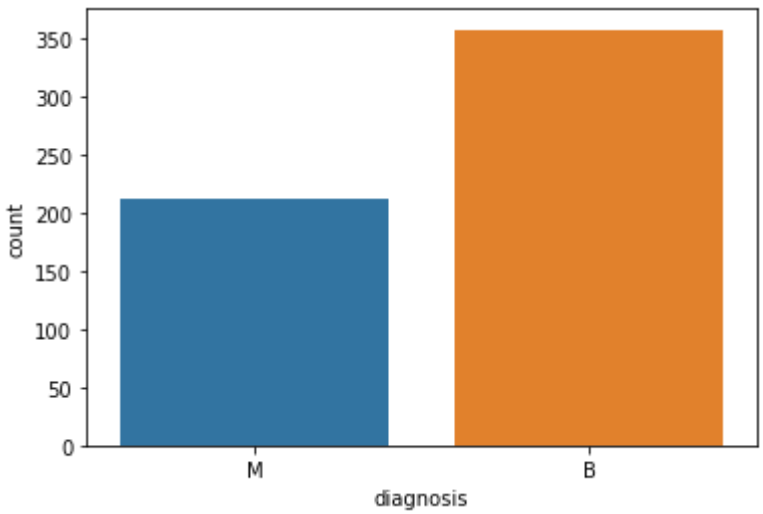
Plot Diagnosis Distributions

In [6]:

```
ax = sns.countplot(y, label="Count")
B, M = y.value_counts()
print('Number of Bengin Tumors', B)
print('Number OF Malignant Tumors', M)
```

```
Number of Bengin Tumors 357
Number OF Malignant Tumors 212

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [7]:

```
x.describe()
```

Out[7]:

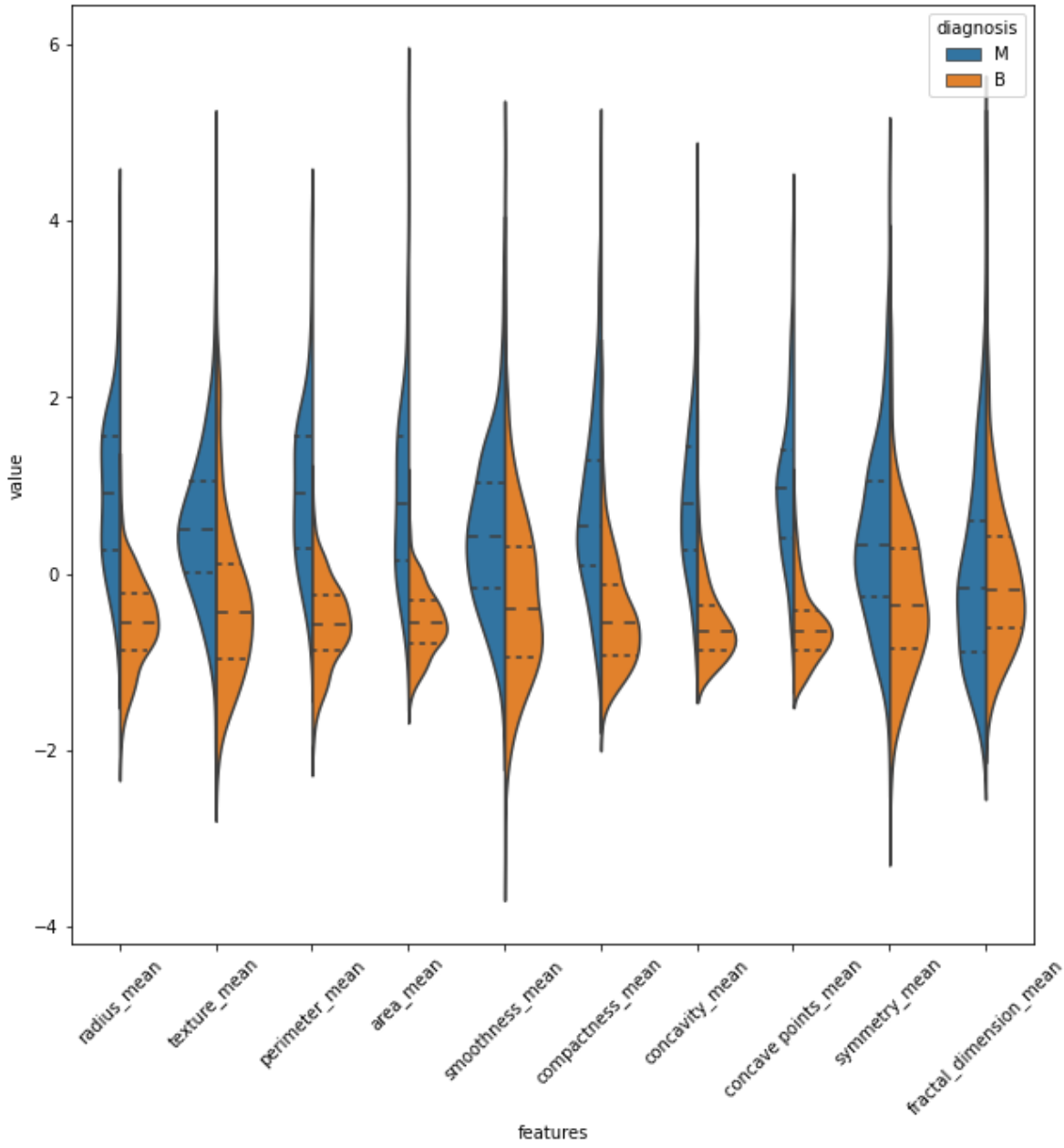
area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
654.889104	0.096360	0.104341	0.088799	0.048919	0.181162
351.914129	0.014064	0.052813	0.079720	0.038803	0.027414
143.500000	0.052630	0.019380	0.000000	0.000000	0.106000
420.300000	0.086370	0.064920	0.029560	0.020310	0.161900
551.100000	0.095870	0.092630	0.061540	0.033500	0.179200
782.700000	0.105300	0.130400	0.130700	0.074000	0.195700
501.000000	0.163400	0.345400	0.426800	0.201200	0.304000

Data Visualization

Visualizing Standardized Data with Seaborn

In [8]:

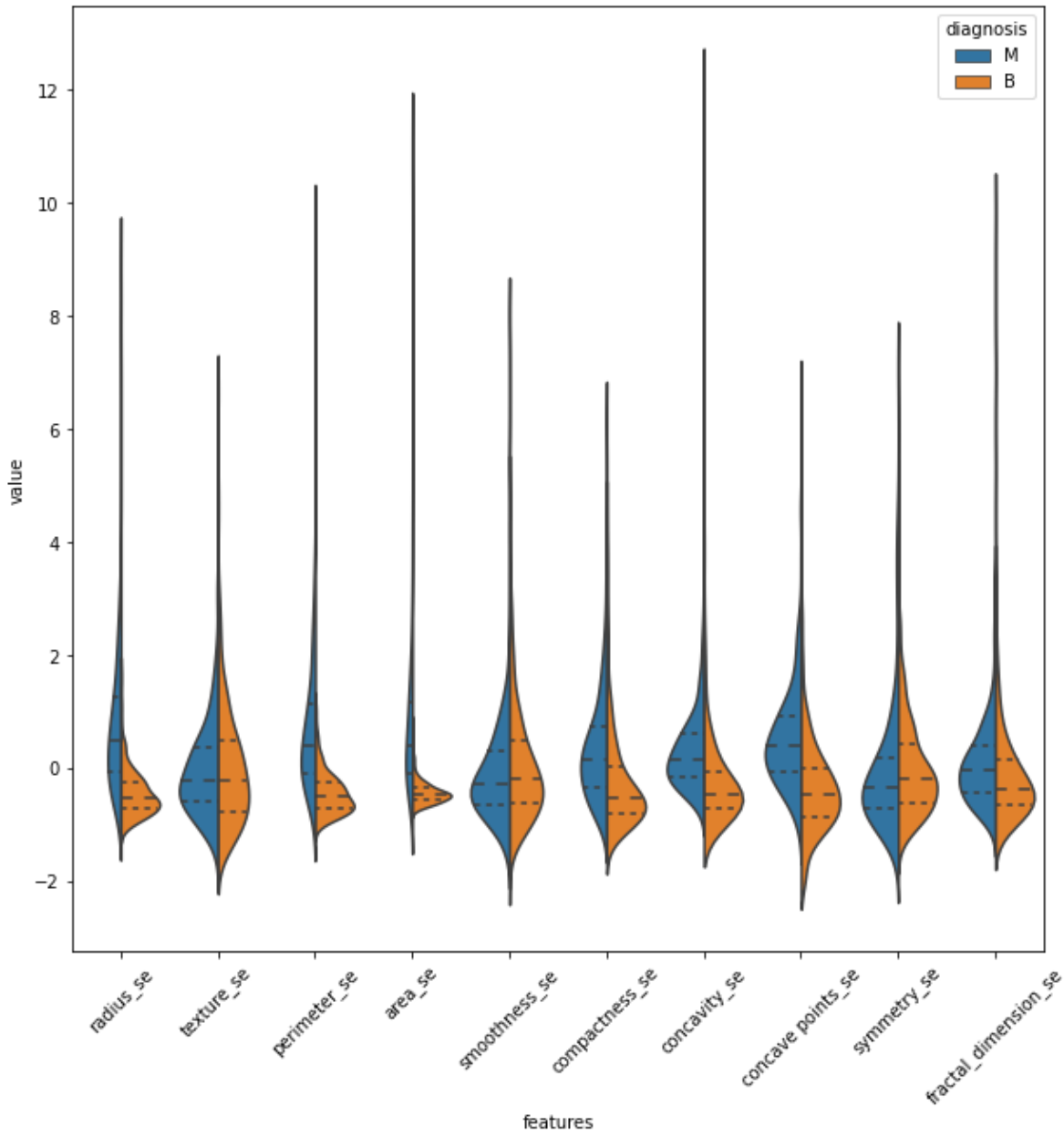
```
data = x
data_std = (data - data.mean()) / data.std()
data = pd.concat([y, data_std.iloc[:, 0:10]], axis=1)
data = pd.melt(data, id_vars='diagnosis',
               var_name='features',
               value_name='value')
plt.figure(figsize=(10, 10))
sns.violinplot(x='features', y='value', hue='diagnosis', data=data, split=True, inner='quar
plt.xticks(rotation=45);
```



Violin Plots and Box Plots

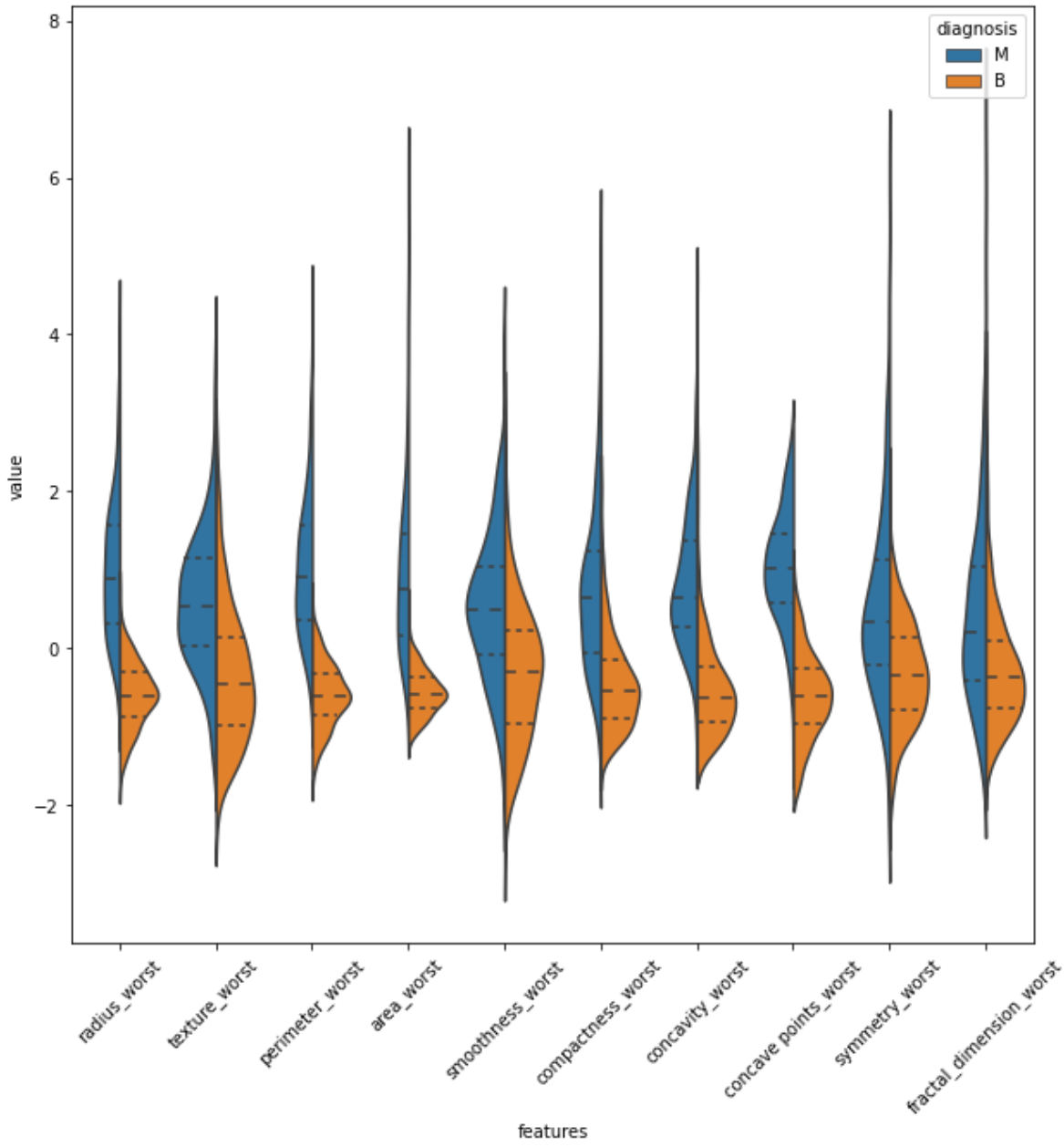
In [9]:

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



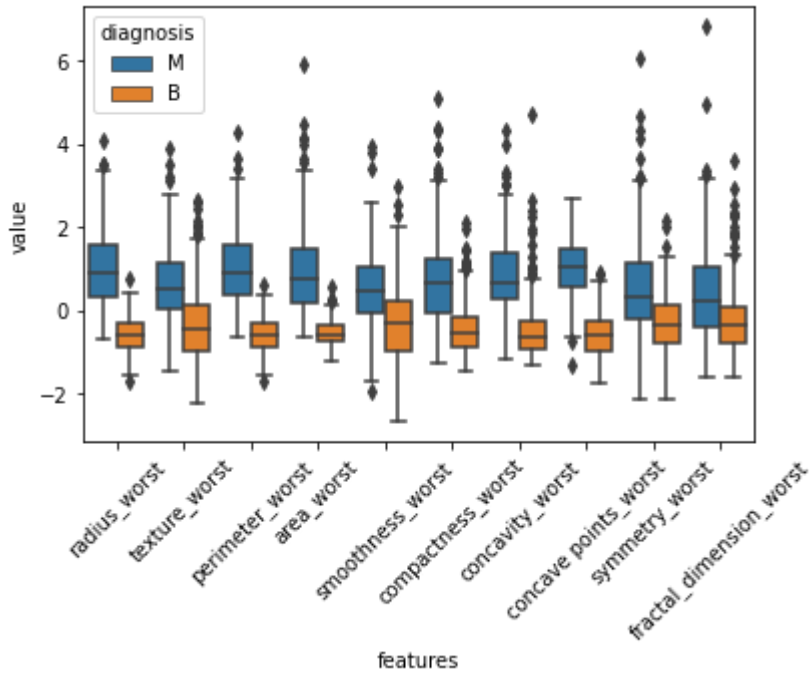
In [10]:

```
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', data=data, split=True, inner='quar
plt.xticks(rotation=45);
```



In [11]:

```
sns.boxplot(x='features', y='value', hue='diagnosis', data=data)
plt.xticks(rotation=45);
```

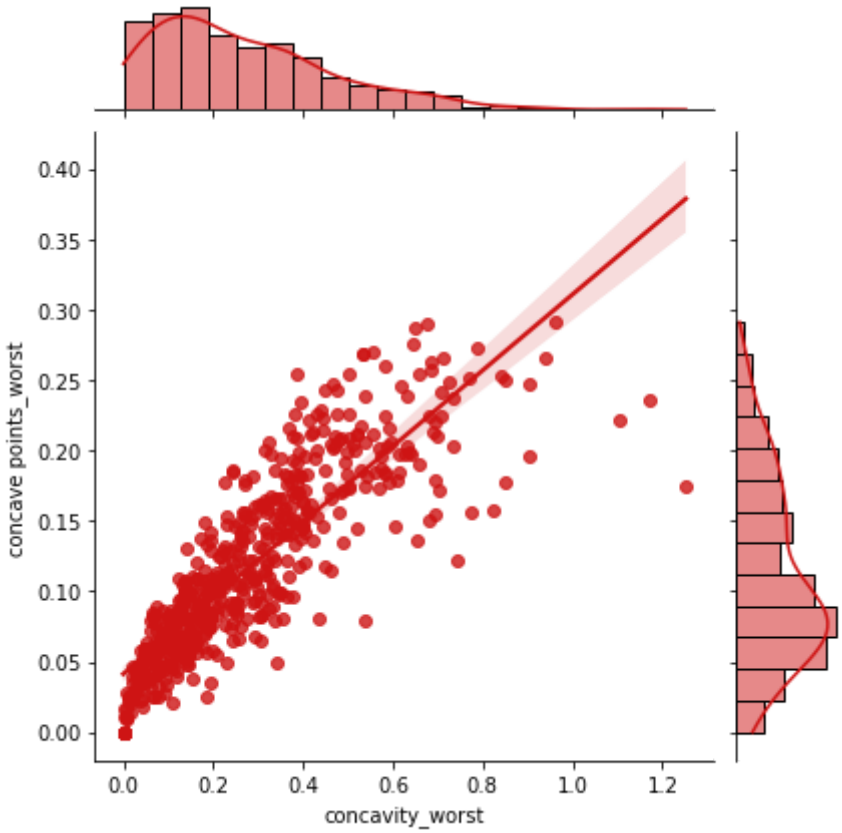


Using Joint Plots for Feature Comparison

In [12]:

```
sns.jointplot(x.loc[:, 'concavity_worst'],
              x.loc[:, 'concave points_worst'],
              kind='reg',
              color='#ce1414');
```

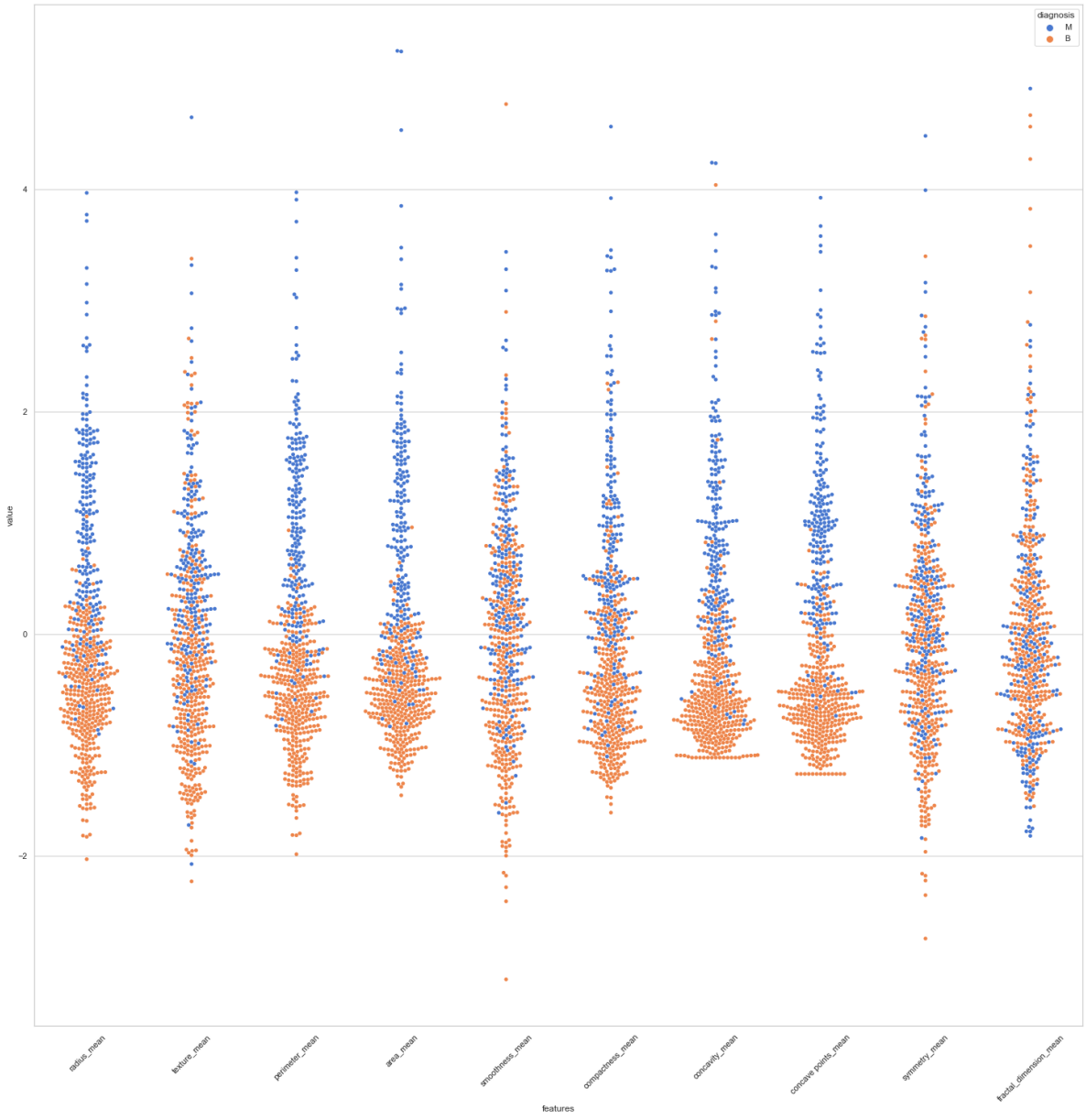
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.
12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```



Observing the Distribution of Values and their Variance with Swarm Plots

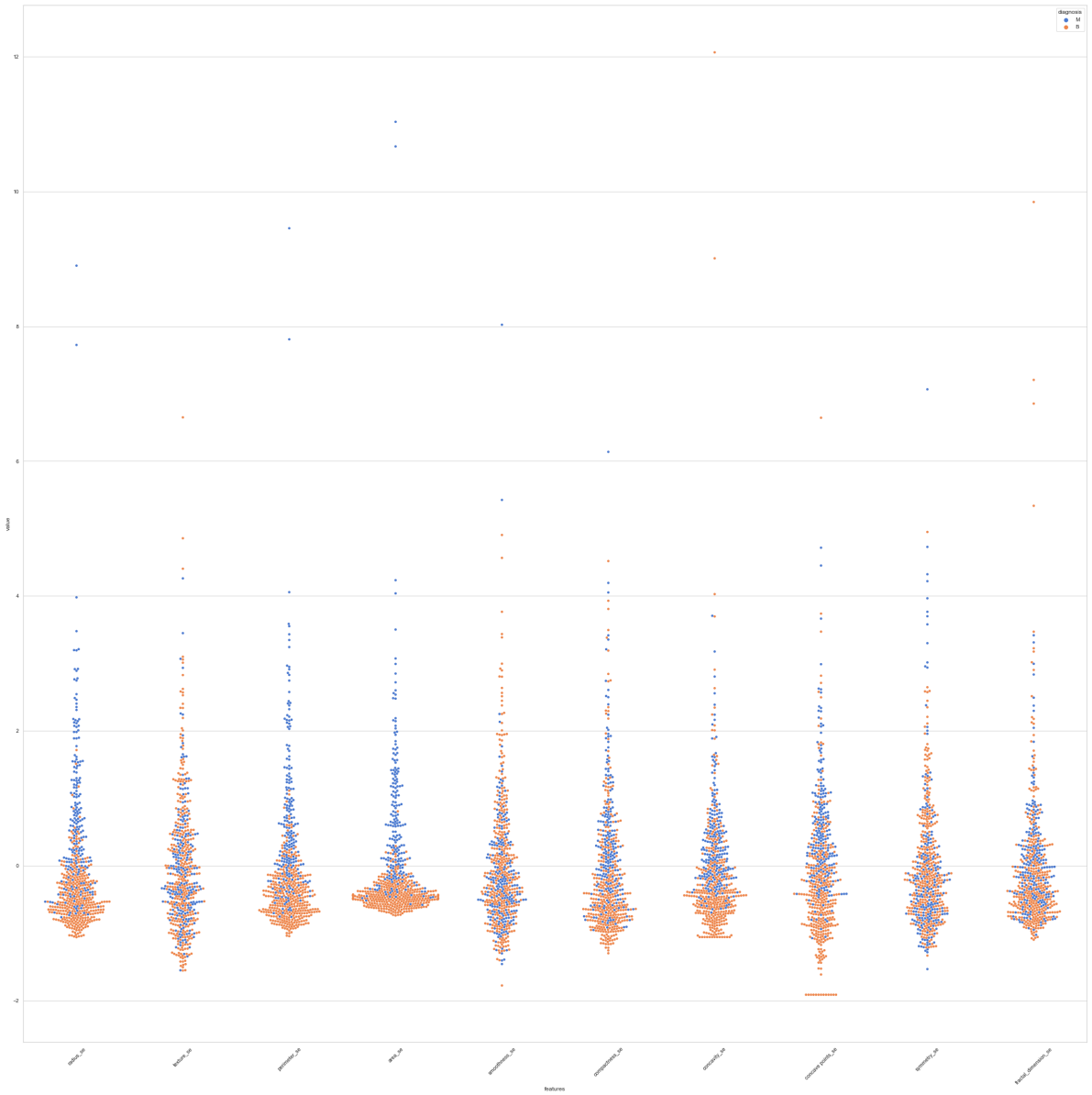
In [13]:

```
sns.set(style='whitegrid', palette='muted')
data = x
data_std = (data - data.mean()) / data.std()
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=(25, 25))
sns.swarmplot(x='features', y='value', hue='diagnosis', data=data)
plt.xticks(rotation=45);
```



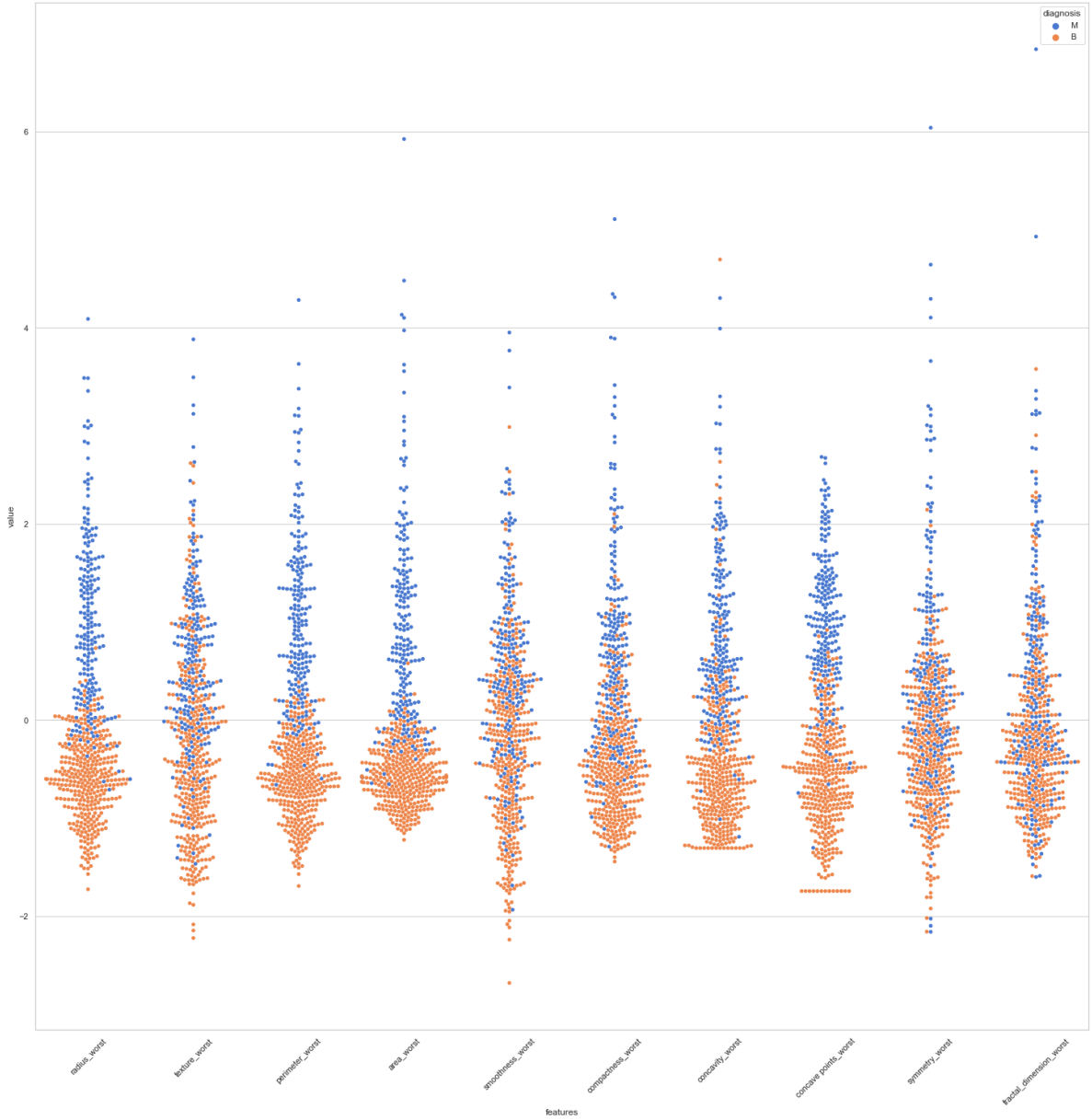
In [18]:

```
sns.set(style='whitegrid', palette='muted')
data = x
data_std = (data - data.mean()) / data.std()
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=(40, 40))
sns.swarmplot(x='features', y='value', hue='diagnosis', data=data)
plt.xticks(rotation=45);
```



In [15]:

```
sns.set(style='whitegrid', palette='muted')
data = x
data_std = (data - data.mean()) / data.std()
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=(25, 25))
sns.swarmplot(x='features', y='value', hue='diagnosis', data=data)
plt.xticks(rotation=45);
```



Observing all Pair-wise Correlations

In [16]:

```
f, ax = plt.subplots(figsize=(18,18))
sns.heatmap(x.corr(), annot=True, linewidth=.5, fmt='.1f', ax=ax);
```