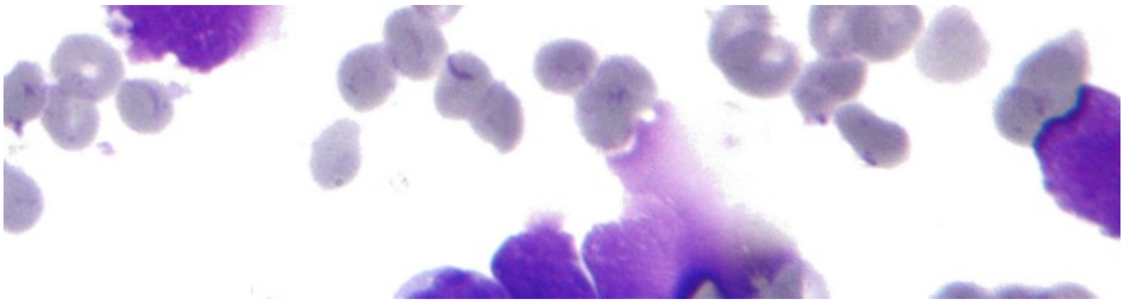
Tumor Diagnosis (Part 1): Exploratory Data Ar



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 Machi Repository. This database is also available through the UW CS ftp server (http:/ /math-prog/cpo-dataset/machine-learn/cancer/WDBC/).

Features are computed from a digitized image of a fine needle aspirate (FNA) o They describe characteristics of the cell nuclei present in the image. n the 3-di that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Program Discrimination of Two Linearly Inseparable Sets", Optimization Methods and So 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 val features were computed for each image, resulting in 30 features. For instance, 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 [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/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 in the kernel by going to the top menu and Kernel > Restart and Run All

In [3]:
```
data.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smootl
0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 33 columns

In [4]:
```
col=data.columns
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 [5]:
```
y= data.diagnosis
drop_cols=['Unnamed: 32','id','diagnosis']
x= data.drop(drop_cols, axis=1)
x.head()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compac
0	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	11.42	20.38	77.58	386.1	0.14250	0.28390
4	20.29	14.34	135.10	1297.0	0.10030	0.13280

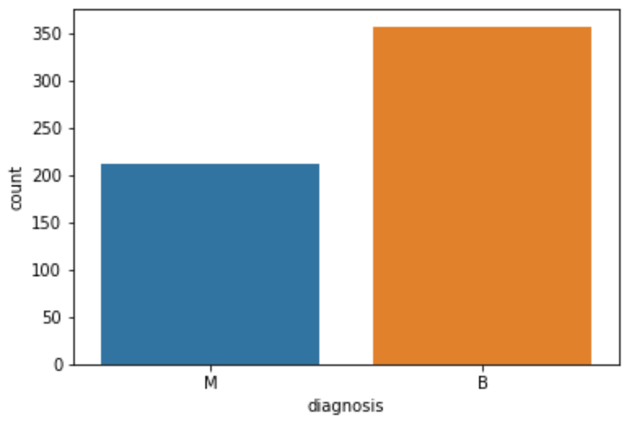
```
5 rows × 30 columns
```

Task 3: Plot Diagnosis Distributions

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

In [6]:
```
ax= sns.countplot(y, label='Count')
B, M= y.value_counts()
print('Number of Benign Tumours', B)
print('Number of Malignant Tumours', M)
```

```
Number of Benign Tumours 357
Number of Malignant Tumours 212
```



In [8]:
```
y.value_counts()
```

```
B    357
M    212
Name: diagnosis, dtype: int64
```

In [9]:
```
x.describe()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	con
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.10
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.05
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.01
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.06
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.09
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.13
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.34

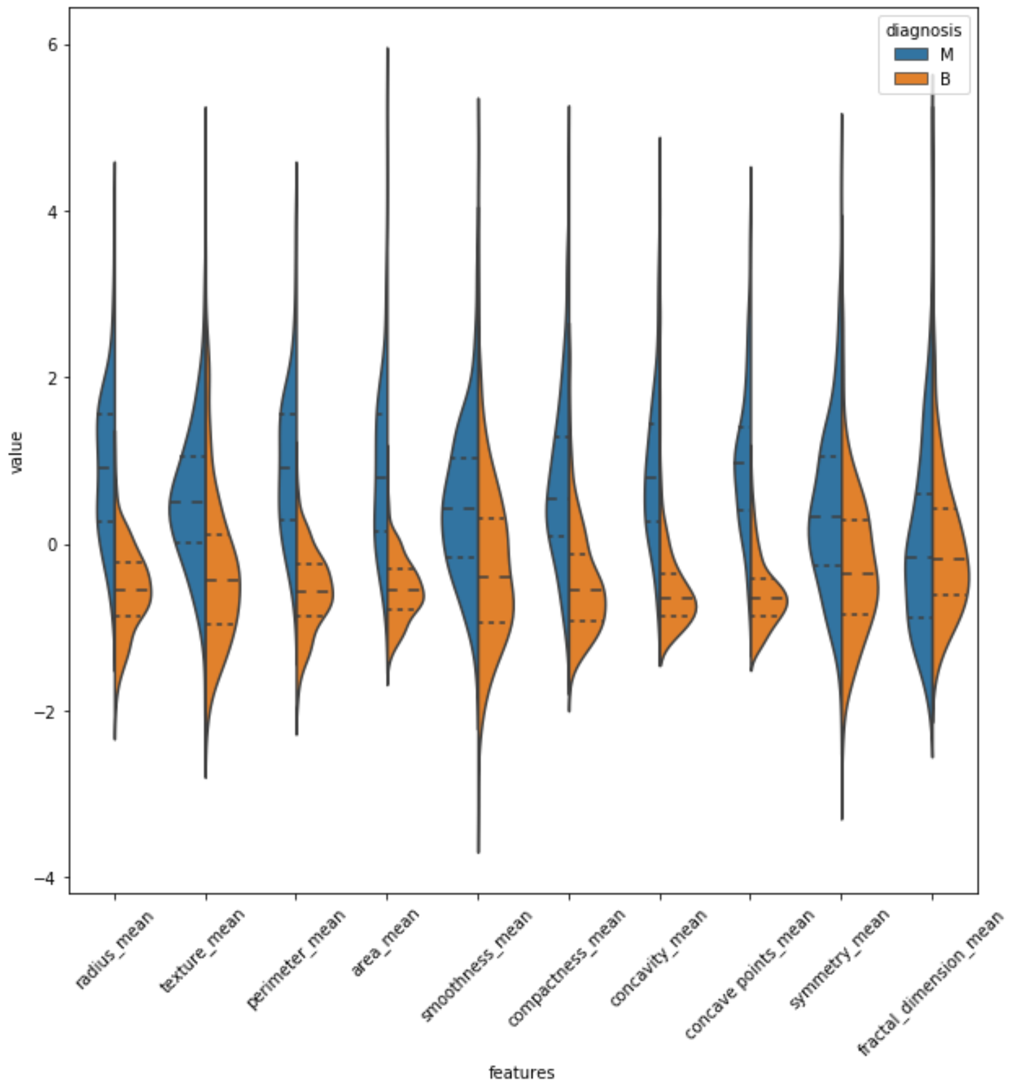
8 rows × 30 columns

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 in the kernel by going to the top menu and Kernel > Restart and Run All

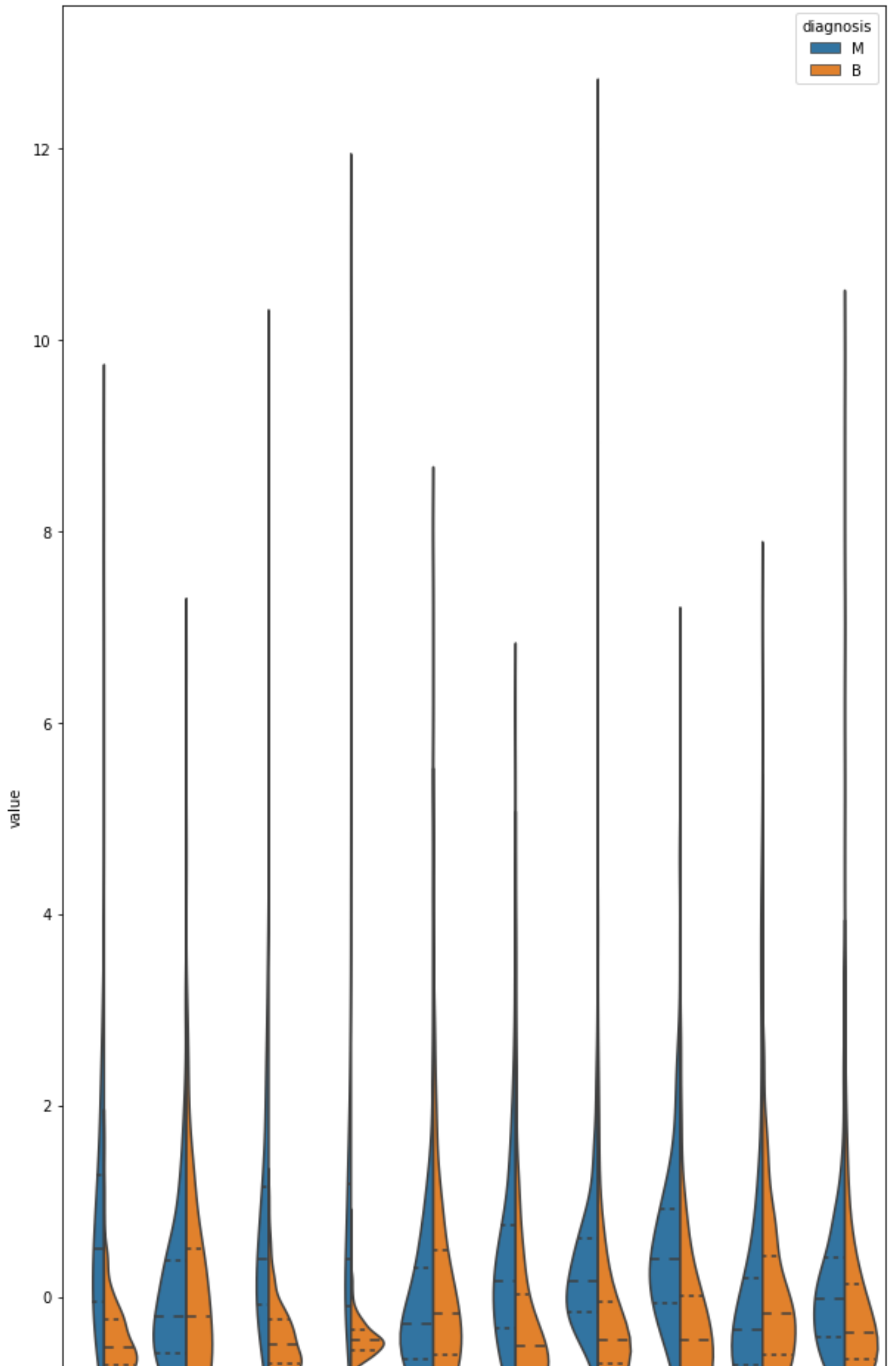
In [10]:
```
data= x
data_std= (data - data.mean())/ data.std()
data= pd.concat([y, data_std.iloc[:, 0:10]], axis=1)
data= pd.melt(data, id_vars='diagnosis',
              var_name='features',
              value_name='value')#convert data to right format
plt.figure(figsize=(10,10))
sns.violinplot(x='features',y='value',hue='diagnosis',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



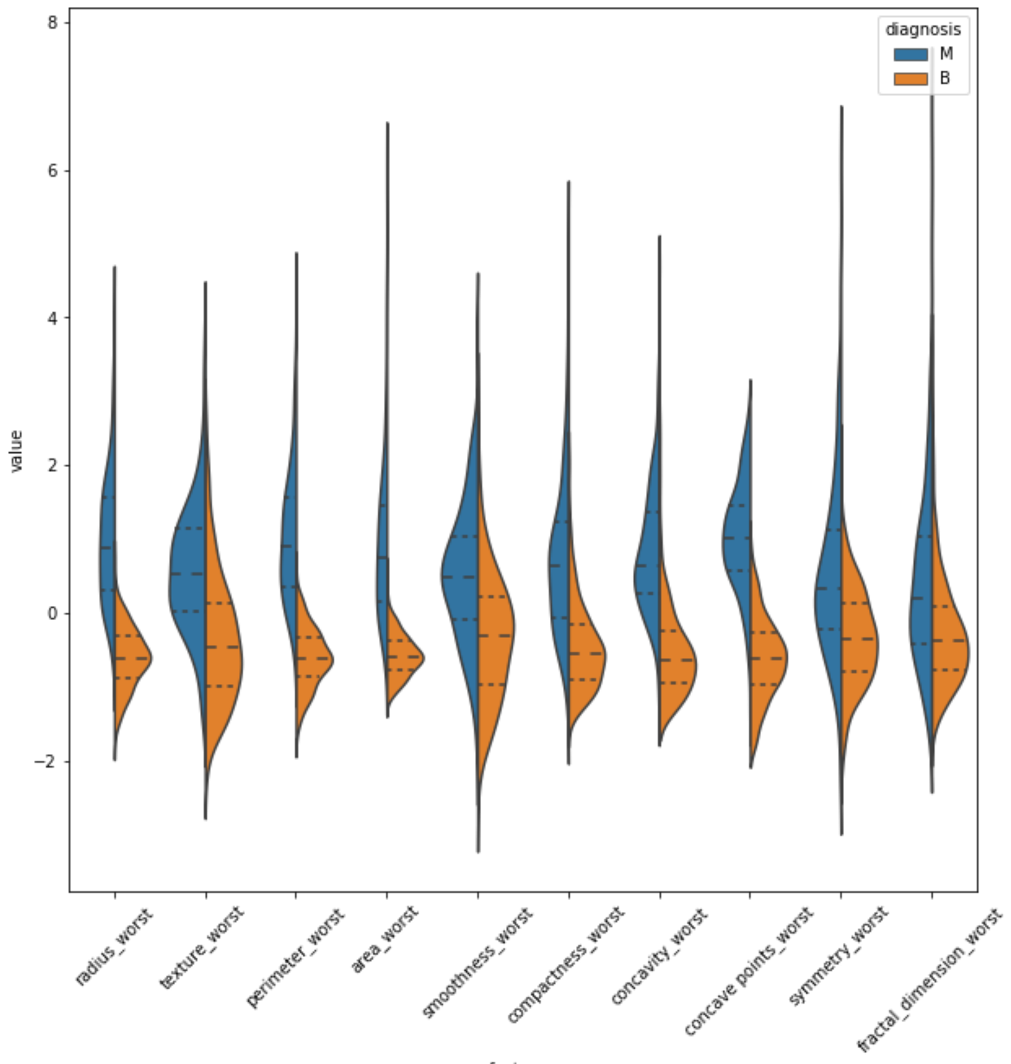
Task 5: Violin Plots and Box Plots

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

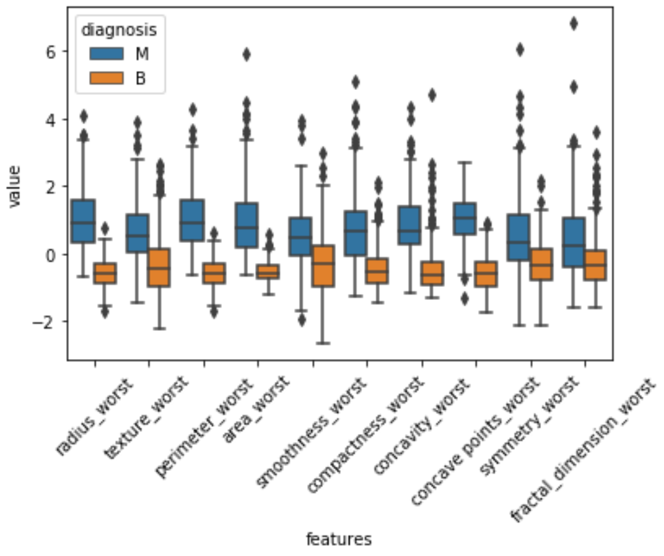
In [13]:
```
data= pd.concat([y, data_std.iloc[:, 10:20]], axis=1)
data= pd.melt(data, id_vars='diagnosis',
             var_name='features',
             value_name='value')#convert data to right format
plt.figure(figsize=(10,20))
sns.violinplot(x='features',y='value',hue='diagnosis', data=data, split=True,
              inner='quart')
plt.xticks(rotation=45);
```



```
In [14]:  data= pd.concat([y, data_std.iloc[:, 20:30]], axis=1)
          data= pd.melt(data, id_vars='diagnosis',
                        var_name='features',
                        value_name='value')#convert data to right format
          plt.figure(figsize=(10,10))
          sns.violinplot(x='features',y='value',hue='diagnosis', data=data, split=True,
                         inner='quart')
          plt.xticks(rotation=45);
```



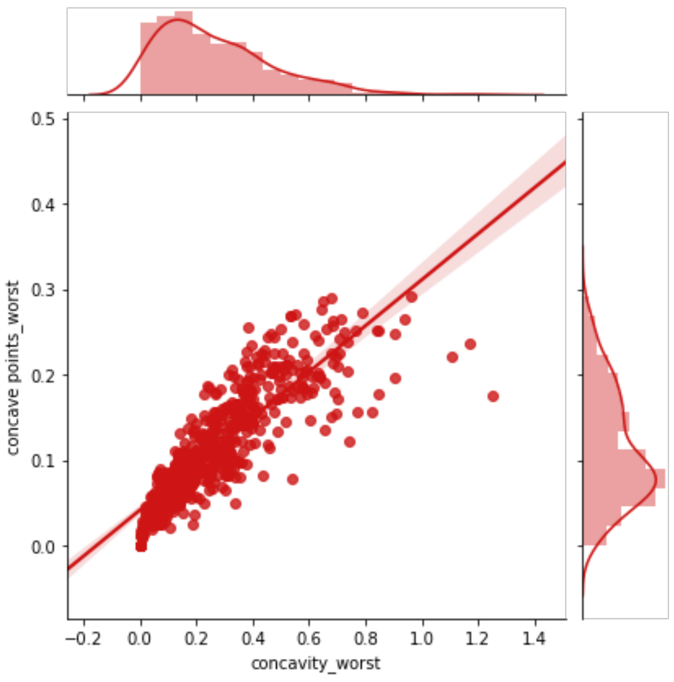
In [16]:
```
sns.boxplot(x='features',y='value',hue='diagnosis',data=data)
#hue to color the points by a specific variable
plt.xticks(rotation=45);
```



Task 6: Using Joint Plots for Feature Comparison

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

In [19]:
```
sns.jointplot(x.loc[:, 'concavity_worst'],
              x.loc[:, 'concave points_worst'],
              kind='regg',
              color='#ce1414');#regression
```

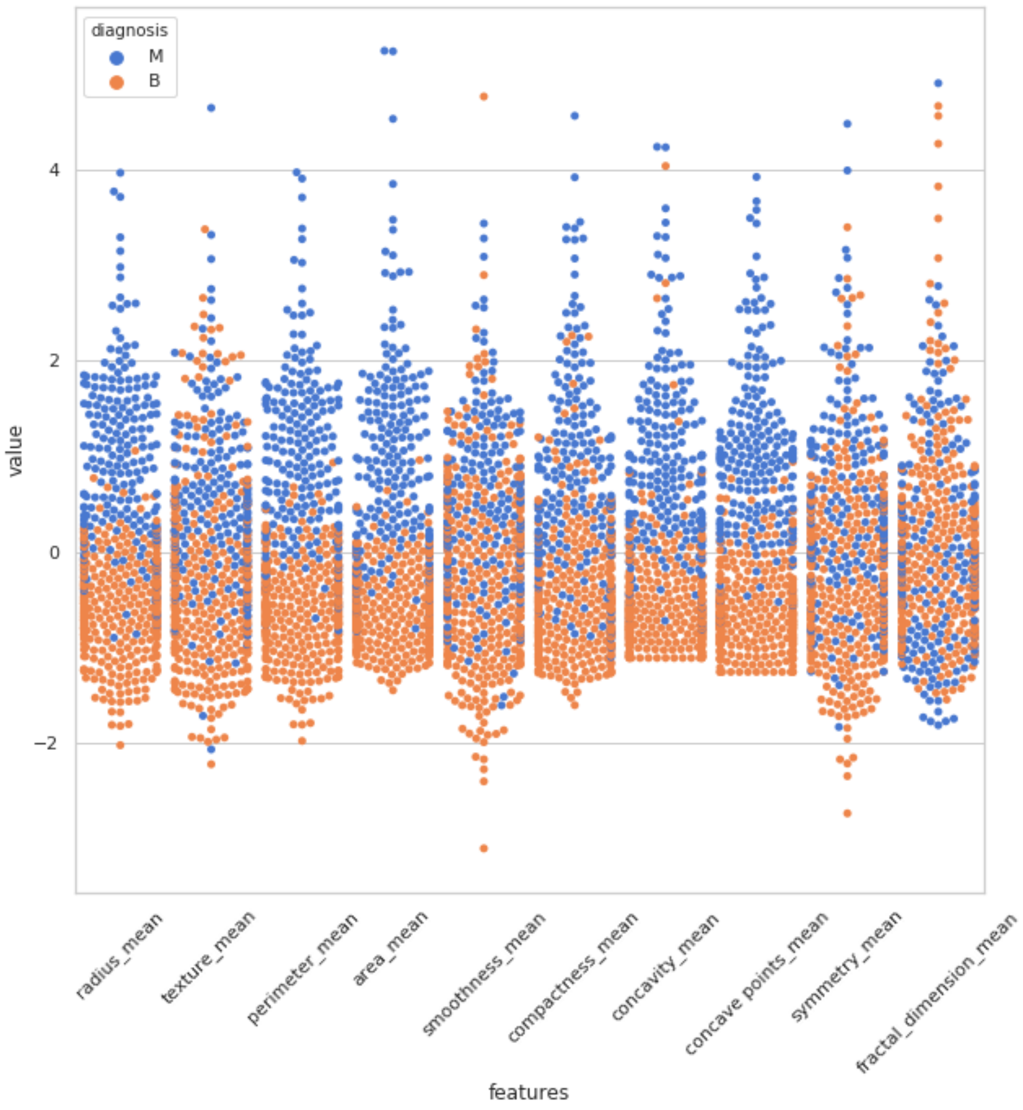


Task 7: Observing the Distribution of Values and their Va
Swarm Plots

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

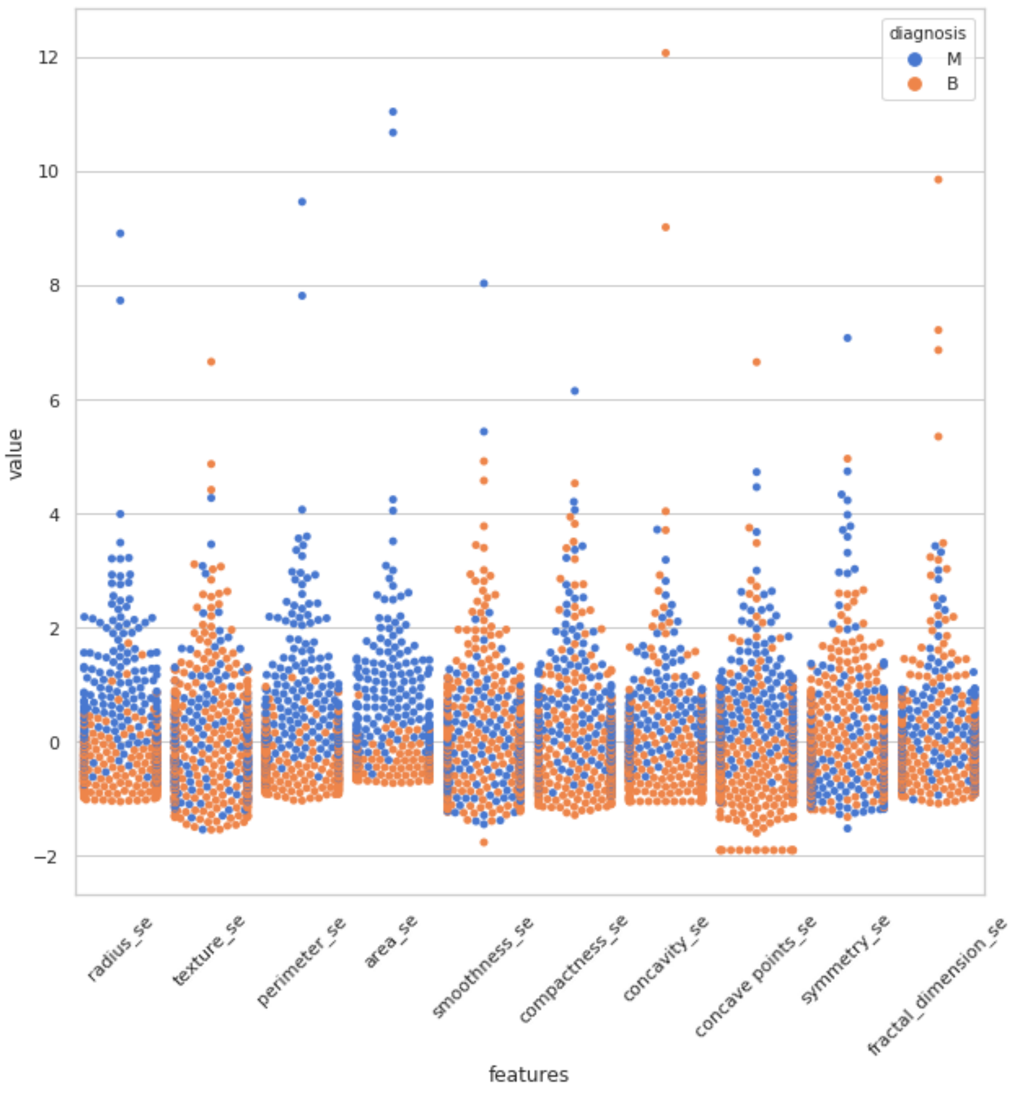
In [20]:

```
sns.set(style='whitegrid', palette='muted')
data= x
data_std= (data - data.mean())/ data.std() #normalizing
data= pd.concat([y, data_std.iloc[:, 0:10]], axis=1)
data= pd.melt(data, id_vars='diagnosis',
             var_name='features',
             value_name='value')#convert data to right format
plt.figure(figsize=(10,10))
sns.swarmplot(x='features',y='value',hue='diagnosis', data=data)
plt.xticks(rotation=45);
```

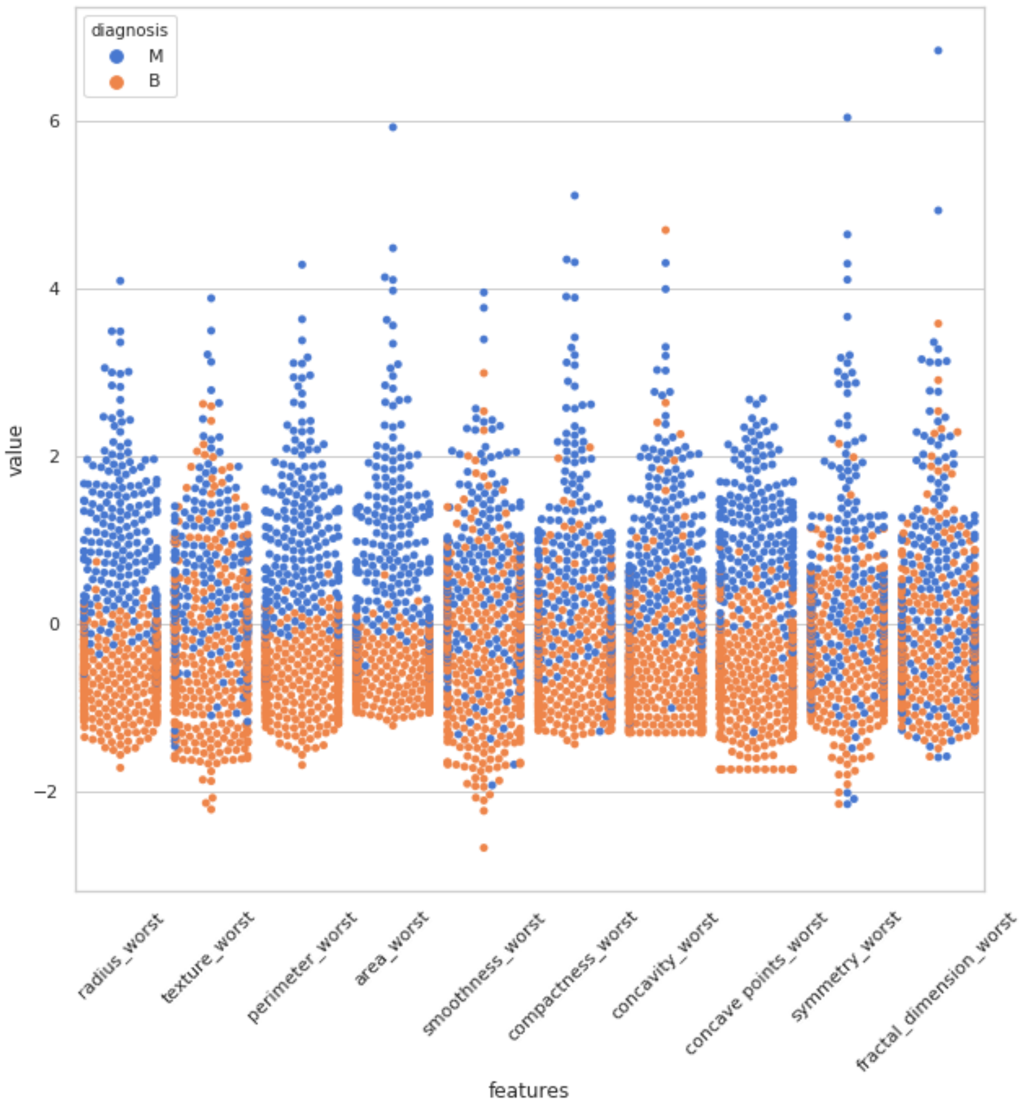


In [21]:

```
sns.set(style='whitegrid', palette='muted')
data= x
data_std= (data - data.mean())/ data.std() #normalizing
data= pd.concat([y, data_std.iloc[:, 10:20]], axis=1)
data= pd.melt(data, id_vars='diagnosis',
              var_name='features',
              value_name='value')#convert data to right format
plt.figure(figsize=(10,10))
sns.swarmplot(x='features',y='value',hue='diagnosis', data=data)
plt.xticks(rotation=45);
```



In [22]:
```
sns.set(style='whitegrid', palette='muted')
data= x
data_std= (data - data.mean())/ data.std() #normalizing
data= pd.concat([y, data_std.iloc[:, 20:30]], axis=1)
data= pd.melt(data, id_vars='diagnosis',
              var_name='features',
              value_name='value')#convert data to right format
plt.figure(figsize=(10,10))
sns.swarmplot(x='features',y='value',hue='diagnosis', data=data)
plt.xticks(rotation=45);
```

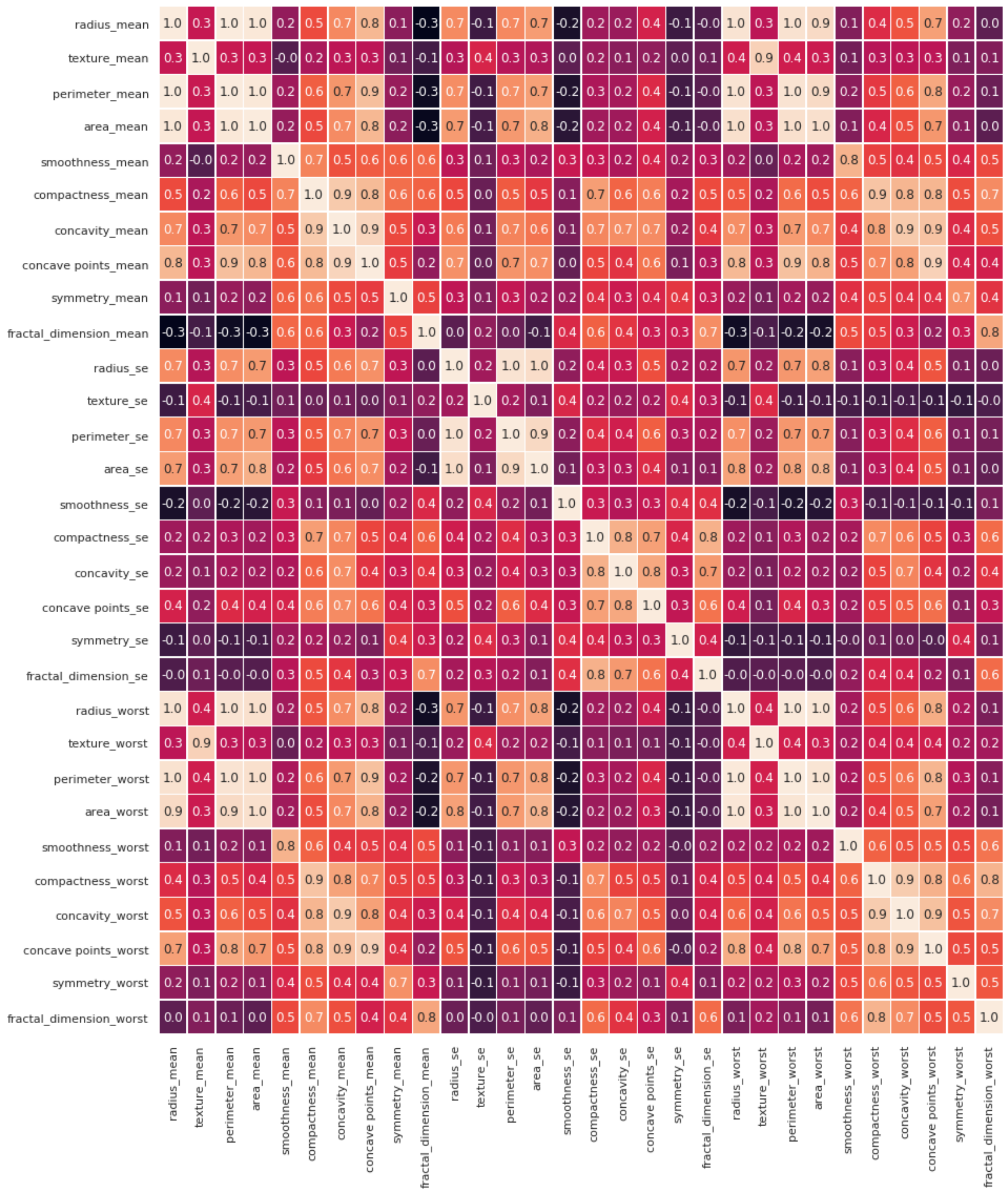


Task 8: Observing all Pair-wise Correlations

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

In [23]:

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


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