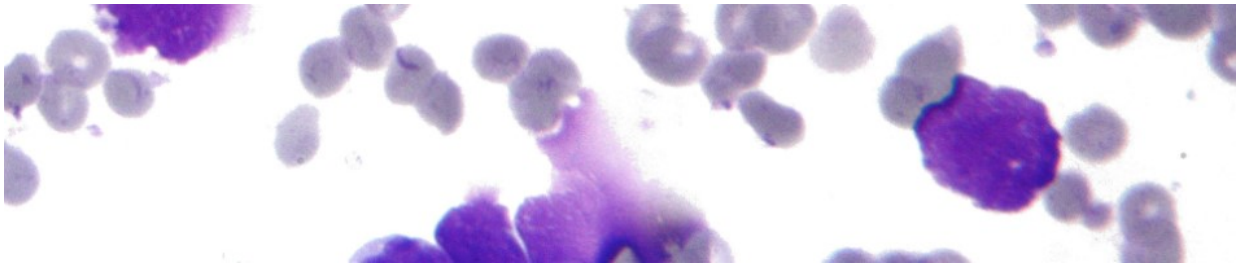
Tumor Diagnosis : Exploratory Data Analysis



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.

Missing attribute values: none

Class distribution: 357 benign, 212 malignant

Task: 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: Separate Target from Features

In [3]:

```
data.head()
data.shape
```

Out[3]:

```
(569, 33)
```

In [4]:

```
col = data.columns
print(col)
```

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimete
r_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concav
ity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_
mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoo
thness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'sym
metry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_wor
st',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 3
2'],
      dtype='object')
```

In [5]:

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

Out[5]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	coı
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	

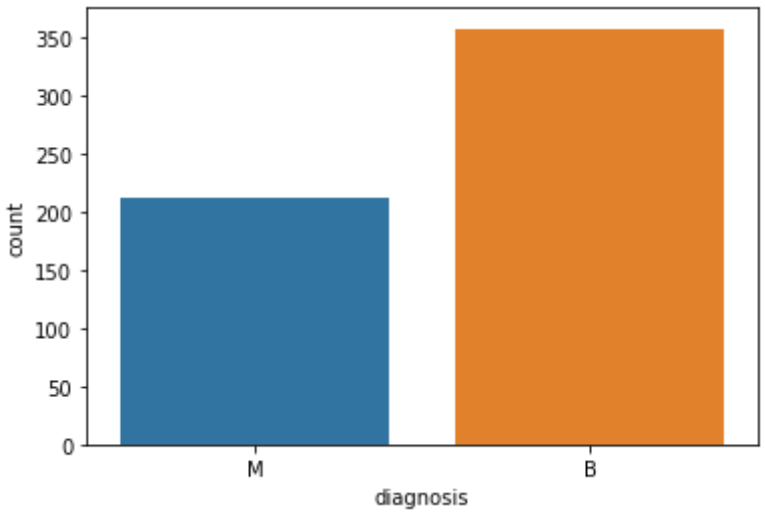
5 rows × 30 columns

Task: Plot Diagnosis Distributions

In [6]:

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

```
Number of Benign Tumors 357
Number of Malignant Tumors 212
```



In [7]:

```
x.describe()
```

Out[7]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
count	569.000000	569.000000	569.000000	569.000000	569.00000
mean	14.127292	19.289649	91.969033	654.889104	0.09636
std	3.524049	4.301036	24.298981	351.914129	0.01406
min	6.981000	9.710000	43.790000	143.500000	0.05263
25%	11.700000	16.170000	75.170000	420.300000	0.08637
50%	13.370000	18.840000	86.240000	551.100000	0.09587
75%	15.780000	21.800000	104.100000	782.700000	0.10530
max	28.110000	39.280000	188.500000	2501.000000	0.16340

8 rows × 30 columns

Data Visualization

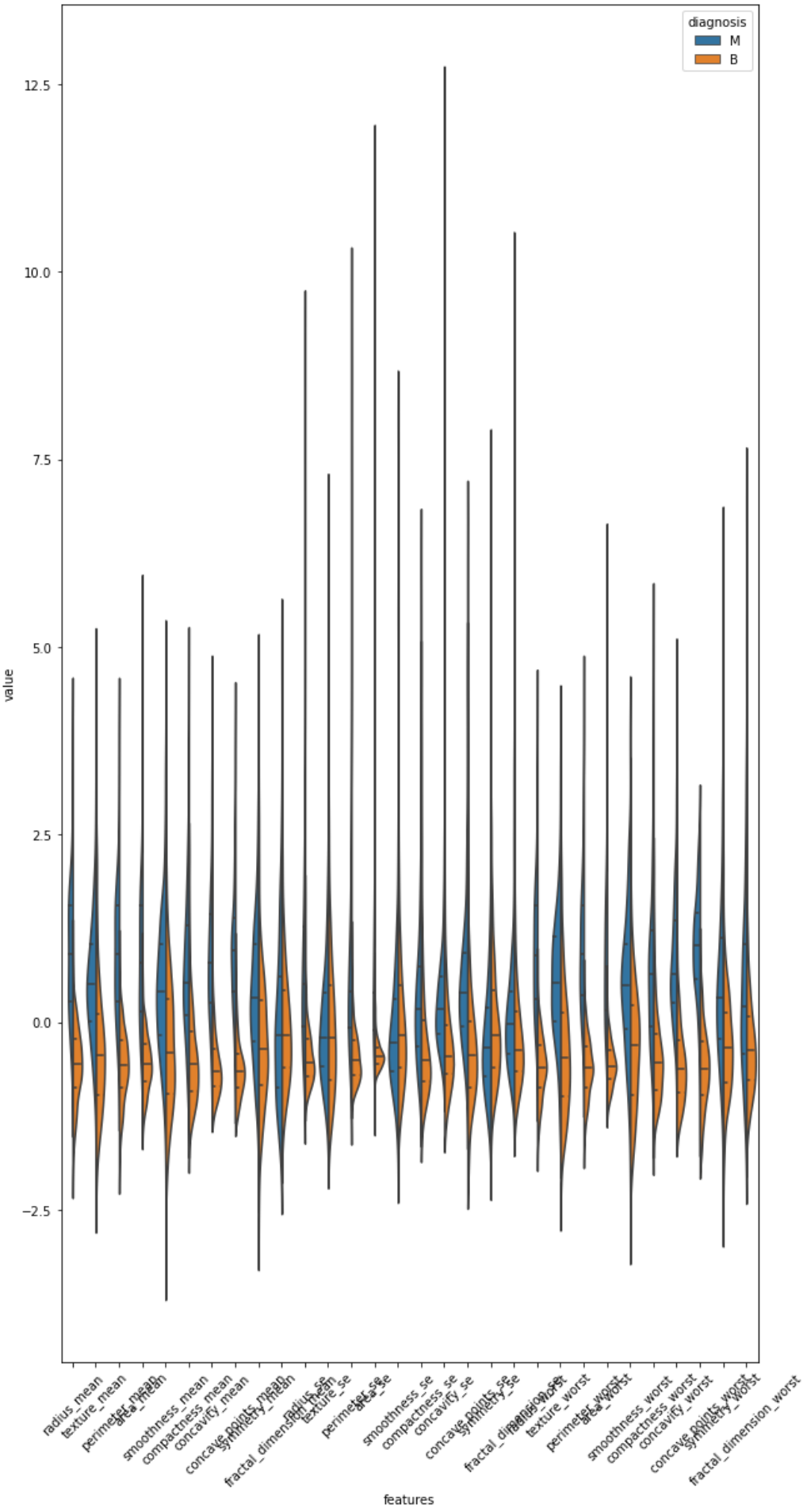
Task: Visualizing Standardized Data with Seaborn

In [8]:

```
data = x
data_std = (data - data.mean()) / data.std()
data = pd.concat([y, data_std.iloc[:, 0 : 30]], axis = 1)
data = pd.melt(data, id_vars="diagnosis",
               var_name = "features",
               value_name = "value")
plt.figure(figsize = (10,20))
sns.violinplot(x="features", y = "value", hue = "diagnosis",
                data = data, split = True, inner = "quart")
plt.xticks(rotation = 45)
```

Out[8]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 1
4, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]),
 <a list of 30 Text major ticklabel objects>)
```

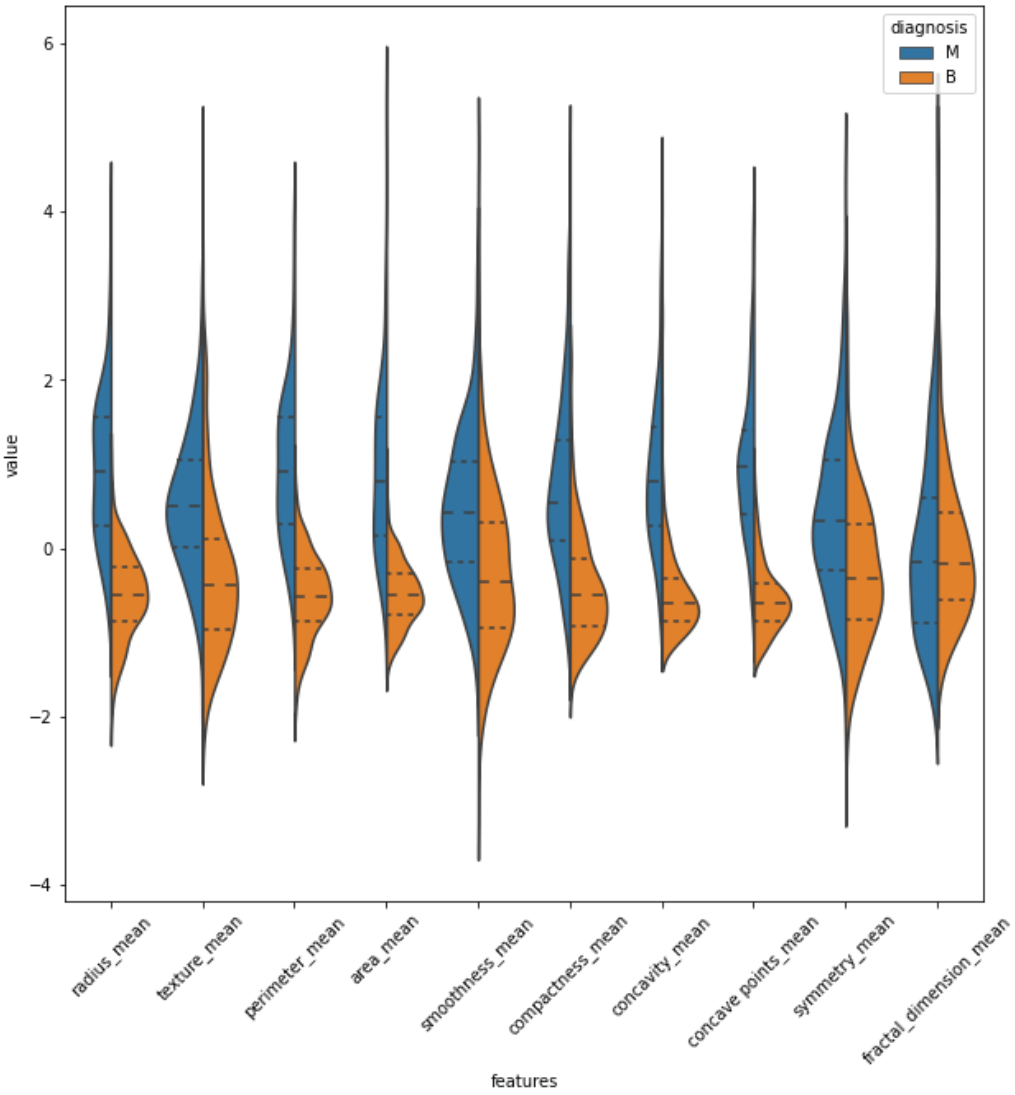

diagnosis
M
B
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
value
features
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave 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

In [9]:

```
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 = 45)
```

Out[9]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 <a list of 10 Text major ticklabel objects>)
```



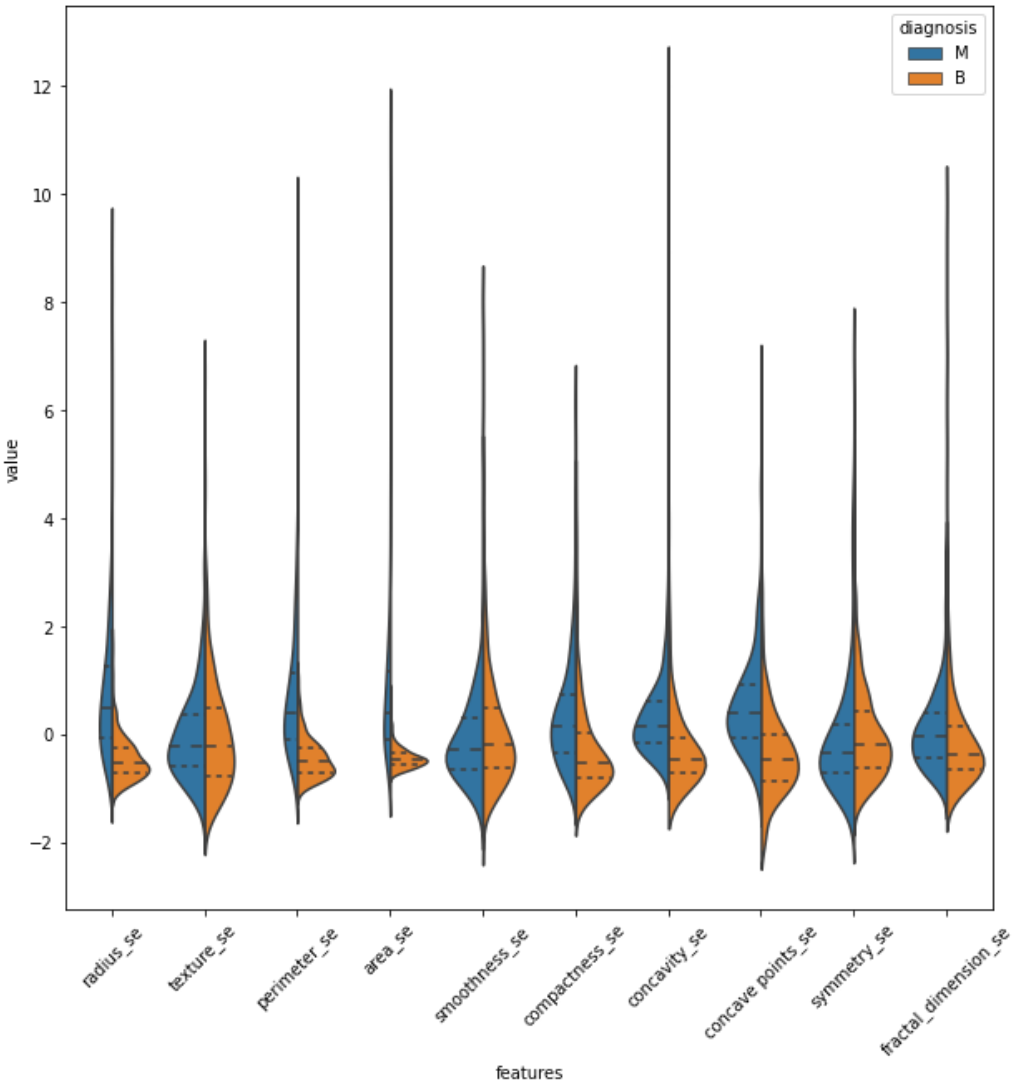
Task: Violin Plots and Box Plots

In [10]:

```
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 = 45)
```

Out[10]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 <a list of 10 Text major ticklabel objects>)
```

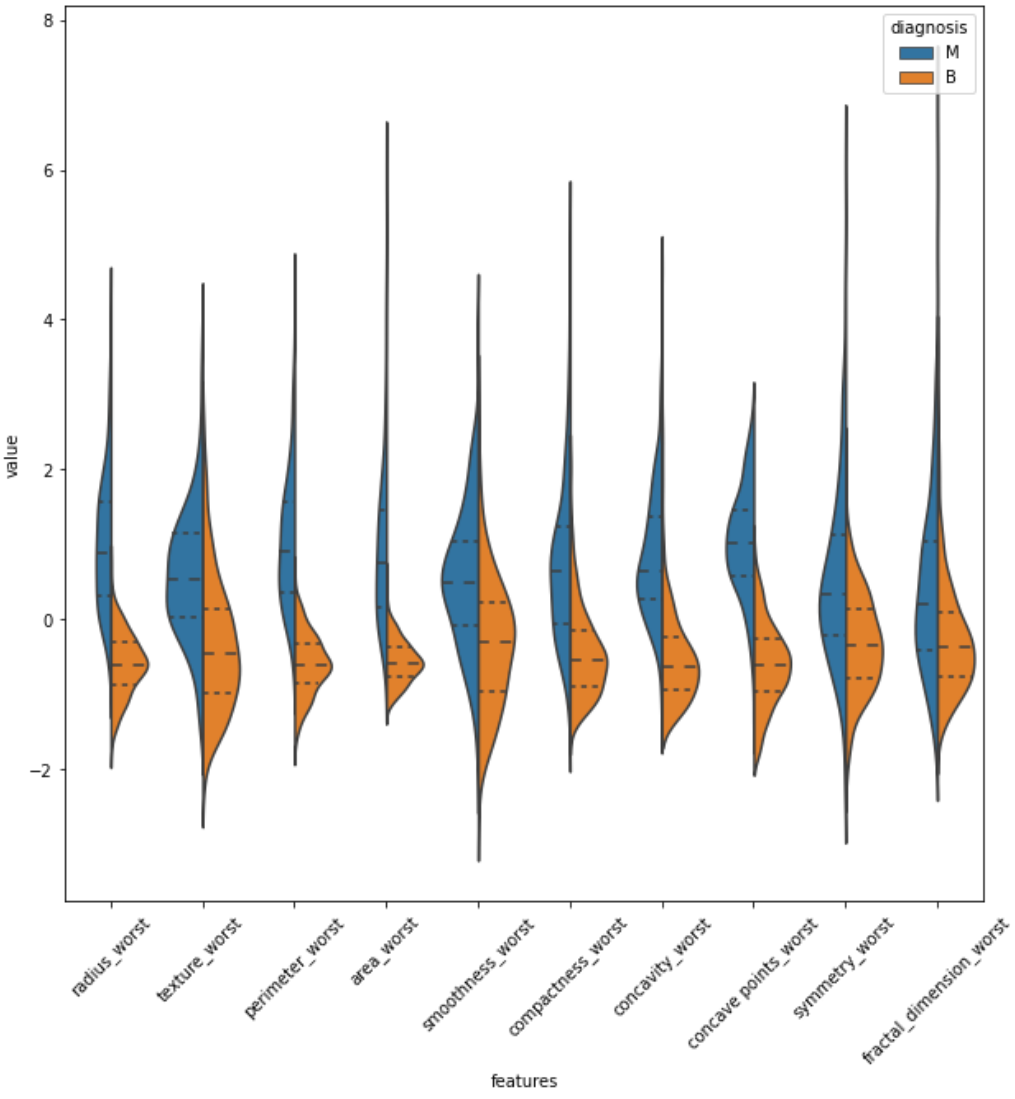


In [11]:

```
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 = "quart")
plt.xticks(rotation = 45)
```

Out[11]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 <a list of 10 Text major ticklabel objects>)
```

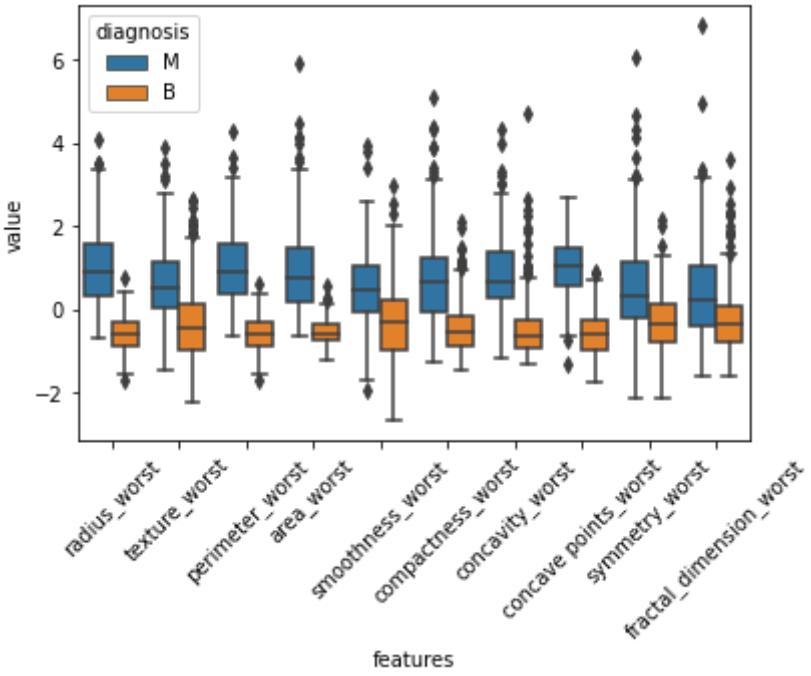


In [12]:

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

Out[12]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 <a list of 10 Text major ticklabel objects>)
```



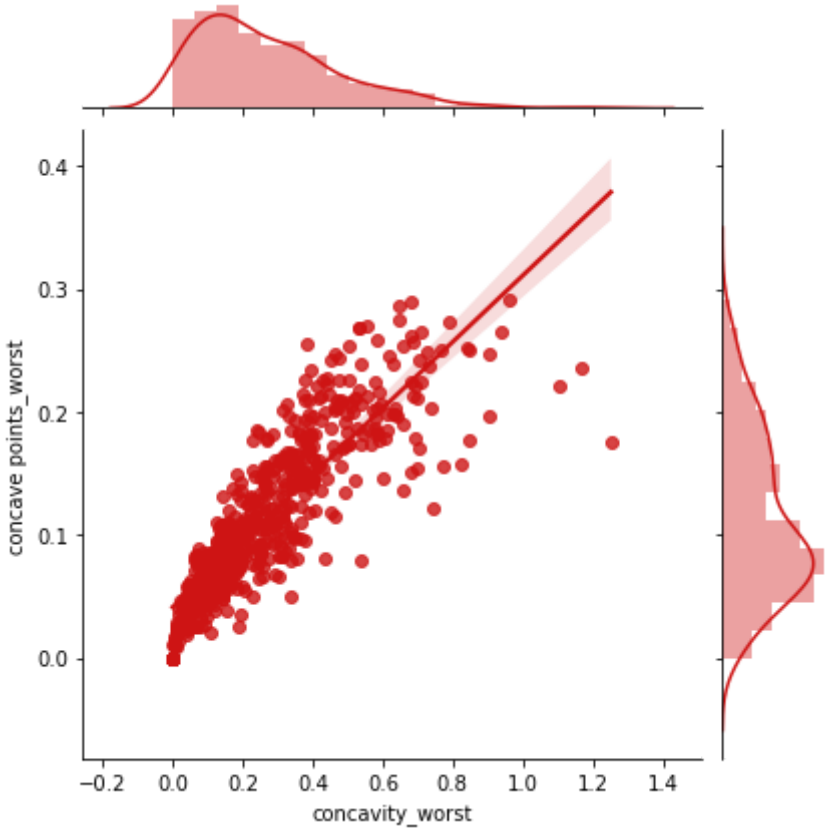
Task: Using Joint Plots for Feature Comparison

In [13]:

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

Out[13]:

```
<seaborn.axisgrid.JointGrid at 0x2c95f4d8b50>
```



In [ ]:

```

```

Task: 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

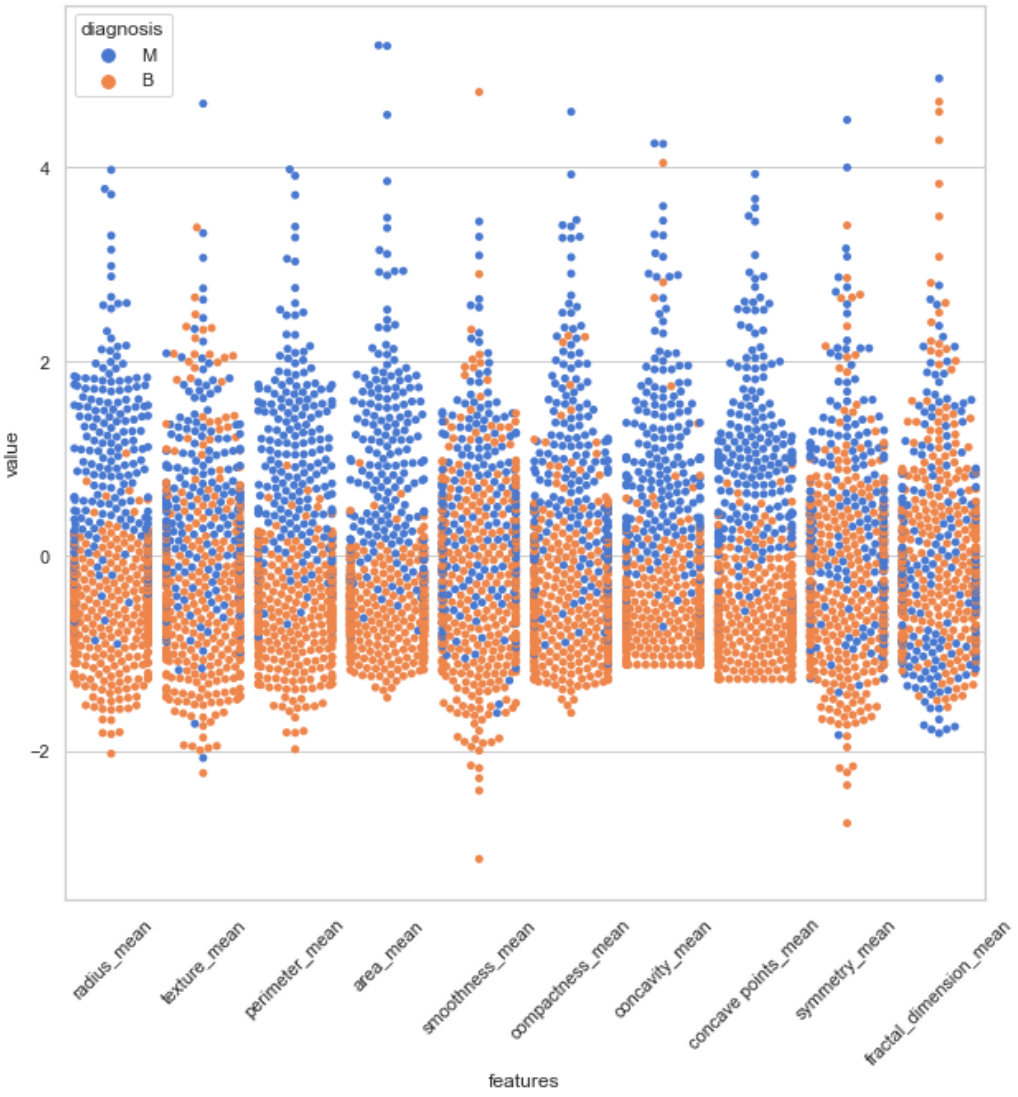
---

In [14]:

```
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 = (10,10))
sns.swarmplot(x="features", y = "value", hue = "diagnosis", data = data)
plt.xticks(rotation = 45)
```

Out[14]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 <a list of 10 Text major ticklabel objects>)
```

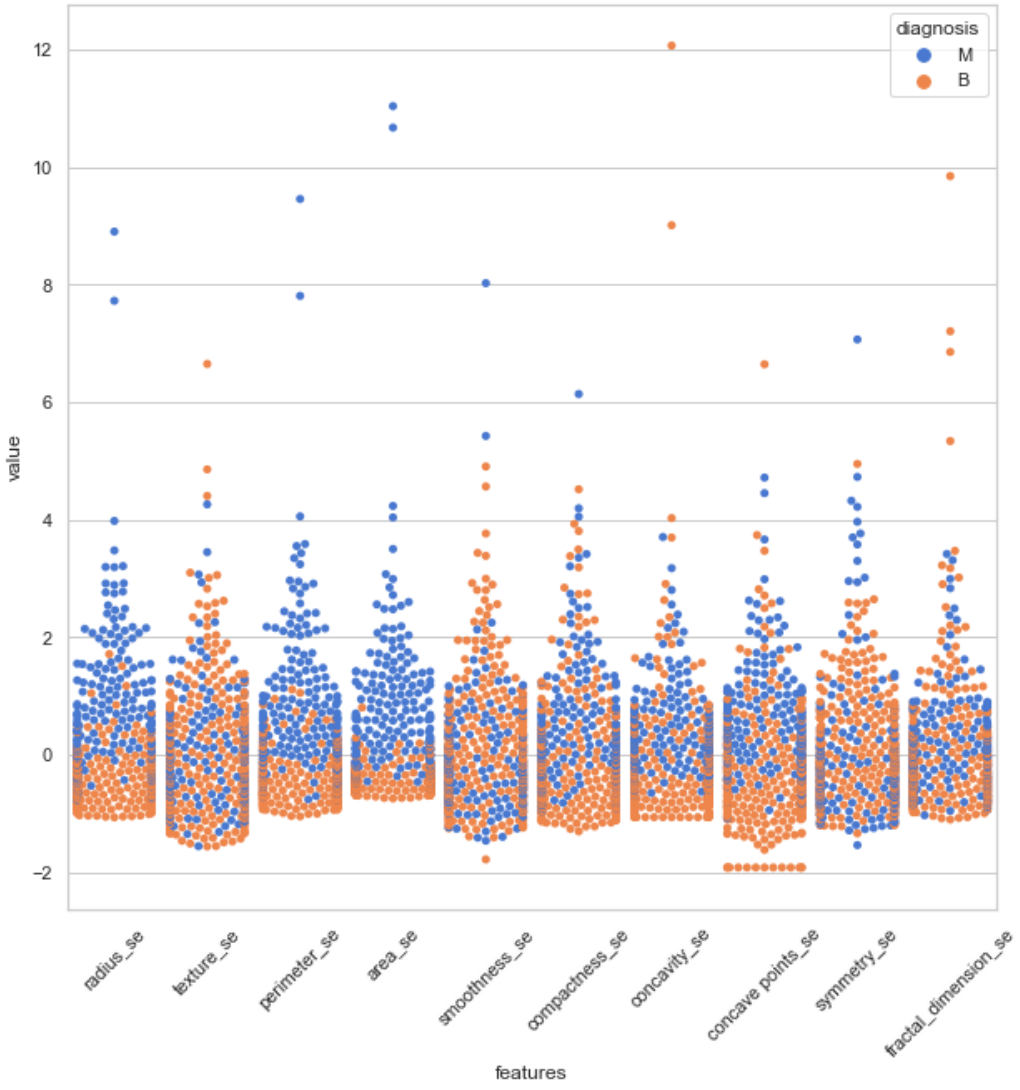

diagnosis
M
B
4
2
0
−2
value
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
features

In [15]:

```
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 = (10,10))
sns.swarmplot(x="features", y = "value", hue = "diagnosis", data = data)
plt.xticks(rotation = 45)
```

Out[15]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 <a list of 10 Text major ticklabel objects>)
```

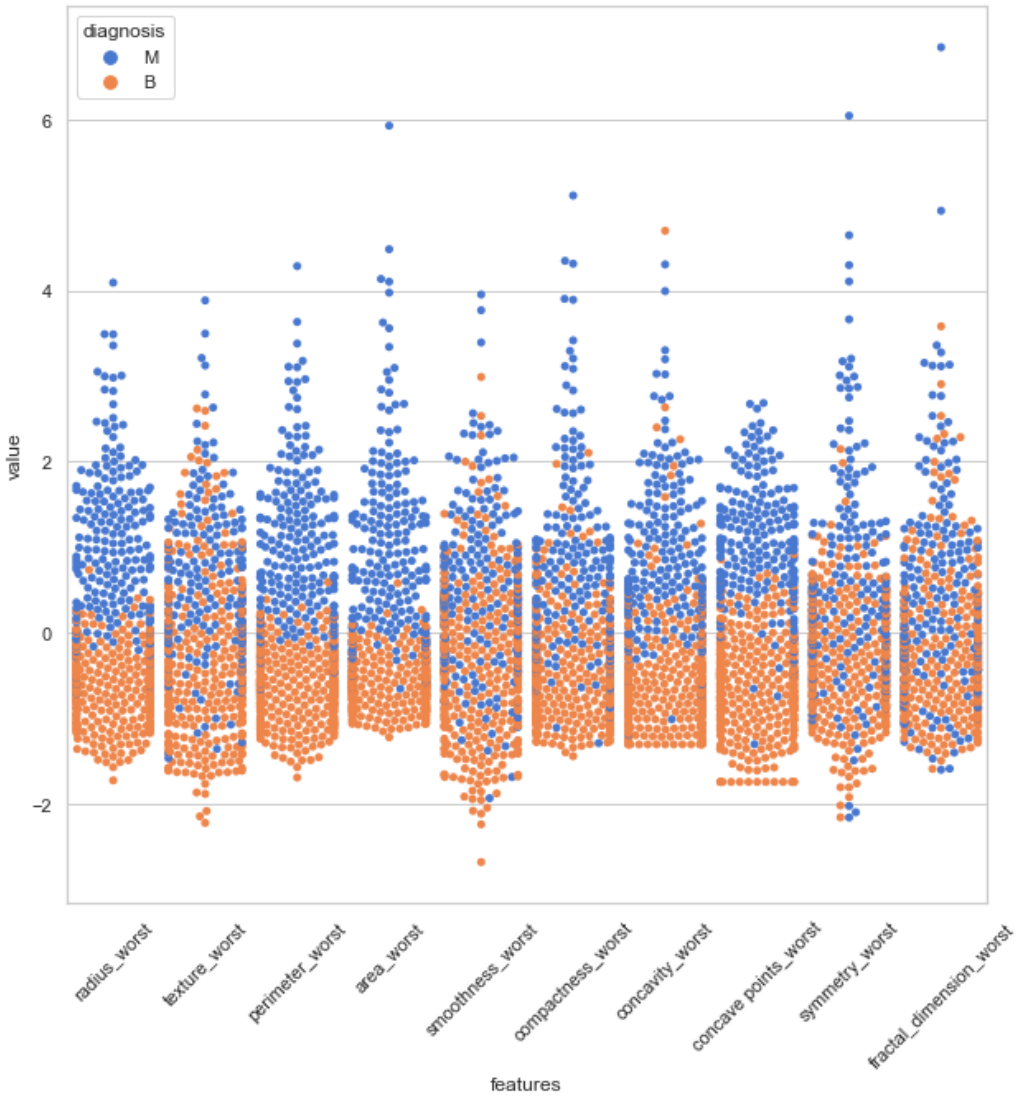


In [16]:

```
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 = (10,10))
sns.swarmplot(x="features", y = "value", hue = "diagnosis", data = data)
plt.xticks(rotation = 45)
```

Out[16]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 <a list of 10 Text major ticklabel objects>)
```



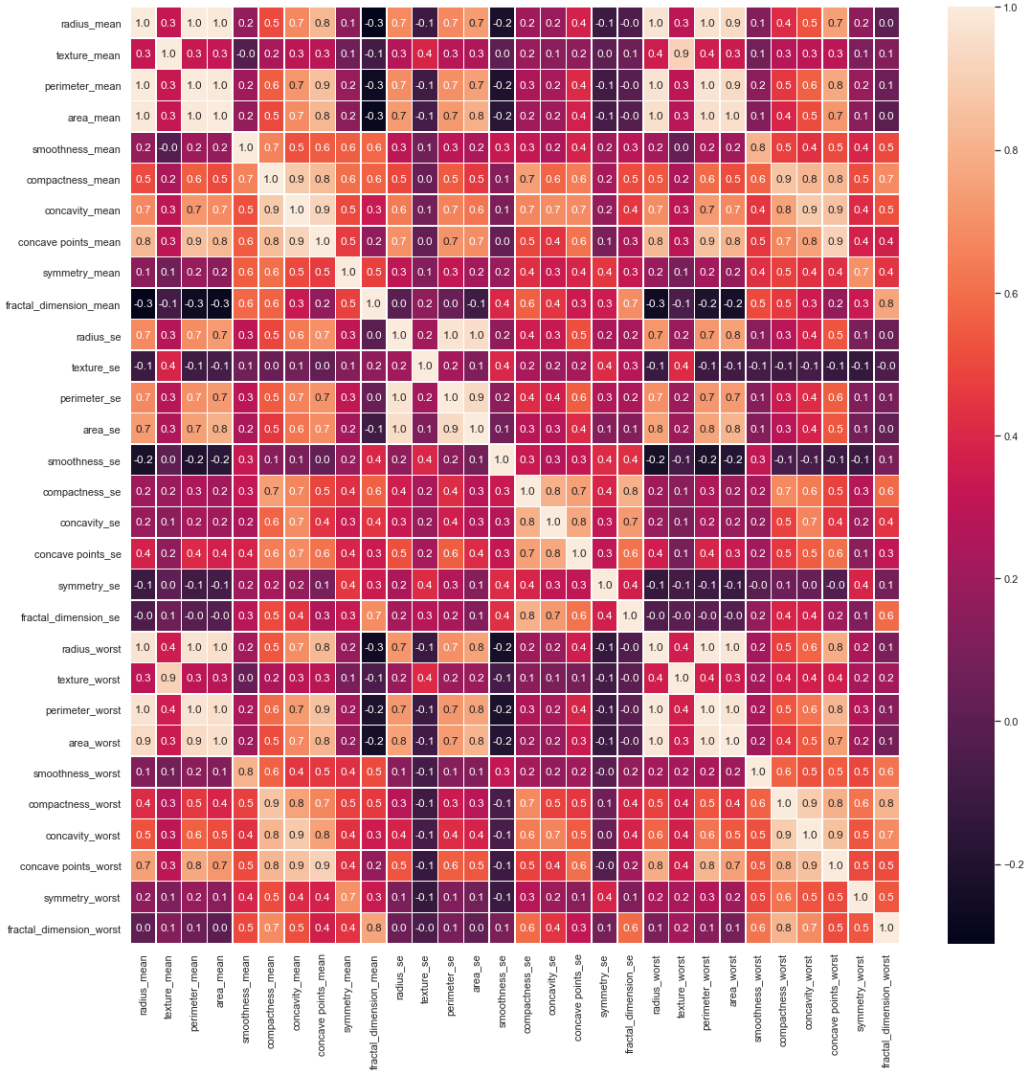
Task: Observing all Pair-wise Correlations

In [17]:

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

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2c96015ba00>
```



In [ ]:

```

```

Task: Dropping Correlated Columns from Feature Matrix

In [18]:

```
drop_cols = ["perimeter_mean", "radius_mean", "compactness_mean",
             "concave points_mean", "radius_se", "perimeter_se",
             "radius_worst", "perimeter_worst", "compactness_worst",
             "concave points_worst", "compactness_se", "concave points_se",
             "texture_worst", "area_worst"]
df = x.drop(drop_cols, axis = 1)
df.head()
```

Out[18]:

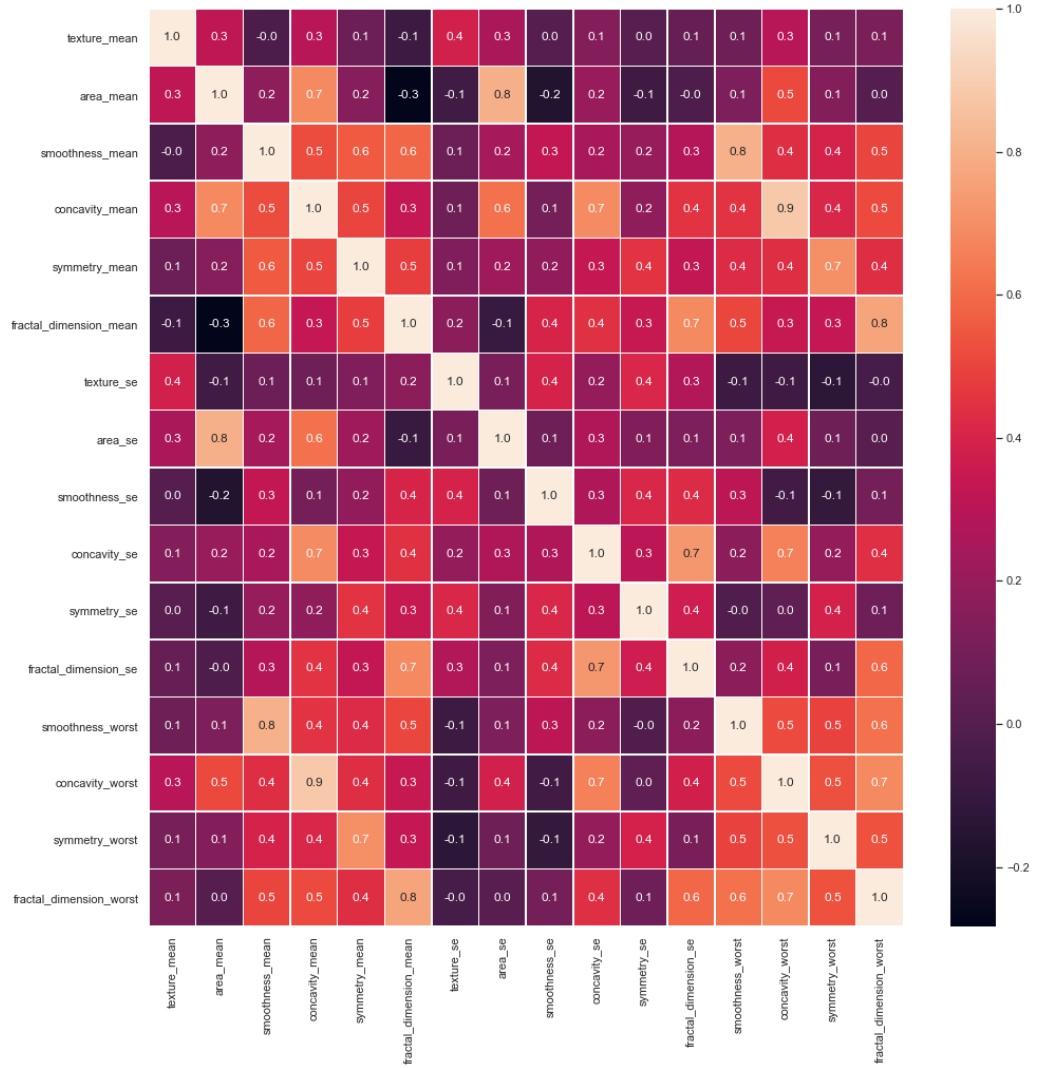
	texture_mean	area_mean	smoothness_mean	concavity_mean	symmetry_mean
0	10.38	1001.0	0.11840	0.3001	0.2419
1	17.77	1326.0	0.08474	0.0869	0.1812
2	21.25	1203.0	0.10960	0.1974	0.2069
3	20.38	386.1	0.14250	0.2414	0.2597
4	14.34	1297.0	0.10030	0.1980	0.1809

In [19]:

```
f, ax = plt.subplots(figsize = (16,16))
sns.heatmap(df.corr(), annot = True, linewidth = 0.5, fmt = ".1f", ax = ax)
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2c9601c42b0>
```


texture_mean 1.0 0.3 -0.0 0.3 0.1 -0.1 0.4 0.3 0.0 0.1 0.0 0.1 0.1 0.3 0.1 0.1
area_mean 0.3 1.0 0.2 0.7 0.2 -0.3 -0.1 0.8 -0.2 0.2 -0.1 -0.0 0.1 0.5 0.1 0.0
smoothness_mean -0.0 0.2 1.0 0.5 0.6 0.6 0.1 0.2 0.3 0.2 0.2 0.3 0.8 0.4 0.4 0.5
concavity_mean 0.3 0.7 0.5 1.0 0.5 0.3 0.1 0.6 0.1 0.7 0.2 0.4 0.4 0.9 0.4 0.5
symmetry_mean 0.1 0.2 0.6 0.5 1.0 0.5 0.1 0.2 0.2 0.3 0.4 0.3 0.4 0.4 0.7 0.4
fractal_dimension_mean -0.1 -0.3 0.6 0.3 0.5 1.0 0.2 -0.1 0.4 0.4 0.3 0.7 0.5 0.3 0.3 0.8
texture_se 0.4 -0.1 0.1 0.1 0.1 0.2 1.0 0.1 0.4 0.2 0.4 0.3 -0.1 -0.1 -0.1 -0.0
area_se 0.3 0.8 0.2 0.6 0.2 -0.1 0.1 1.0 0.1 0.3 0.1 0.1 0.1 0.4 0.1 0.0
smoothness_se 0.0 -0.2 0.3 0.1 0.2 0.4 0.4 0.1 1.0 0.3 0.4 0.4 0.3 -0.1 -0.1 0.1
concavity_se 0.1 0.2 0.2 0.7 0.3 0.4 0.2 0.3 0.3 1.0 0.3 0.7 0.2 0.7 0.2 0.4
symmetry_se 0.0 -0.1 0.2 0.2 0.4 0.3 0.4 0.1 0.4 0.3 1.0 0.4 -0.0 0.0 0.4 0.1
fractal_dimension_se 0.1 -0.0 0.3 0.4 0.3 0.7 0.3 0.1 0.4 0.7 0.4 1.0 0.2 0.4 0.1 0.6
smoothness_worst 0.1 0.1 0.8 0.4 0.4 0.5 -0.1 0.1 0.3 0.2 -0.0 0.2 1.0 0.5 0.5 0.6
concavity_worst 0.3 0.5 0.4 0.9 0.4 0.3 -0.1 0.4 -0.1 0.7 0.0 0.4 0.5 1.0 0.5 0.7
symmetry_worst 0.1 0.1 0.4 0.4 0.7 0.3 -0.1 0.1 -0.1 0.2 0.4 0.1 0.5 0.5 1.0 0.5
fractal_dimension_worst 0.1 0.0 0.5 0.5 0.4 0.8 -0.0 0.0 0.1 0.4 0.1 0.6 0.6 0.7 0.5 1.0
texture_mean
area_mean
smoothness_mean
concavity_mean
symmetry_mean
fractal_dimension_mean
texture_se
area_se
smoothness_se
concavity_se
symmetry_se
fractal_dimension_se
smoothness_worst
concavity_worst
symmetry_worst
fractal_dimension_worst
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Task: Classification using XGBoost (Minimal Feature Selection)

In [22]:

```
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import f1_score,confusion_matrix
from sklearn.metrics import accuracy_score
```

In [23]:

```
x_train, x_test, y_train, y_test = train_test_split(df, y, test_size = 0.3, ran
dom_state = 42)

clf_1 = xgb.XGBClassifier(random_state = 42)
clf_1 = clf_1.fit(x_train, y_train)
```

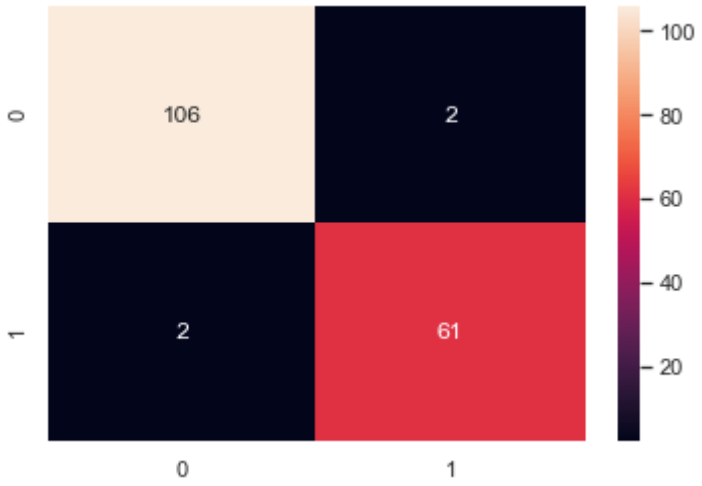
In [25]:

```
print('Accuracy: ', accuracy_score(y_test, clf_1.predict(x_test)))
cm = confusion_matrix(y_test, clf_1.predict(x_test))
sns.heatmap(cm, annot = True, fmt = "d")
```

```
Accuracy:  0.9766081871345029
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2c962a5d070>
```



Task: Univariate Feature Selection and XGBoost

In [26]:

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

In [27]:

```
select_feature = SelectKBest(chi2, k = 10).fit(x_train, y_train)
print("Score List: ", select_feature.scores_)
print("Feature List: ", x_train.columns)
```

```
Score List:  [6.06916433e+01 3.66899557e+04 1.00015175e-01 1.30547
650e+01
 1.95982847e-01 3.42575072e-04 4.07131026e-02 6.12741067e+03
 1.32470372e-03 6.92896719e-01 1.39557806e-03 2.65927071e-03
 2.63226314e-01 2.58858117e+01 1.00635138e+00 1.23087347e-01]
Feature List:  Index(['texture_mean', 'area_mean', 'smoothness_mea
n', 'concavity_mean',
       'symmetry_mean', 'fractal_dimension_mean', 'texture_se', 'a
rea_se',
       'smoothness_se', 'concavity_se', 'symmetry_se', 'fractal_di
mension_se',
       'smoothness_worst', 'concavity_worst', 'symmetry_worst',
       'fractal_dimension_worst'],
      dtype='object')
```

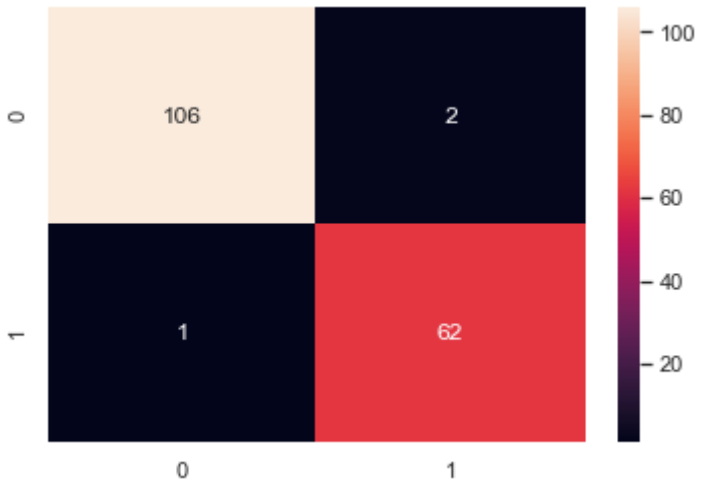
In [29]:

```
x_train_2 = select_feature.transform(x_train)
x_test_2 = select_feature.transform(x_test)

clf_2 = xgb.XGBClassifier().fit(x_train_2, y_train)

print("Accuracy: ", accuracy_score(y_test, clf_2.predict(x_test_2)))

cm2 = confusion_matrix(y_test, clf_2.predict(x_test_2))
sns.heatmap(cm2, annot = True, fmt = 'd')
```

```
Accuracy:  0.9824561403508771
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2c962ddf670>
```



Task: Recursive Feature Elimination with Cross-Validation

In [31]:

```
from sklearn.feature_selection import RFECV

clf_3 = xgb.XGBClassifier()
rfecv = RFECV(estimator = clf_3, step = 1, cv = 5,
              scoring = "accuracy", n_jobs = -1).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:  14
Best Features:  Index(['texture_mean', 'area_mean', 'smoothness_me
an', 'concavity_mean',
       'symmetry_mean', 'texture_se', 'area_se', 'smoothness_se',
       'concavity_se', 'symmetry_se', 'fractal_dimension_se',
       'smoothness_worst', 'concavity_worst', 'symmetry_worst'],
      dtype='object')
```

In [34]:

```
print("Accuracy: ", accuracy_score(y_test, rfecv.predict(x_test)))
```

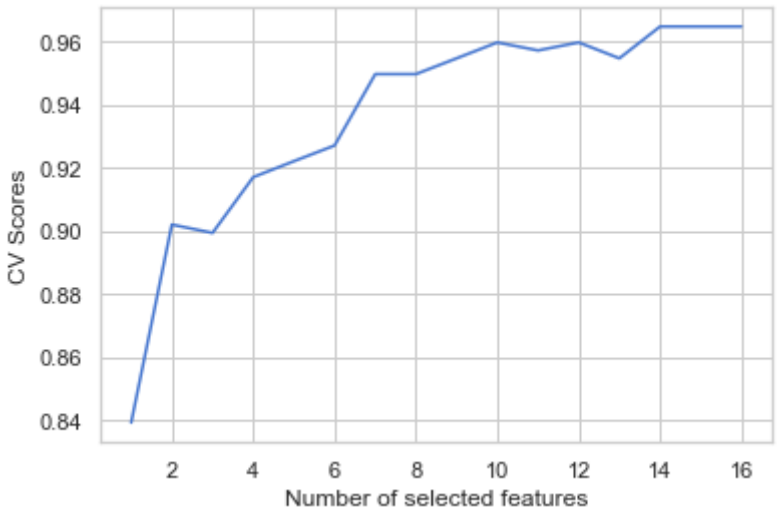
```
Accuracy:  0.9824561403508771
```

In [36]:

```
num_features = [i for i in range(1, len(rfecv.grid_scores_) + 1)]
cv_scores = rfecv.grid_scores_
ax = sns.lineplot(x = num_features, y = cv_scores)
ax.set(xlabel = "Number of selected features",
       ylabel = "CV Scores")
```

Out[36]:

```
[Text(0, 0.5, 'CV Scores'), Text(0.5, 0, 'Number of selected featu
res')]
```



Task: Feature Extraction using Principal Component Analysis

In [40]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, rand
om_state = 42)

x_train_norm = (x_train - x_train.mean()) / (x_train.max() - x_train.min())
x_test_norm = (x_test - x_test.mean()) / (x_test.max() - x_test.min())
```

In [41]:

```
from sklearn.decomposition import PCA

pca = PCA()
pca.fit(x_train_norm)

plt.figure(1, figsize=(10,8))
sns.lineplot(data = np.cumsum(pca.explained_variance_ratio_))
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
```

Out[41]:

```
Text(0, 0.5, 'Cumulative Explained Variance')
```