Ozan MÖHÜRCÜ

Data Analyst | Data Scientist

👋 Hello! I am Ozan, a data analyst who is open to learning and who improves myself in analytical thinking and producing data-driven solutions. I have successfully completed my analyst training and am currently focusing on data science and increasing my competencies in this field.

📊 What Do I Know?

I can extract meaningful results from data by working with Python, SQL and data visualization tools. I am constantly improving myself in statistical analysis and reporting. I aim to solve problems and support decision processes with the insights I obtain.

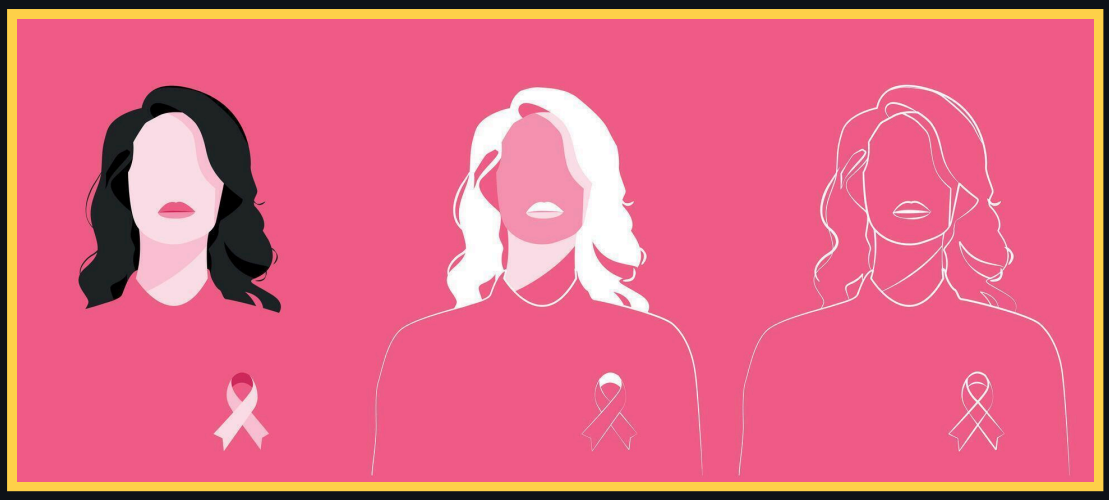
📚 What Am I Doing Right Now? In my data science training, I am gaining knowledge on topics such as machine learning and big data analytics. In addition, I am looking for opportunities to put my theoretical knowledge into practice by gaining experience in real-world projects.

🎯 My Goal: To contribute to the growth goals of companies by using my talents in data analysis and data science in a way that will create value in the business world. I am here to learn new information and to constantly improve by sharing my experiences.

If you would like to discuss projects, collaborate or share experiences, I would be happy to connect!

LinkedIn GitHub

Women's Cancer: Innovative Approaches to Detection and Analysis



Attribute Information

ID number: Unique identifier for each sample.
Diagnosis: (M = malignant, B = benign)

Ten Real-Valued Features for Each Cell Nucleus:

- **Radius:** Mean of distances from the center to points on the perimeter.
- **Texture:** Standard deviation of gray-scale values.
- **Perimeter:** The perimeter of the nucleus.
- **Area:** The area of the nucleus.
- **Smoothness:** Local variation in radius lengths.
- **Compactness:** (Perimeter² / Area - 1.0).
- **Concavity:** Severity of concave portions of the contour.
- **Concave Points:** Number of concave portions of the contour.
- **Symmetry:** Measure of symmetry of the nucleus.
- **Fractal Dimension:** "Coastline approximation" - 1.

Back to top

1.1 Libraries and Utilities

In [1]:

```
import warnings
warnings.filterwarnings("ignore")

# Base libraries
import os
import numpy as np
import pandas as pd
import re
import string
import glob
import math
from IPython.display import display_html
import tqdm
import wandb


## visualization libraries
import matplotlib.pyplot as plt
import matplotlib as mpl
import matplotlib.patches as patches
import seaborn as sns
!pip install pywaffle > dev null
from pywaffle import Waffle


# stat tools
import statsmodels.api as sm
from scipy.stats import kurtosis, skew

## preprocessing & otherlibraries
from sklearn.model_selection import (train_test_split,
                                     cross_val_score,
                                     StratifiedKFold,
                                     GridSearchCV)


from sklearn.preprocessing import (StandardScaler,
                                   MinMaxScaler,
                                   RobustScaler)


## data sampling and outlier detection libraries

from sklearn.cluster import DBSCAN
from sklearn.svm import OneClassSVM
from sklearn.ensemble import IsolationForest
from sklearn.covariance import EllipticEnvelope
from sklearn.neighbors import LocalOutlierFactor




# modeling
from sklearn.linear_model import (LinearRegression,
                                  LogisticRegression)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier, plot_importance, early_stopping
from sklearn.ensemble import (AdaBoostClassifier,
                              ExtraTreesClassifier,
                              RandomForestClassifier,
```

```
                                GradientBoostingClassifier)


# metrics
from sklearn.metrics import (r2_score,
                             accuracy_score,
                             roc_auc_score,
                             f1_score,
                             recall_score,
                             precision_score,
                             recall_score,
                             confusion_matrix)


#feature selection and model interpretaiton
import shap
import eli5
from eli5.sklearn import PermutationImportance
```

1.2 Loading Dataset

In [2]:

```
class color_class:
    BOLD_COLOR = '\033[1m' + '\033[31m'   # Bold + Red text
    BOLD = '\033[1m'    # Bold text only
    END = '\033[0m'   # Reset to default formatting

print(color_class.BOLD_COLOR + '\nImporting all the required libraries....\n\
```

```
Importing all the required libraries....
```

In [3]:

```
print(color_class.BOLD_COLOR+ '\nloading the dataset....' + color_class.END)
df = pd.read_csv('../input/breast-cancer-wisconsin-data/data.csv', delimiter
print(color_class.BOLD )
print('Done!')
```

```
loading the dataset....

Done!
```

1.3 Non-Null ValueCounts and Feature Datatypes

In [4]:

```
print(color_class.BOLD_COLOR + 'Shape of The Dataset:'+ color_class.END + col
    '\n')
```

```
print(color_class.BOLD_COLOR + 'Null Value Count Details as Follows' + color_c
print(color_class.BOLD)
print(df.isnull().sum())
```

```
Shape of The Dataset: (569, 33)

Note: There is 1 missing values column in the data

Null Value Count Details as Follows


id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32              569
dtype: int64
```

In [5]:

```
df.drop(columns = ['Unnamed: 32'], inplace = True)
df['diagnosis'] = df['diagnosis'].map({'M':1, 'B':0})

print(color_class.BOLD_COLOR + 'Sneak peak into the data...'+ color_class.END
print(color_class.BOLD)
print(df.head(2).T)
```

```
Sneak peak into the data...

                                0             1
id                     842302.000000  842517.000000
diagnosis                   1.000000       1.000000
radius_mean                17.990000      20.570000
texture_mean               10.380000      17.770000
perimeter_mean            122.800000     132.900000
area_mean                1001.000000    1326.000000
```

```
smoothness_mean              0.118400       0.084740
compactness_mean             0.277600       0.078640
concavity_mean               0.300100       0.086900
concave points_mean          0.147100       0.070170
symmetry_mean                0.241900       0.181200
fractal_dimension_mean       0.078710       0.056670
radius_se                    1.095000       0.543500
texture_se                   0.905300       0.733900
perimeter_se                 8.589000       3.398000
area_se                    153.400000      74.080000
smoothness_se                0.006399       0.005225
compactness_se               0.049040       0.013080
concavity_se                 0.053730       0.018600
concave points_se            0.015870       0.013400
symmetry_se                  0.030030       0.013890
fractal_dimension_se         0.006193       0.003532
radius_worst                25.380000      24.990000
texture_worst               17.330000      23.410000
perimeter_worst            184.600000     158.800000
area_worst                2019.000000    1956.000000
smoothness_worst             0.162200       0.123800
compactness_worst            0.665600       0.186600
concavity_worst              0.711900       0.241600
concave points_worst         0.265400       0.186000
symmetry_worst               0.460100       0.275000
fractal_dimension_worst      0.118900       0.089020
```

In [6]:

```
print(color_class.BOLD_COLOR +'\nFinal Check for Null Values in data....\n' +

print(color_class.BOLD_COLOR + '\nTotal Number of Null values: \n' + color_cl
print(color_class.BOLD_COLOR + '\nAny NUll Value columns: \n' + color_class.E
```

```
Final Check for Null Values in data....


Total Number of Null values:
0

Any NUll Value columns:
False
```

In [7]:

```
colors = ['#fe4a49', '#2ab7ca', '#fed766', '#e6e6ea', '#f4f4f8']
stats_df = df.drop(columns=['id']).describe().T.reset_index().rename(columns=
stats_df['count'] = stats_df['count'].astype(int)

# Gradient: light gray to red
style = stats_df.style.set_table_attributes("style='display:inline'")\
    .bar(subset=['mean', 'std', 'min', '25%', '50%', '75%', 'max'],
         axis=1,
         color=['#e6e6ea', '#fe4a49'])\
    .format({
        'mean': "{:20,.3f}",
        'std': "{:20,.3f}",
        'min': "{:20,.3f}",
        '25%': "{:20,.3f}",
        '50%': "{:20,.3f}",
        '75%': "{:20,.3f}",
        'max': "{:20,.3f}"
    })\
    .format({"Features": lambda x: x.upper()})\
    .set_properties(**{
        'background-color': 'white',
```

```
        'color': 'black'
    })

print(color_class.BOLD_COLOR + '''\n Descriptive Statistics of The Dataset \n
display_html(style._repr_html_(), raw=True)
```

Descriptive Statistics of The Dataset

	Features	count	mean	std	min	25%
0	DIAGNOSIS	569	0.372583	0.483918	0.000000	0.000000
1	RADIUS_MEAN	569	14.127292	3.524049	6.981000	11.700000
2	TEXTURE_MEAN	569	19.289649	4.301036	9.710000	16.170000
3	PERIMETER_MEAN	569	91.969033	24.298981	43.790000	75.170000
4	AREA_MEAN	569	654.889104	351.914129	143.500000	420.300000
5	SMOOTHNESS_MEAN	569	0.096360	0.014064	0.052630	0.086370
6	COMPACTNESS_MEAN	569	0.104341	0.052813	0.019380	0.064920
7	CONCAVITY_MEAN	569	0.088799	0.079720	0.000000	0.029560
8	CONCAVE POINTS_MEAN	569	0.048919	0.038803	0.000000	0.020310
9	SYMMETRY_MEAN	569	0.181162	0.027414	0.106000	0.161900
10	FRACTAL_DIMENSION_MEAN	569	0.062798	0.007060	0.049960	0.057700
11	RADIUS_SE	569	0.405172	0.277313	0.111500	0.232400
12	TEXTURE_SE	569	1.216853	0.551648	0.360200	0.833900
13	PERIMETER_SE	569	2.866059	2.021855	0.757000	1.606000
14	AREA_SE	569	40.337079	45.491006	6.802000	17.850000
15	SMOOTHNESS_SE	569	0.007041	0.003003	0.001713	0.005169
16	COMPACTNESS_SE	569	0.025478	0.017908	0.002252	0.013080
17	CONCAVITY_SE	569	0.031894	0.030186	0.000000	0.015090
18	CONCAVE POINTS_SE	569	0.011796	0.006170	0.000000	0.007638
19	SYMMETRY_SE	569	0.020542	0.008266	0.007882	0.015160
20	FRACTAL_DIMENSION_SE	569	0.003795	0.002646	0.000895	0.002248
21	RADIUS_WORST	569	16.269190	4.833242	7.930000	13.010000
22	TEXTURE_WORST	569	25.677223	6.146258	12.020000	21.080000
23	PERIMETER_WORST	569	107.261213	33.602542	50.410000	84.110000
24	AREA_WORST	569	880.583128	569.356993	185.200000	515.300000
25	SMOOTHNESS_WORST	569	0.132369	0.022832	0.071170	0.116600
26	COMPACTNESS_WORST	569	0.254265	0.157336	0.027290	0.147200
27	CONCAVITY_WORST	569	0.272188	0.208624	0.000000	0.114500

28	CONCAVE POINTS_WORST	569	0.114606	0.065732	0.000000	0.064930
29	SYMMETRY_WORST	569	0.290076	0.061867	0.156500	0.250400
30	FRACTAL_DIMENSION_WORST	569	0.083946	0.018061	0.055040	0.071460

Descriptive Statistics of Women's Cancer Dataset

It presents descriptive statistics of various features of breast cancer dataset. The table includes total count (count), mean (mean), standard deviation (std), minimum value (min), 25% quartile (25%), median (50%), 75% quartile (75%) and maximum value (max) information for each feature. For example, the mean for the "DIAGNOSIS" feature is 0.372583, which can represent the proportion of malignant tumors in the dataset (assuming 0 is benign and 1 is malignant). The mean and standard deviation values of basic features such as "RADIUS_MEAN", "TEXTURE_MEAN" and "AREA_MEAN" provide an idea about their distribution in the dataset. Quartile values provide information about the spread of the data and possible outliers. In general, this table summarizes the basic statistical properties of numerical features in the dataset.

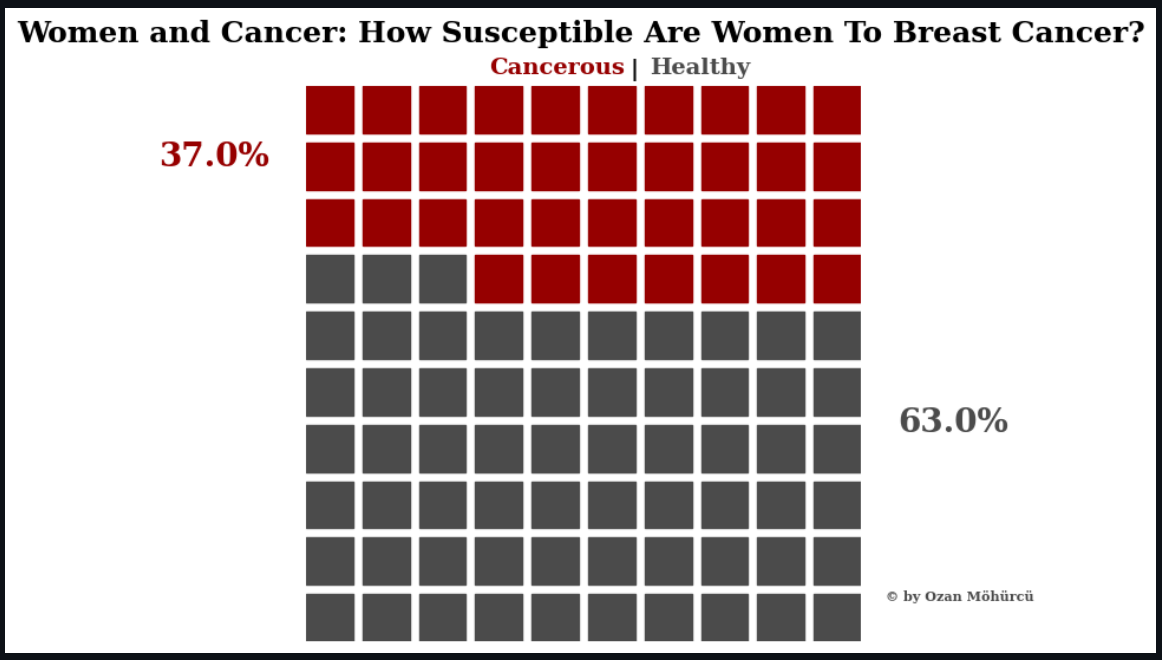
Ozan Möhürcü

Data Visualization

In [8]:

```
# Sample data preparation (replace with your actual DataFrame)
# df = pd.DataFrame(...)  # Your DataFrame here
feat_df = df.drop(columns=['id', 'diagnosis'])
tar_df = df['diagnosis']
cancer_dist = round(tar_df.value_counts(normalize=True), 2) * 100

# Define colors (replace with your actual colors list)
colors = ['#990000', '#ff9999', '#4d4d4d']  # Example colors

# Create the Waffle chart
fig = plt.figure(
    FigureClass=Waffle,
    rows=10,
    columns=10,
    values=[cancer_dist.values[0], cancer_dist.values[1]],
    colors=[colors[2], colors[0]],
    figsize=(8, 5),
    facecolor='white',
    dpi=100,
    vertical=True,
    interval_ratio_y=0.2,
    interval_ratio_x=0.2,
    icon_legend=False,
    icon_size=5,
    plot_anchor='C'
)
```

```
# Labeling - Adjusted positions
# Cancerous percentage (left side)
fig.text(0.05, 0.85, '{}%'.format(cancer_dist.values[1]),
         {'font': 'serif', 'size': 20, 'weight': 'bold', 'color': colors[0]},
         ha='left', va='center')  # Left-aligned, vertically centered

# Healthy percentage (right side)
fig.text(0.95, 0.40, '{}%'.format(cancer_dist.values[0]),
         {'font': 'serif', 'size': 20, 'weight': 'bold', 'color': colors[2]},
         ha='right', va='center')  # Right-aligned, vertically centered

# Titles and text
fig.text(-0.1, 1.045, 'Women and Cancer: How Susceptible Are Women To Breast
         {'font': 'serif', 'size': 18, 'weight': 'bold'}, alpha=1)

# Legend
fig.text(0.40, 0.99, "Cancerous", {'font': 'serif', 'size': 14, 'weight': 'bo
fig.text(0.55, 0.99, '|', {'font': 'serif', 'size': 14, 'weight': 'bold'})
fig.text(0.57, 0.99, "Healthy", {'font': 'serif', 'size': 14, 'weight': 'bold


# Footer
fig.text(0.82, 0.1, '© by Ozan Möhürcü',
         {'font': 'serif', 'size': 8, 'weight': 'bold'}, alpha=0.7)

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



Academic Analysis of Breast Cancer Visualization

This visualization provides a simple waffle chart format of the prevalence of breast cancer among women. The chart shows that 37% of women have cancer (red squares) and 63% have good health (grey squares). This visualization provides an effective way to understand the prevalence of cancer.

However, the chart is based on only two basic categories and does not include

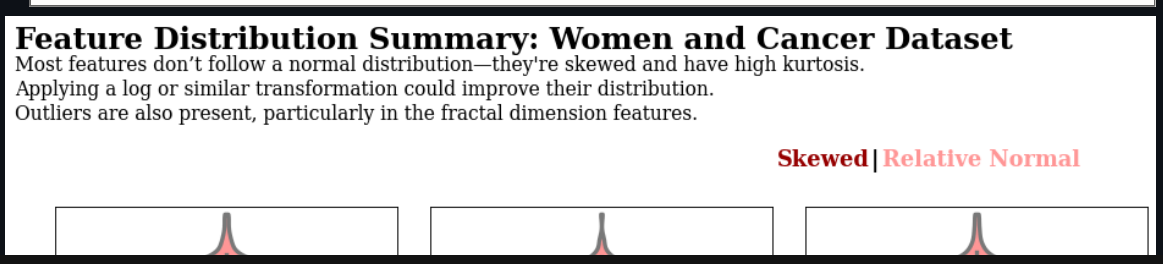
important contextual information such as genetic, environmental, or lifestyle factors. The title statement "women are susceptible to breast cancer" does not adequately reflect the complex nature of the risk.

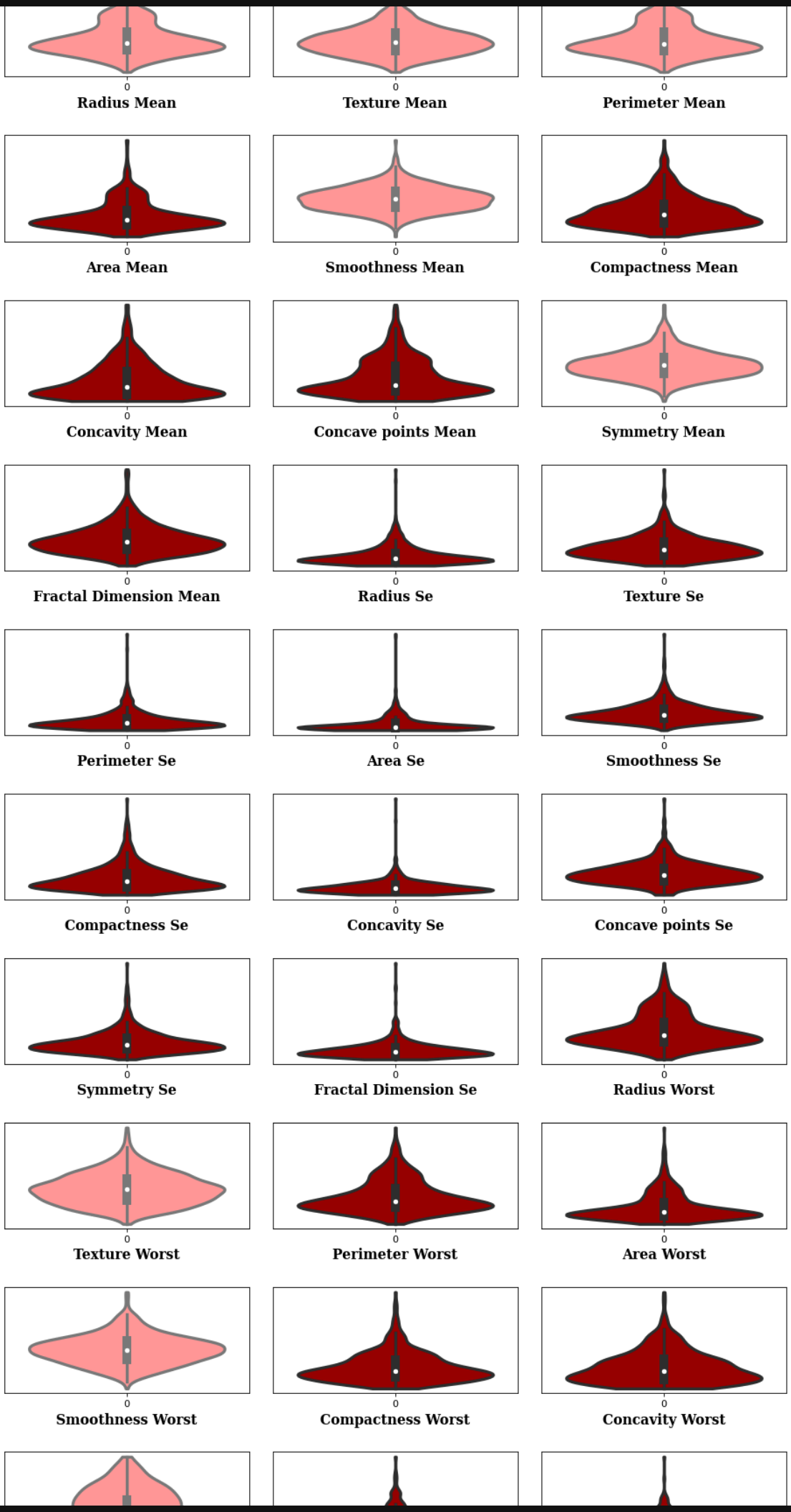
For a more accurate analysis, epidemiological data and demographic factors should be considered.

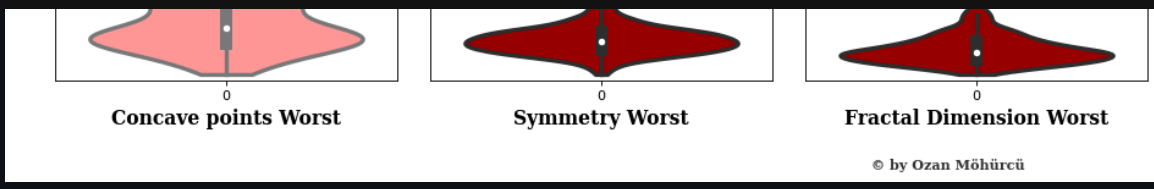
Ozan Möhürcü

In [9]:

```
fig,ax = plt.subplots(nrows = 10, ncols = 3, figsize = (12,24),dpi=80)
#fig.patch.set_facecolor(colors[-1])
axes = ax.ravel()

for col,ax in zip(feat_df.columns,axes):

    # skewness and kurtosis
    if skew(feat_df[col])>1:
        color = colors[0]
    else:
        color = colors[1]

    ## plots
    #sns.kdeplot(feat_df[col], ax= ax, fill = True , color = color, alpha = 1
    sns.violinplot(feat_df[col], ax =ax,
                   color = color, cut =0,
                   inner = 'box',
                   alpha = 1,linewidth = 3, edgecolor = 'solidblack', saturat

    ## plot setting
    xlabel = ' '.join([value.capitalize() for value in str(col).split('_') ])
    #ax.set_facecolor(colors[-1])
    ax.axes.get_yaxis().set_visible(False)
    ax.axes.set_xlabel(xlabel,{'font':'serif','size':14, 'weight':'bold'}, al



plt.tight_layout(pad= 3,h_pad = 2.5, w_pad = 2.5)


## titles and text
fig.text(0,1.05,'Feature Distribution Summary: Women and Cancer Dataset', {'f
fig.text(0,1.02,'''Most features don’t follow a normal distribution—they're s
Applying a log or similar transformation could improve their distribution.
Outliers are also present, particularly in the fractal dimension features.  '

fig.text(0.65,1, "Skewed",{'font':'serif','size':16, 'weight':'bold', 'color'
fig.text(0.73,1, '|',{'font':'serif','size':16, 'weight':'bold'})
fig.text(0.74,1, "Relative Normal",{'font':'serif','size':16, 'weight':'bold'

fig.text(0.73,0,'© by Ozan Möhürcü',{'font':'serif', 'size':10,'weight':'bold

fig.show()
```




0
Radius Mean
0
Texture Mean
0
Perimeter Mean
0
Area Mean
0
Smoothness Mean
0
Compactness Mean
0
Concavity Mean
0
Concave points Mean
0
Symmetry Mean
0
Fractal Dimension Mean
0
Radius Se
0
Texture Se
0
Perimeter Se
0
Area Se
0
Smoothness Se
0
Compactness Se
0
Concavity Se
0
Concave points Se
0
Symmetry Se
0
Fractal Dimension Se
0
Radius Worst
0
Texture Worst
0
Perimeter Worst
0
Area Worst
0
Smoothness Worst
0
Compactness Worst
0
Concavity Worst



Analysis of Feature Distribution in Women's Cancer Dataset

The violin plots display feature distributions across mean values, standard errors, and worst measurements of cell characteristics in breast tissue samples. Most features exhibit significant right-skewed distributions with high kurtosis, particularly in fractal dimension, compactness, and concavity measurements.

Quantitative feature analysis shows darker red plots indicating more extreme distribution patterns, suggesting these measurements may hold greater diagnostic value in distinguishing malignant from benign samples. The distribution patterns indicate that logarithmic transformation would be appropriate for data normalization.

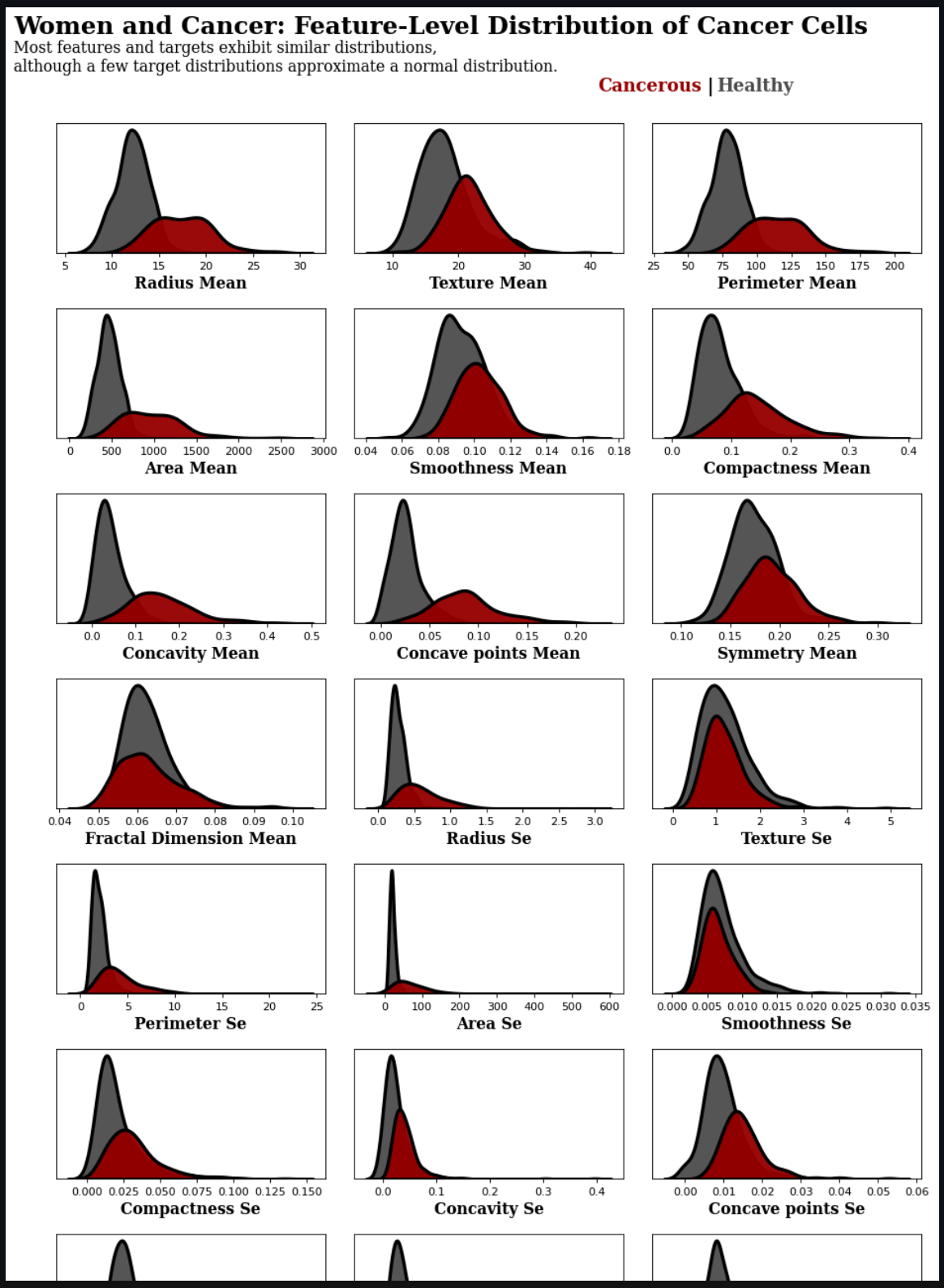
Features demonstrating highest skewness (concavity, fractal dimension, compactness) appear consistently non-normal across mean, standard error, and worst-case measurements, highlighting these as potentially critical diagnostic indicators in computerized analysis of breast cancer cell morphology Ozan Möhürcü

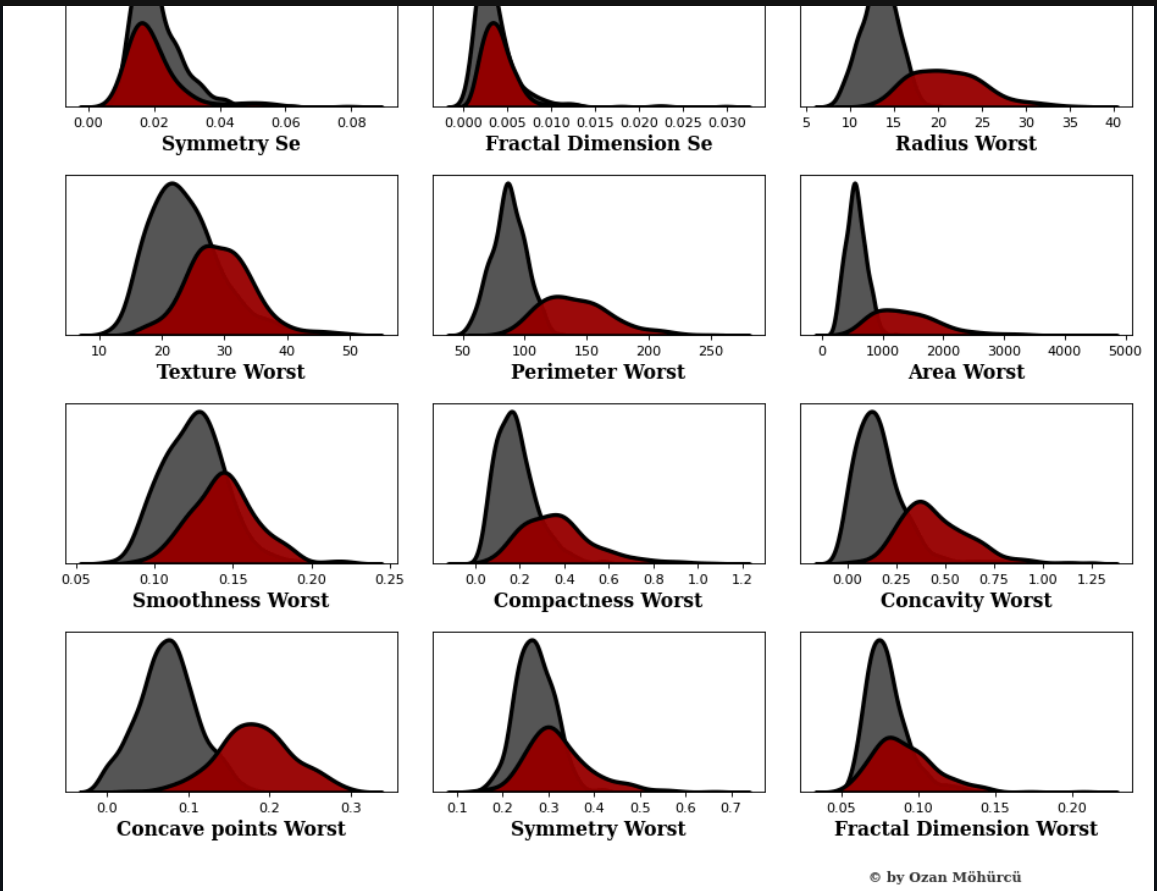
Analysis prepared on April 4, 2025

In [10]:

```
# Assuming df and colors are defined
# df = pd.DataFrame(...)  # Your DataFrame
colors = ['#990000', '#ff9999', '#4d4d4d']  # Example colors
feat_df = df.drop(columns=['id', 'diagnosis'])

# Create subplots
fig, ax = plt.subplots(nrows=10, ncols=3, figsize=(12, 24), dpi=80)
axes = ax.ravel()

# Plot KDEs
for col, ax in zip(feat_df.columns, axes):
    ## Plots
    sns.kdeplot(data=df, x=col, ax=ax, fill=True,  # Changed 'shade' to 'fill
                palette=[colors[0], colors[2]],
                alpha=0.95, linewidth=3, ec='black',
                hue='diagnosis', hue_order=[1, 0],
                legend=False)

    ## Plot settings
    xlabel = ' '.join([value.capitalize() for value in str(col).split('_')])
    ax.axes.get_yaxis().set_visible(False)
    ax.axes.set_xlabel(xlabel, {'font': 'serif', 'size': 14, 'weight': 'bold'

# Adjust layout
plt.tight_layout(pad=3, h_pad=1.5, w_pad=1.5)

# Titles and text
```

```
fig.text(0, 1.03, 'Women and Cancer: Feature-Level Distribution of Cancer Cel
         {'font': 'serif', 'size': 22, 'weight': 'bold'}, alpha=1)
fig.text(0, 1.01, '''Most features and targets exhibit similar distributions,
although a few target distributions approximate a normal distribution.''',
         {'font': 'serif', 'size': 14, 'weight': 'normal'}, alpha=1)

fig.text(0.615, 1, "Cancerous", {'font': 'serif', 'size': 16, 'weight': 'bold
fig.text(0.73, 1, '|', {'font': 'serif', 'size': 16, 'weight': 'bold'})
fig.text(0.74, 1, "Healthy", {'font': 'serif', 'size': 16, 'weight': 'bold',

fig.text(0.73, 0, '© by Ozan Möhürcü', {'font': 'serif', 'size': 10, 'weight'

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





Analysis of Feature Distribution in Women's Cancer Dataset

The graphs in the image compare the distributions of various features of breast cancer and healthy cells. Red curves represent cancer cells, and gray curves represent healthy cells. In many features, the values of cancer cells exhibit a different distribution compared to healthy cells. In particular, in basic measurements such as "Radius Mean", "Texture Mean" and "Area Mean", the values of cancer cells are concentrated in higher ranges. These differences indicate that these features may be important distinguishing factors in cancer diagnosis.

Ozan Möhürcü

Analysis prepared on April 4, 2025

In [11]:
```
print(color_class.BOLD_COLOR + '\nSegregating Features Based On Category....\

### measurement and characteristics keyword lists
measure_keyword = ['radius','perimeter','area','concavity', 'concave points']
character_keyword = ['texture','smoothness','compactness','symmetry','fractal

### mean, standard error, and worst measure feature lists
mean_measure, mean_character = ['diagnosis'],['diagnosis']
se_measure, se_character = ['diagnosis'],['diagnosis']
worst_measure,worst_character = ['diagnosis'],['diagnosis']

### requrired mean, standard errror,and worst measure feature creating loop
for col in feat_df.columns:

    name_list = str(col).split('_')
```

```
    if name_list[0] in measure_keyword:
        if 'mean' in name_list:
            mean_measure.append(col)

        elif 'se' in name_list:
            se_measure.append(col)

        else:
            worst_measure.append(col)

    if name_list[0] in character_keyword:
        if 'mean' in name_list:
            mean_character.append(col)
        elif 'se' in name_list:
            se_character.append(col)
        else:
            worst_character.append(col)

###### descriptions and lists
print(color_class.BOLD + 'Done!' +color_class.END)
print(color_class.BOLD_COLOR + '\nSeperated Features are stored into lists:\n
print(color_class.BOLD_COLOR + 'Mean of Measurements: '+color_class.END \
      + color_class.BOLD +str(' , '.join(mean_measure[1:])) + color_class.END
print(color_class.BOLD_COLOR + 'Mean of Characteristics: '+color_class.END \
      + color_class.BOLD +str(' , '.join(mean_character[1:])) + color_class.E
print(color_class.BOLD_COLOR + 'Standard Error of Measurements: '+color_class
      + color_class.BOLD +str(' , '.join(se_measure[1:])) + color_class.END
```



```
Segregating Features Based On Category....

Done!

Seperated Features are stored into lists:

Mean of Measurements: radius_mean , perimeter_mean , area_mean , concavity_mean
, concave points_mean

Mean of Characteristics: texture_mean , smoothness_mean , compactness_mean , sy
mmetry_mean , fractal_dimension_mean

Standard Error of Measurements: radius_se , perimeter_se , area_se , concavity_
se , concave points_se

Standard Error of Characteristics: texture_se , smoothness_se , compactness_se
, symmetry_se , fractal_dimension_se

Worst of Measurements: radius_worst , perimeter_worst , area_worst , concavity_
worst , concave points_worst

Worst of Characteristics: texture_worst , smoothness_worst , compactness_worst
, symmetry_worst , fractal_dimension_worst
```

In [12]:

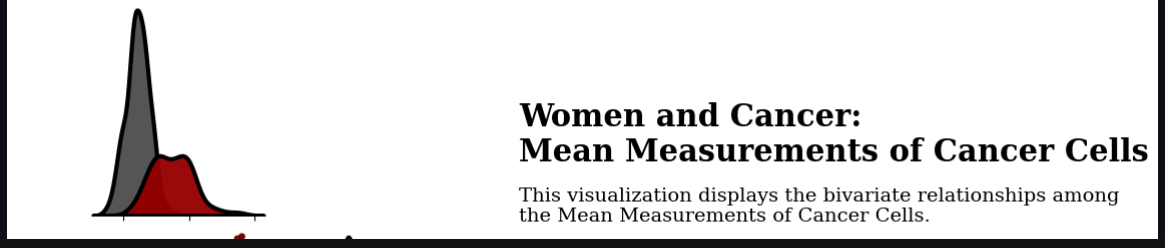
```
print(color_class.BOLD_COLOR +'\nFinally, we include a helper function for vi
```

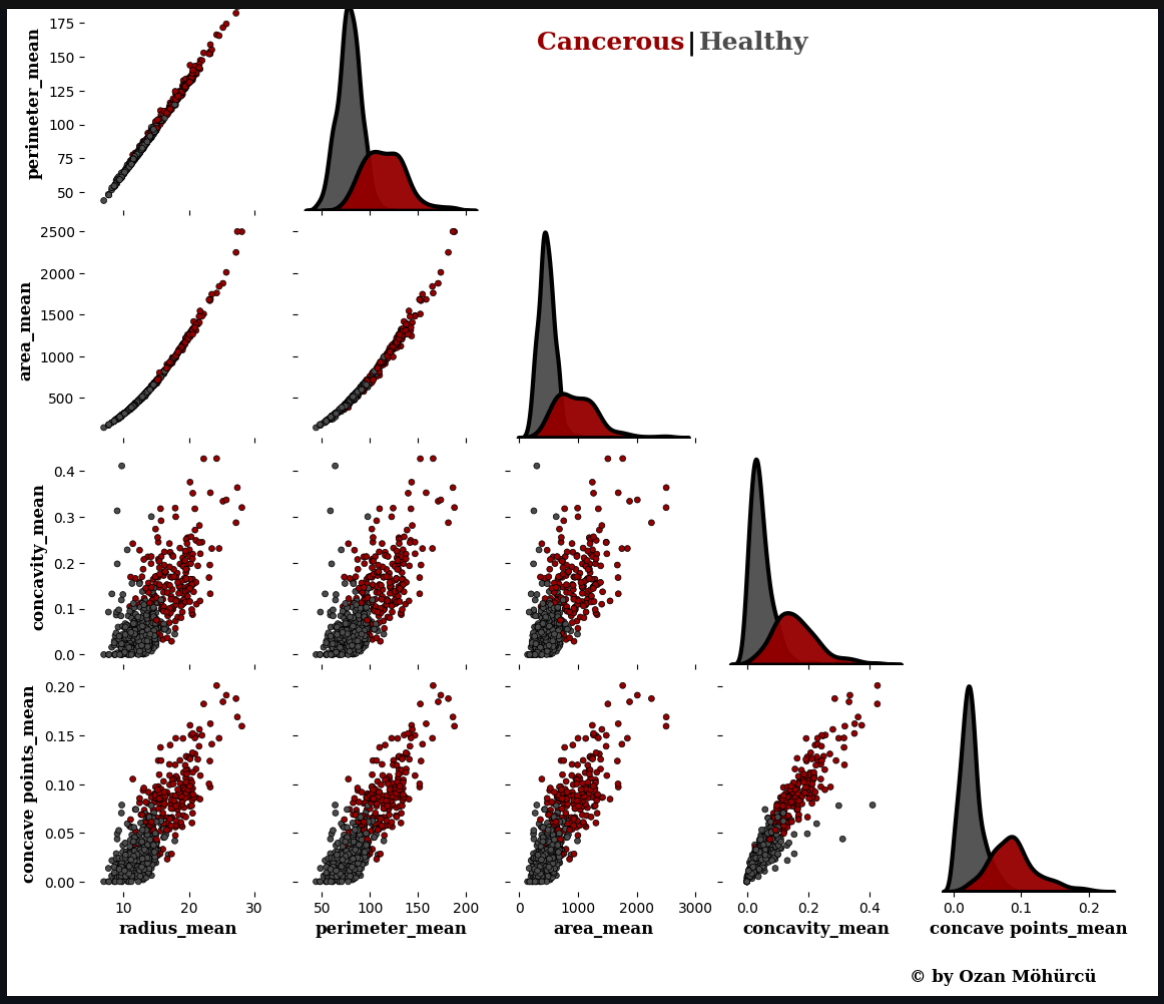
```
### bivariate cross relations visualizations function

def cust_pairplot(df,var,title, diag_kind = 'kde',corner = True,sign = 'off')

    ## plot
    g = sns.pairplot(data = df[var],
                 hue= 'diagnosis',hue_order = [1,0],
                 height = 2.5,aspect = 1,
                 corner = True, diag_kind= diag_kind,
                 palette = [colors[0],colors[2]],
                 plot_kws = {'alpha':1, 'size' : 1, 'linewidth' : 0.5, 'ec':'
                 diag_kws = {'alpha':0.95,'ec':'black','linewidth':3 });

    ### plot setting
    g._legend.remove();

    plt.gcf().patch.set_facecolor('white');
    plt.gcf().patch.set_alpha(1)
    plt.gcf().set_size_inches(12,12);
    #plt.gcf().set_dpi(55);

    for ax in plt.gcf().axes:
        ax.set_facecolor('white')
        for loc in ['left','right','top','bottom']:
            ax.spines[loc].set_visible(False)
        #ax.set_xticks(ticks = [])
        #ax.set_yticks(ticks = [])
        ax.set_xlabel(xlabel = ax.get_xlabel(), **{'font':'serif', 'size':12,
        ax.set_ylabel(ylabel = ax.get_ylabel(), **{'font':'serif', 'size':12,

    ### titles and descriptions

    plt.gcf().text(0.425,0.85, 'Women and Cancer:\n{}'.format(title),{'font':


    plt.gcf().text(0.425,0.8,'''This visualization displays the bivariate rel

    plt.gcf().text(0.44,0.75, "Cancerous",{'font':'serif','size':18, 'weight'
    plt.gcf().text(0.565,0.75, '|',{'font':'serif','size':18, 'weight':'bold'
    plt.gcf().text(0.575,0.75, "Healthy",{'font':'serif','size':18, 'weight':

    ## legend
    if sign == 'on':
        plt.gcf().text(0.75,-0.025,'© by Ozan Möhürcü',{'font':'serif', 'size

    plt.gca().margins(x =0)
    plt.gcf().show();
```

Finally, we include a helper function for visualizing bivariate features....

In [13]:

```
cust_pairplot(df, mean_measure, 'Mean Measurements of Cancer Cells', sign = '
```





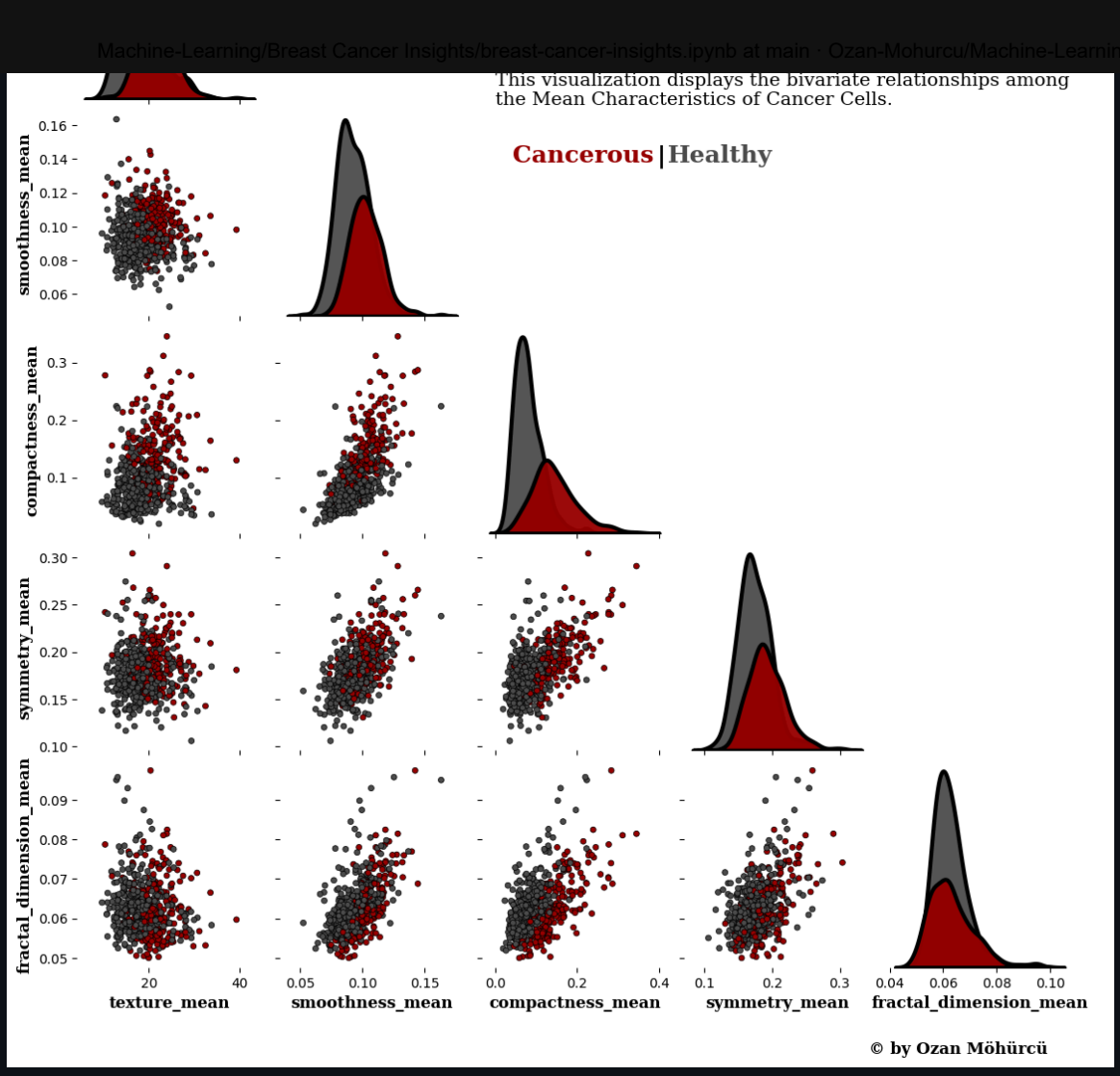
Analysis of Mean Feature Relationships in Women's Cancer Dataset

It shows the relationships between various features of the mean values of breast cancer and healthy cells. The curves on the diagonal show the distribution of each feature, while the other graphs represent the scatter between the two features. Red dots represent cancerous cells, and gray dots represent healthy cells. It is seen that there is a strong positive correlation between the features "radius_mean", "perimeter_mean" and "area_mean" in particular, and that cancerous cells generally have higher values in these features. The distributions of cancerous cells are also clearly different from healthy cells in shape features such as "concavity_mean" and "concave points_mean".

Ozan Möhürcü

Analysis prepared on April 4, 2025

In [14]:
```
cust_pairplot(df, mean_character, 'Mean Characteristics of Cancer Cells', sig
```



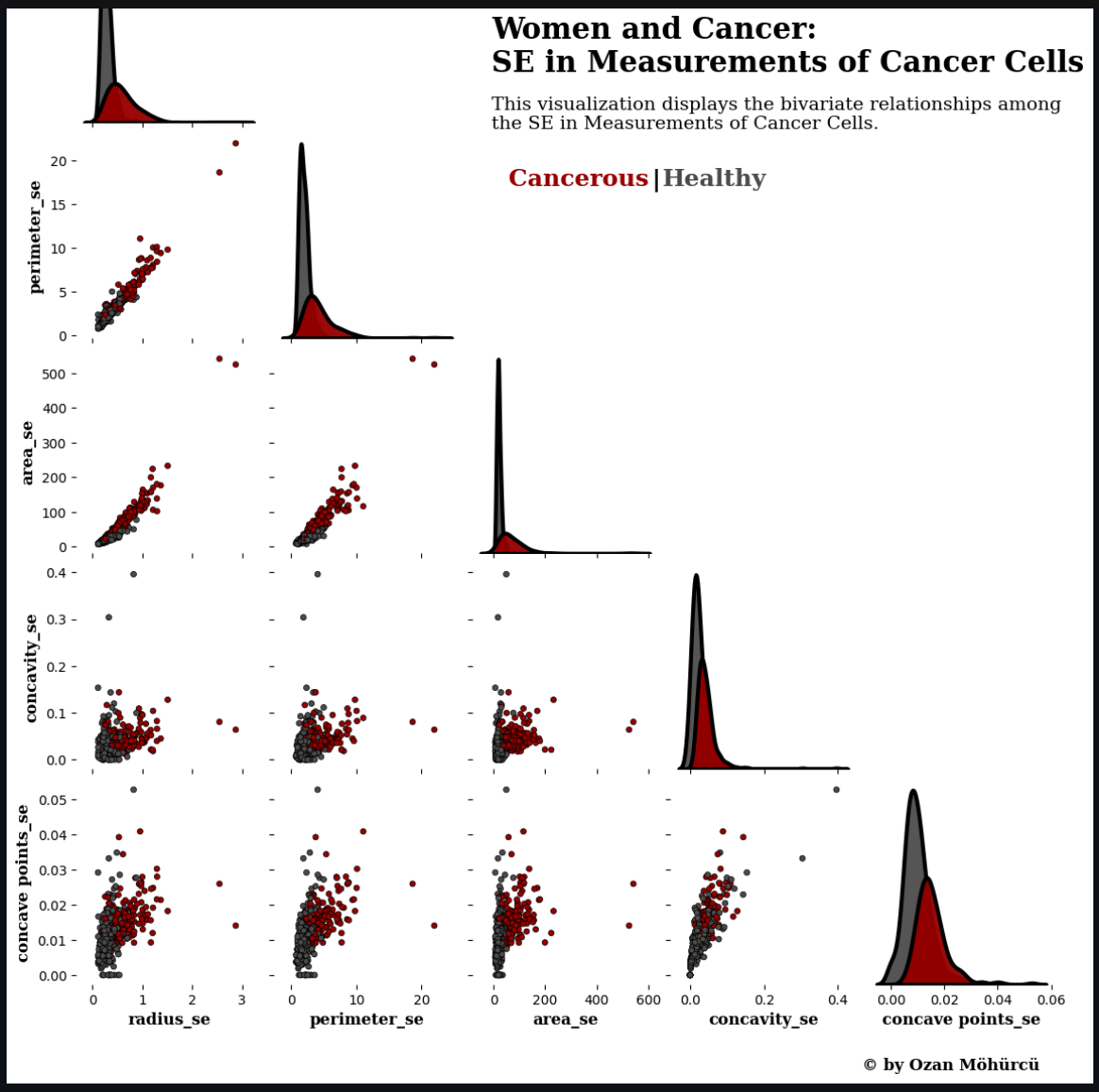
Analysis of Mean Feature Relationships in Women's Cancer Dataset

It shows the relationships between the mean values of breast cancer and healthy cells in the features "texture_mean", "smoothness_mean", "compactness_mean", "symmetry_mean" and "fractal_dimension_mean". The curves on the diagonal present the distribution of each feature, while the other graphs visualize the scatter between the two features. Red dots represent cancerous cells, and gray dots represent healthy cells. While there is no significant correlation between "texture_mean" and other features, it is observed that cancerous cells have higher values and their distributions differ from healthy cells in shape features such as "compactness_mean" and "concavity_mean". For "fractal_dimension_mean", the distributions of cancerous and healthy cells appear closer to each other.

Ozan Möhürcü

Analysis prepared on April 4, 2025

In [15]:
```
cust_pairplot(df, se_measure, 'SE in Measurements of Cancer Cells',sign = 'on
```



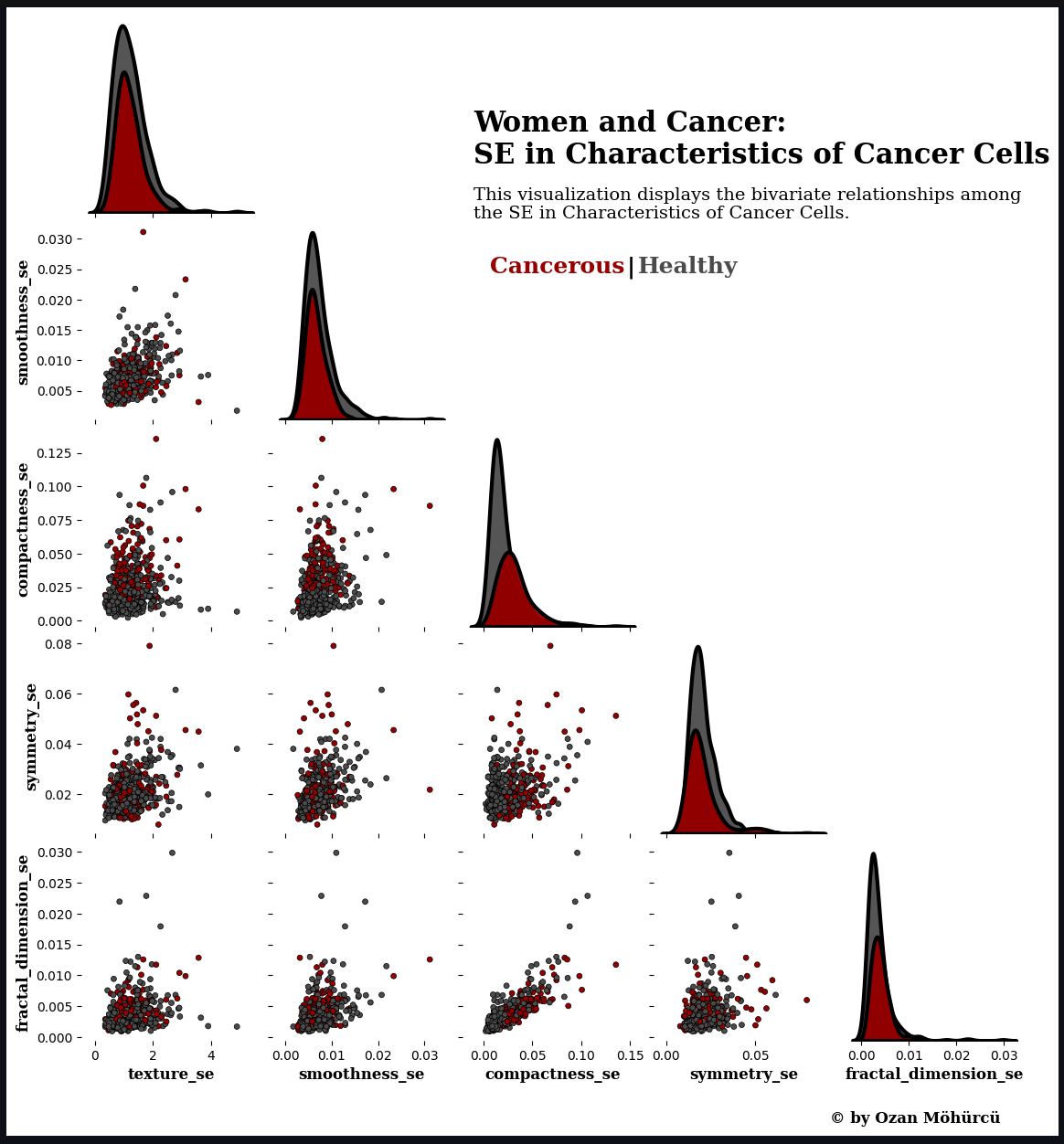
Analysis of Standard Error (SE) Feature Relationships in Women's Cancer Dataset

It shows the relationships between various features of standard errors (SE) of breast cancer and healthy cells. The curves on the diagonal show the distribution of SE values of each feature, while the other graphs represent the scatter between SE values of two features. Red dots represent cancerous cells, and gray dots represent healthy cells. It is observed that there is a positive correlation between the features "radius_se", "perimeter_se" and "area_se" in particular, and that cancerous cells generally have higher values in the standard errors of these features. The distributions of cancerous cells are also separated from healthy cells in the standard errors of shape features such as "concavity_se" and "concave points_se", but this separation is not as clear as in the mean values.

Ozan Möhürcü

Analysis prepared on April 4, 2025

In [16]:
```
cust_pairplot(df, se_character, 'SE in Characteristics of Cancer Cells',sign
```



Analysis of Standard Error (SE) Feature Relationships in Women's Cancer Dataset

shows the relationships between the standard errors (SE) of breast cancer and healthy cells, "texture_se", "smoothness_se", "compactness_se", "symmetry_se" and "fractal_dimension_se" features. The curves on the diagonal present the distribution of SE values of each feature, while the other graphs visualize the scatter between the SE values of the two features. Red dots represent cancerous cells, and gray dots represent healthy cells. While no significant correlation is observed between the SE values of "texture_se" and other features, it is seen that cancerous cells generally have higher values in SE values of features such as "compactness_se" and "smoothness_se" and their distributions are partially separated from healthy cells. For "fractal_dimension_se", the distributions of SE values of cancerous and healthy cells are closer to each other.

Ozan Möhürcü

Analysis prepared on April 4, 2025

In [17]:
```
print(color_class.BOLD_COLOR + '\nExtracting Features with High Positive and

temp_df = df.corr().unstack().reset_index()


### cross relational positive features

positive_corr_df = (temp_df[(temp_df[0]>0.9) &
         (temp_df['level_0'] != temp_df['level_1']) &
         ((temp_df['level_0'].apply(lambda x: str(x).split('_')[-1])) != (tem

positive_corr_df['z'] = positive_corr_df.apply(lambda x: tuple(sorted([x['lev
positive_corr_df.drop_duplicates(subset="z", keep="first" , inplace = True )
positive_corr_df.drop(columns = ['z'], inplace = True)



### cross relational negative features

negative_corr_df = (temp_df[(temp_df[0]<-0.2) &
         (temp_df['level_0'] != temp_df['level_1']) &
         ((temp_df['level_0'].apply(lambda x: str(x).split('_')[-1])) != (tem

negative_corr_df['z'] = negative_corr_df.apply(lambda x: tuple(sorted([x['lev
negative_corr_df.drop_duplicates(subset="z", keep="first" , inplace = True )
negative_corr_df.drop(columns = ['z'], inplace = True)

print(color_class.BOLD +'Done!')
```

Extracting Features with High Positive and Negative Correlations through Cross-Categorical Analysis..

Done!

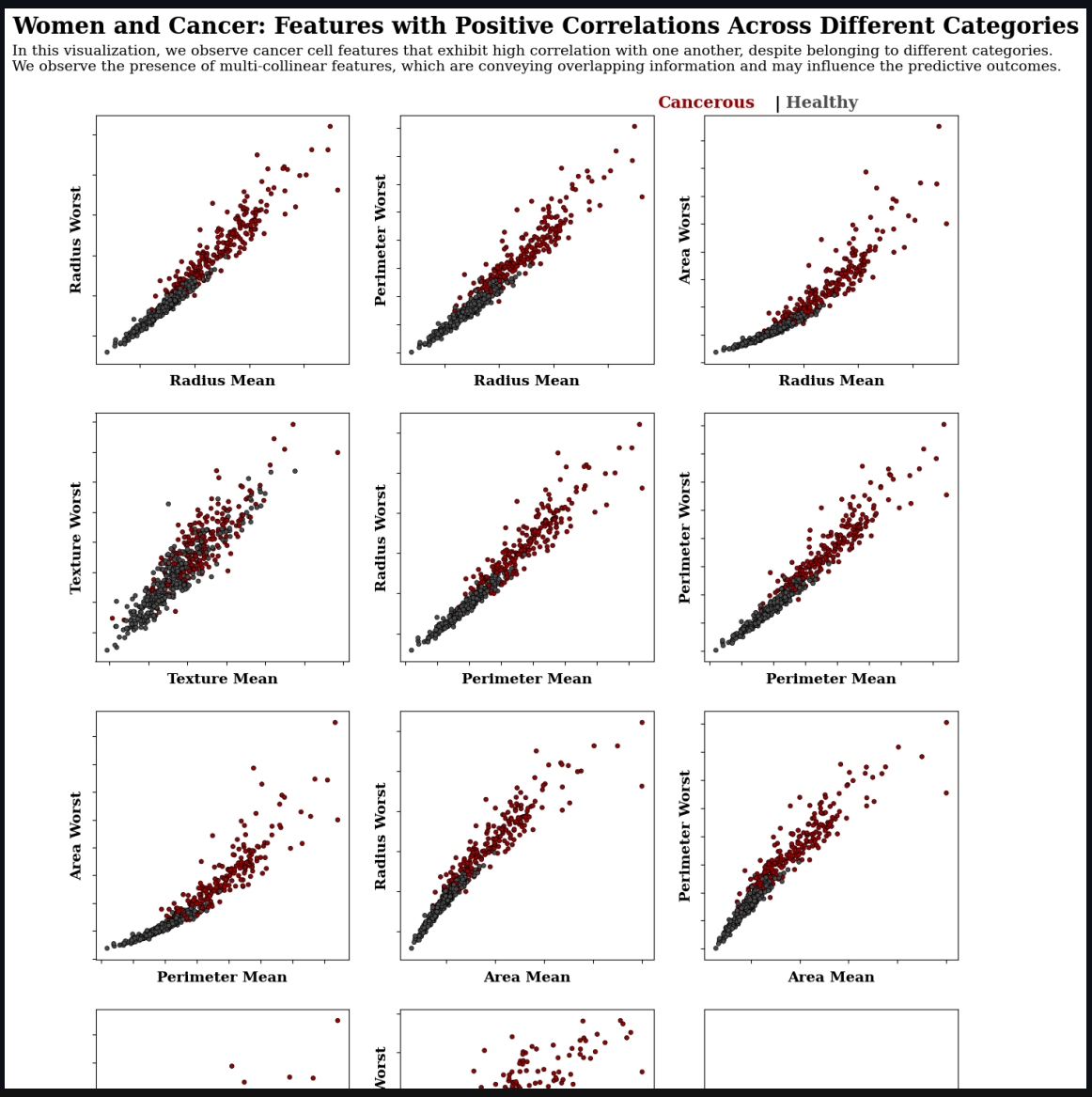
In [18]:
```
print(color_class.BOLD_COLOR + '\nA helper function designed to visualize cro
def plot_cross_scatter(corr_df, data =df,title = None,des = None,nrows = 4, n

    col1_list = corr_df['level_0'].values.tolist()
    col2_list = corr_df['level_1'].values.tolist()

    ## plotting
    fig,axes = plt.subplots(nrows,ncols, figsize = (15,20))

    # removing the last axes
    axes.ravel()[-1].axes.get_xaxis().set_visible(False)
    axes.ravel()[-1].axes.get_yaxis().set_visible(False)

    for ax,col1,col2 in zip(axes.ravel(), col1_list,col2_list):

        sns.scatterplot(x= data[col1], y = data[col2], ax = ax,size = 100,
                        linewidth= 0.5, edgecolor = 'black',
                        hue = data['diagnosis'], hue_order = [1,0],
                        palette = [colors[0],colors[2]], legend = False )

        ## plot setting
        xlabel = ' '.join([value.capitalize() for value in str(col1).split('_
        ylabel = ' '.join([value.capitalize() for value in str(col2).split('_

        ax.axes.set_xlabel(xlabel,{'font':'serif','size':14, 'weight':'bold'}
        ax.axes.set_ylabel(ylabel,{'font':'serif','size':14, 'weight':'bold'}
```

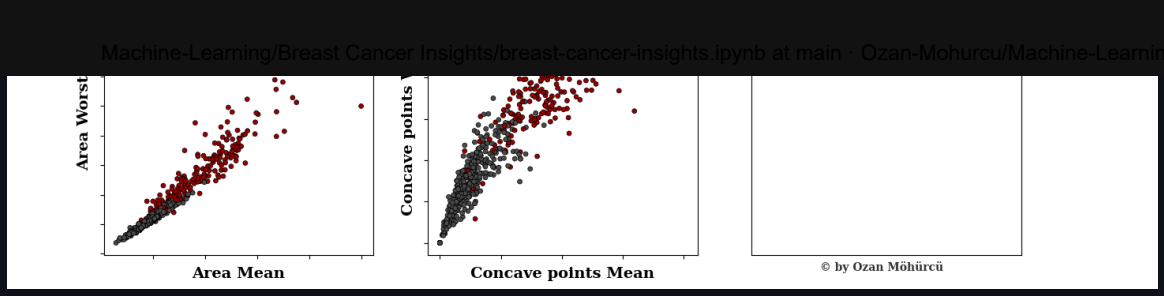
```
        ax.set_xticklabels('')
        ax.set_yticklabels('')


    ## titles and text
    fig.text(0.05,0.935,'Women and Cancer: {}'.format(title), {'font':'serif'
    fig.text(0.05,0.91,'''{}'''.format(des),{'font':'serif','size':14, 'weigh

    fig.text(0.63,0.885, "Cancerous",{'font':'serif','size':16, 'weight':'bol
    fig.text(0.735,0.885, '|',{'font':'serif','size':16, 'weight':'bold'})
    fig.text(0.745,0.885, "Healthy",{'font':'serif','size':16, 'weight':'bold

    fig.text(0.73,0.1,'© by Ozan Möhürcü',{'font':'serif', 'size':10,'weight'

    fig.show()

    return None
```

A helper function designed to visualize cross-categorical feature relationships.

In [19]:

```
des = 'In this visualization, we observe cancer cell features that exhibit hi
plot_cross_scatter(positive_corr_df, title = 'Features with Positive Correlat
```





Analysis of Mean vs. Worst Feature Relationships in Women's Cancer Dataset

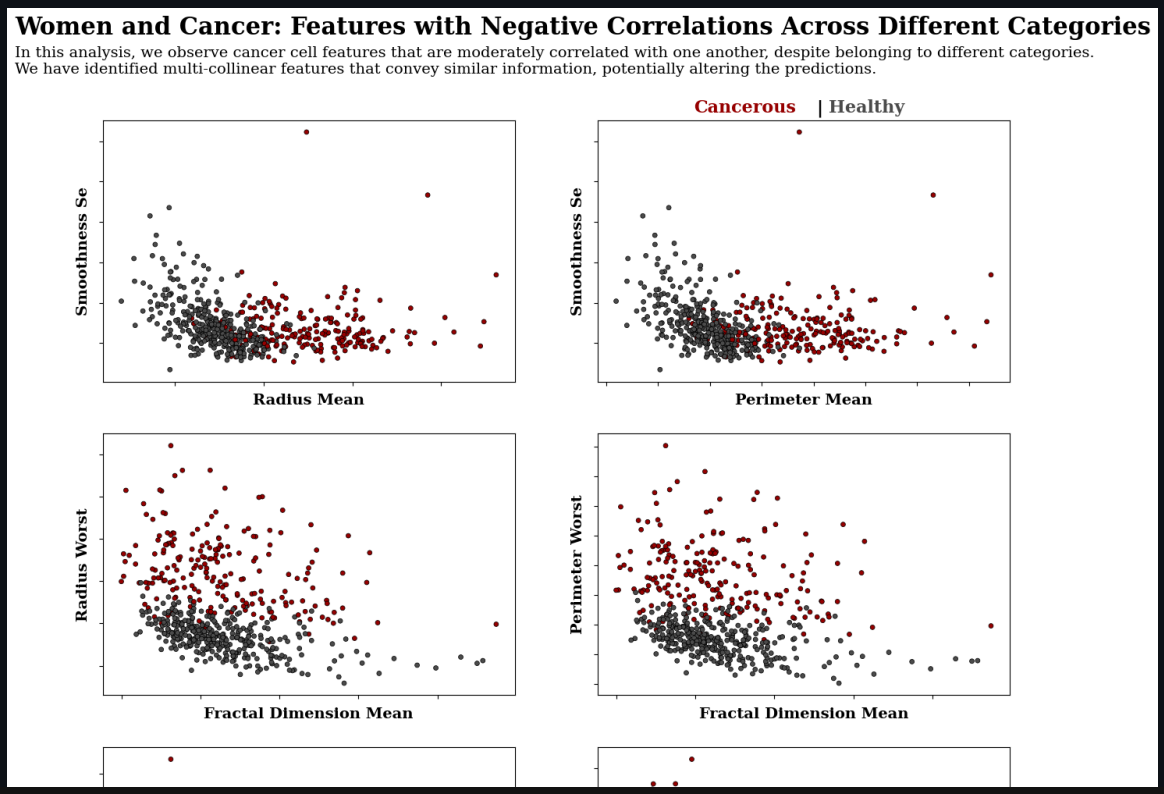
It shows the relationships between the mean values and the worst values of breast cancer and healthy cells. Red dots represent cancerous cells, and gray dots represent healthy cells. For many features, a significant positive correlation is observed, with the worst value increasing as the mean value increases. Cancerous cells generally have higher values than healthy cells, both in the mean and worst values. This distinction is especially clear for the

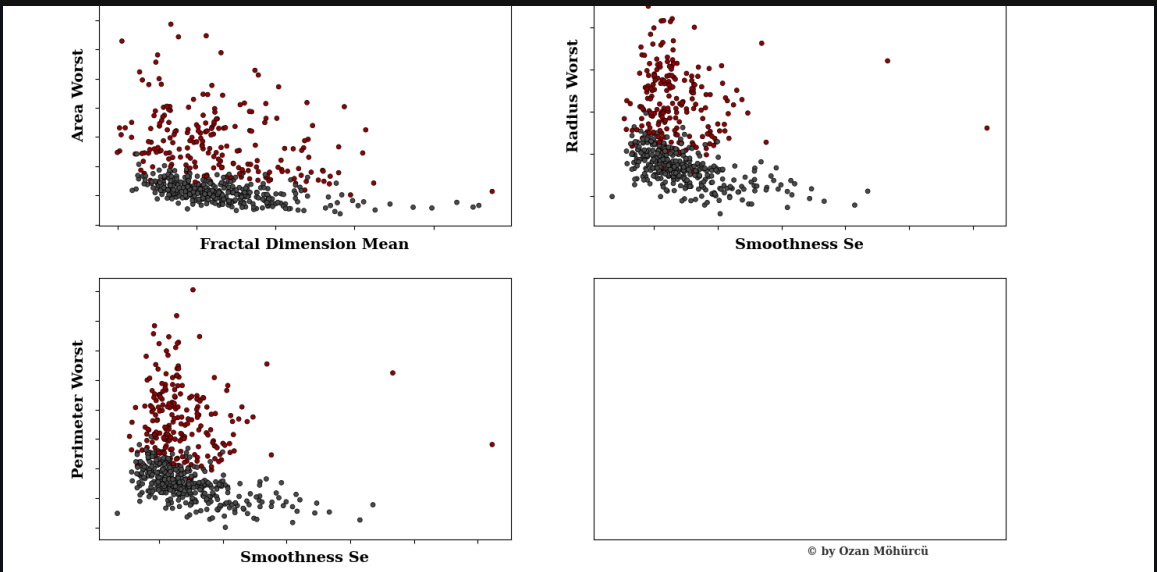
"Radius", "Perimeter" and "Area" features, suggesting that these features may play an important role in cancer diagnosis.

Ozan Möhürcü

Analysis prepared on April 4, 2025

In [20]:
```
des = 'In this analysis, we observe cancer cell features that are moderately
plot_cross_scatter(negative_corr_df,nrows = 4,ncols = 2, figsize=(12,6)
                   ,title = 'Features with Negative Correlations Across Diffe
```





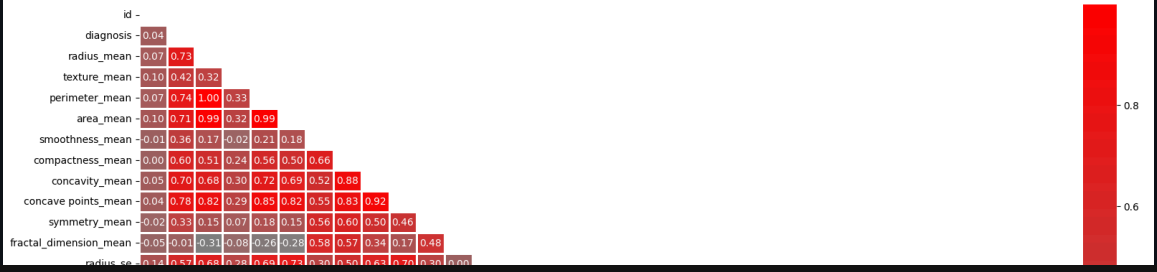
Analysis of Various Feature Relationships in Women's Cancer Dataset

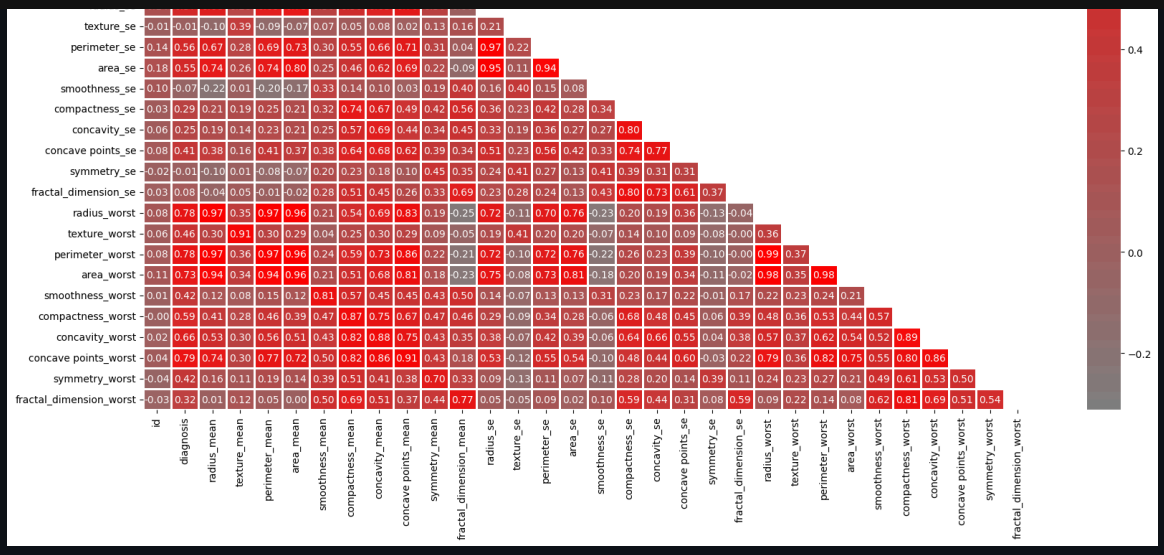
It shows the comparison of different features of breast cancer and healthy cells. Red dots represent cancerous cells, gray dots represent healthy cells. In some graphs, such as the comparison of "Radius Mean" and "Smoothness Se", it is seen that cancerous cells spread over a wider area and are partially separated from healthy cells. In the comparisons of "Fractal Dimension Mean" and "Radius Worst", "Perimeter Worst" and "Area Worst", it is noted that cancerous cells generally have higher "worst" values. This situation suggests that the tumor may exhibit more aggressive features as the "fractal dimension" average increases.

Ozan Möhürcü

Analysis prepared on April 4, 2025

In [21]:
```
# heatmap

from matplotlib.colors import LinearSegmentedColormap

cmap = LinearSegmentedColormap.from_list("red_to_gray", ["gray", "red"])

plt.figure(figsize=(20, 12))

corr = df.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))

sns.heatmap(corr, mask=mask, linewidths=1, annot=True, fmt=".2f", cmap=cmap)

plt.show()
```





Correlation Heatmap of Women's Cancer Dataset Features

It shows the correlation coefficients between various features in the breast cancer dataset. Red tones indicate positive correlation, while gray tones indicate negative correlation. The darker the color, the stronger the correlation. The strongest positive correlations with "diagnosis" are seen in basic tumor features such as "radius_mean", "perimeter_mean", "area_mean", "compactness_mean", "concavity_mean" and "concave points_mean". This shows that these features are strongly associated with malignant tumors. On the negative side, there are weaker correlations with "smoothness_mean" and "fractal_dimension_mean". In addition, high positive correlations are also noted between features in the same category (e.g. mean, se, worst). This heatmap visualizes the relationships between features and gives an idea about which features may be more important for diagnosis.

Ozan Möhürcü

Analysis prepared on April 4, 2025

In [22]:

```
# removing highly correlated features

corr_matrix = df.corr().abs()

mask = np.triu(np.ones_like(corr_matrix, dtype = bool))
tri_df = corr_matrix.mask(mask)

to_drop = [x for x in tri_df.columns if any(tri_df[x] > 0.92)]

df = df.drop(to_drop, axis = 1)

print(f"The reduced dataframe has {df.shape[1]} columns.")
```

```
The reduced dataframe has 24 columns.
```

Advanced Feature Engineering and

Advanced Feature Engineering and Impact Analysis

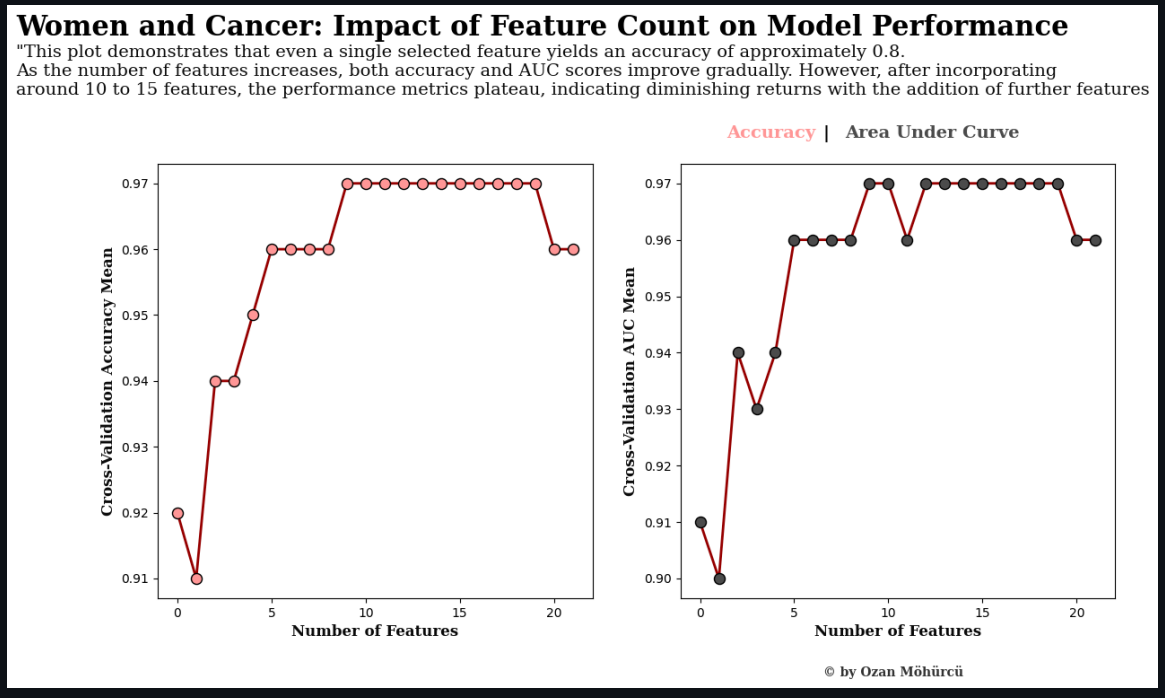
In [23]:
```
# creating features and label

X = df.drop('diagnosis', axis = 1)
y = df['diagnosis']
```

In [24]:
```
X_temp = df.drop(columns = ['diagnosis'])
y_temp = df['diagnosis']
temp_X_train,temp_X_val,temp_y_train,temp_y_val = train_test_split(X_temp,y_t

temp_model= XGBClassifier(eval_metric='logloss').fit(temp_X_train,temp_y_trai

perm = PermutationImportance(temp_model, scoring = 'roc_auc').fit(temp_X_val,
eli5_feature_importance2 = (pd.DataFrame({'Features':temp_X_train.columns.tol
                            .sort_values(by = 'Importance'))
perm_imp_feats_auc = (eli5_feature_importance2.sort_values(by = 'Importance',
                        .reset_index(drop = True))['Features']


## dataframe as per feature selection from permutation importance
temp_X_df = df.drop(columns ='diagnosis').copy()
temp_X_df = temp_X_df[perm_imp_feats_auc]
temp_y_df = df['diagnosis']

## crossvalidation with repeated feature selection
stratified = StratifiedKFold(n_splits = 5,shuffle = True, random_state = 2021

feat_acc = []
feat_auc = []
feat_f1 = []
for idx,feat in enumerate (perm_imp_feats_auc):
    temp = temp_X_df.iloc[:,:idx]
    temp['all_other'] = temp_X_df.iloc[:,idx:len(perm_imp_feats_auc)].sum(axi
    X_ = temp
    y_ = temp_y_df

    if idx == 0:
        continue
    else:
        fold_acc = []
        fold_auc = []
        fold_f1 = []
        for train_idx,valid_idx in stratified.split(X_,y_):

            xtrain,xvalid = X_.iloc[train_idx],X_.iloc[valid_idx]
            ytrain,yvalid = y_.iloc[train_idx],y_.iloc[valid_idx]

            model = XGBClassifier(eval_metric = 'logloss').fit(xtrain.values,
            preds = model.predict(xvalid.values)

            acc_score= accuracy_score(yvalid,preds)
            auc_score = roc_auc_score(yvalid,preds)
            f1 = f1_score(yvalid,preds)
            fold_acc.append(acc_score)
            fold_auc.append(auc_score)
            fold_f1.append(f1)


    feat_acc.append(round(np.mean(fold_acc),2))
```

```
    feat_auc.append(round(np.mean(fold_auc),2))
    feat_f1.append(round(np.mean(fold_f1),2))
```

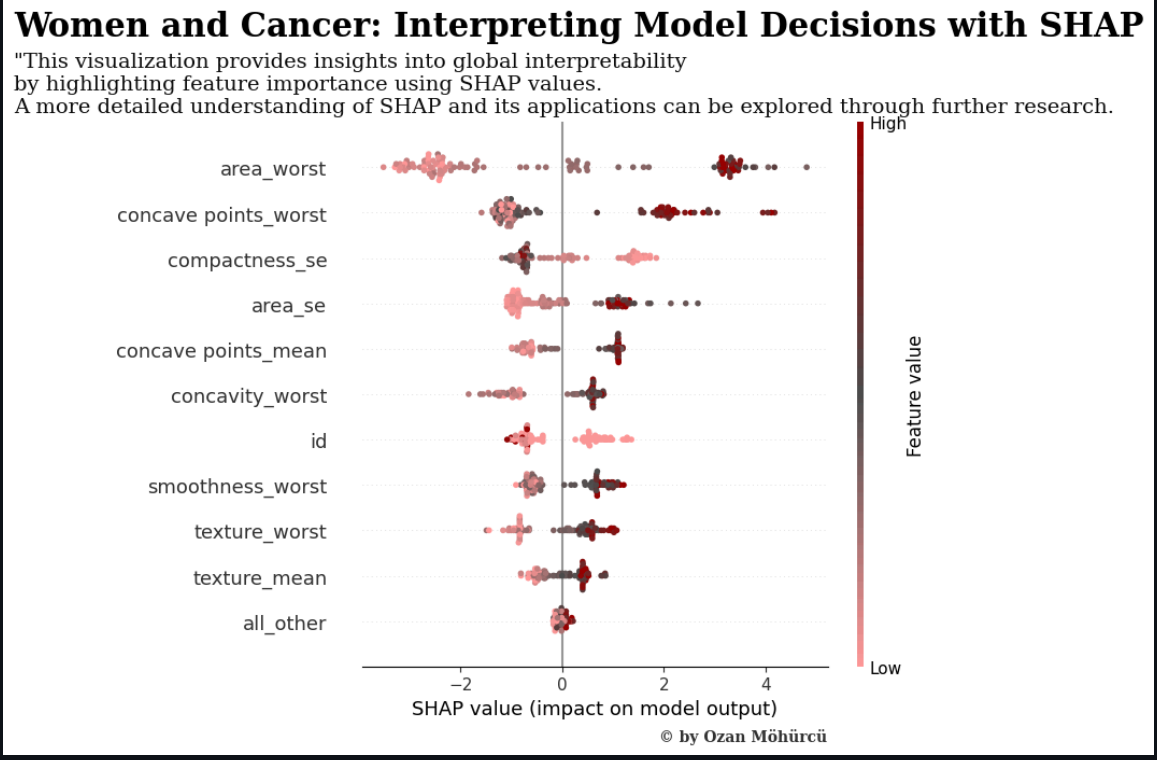
In [25]:

```
fig,ax = plt.subplots(1,2,figsize =(12,6))

## accuracy vs number of features from permutation importance based feature s
ax[0].plot(np.arange(0,len(feat_acc),1),feat_acc, color = colors[0], linewidt
ax[0].scatter(x =np.arange(0,len(feat_acc),1),y=feat_acc,
              color = colors[1], s = 75,zorder = 3,
              linewidth = 1,ec = 'black')
ax[0].set_ylabel('Cross-Validation Accuracy Mean',{'font':'serif','size':12,
ax[0].set_xlabel('Number of Features',{'font':'serif','size':12, 'weight':'bo

## area under curve vs number of features from permutation importance based f
ax[1].plot(np.arange(0,len(feat_auc),1),feat_auc, color=colors[0], linewidth
ax[1].scatter(x =np.arange(0,len(feat_auc),1),y=feat_auc,
              color= colors[2], s= 75,zorder =3,
               linewidth = 1,ec = 'black')
ax[1].set_ylabel('Cross-Validation AUC Mean',{'font':'serif','size':12, 'weig
ax[1].set_xlabel('Number of Features',{'font':'serif','size':12, 'weight':'bo

### title and annotations
## titles and text
fig.text(-0.05,1.18,'Women and Cancer: Impact of Feature Count on Model Perfo
fig.text(-0.05,1.07,'''"This plot demonstrates that even a single selected fe
As the number of features increases, both accuracy and AUC scores improve gra
around 10 to 15 features, the performance metrics plateau, indicating diminis

fig.text(0.61,0.99, "Accuracy",{'font':'serif','size':14, 'weight':'bold', 'c
fig.text(0.7,0.99, '|',{'font':'serif','size':14, 'weight':'bold'})
fig.text(0.72,0.99, "Area Under Curve",{'font':'serif','size':14, 'weight':'b


fig.text(0.70,-0.01,'© by Ozan Möhürcü',{'font':'serif', 'size':10,'weight':'

fig.tight_layout(pad = 2.5, w_pad = 2.5)

fig.show()
```



In [26]:

```
temp_df = temp_X_df.iloc[:,0:10]
temp_df['all_other'] = temp_X_df.iloc[:,10:len(perm_imp_feats_auc)].sum(axis
cols = temp_df.columns
temp_xtrain,temp_xtest, temp_ytrain,temp_ytest = train_test_split(temp_df, te

temp_model = XGBClassifier(eval_metric = 'logloss')
temp_model.fit(temp_xtrain,temp_ytrain)

### shapvalues
explainer = shap.TreeExplainer(temp_model)

shap_values = explainer.shap_values(temp_xtest)

cmap = mpl.colors.LinearSegmentedColormap.from_list("",[colors[1],colors[2],c
shap.summary_plot(shap_values,temp_xtest,
                  show = False,cmap = cmap)

# plot settings
## titles and text
plt.gcf().text(-0.1,1.1,'Women and Cancer: Interpreting Model Decisions with
plt.gcf().text(-0.1,0.98,'''"This visualization provides insights into global
by highlighting feature importance using SHAP values.
A more detailed understanding of SHAP and its applications can be explored th


plt.gcf().text(0.65,-0.01,'© by Ozan Möhürcü',{'font':'serif', 'size':10,'wei



plt.gcf().show()
```



Modeling

In [27]:

```
# splitting data into training and test set
```

```
# splitting data into training and test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, r
```

In [28]:
```
# scaling data

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Logistic Regression

In [29]:
```
# fitting data to model

from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
```

Out[29]:
```
LogisticRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [30]:
```
# model predictions

y_pred = log_reg.predict(X_test)
```

In [31]:
```
# accuracy score

from sklearn.metrics import accuracy_score, confusion_matrix, classification_

print(accuracy_score(y_train, log_reg.predict(X_train)))

log_reg_acc = accuracy_score(y_test, log_reg.predict(X_test))
print(log_reg_acc)
```

```
0.9849246231155779
0.9707602339181286
```

In [32]:
```
# confusion matrix

print(confusion_matrix(y_test, y_pred))
```

```
[[106   2]
 [  3  60]]
```

In [33]:
```
# classification report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       108
           1       0.97      0.95      0.96        63

    accuracy                           0.97       171
   macro avg       0.97      0.97      0.97       171
weighted avg       0.97      0.97      0.97       171
```

Decision Tree Classifier

In [34]:
```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()

parameters = {
    'criterion' : ['gini', 'entropy'],
    'max_depth' : range(2, 32, 1),
    'min_samples_leaf' : range(1, 10, 1),
    'min_samples_split' : range(2, 10, 1),
    'splitter' : ['best', 'random']
}

grid_search_dt = GridSearchCV(dtc, parameters, cv = 5, n_jobs = -1, verbose =
grid_search_dt.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 8640 candidates, totalling 43200 fits
```

Out[34]:
```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 32),
                         'min_samples_leaf': range(1, 10),
                         'min_samples_split': range(2, 10),
                         'splitter': ['best', 'random']},
             verbose=1)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [35]:
```
# best parameters

grid_search_dt.best_params_
```

Out[35]:
```
{'criterion': 'gini',
 'max_depth': 22,
 'min_samples_leaf': 3,
 'min_samples_split': 5,
 'splitter': 'random'}
```

In [36]:
```
# best score

grid_search_dt.best_score_
```

Out[36]: 0.9598101265822784

In [37]:
```
dtc = DecisionTreeClassifier(criterion = 'entropy', max_depth = 28, min_sampl
dtc.fit(X_train, y_train)
```

Out[37]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=28, min_samples_
split=8,
                       splitter='random')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [38]:
```
y_pred = dtc.predict(X_test)
```

In [39]:
```
# accuracy score

print(accuracy_score(y_train, dtc.predict(X_train)))

dtc_acc = accuracy_score(y_test, dtc.predict(X_test))
print(dtc_acc)
```

```
0.9874371859296482
0.935672514619883
```

In [40]:
```
# confusion matrix

print(confusion_matrix(y_test, y_pred))
```

```
[[102   6]
 [  5  58]]
```

In [41]:
```
# classification report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       108
           1       0.91      0.92      0.91        63

    accuracy                           0.94       171
   macro avg       0.93      0.93      0.93       171
weighted avg       0.94      0.94      0.94       171
```

In [42]:
```
models = pd.DataFrame({
    'Model': ['Logistic Regression', 'Decision Tree Classifier', ],
    'Score': [log_reg_acc, dtc_acc,]
})
```