pairplot using seaborn

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
```

In [2]:
```
from sklearn.datasets import load_breast_cancer
canncer_dataset=load_breast_cancer()
canncer_dataset
```

Out[2]:
```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
        1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
```

```
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnostic) dataset\n--------------------
--------------------\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 569\n\n    :Number of A
ibutes: 30 numeric, predictive attributes and the class\n\n    :Attribute Information:\n        - radius (m
of distances from center to points on the perimeter)\n        - texture (standard deviation of gray-scale v
es)\n        - perimeter\n        - area\n        - smoothness (local variation in radius lengths)\n
compactness (perimeter^2 / area - 1.0)\n        - concavity (severity of concave portions of the contour)\n
- concave points (number of concave portions of the contour)\n        - symmetry\n        - fractal dimensi
("coastline approximation" - 1)\n\n        The mean, standard error, and "worst" or largest (mean of the th
\n        worst/largest values) of these features were computed for each image,\n        resulting in 30 fe
res.  For instance, field 0 is Mean Radius, field\n        10 is Radius SE, field 20 is Worst Radius.\n\n
- class:\n                - WDBC-Malignant\n                - WDBC-Benign\n\n    :Summary Statistics:\n\n
===================================== ====== ======\n                                          Min    Max\n
===================================== ====== ======\n    radius (mean):                        6.981  28.11
texture (mean):                        9.71   39.28\n    perimeter (mean):                    43.79  188.5\
area (mean):                           143.5  2501.0\n    smoothness (mean):                   0.053  0.163
compactness (mean):                    0.019  0.345\n    concavity (mean):                    0.0    0.427\
concave points (mean):                 0.0    0.201\n    symmetry (mean):                     0.106  0.304\
fractal dimension (mean):              0.05   0.097\n    radius (standard error):             0.112  2.873\
texture (standard error):              0.36   4.885\n    perimeter (standard error):          0.757  21.98\
area (standard error):                 6.802  542.2\n    smoothness (standard error):         0.002  0.031\
compactness (standard error):          0.002  0.135\n    concavity (standard error):          0.0    0.396\
concave points (standard error):       0.0    0.053\n    symmetry (standard error):           0.008  0.079\
fractal dimension (standard error):    0.001  0.03\n    radius (worst):                       7.93   36.04\n
texture (worst):                       12.02  49.54\n    perimeter (worst):                   50.41  251.2\
area (worst):                          185.2  4254.0\n    smoothness (worst):                  0.071  0.223
compactness (worst):                   0.027  1.058\n    concavity (worst):                   0.0    1.252\
concave points (worst):                0.0    0.291\n    symmetry (worst):                    0.156  0.664\
fractal dimension (worst):             0.055  0.208\n    ===================================== ====== ======
\n    :Missing Attribute Values: None\n\n    :Class Distribution: 212 - Malignant, 357 - Benign\n\n    :Cre
r:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian\n\n    :Donor: Nick Street\n\n    :Date: No
ber, 1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.\nhttps://goo.gl/U2Uwz2
\nFeatures are computed from a digitized image of a fine needle\naspirate (FNA) of a breast mass.  They des
```

```
be\ncharacteristics of the cell nuclei present in the image.\n\nSeparating plane described above was obtain
using\nMultisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via Linear Programming
Proceedings of the 4th\nMidwest Artificial Intelligence and Cognitive Science Society,\npp. 97-101, 1992],
lassification method which uses linear\nprogramming to construct a decision tree.  Relevant features\nwere
ected using an exhaustive search in the space of 1-4\nfeatures and 1-3 separating planes.\n\nThe actual lin
program used to obtain the separating plane\nin the 3-dimensional space is that described in:\n[K. P. Benne
and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two Linearly Inseparable Sets",\nOptim
tion Methods and Software 1, 1992, 23-34].\n\nThis database is also available through the UW CS ftp server:
\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. topic:: References\n\n   - W.N. S
et, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction \n     for breast tumor diagnosis. IS&T/S
1993 International Symposium on \n     Electronic Imaging: Science and Technology, volume 1905, pages 861-8
0,\n     San Jose, CA, 1993.\n   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis
\n     prognosis via linear programming. Operations Research, 43(4), pages 570-577, \n     July-August 1995
- W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques\n     to diagnose breast can
from fine-needle aspirates. Cancer Letters 77 (1994) \n     163-171.',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
 'filename': 'c:\\python3.8.3\\lib\\site-packages\\sklearn\\datasets\\data\\breast_cancer.csv'}
```

In [3]:
```
canner_df = pd.DataFrame(np.c_[canncer_dataset["data"],canncer_dataset["target"]],
                         columns=np.append(canncer_dataset["feature_names"],["targget"]))
canner_df
```

1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80	1
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50	1
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	2
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	1
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70	1
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60	1
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.16	

569 rows × 31 columns

```
# Signature:
sns.pairplot(
    data,
    hue=None,
    hue_order=None,
    palette=None,
    vars=None,
    x_vars=None,
    y_vars=None,
    kind='scatter',
    diag_kind='auto',
    markers=None,
    height=2.5,
    aspect=1,
```

```
    corner=False,
    dropna=True,
    plot_kws=None,
    diag_kws=None,
    grid_kws=None,
    size=None,
)
Docstring:
Plot pairwise relationships in a dataset.

By default, this function will create a grid of Axes such that each numeric
variable in ``data`` will by shared in the y-axis across a single row and
in the x-axis across a single column. The diagonal Axes are treated
differently, drawing a plot to show the univariate distribution of the data
for the variable in that column.

It is also possible to show a subset of variables or plot different
variables on the rows and columns.

This is a high-level interface for :class:`PairGrid` that is intended to
make it easy to draw a few common styles. You should use :class:`PairGrid`
directly if you need more flexibility.

Parameters
----------
data : DataFrame
    Tidy (long-form) dataframe where each column is a variable and
    each row is an observation.
hue : string (variable name), optional
    Variable in ``data`` to map plot aspects to different colors.
hue_order : list of strings
    Order for the levels of the hue variable in the palette
palette : dict or seaborn color palette
    Set of colors for mapping the ``hue`` variable. If a dict, keys
    should be values  in the ``hue`` variable.
vars : list of variable names, optional
    Variables within ``data`` to use, otherwise use every column with
    a numeric datatype.
{x, y}_vars : lists of variable names, optional
```

```
    Variables within ``data`` to use separately for the rows and
    columns of the figure; i.e. to make a non-square plot.
kind : {'scatter', 'reg'}, optional
    Kind of plot for the non-identity relationships.
diag_kind : {'auto', 'hist', 'kde', None}, optional
    Kind of plot for the diagonal subplots. The default depends on whether
    ``"hue"`` is used or not.
markers : single matplotlib marker code or list, optional
    Either the marker to use for all datapoints or a list of markers with
    a length the same as the number of levels in the hue variable so that
    differently colored points will also have different scatterplot
    markers.
height : scalar, optional
    Height (in inches) of each facet.
aspect : scalar, optional
    Aspect * height gives the width (in inches) of each facet.
corner : bool, optional
    If True, don't add axes to the upper (off-diagonal) triangle of the
    grid, making this a "corner" plot.
dropna : boolean, optional
    Drop missing values from the data before plotting.
{plot, diag, grid}_kws : dicts, optional
    Dictionaries of keyword arguments. ``plot_kws`` are passed to the
    bivariate plotting function, ``diag_kws`` are passed to the univariate
    plotting function, and ``grid_kws`` are passed to the :class:`PairGrid`
    constructor.

Returns
-------
grid : :class:`PairGrid`
    Returns the underlying :class:`PairGrid` instance for further tweaking.

See Also
--------
PairGrid : Subplot grid for more flexible plotting of pairwise
           relationships.

Examples
--------
```

```
Draw scatterplots for joint relationships and histograms for univariate
distributions:

.. plot::
    :context: close-figs

    >>> import seaborn as sns; sns.set(style="ticks", color_codes=True)
    >>> iris = sns.load_dataset("iris")
    >>> g = sns.pairplot(iris)

Show different levels of a categorical variable by the color of plot
elements:

.. plot::
    :context: close-figs

    >>> g = sns.pairplot(iris, hue="species")

Use a different color palette:

.. plot::
    :context: close-figs

    >>> g = sns.pairplot(iris, hue="species", palette="husl")

Use different markers for each level of the hue variable:

.. plot::
    :context: close-figs

    >>> g = sns.pairplot(iris, hue="species", markers=["o", "s", "D"])

Plot a subset of variables:

.. plot::
    :context: close-figs

    >>> g = sns.pairplot(iris, vars=["sepal_width", "sepal_length"])
```

```
Draw larger plots:

.. plot::
    :context: close-figs

    >>> g = sns.pairplot(iris, height=3,
    ...                  vars=["sepal_width", "sepal_length"])

Plot different variables in the rows and columns:

.. plot::
    :context: close-figs

    >>> g = sns.pairplot(iris,
    ...                  x_vars=["sepal_width", "sepal_length"],
    ...                  y_vars=["petal_width", "petal_length"])

Plot only the lower triangle of bivariate axes:

.. plot::
    :context: close-figs

    >>> g = sns.pairplot(iris, corner=True)

Use kernel density estimates for univariate plots:

.. plot::
    :context: close-figs

    >>> g = sns.pairplot(iris, diag_kind="kde")

Fit linear regression models to the scatter plots:

.. plot::
    :context: close-figs

    >>> g = sns.pairplot(iris, kind="reg")
```

```
Pass keyword arguments down to the underlying functions (it may be easier
to use :class:`PairGrid` directly):

.. plot::
    :context: close-figs
```

```
sns.pairplot(
    data,
    hue=None,
    hue_order=None,
    palette=None,
    vars=None,
    x_vars=None,
    y_vars=None,
    kind='scatter',
    diag_kind='auto',
    markers=None,
    height=2.5,
    aspect=1,
    corner=False,
    dropna=True,
    plot_kws=None,
    diag_kws=None,
    grid_kws=None,
    size=None,
)
```

In [ ]:

```

```

In [4]:

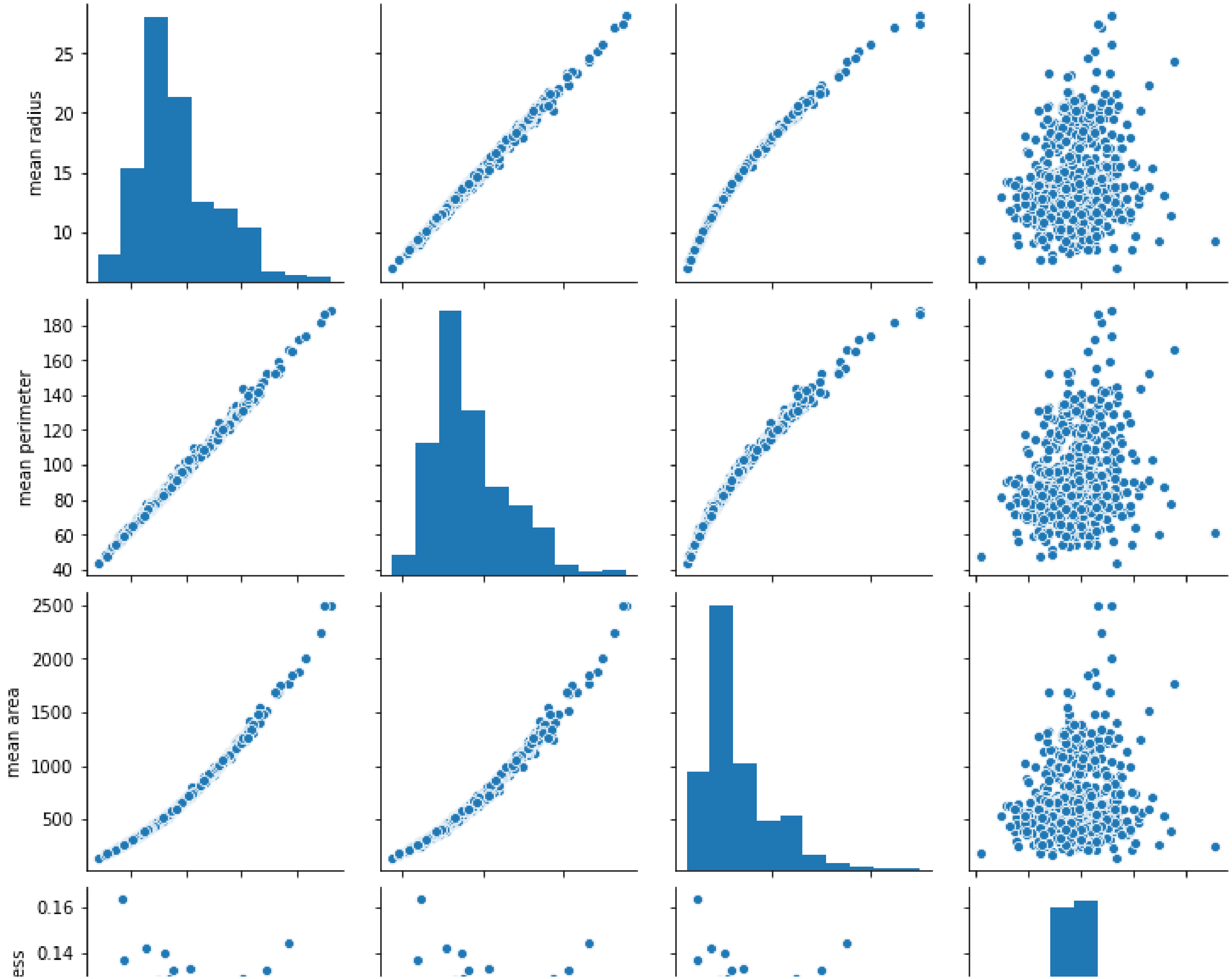
```
# sns.pairplot(canner_df)
```

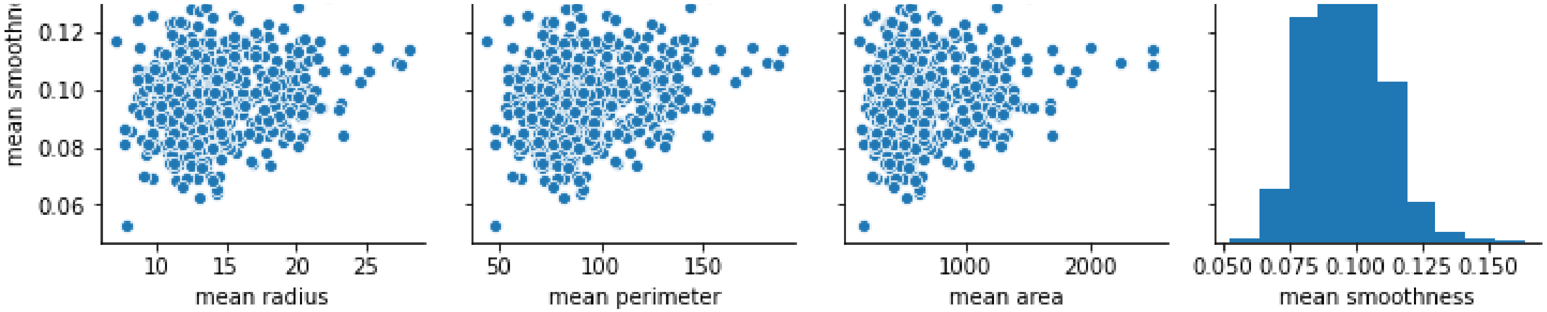
In [5]:
```
sns.pairplot(canner_df,vars=['mean radius', 'mean texture', 'mean perimeter', 'mean area',
        'mean smoothness'])
```

In [39]:
```
# sns.pairplot(canner_df,vars=['mean radius',  'mean perimeter', 'mean area',
#          'mean smoothness'],hue="traget")
```

In [7]:
```
sns.pairplot(canner_df,vars=['mean radius',  'mean perimeter', 'mean area',
        'mean smoothness'],hue_order=(1.0,0.0))
```

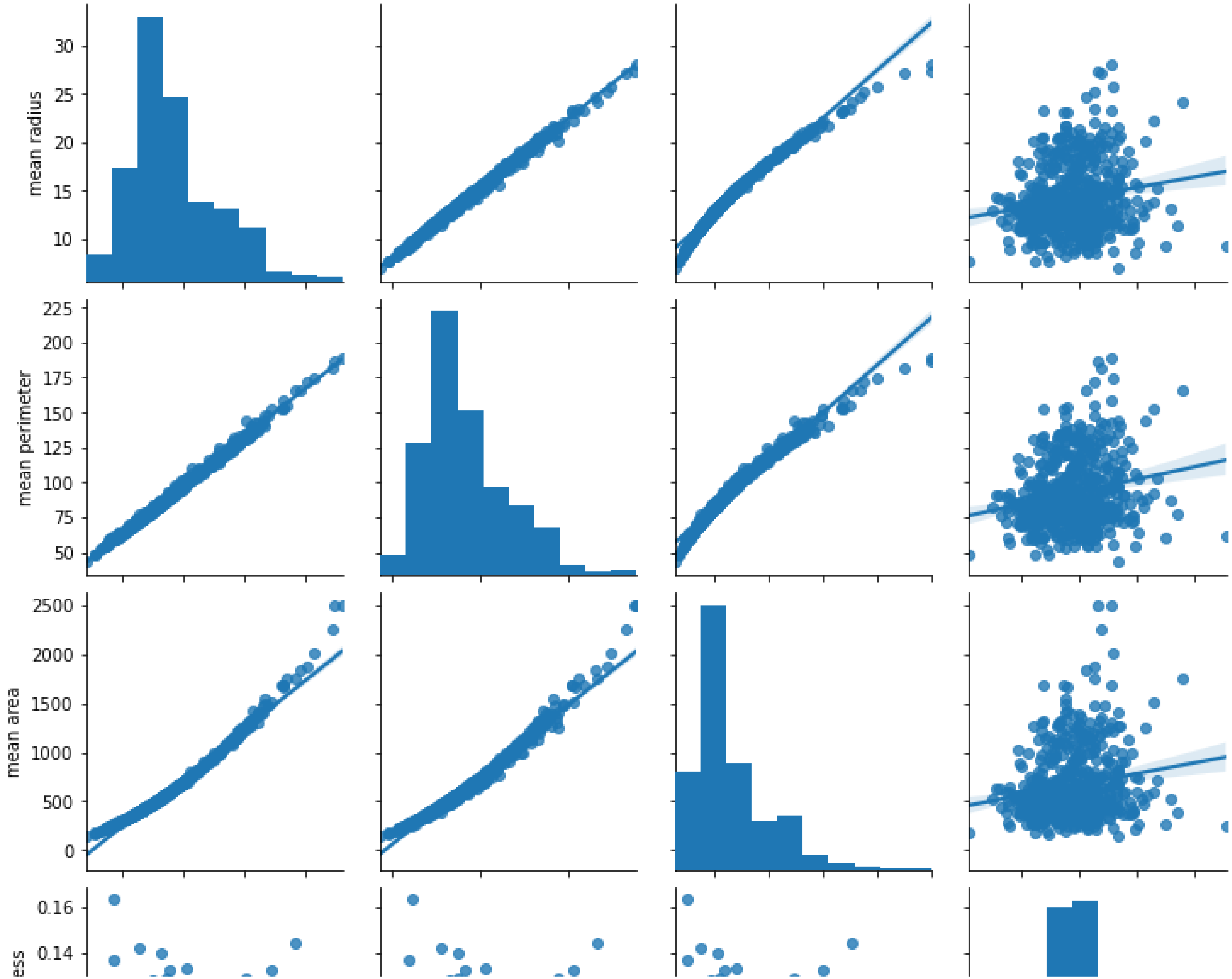
Out[7]: <seaborn.axisgrid.PairGrid at 0x1050fa00>

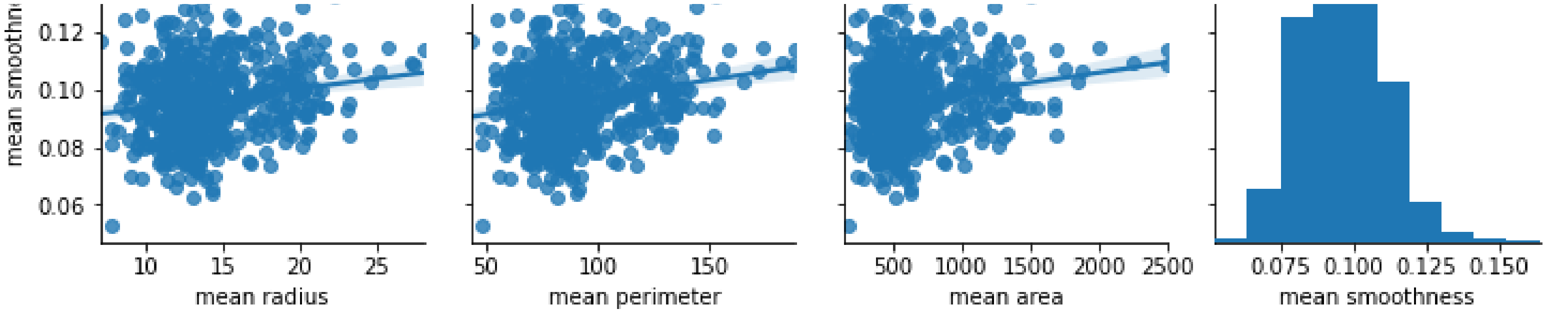



mean smoothness
0.12
0.10
0.08
0.06
10 15 20 25
mean radius
50 100 150
mean perimeter
1000 2000
mean area
0.050 0.075 0.100 0.125 0.150
mean smoothness

In [12]:
```
sns.pairplot(canner_df,vars=['mean radius',  'mean perimeter', 'mean area',
        'mean smoothness'],palette="Drak2",kind="reg")
```

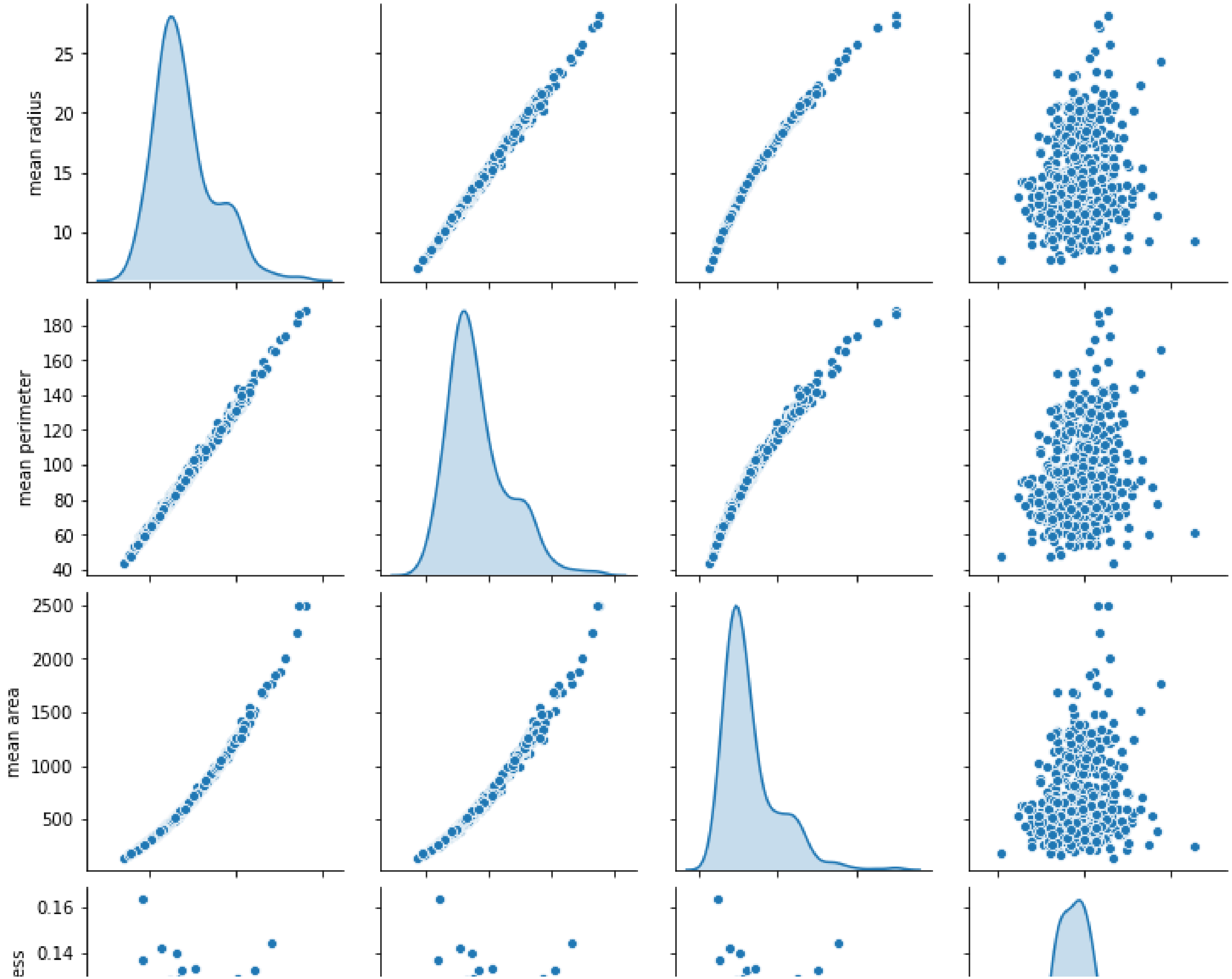
Out[12]: <seaborn.axisgrid.PairGrid at 0x12145fe8>

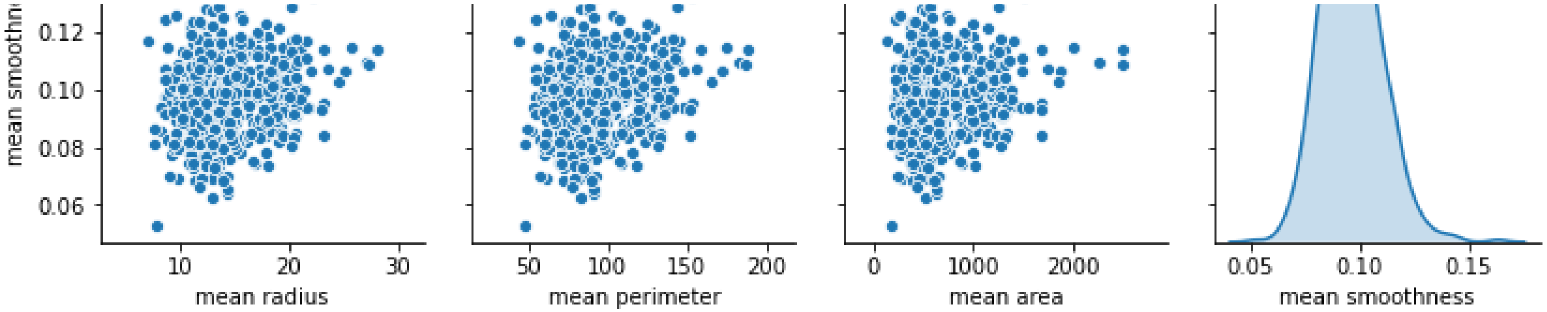



mean smoothness
0.12
0.10
0.08
0.06
10
15
20
25
mean radius
50
100
150
mean perimeter
500
1000
1500
2000
2500
mean area
0.075
0.100
0.125
0.150
mean smoothness

In [14]:
```
sns.pairplot(canner_df,vars=['mean radius',  'mean perimeter', 'mean area',
         'mean smoothness'],palette="Drak2",diag_kind="kde")
```

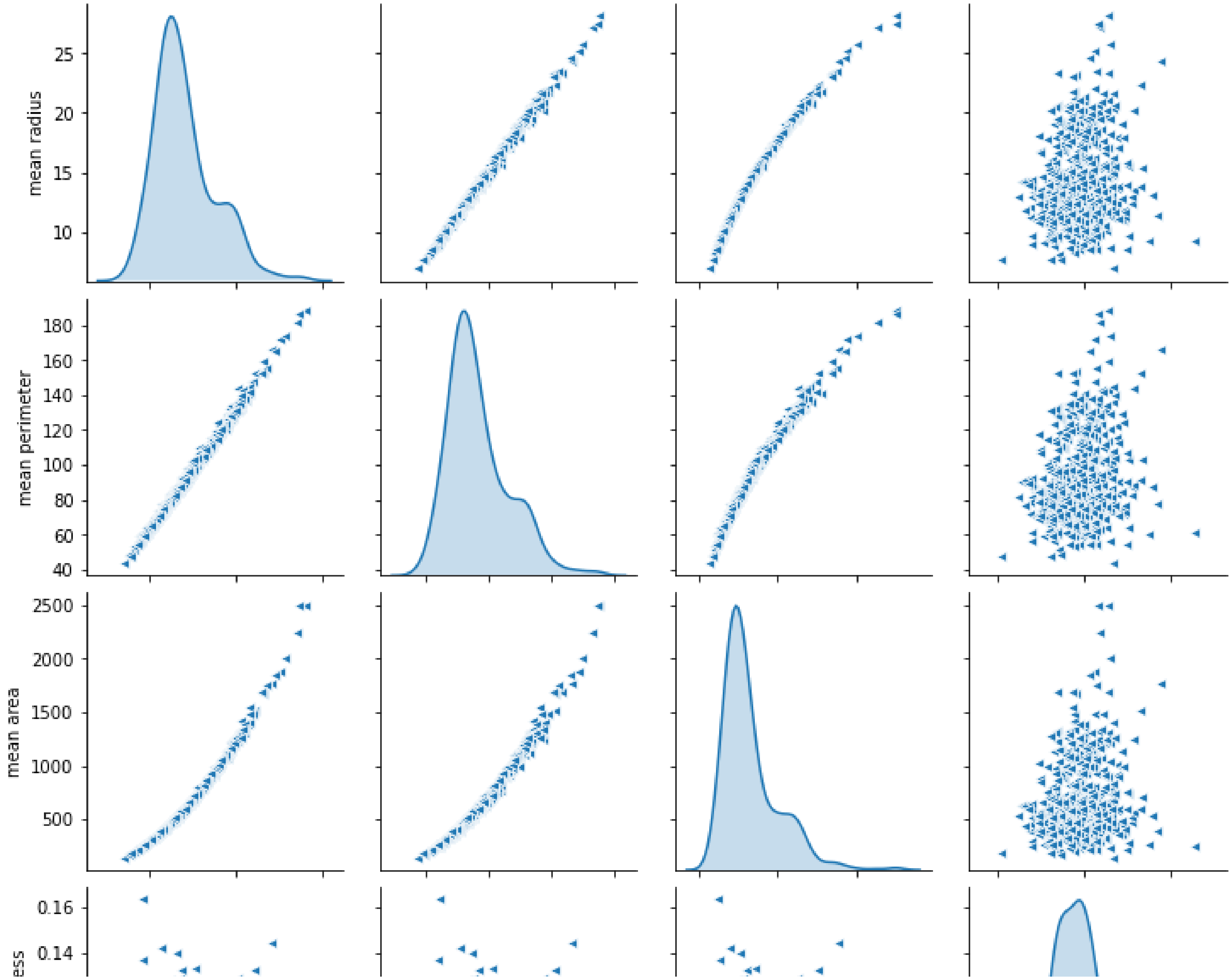
Out[14]: <seaborn.axisgrid.PairGrid at 0x13f298f8>

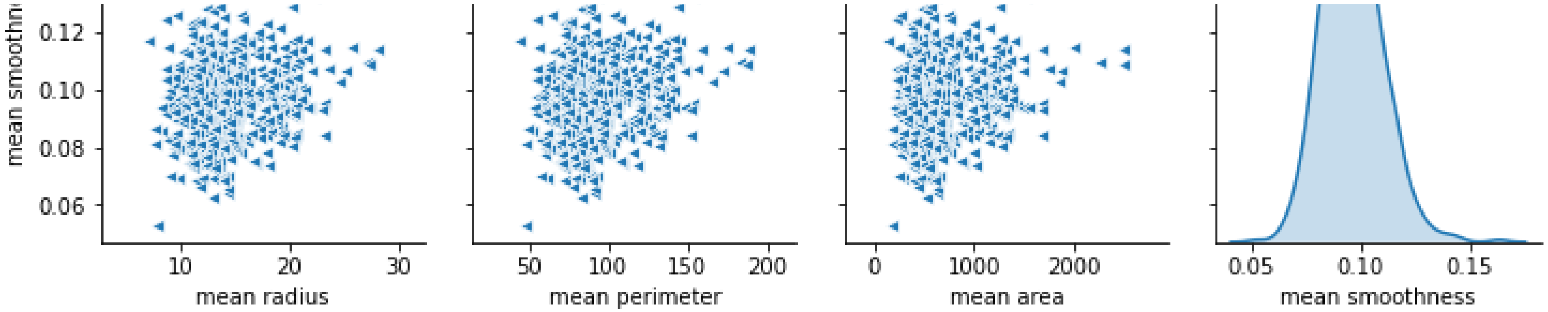



mean smoothne
0.12
0.10
0.08
0.06
10
20
30
mean radius
50
100
150
200
mean perimeter
0
1000
2000
mean area
0.05
0.10
0.15
mean smoothness

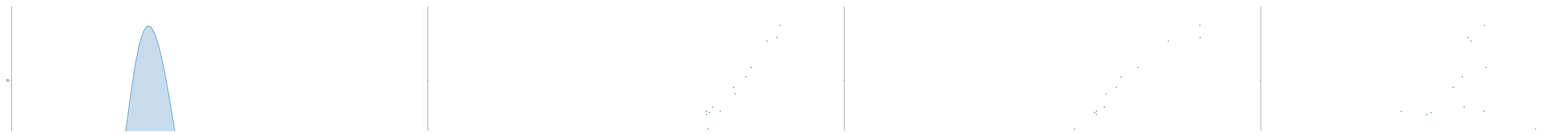
In [19]:
```
sns.pairplot(canner_df,vars=['mean radius',  'mean perimeter', 'mean area',
         'mean smoothness'],palette="Drak2",diag_kind="kde",markers=["<"])
```

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x15f7d4f0>
```





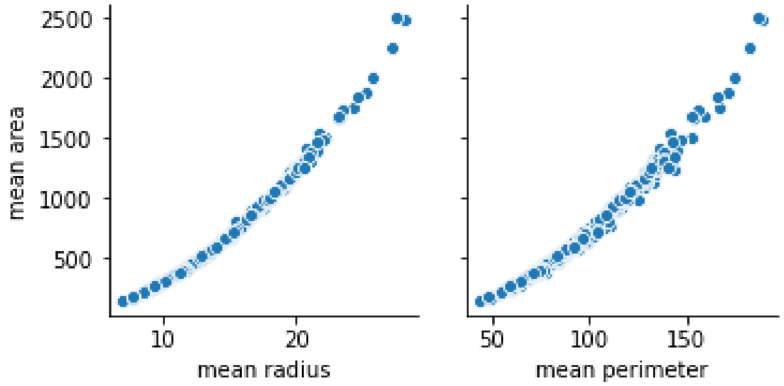
In [20]:
```
sns.pairplot(canner_df,vars=['mean radius',  'mean perimeter', 'mean area',
         'mean smoothness'],palette="Drak2",diag_kind="kde",markers=["<"],height=20)
```



In [37]:

```
sns.pairplot(canner_df,
             x_vars=['mean radius','mean perimeter'],y_vars=['mean area'] )
```

Out[37]: <seaborn.axisgrid.PairGrid at 0x271c7790>



In [ ]: