In [1]:

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
import os
```

In [2]:

```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=FutureWarning)
```

In [3]:

```
bcancer = pd.read_csv("data.csv")
bcancer.head(4)
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	

4 rows × 33 columns

In [4]:

```
bcancer.shape
```

Out[4]:

```
(569, 33)
```

In [5]:

```
bcancer.dtypes
```

Out[5]:

```
id                           int64
diagnosis                   object
radius_mean                float64
texture_mean               float64
perimeter_mean             float64
area_mean                  float64
smoothness_mean            float64
compactness_mean           float64
concavity_mean             float64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
concave points_se          float64
symmetry_se                float64
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
Unnamed: 32                float64
dtype: object
```

In [6]:

```
bcancer.drop('Unnamed: 32', axis = 1, inplace = True)
```

In [7]:

```
bcancer.drop('id',axis = 1, inplace=True)
```

In [8]:

```
bcancer.isnull().sum()
```

Out[8]:

```
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
dtype: int64
```

In [9]:

```
bcancer.describe(include='object')
```

Out[9]:

	diagnosis
count	569
unique	2
top	B
freq	357

In [10]:

```
bcancer.describe()
```

Out[10]:

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

8 rows × 30 columns

In [11]:

```
bcancer.groupby('diagnosis').mean()
```

Out[11]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
diagnosis						
B	12.146524	17.914762	78.075406	462.790196	0.092478	
M	17.462830	21.604906	115.365377	978.376415	0.102898	

2 rows × 30 columns

In [12]:

```
bcancer['diagnosis'] = (bcancer['diagnosis'] == 'M').astype('int')
```

In [13]:

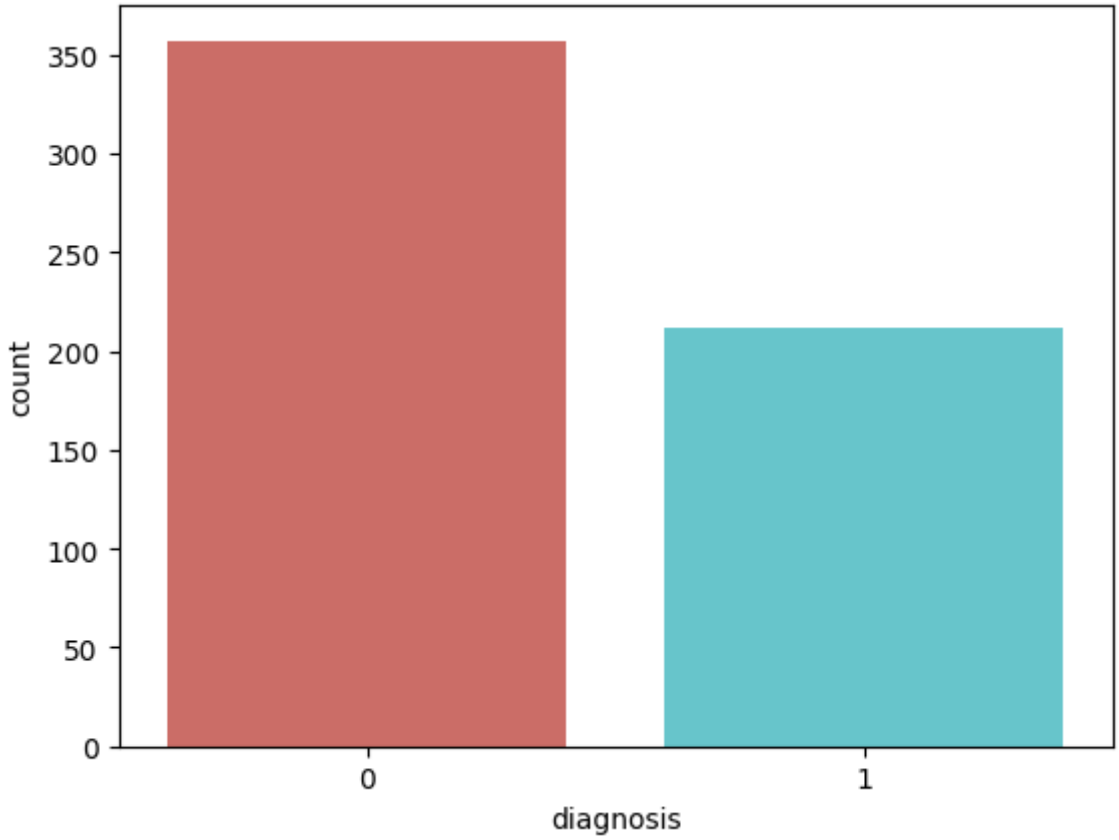
```
bcancer['diagnosis'].value_counts()
```

Out[13]:

```
diagnosis
0    357
1    212
Name: count, dtype: int64
```

In [14]:

```
sns.countplot(x='diagnosis',data = bcancer,palette='hls')
plt.show()
```

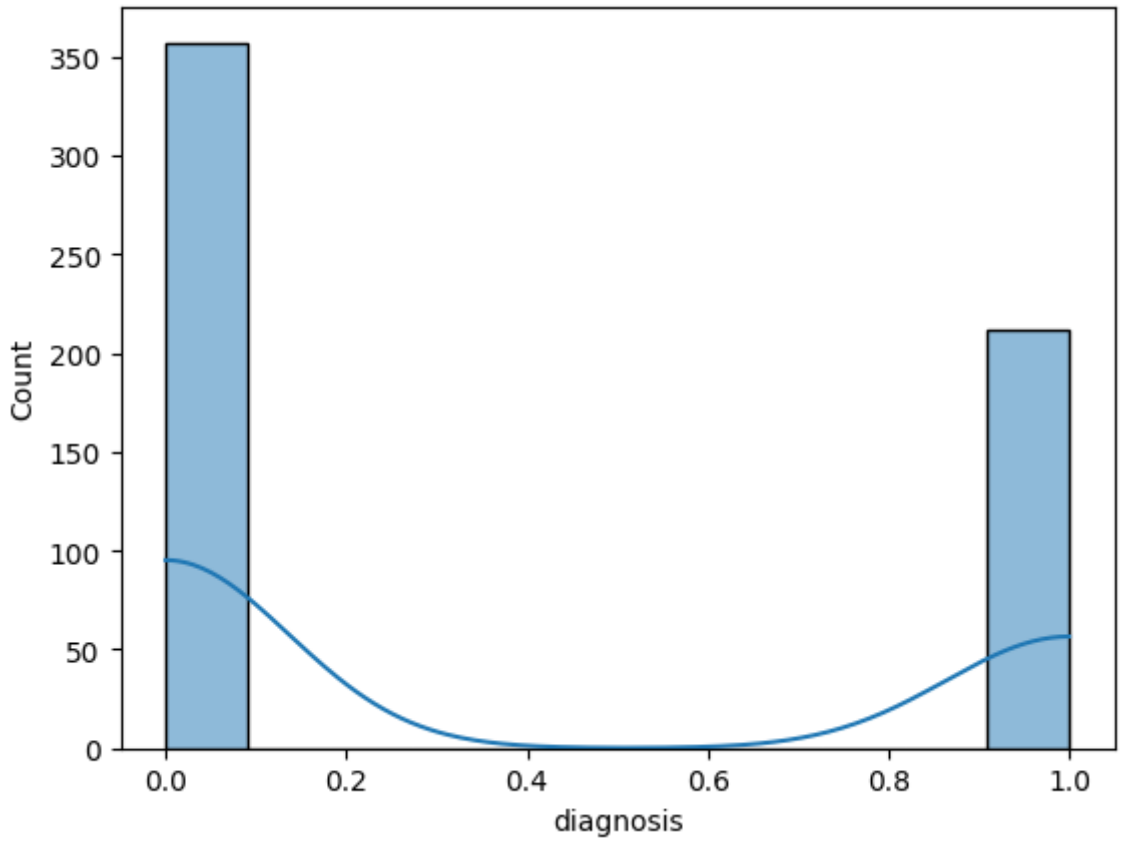


In [22]:

```
sns.histplot(bcancer['diagnosis'], kde=True)
```

Out[22]:

```
<Axes: xlabel='diagnosis', ylabel='Count'>
```

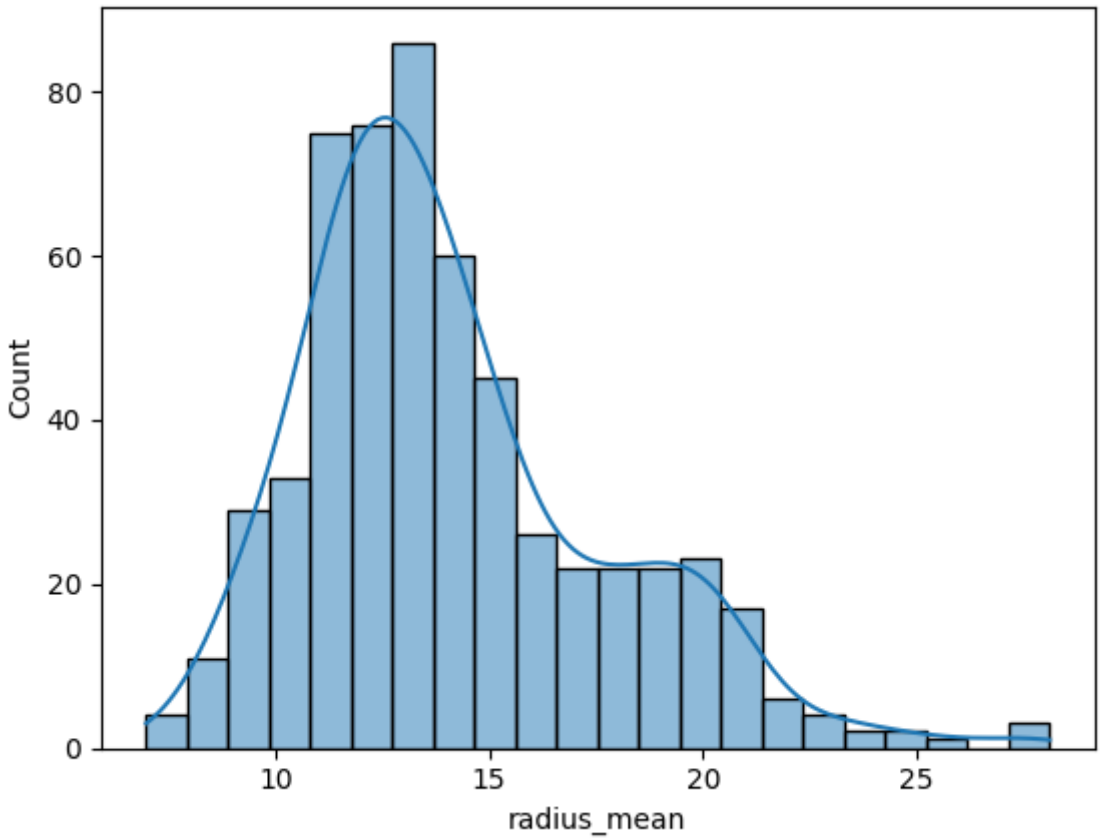


In [21]:

```
sns.histplot(bcancer['radius_mean'], kde=True)
```

Out[21]:

```
<Axes: xlabel='radius_mean', ylabel='Count'>
```

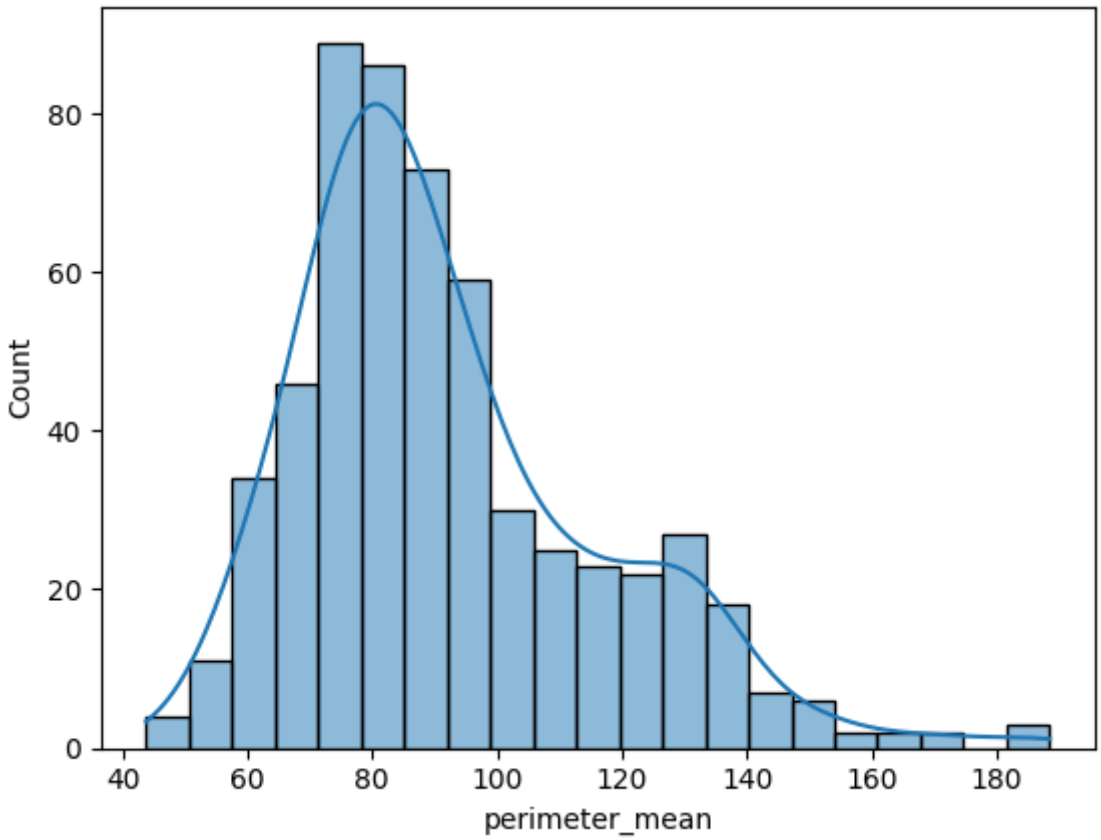


In [23]:

```
sns.histplot(bcancer['perimeter_mean'], kde=True)
```

Out[23]:

```
<Axes: xlabel='perimeter_mean', ylabel='Count'>
```



In [24]:

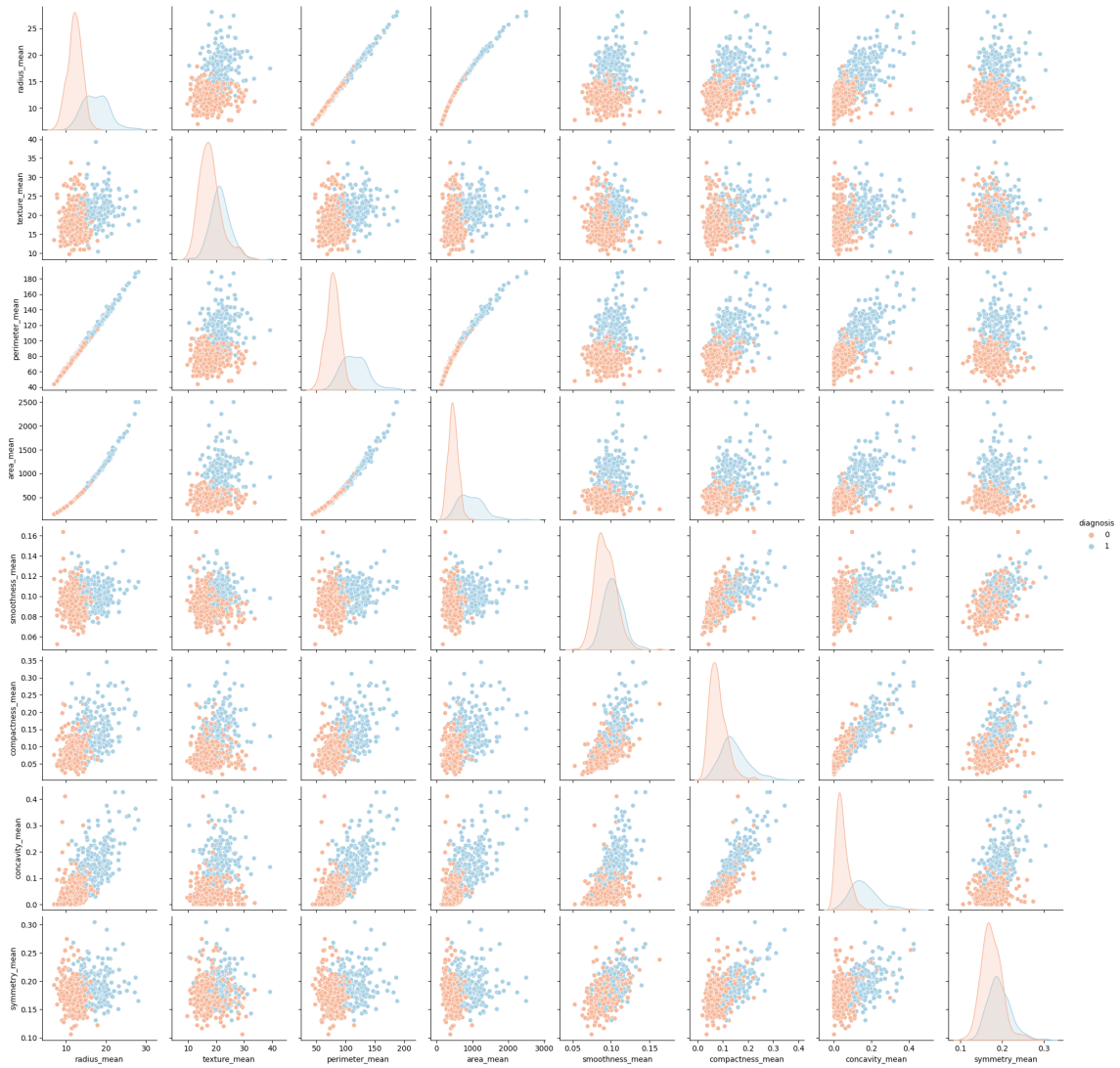
```
features_mean=list(bcancer.columns[1:11])


bcancer_M = bcancer[bcancer['diagnosis'] == 1]
bcancer_B = bcancer[bcancer['diagnosis'] == 0]
```

In [25]:

```
cols = ['diagnosis',
        'radius_mean',
        'texture_mean',
        'perimeter_mean',
        'area_mean',
        'smoothness_mean',
        'compactness_mean',
        'concavity_mean',
        'symmetry_mean']

sns.pairplot(data = bcancer[cols], hue = 'diagnosis', palette = 'RdBu')
```

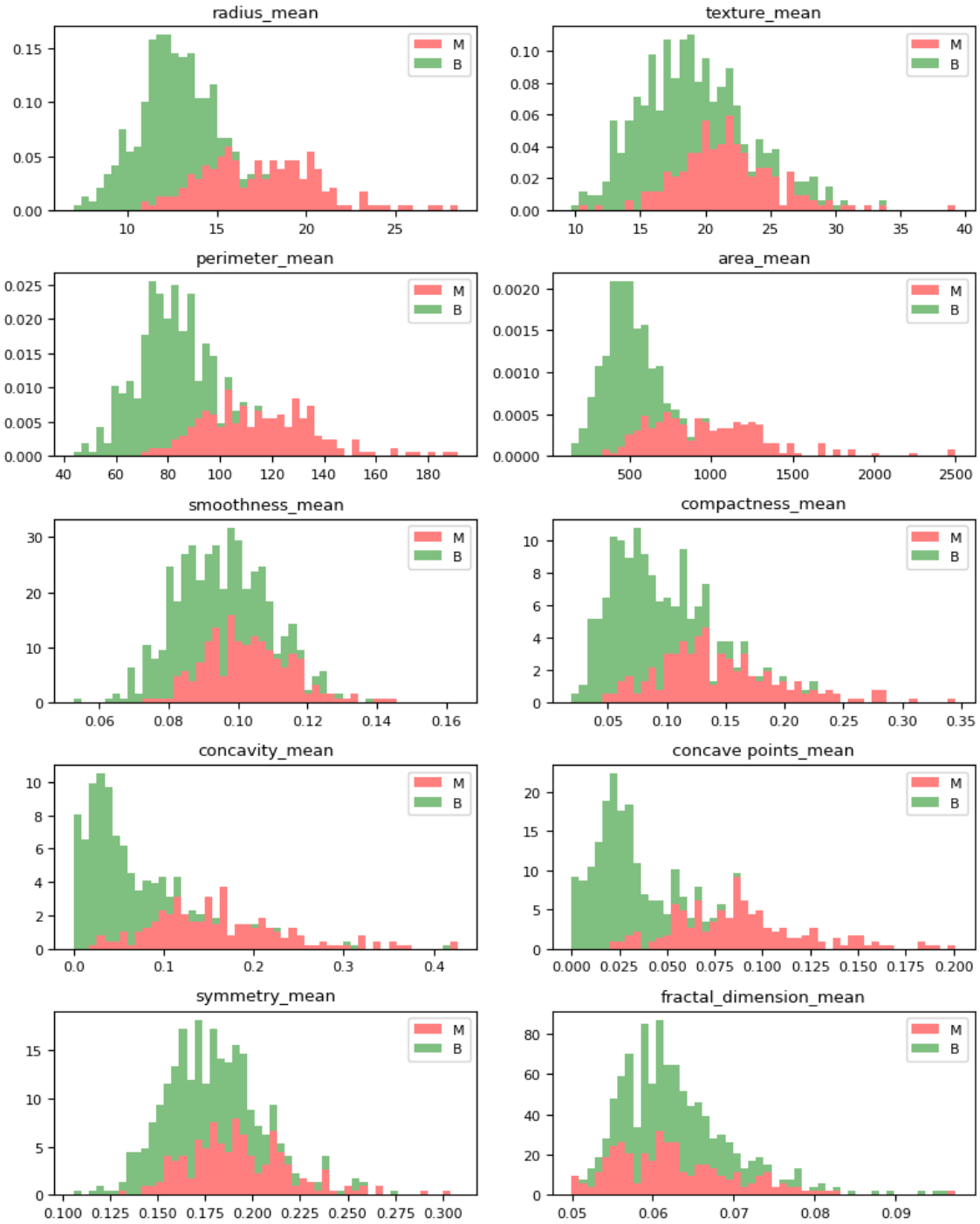
Out[25]:

```
<seaborn.axisgrid.PairGrid at 0x20a331d7850>
```



In [30]:

```
import matplotlib.pyplot as plt

plt.rcParams.update({'font.size': 8})
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(8, 10))
axes = axes.ravel()

for idx, ax in enumerate(axes):
    binwidth = (max(bcancer[features_mean[idx]]) - min(bcancer[features_mean[idx]])) / 5
    ax.hist([bcancer_M[features_mean[idx]], bcancer_B[features_mean[idx]]], bins=np.aran
            max(bcancer[features_mean[idx]]) + binwidth, binwidth), alpha=0.5, stacked=T
            density=True, label=['M', 'B'], color=['r', 'g'])
    ax.legend(loc='upper right')
    ax.set_title(features_mean[idx])

plt.tight_layout()
plt.show()
```



In [31]:

```
# Building the model with all the attributes

x=bcancer.iloc[:,1:31]
y=bcancer['diagnosis']
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = .2, random_state=10)
```

In [32]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(455, 30)
(114, 30)
(455,)
(114,)
```

In [33]:

```
pd.options.display.max_columns = None
x_train.head(3)
```

Out[33]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactne
567	20.60	29.33	140.10	1265.0	0.11780	
295	13.77	13.27	88.06	582.7	0.09198	
91	15.37	22.76	100.20	728.2	0.09200	

In [45]:

```
#Creating an instance of logistic regression model
from sklearn.linear_model import LogisticRegression
logistic_model1 = LogisticRegression()

#We fit our model to data
fitted_model1 = logistic_model1.fit(x_train,y_train)

#We use predict_proba() to predict the probabilities
predictedvalues1 = fitted_model1.predict(x_test)

#We print the probabilites to take a glance
print(predictedvalues1)
```

```
[1 0 0 1 0 0 0 1 1 1 0 0 1 0 1 1 0 0 0 1 1 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 1 1 0 0 0 0
 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 0 0 1 1 0 0 0
 0 1 1]
```

```
C:\Users\CHINMAY\AppData\Roaming\Python\Python311\site-packages\sklearn\li
near_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

In [47]:

```
#Accuracy of the mode:-

print('Accuracy of logistic regression classifier on test set: {:.3f}'.format(logistic_m
```

```
Accuracy of logistic regression classifier on test set: 0.930
```

In [48]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix1 = confusion_matrix(y_test,predictedvalues1)
print(confusion_matrix1)
```

```
[[70  5]
 [ 3 36]]
```

In [49]:

```
#Generating the classification report.
from sklearn.metrics import classification_report
print(classification_report(y_test, predictedvalues1))
```

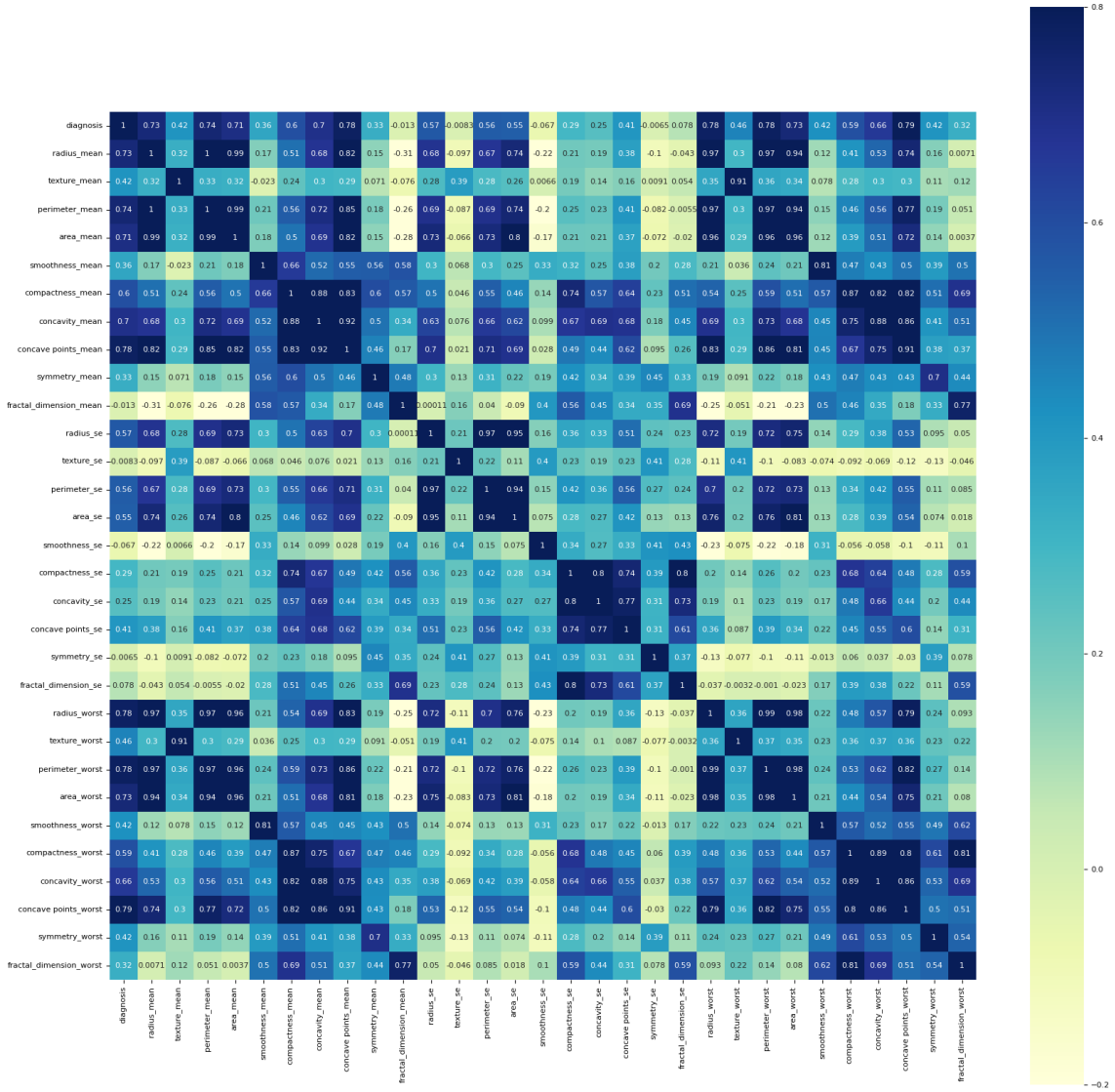
```
              precision    recall  f1-score   support

           0       0.96      0.93      0.95        75
           1       0.88      0.92      0.90        39

    accuracy                           0.93       114
   macro avg       0.92      0.93      0.92       114
weighted avg       0.93      0.93      0.93       114
```

In [51]:

```
#Generating the correlation matrix.
fig, ax=plt.subplots(figsize=(20,20))
correlation=bcancer.corr()
sns.heatmap(correlation,square=True, vmin=-0.2, vmax=0.8,cmap="YlGnBu", annot=True)
```

Out[51]:

<Axes: >



In [ ]:

```
```