In [73]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [74]:
```
df = pd.read_csv("breast cancer.csv")
```

In [75]:
```
df.head()
```

Out[75]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mea
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.300
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.086
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.197
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.241
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.198

5 rows × 33 columns

In [76]:
```
df.shape
```

Out[76]:
```
(569, 33)
```

In [77]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [78]:
```
df['diagnosis'].unique()
```

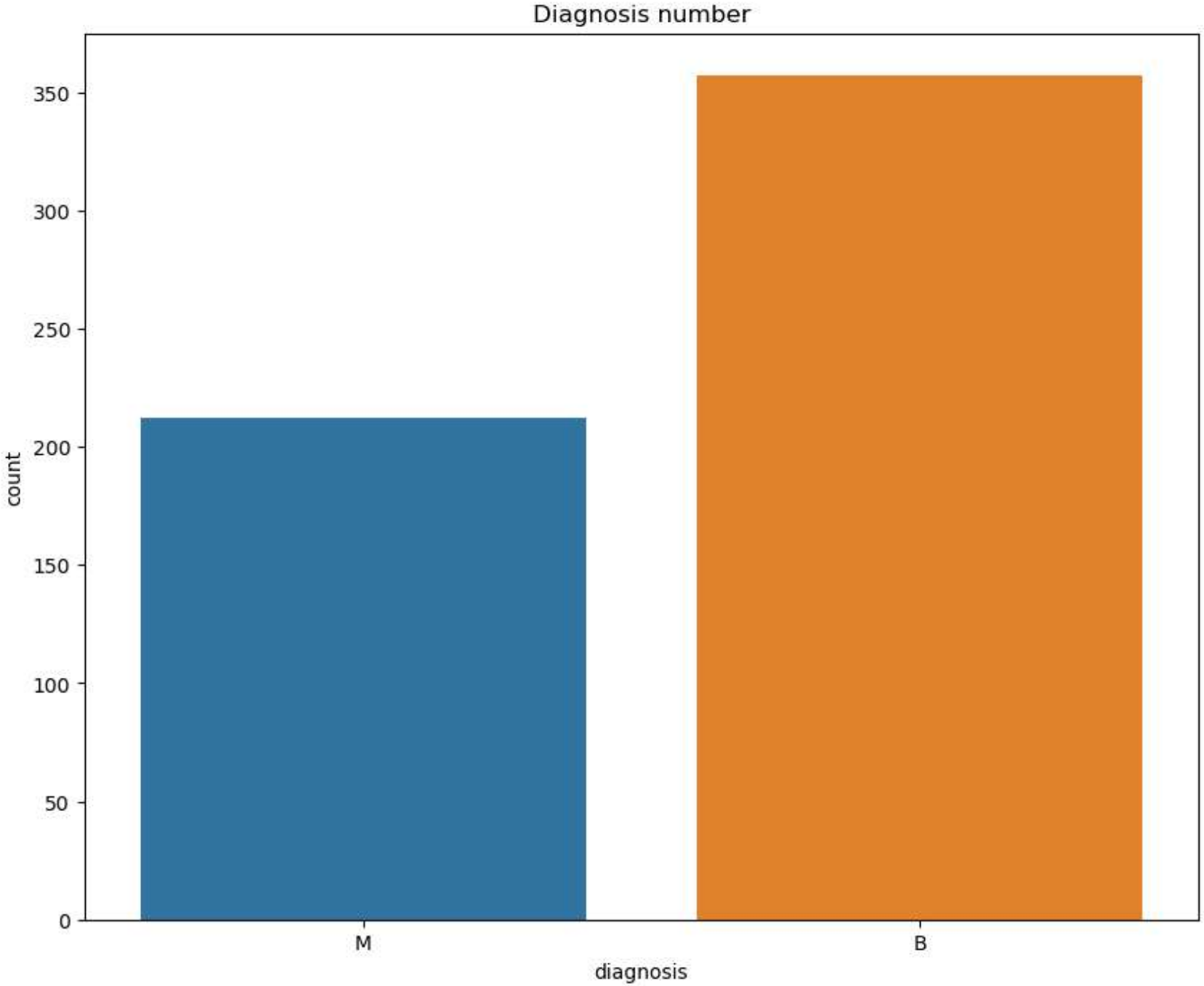
Out[78]:
```
array(['M', 'B'], dtype=object)
```

In [79]:
```
df['diagnosis'].value_counts()
```

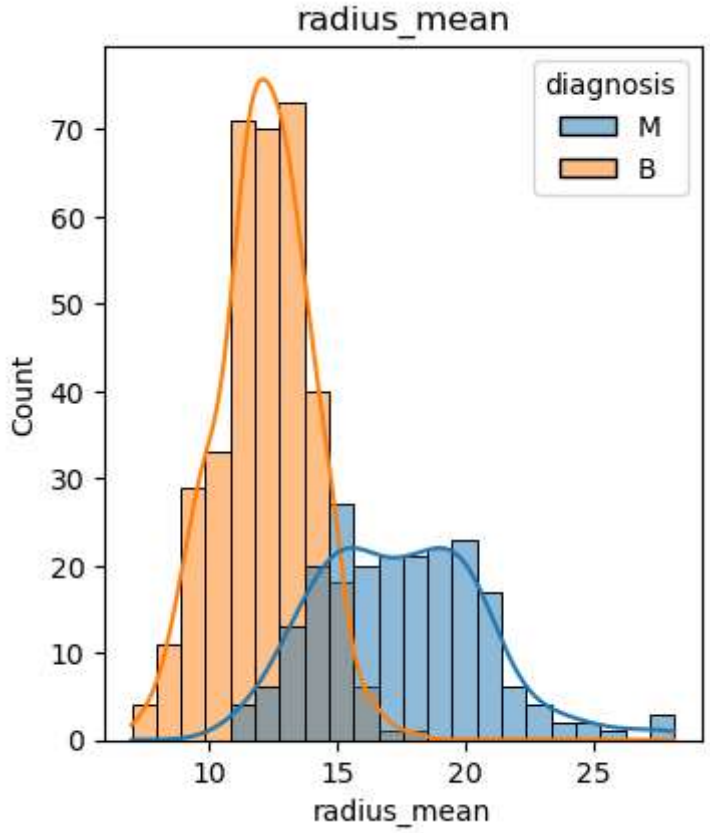
Out[79]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

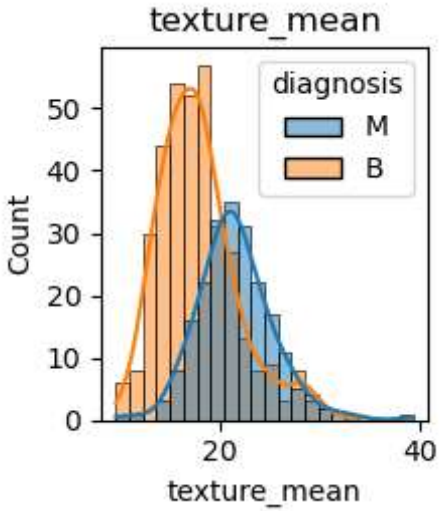
In [80]:

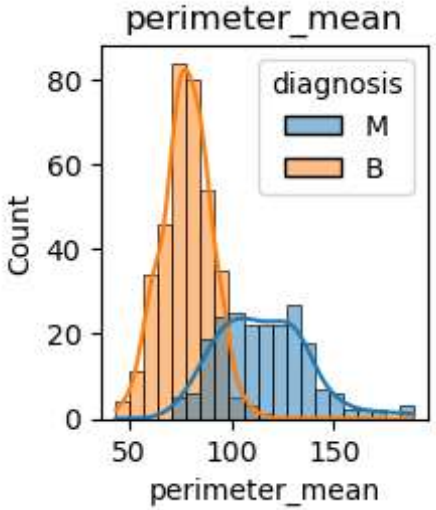
```
plt.figure(figsize=(10,8))

sns.countplot(data=df,x='diagnosis')
plt.title("Diagnosis number")
plt.show()
```

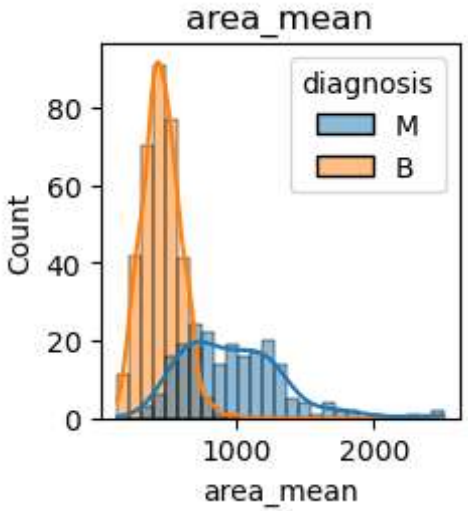

Diagnosis number
350
300
250
200
150
100
50
0
count
M
B
diagnosis

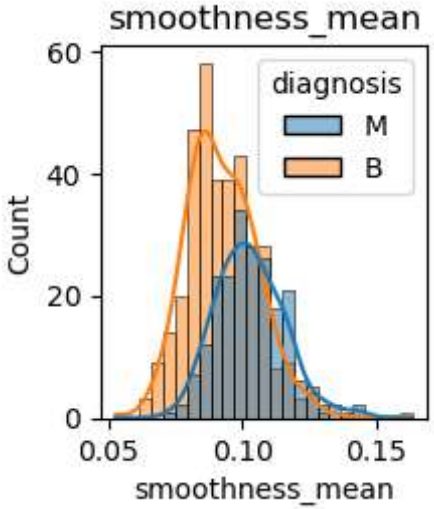
In [81]:
```
plt.figure(figsize=(10,8))

temp_df = ['radius_mean','texture_mean','perimeter_mean','area_mean','smoothness_mean']
n_temp_df = len(temp_df)


for i in range(n_temp_df):
    plt.subplot(2,3,i+1)
    sns.histplot(df,x=temp_df[i],hue='diagnosis',kde=True)
    plt.title(temp_df[i])
    plt.tight_layout()
    plt.show()
```




texture_mean
diagnosis
M
B
Count
0
10
20
30
40
50
20
40
texture_mean

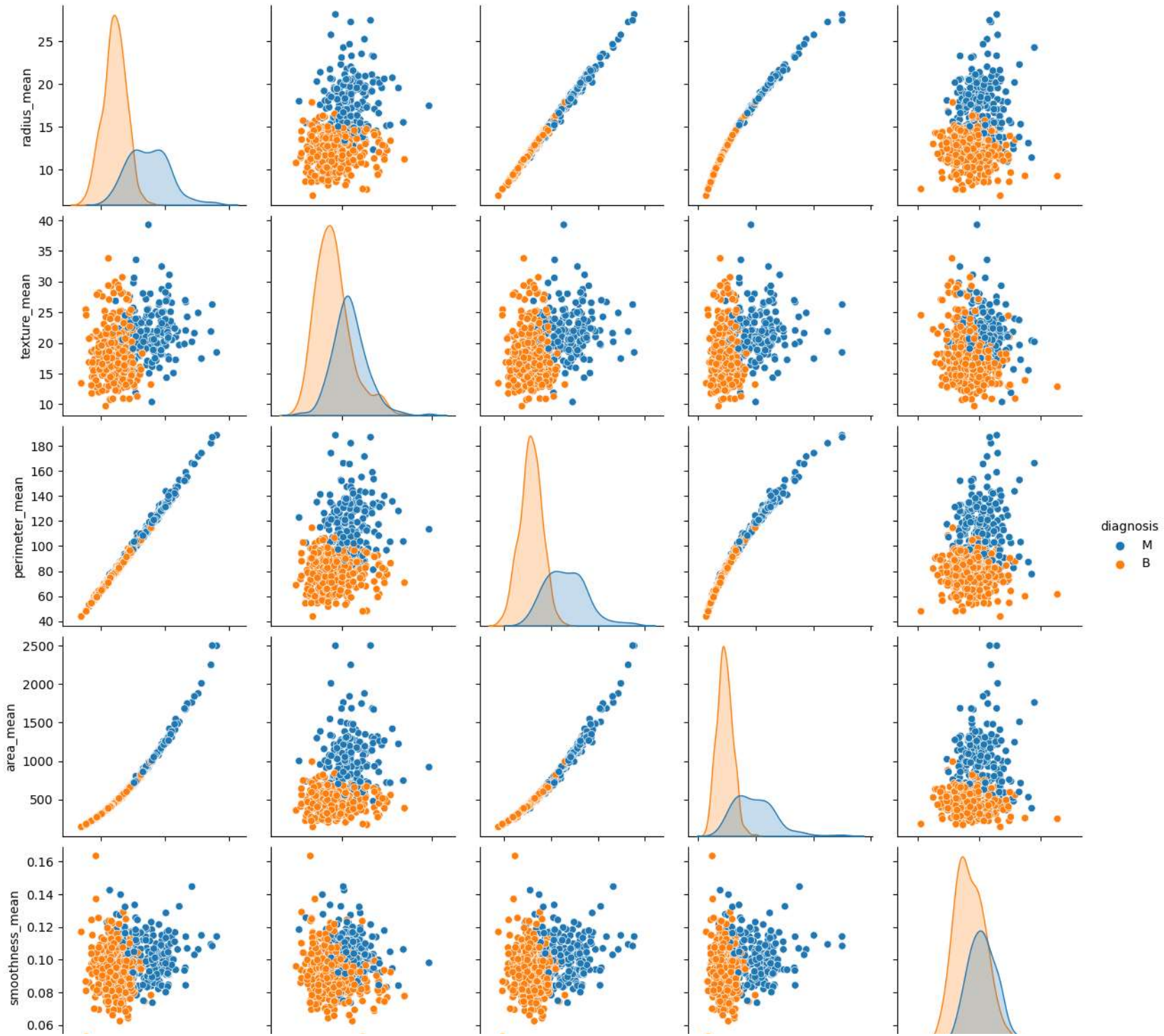

perimeter_mean
diagnosis
M
B
Count
0
20
40
60
80
50
100
150
perimeter_mean

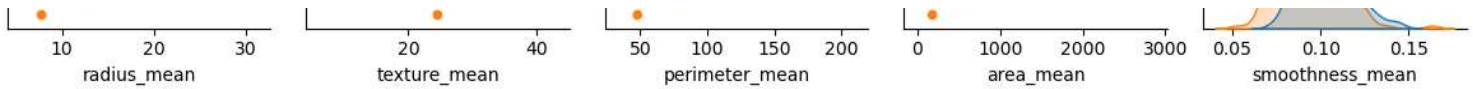

area_mean
diagnosis
M
B
Count
0
20
40
60
80
1000
2000
area_mean


smoothness_mean
diagnosis
M
B
Count
0
20
40
60
0.05
0.10
0.15
smoothness_mean

In [82]:
```
sns.pairplot(df.iloc[:,1:7],hue='diagnosis')
```

Out[82]:
```
<seaborn.axisgrid.PairGrid at 0x1c96607d4d0>
```

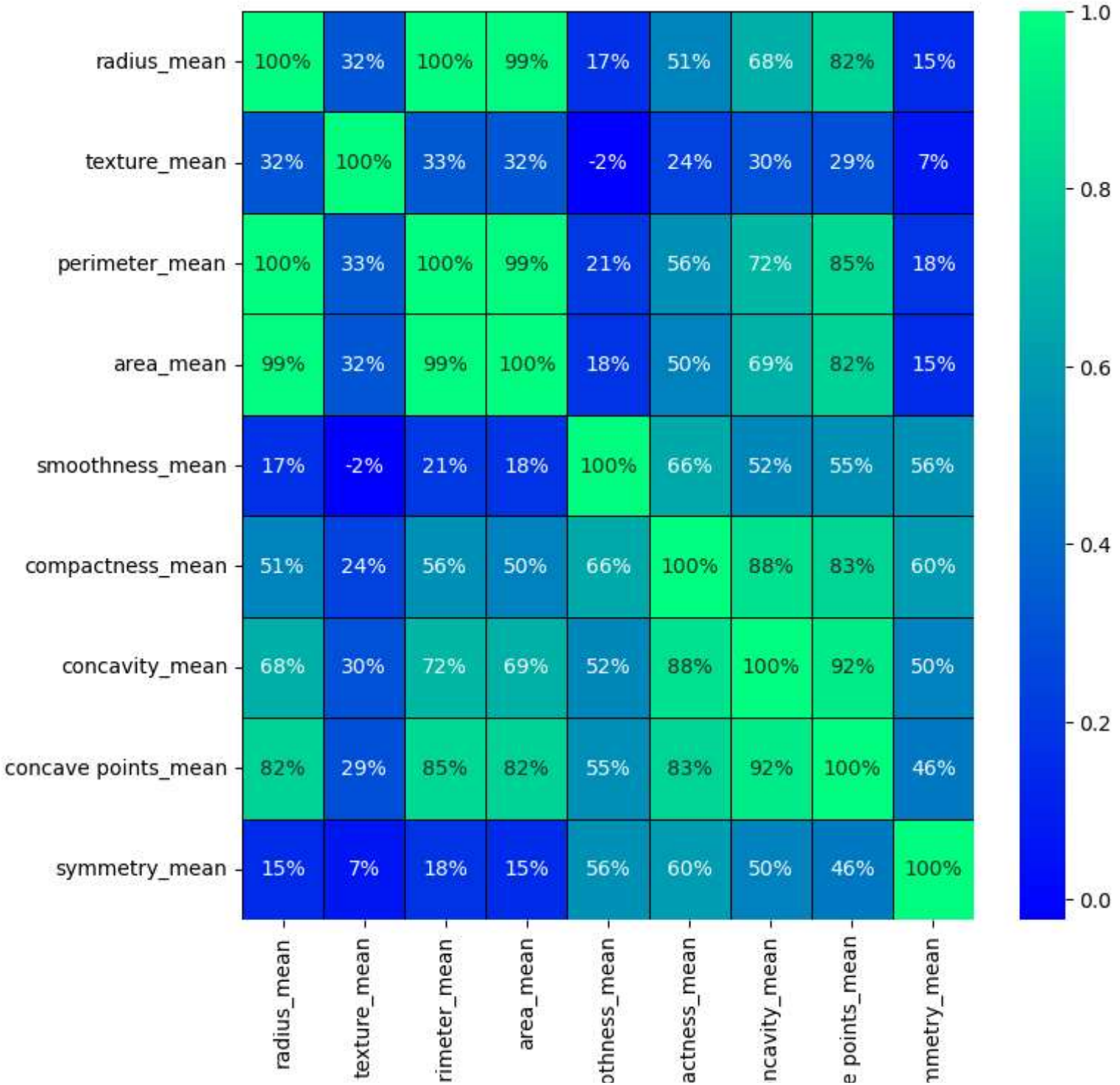

radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
diagnosis
M
B


10
20
30
radius_mean
20
40
texture_mean
50
100
150
200
perimeter_mean
0
1000
2000
3000
area_mean
0.05
0.10
0.15
smoothness_mean

In [83]:
```
plt.figure(figsize=(8,8))

sns.heatmap(df.iloc[:,1:11].corr(),annot=True,linecolor='black',linewidths=0.5,fmt='.0%',cmap='winter')
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_14932\913893192.py:3: FutureWarning: The default value of numeric_
only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid column
s or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.iloc[:,1:11].corr(),annot=True,linecolor='black',linewidths=0.5,fmt='.0%',cmap='winter')
```

Out[83]:
```
<Axes: >
```


radius_mean 100% 32% 100% 99% 17% 51% 68% 82% 15%
texture_mean 32% 100% 33% 32% -2% 24% 30% 29% 7%
perimeter_mean 100% 33% 100% 99% 21% 56% 72% 85% 18%
area_mean 99% 32% 99% 100% 18% 50% 69% 82% 15%
smoothness_mean 17% -2% 21% 18% 100% 66% 52% 55% 56%
compactness_mean 51% 24% 56% 50% 66% 100% 88% 83% 60%
concavity_mean 68% 30% 72% 69% 52% 88% 100% 92% 50%
concave points_mean 82% 29% 85% 82% 55% 83% 92% 100% 46%
symmetry_mean 15% 7% 18% 15% 56% 60% 50% 46% 100%
radius_mean
texture_mean
rimeter_mean
area_mean
othness_mean
actness_mean
ncavity_mean
e points_mean
mmetry_mean
1.0
0.8
0.6
0.4
0.2
0.0

pe smoo compa co concav syr

In [84]:
```
from sklearn.model_selection import train_test_split

x = df.iloc[:,2:31]
y= df.iloc[:,:1]
```

In [85]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=0)
```

In [86]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[86]: ((398, 29), (171, 29), (398, 1), (171, 1))

In [87]:
```
from sklearn.preprocessing import StandardScaler
```

In [88]:
```
sc= StandardScaler()
sc.fit(x_train)

new_x_train = sc.transform(x_train)
new_x_test = sc.transform(x_test)
```

In [89]:
```
new_x_train = pd.DataFrame(new_x_train,columns=x_train.columns)
```

In [90]:

```
def models(X_train,y_train):
    from sklearn.linear_model import LogisticRegression

    log = LogisticRegression()
    log.fit(x_train,y_train)



    from sklearn.ensemble import RandomForestClassifier

    rm = RandomForestClassifier(n_estimators=10,criterion='entropy')
    rm.fit(x_train,y_train)


    print('[0] Logistic Regression Training Accuracy : ', log.score(x_train,y_train))

    print("[2] Random Forest Classifier Accuracy : ",rm.score(x_train,y_train))


    return log,rm
```

In [91]:
```
model = models(x_train,y_train)
```

```
ple using ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_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 in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/pre
processing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.or
g/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
C:\Users\DELL\AppData\Local\Temp\ipykernel_14932\2968574983.py:12: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ra
vel().
  rm.fit(x_train,y_train)

[0] Logistic Regression Training Accuracy :  0.49748743718592964
[2] Random Forest Classifier Accuracy :  1.0
```

In [116]:
```
from sklearn.svm import SVC

svc = SVC (kernel='rbf',gamma=0.5)

svc1 = svc.fit(x_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example
using ravel().
  y = column_or_1d(y, warn=True)
```

In [117]:
```
print(f'Support Vector Classifier Accuracy : {svc.score(x_train,y_train)}')
```

```
Support Vector Classifier Accuracy : 1.0
```

In [ ]: