In [1]:
```
#Importing all the required library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#Dataset:'/kaggle/input/breast-cancer-wisconsin-data/data.csv'
df=pd.read_csv("D:\data.csv")
```

In [3]:
```
df
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness
0	842302	M	17.99	10.38	122.80	1001.0	0
1	842517	M	20.57	17.77	132.90	1326.0	0
2	84300903	M	19.69	21.25	130.00	1203.0	0
3	84348301	M	11.42	20.38	77.58	386.1	0
4	84358402	M	20.29	14.34	135.10	1297.0	0
...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0
565	926682	M	20.13	28.25	131.20	1261.0	0
566	926954	M	16.60	28.08	108.30	858.1	0
567	927241	M	20.60	29.33	140.10	1265.0	0
568	92751	B	7.76	24.54	47.92	181.0	0

569 rows × 33 columns

In [4]:
```
df.shape
```

Out[4]:
```
(569, 33)
```

In [5]:
```
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 [6]:
```
#descriptive statistics of data
df.describe()
```

Out[6]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.0000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.0963
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.0140
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.0526
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.0863
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.0958
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.1053
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.1634

8 rows × 32 columns

In [8]:
```
df.drop({'Unnamed: 32'},axis=1,inplace=True)
```

In [9]:
```
df.columns
```

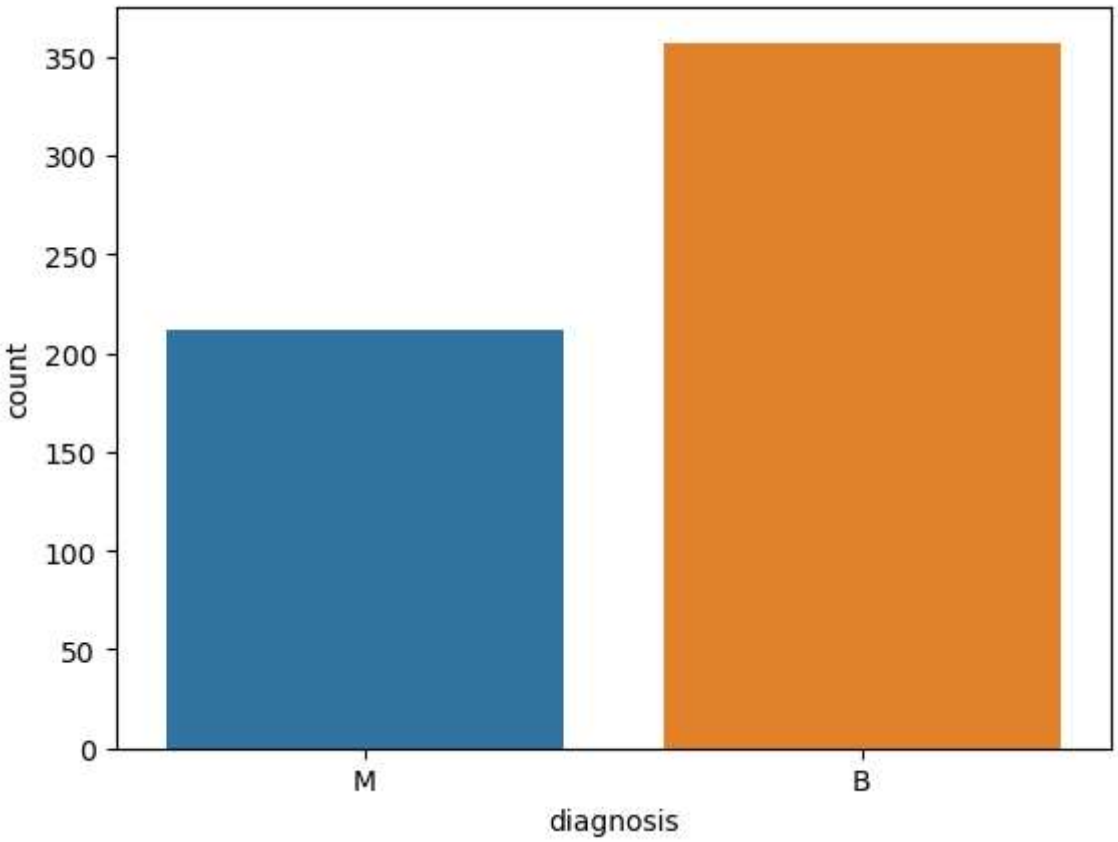
Out[9]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       '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'],
      dtype='object')
```

In [10]:
```
df['diagnosis'].value_counts()
```

Out[10]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

In [11]:
```
sns.countplot(df['diagnosis'])
```

Out[11]:
```
<AxesSubplot:xlabel='diagnosis', ylabel='count'>
```



In [12]:
```
#mapping categorical values to numerical values
df['diagnosis']=df['diagnosis'].map({"M": 0,"B":1},)
```

In [13]:
```
df.dtypes
```

Out[13]:
```
id                         int64
diagnosis                  int64
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
dtype: object
```

In [14]:
```
df.drop({"id"},axis=1,inplace=True)
```

In [15]:
```
df.columns
```

Out[15]:
```
Index(['diagnosis', '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'],
      dtype='object')
```

In [16]:
```
#checking for null values
df.isnull().sum()
```

Out[16]:
```
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 [17]:
```
df.corr()
```

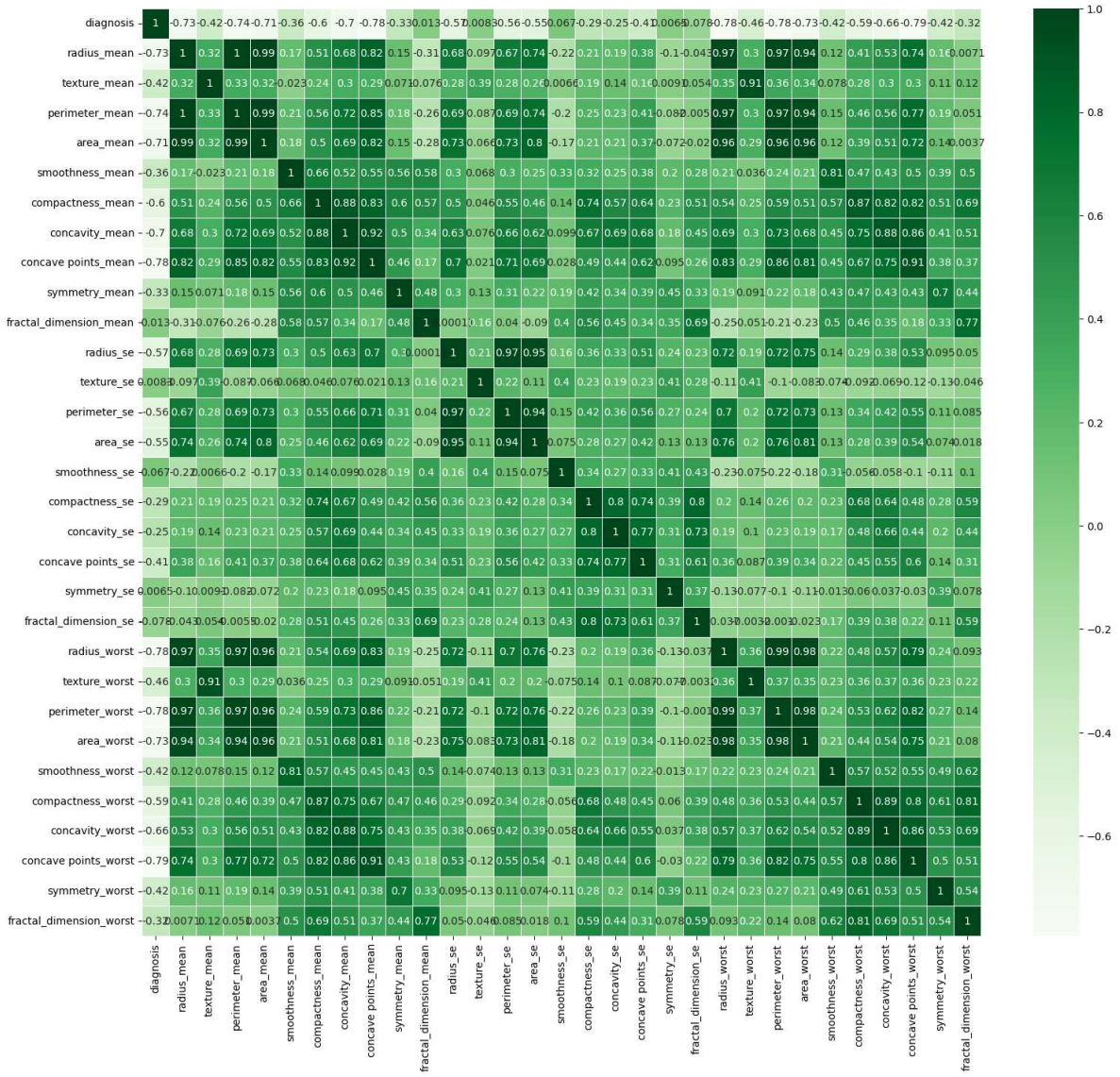
Out[17]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	sm
diagnosis	1.000000	-0.730029	-0.415185	-0.742636	-0.708984	
radius_mean	-0.730029	1.000000	0.323782	0.997855	0.987357	
texture_mean	-0.415185	0.323782	1.000000	0.329533	0.321086	
perimeter_mean	-0.742636	0.997855	0.329533	1.000000	0.986507	
area_mean	-0.708984	0.987357	0.321086	0.986507	1.000000	
smoothness_mean	-0.358560	0.170581	-0.023389	0.207278	0.177028	
compactness_mean	-0.596534	0.506124	0.236702	0.556936	0.498502	
concavity_mean	-0.696360	0.676764	0.302418	0.716136	0.685983	
concave points_mean	-0.776614	0.822529	0.293464	0.850977	0.823269	
symmetry_mean	-0.330499	0.147741	0.071401	0.183027	0.151293	
fractal_dimension_mean	0.012838	-0.311631	-0.076437	-0.261477	-0.283110	
radius_se	-0.567134	0.679090	0.275869	0.691765	0.732562	
texture_se	0.008303	-0.097317	0.386358	-0.086761	-0.066280	
perimeter_se	-0.556141	0.674172	0.281673	0.693135	0.726628	
area_se	-0.548236	0.735864	0.259845	0.744983	0.800086	
smoothness_se	0.067016	-0.222600	0.006614	-0.202694	-0.166777	
compactness_se	-0.292999	0.206000	0.191975	0.250744	0.212583	
concavity_se	-0.253730	0.194204	0.143293	0.228082	0.207660	
concave points_se	-0.408042	0.376169	0.163851	0.407217	0.372320	
symmetry_se	0.006522	-0.104321	0.009127	-0.081629	-0.072497	
fractal_dimension_se	-0.077972	-0.042641	0.054458	-0.005523	-0.019887	
radius_worst	-0.776454	0.969539	0.352573	0.969476	0.962746	
texture_worst	-0.456903	0.297008	0.912045	0.303038	0.287489	
perimeter_worst	-0.782914	0.965137	0.358040	0.970387	0.959120	
area_worst	-0.733825	0.941082	0.343546	0.941550	0.959213	
smoothness_worst	-0.421465	0.119616	0.077503	0.150549	0.123523	
compactness_worst	-0.590998	0.413463	0.277830	0.455774	0.390410	
concavity_worst	-0.659610	0.526911	0.301025	0.563879	0.512606	
concave points_worst	-0.793566	0.744214	0.295316	0.771241	0.722017	
symmetry_worst	-0.416294	0.163953	0.105008	0.189115	0.143570	
fractal_dimension_worst	-0.323872	0.007066	0.119205	0.051019	0.003738	

31 rows × 31 columns

In [18]:
```
plt.figure(figsize=(18,16))
sns.heatmap(df.corr(),annot=True,linewidth=.5,cmap='Greens')
```

Out[18]: <AxesSubplot:>



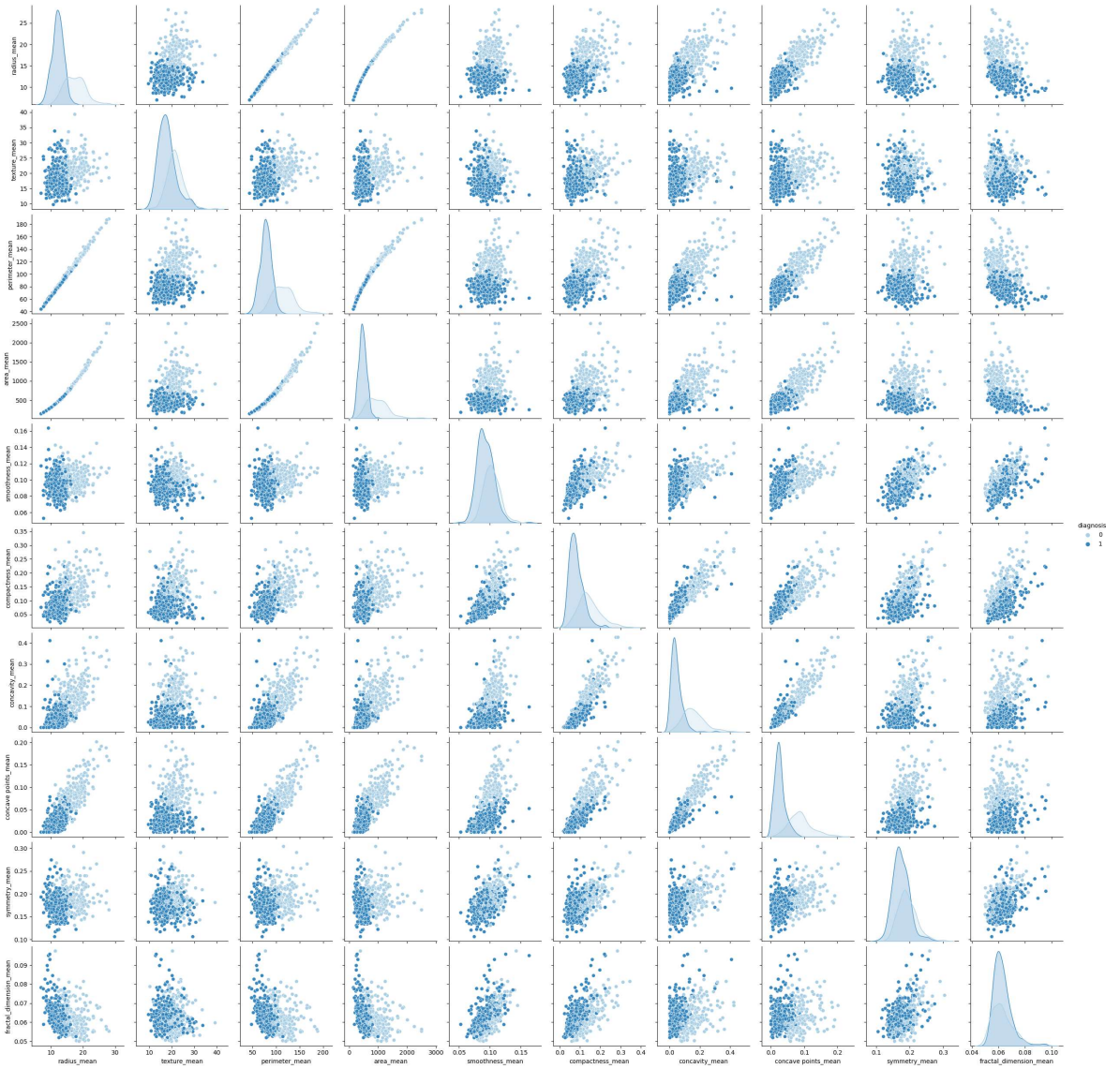
we can observe from the heatmap that there are many negative correlations in the dataset

In [19]:
```
#getting Mean columns with diagnosis
m_col = ['diagnosis','radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean']

#getting Se columns with diagnosis
s_col= ['diagnosis','radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smo
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se']

#getting Worst column with diagnosis
w_col=['diagnosis','radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst']
```

For Mean Columns

In [20]:
```
sns.pairplot(df[m_col],hue='diagnosis',palette='Blues')
```

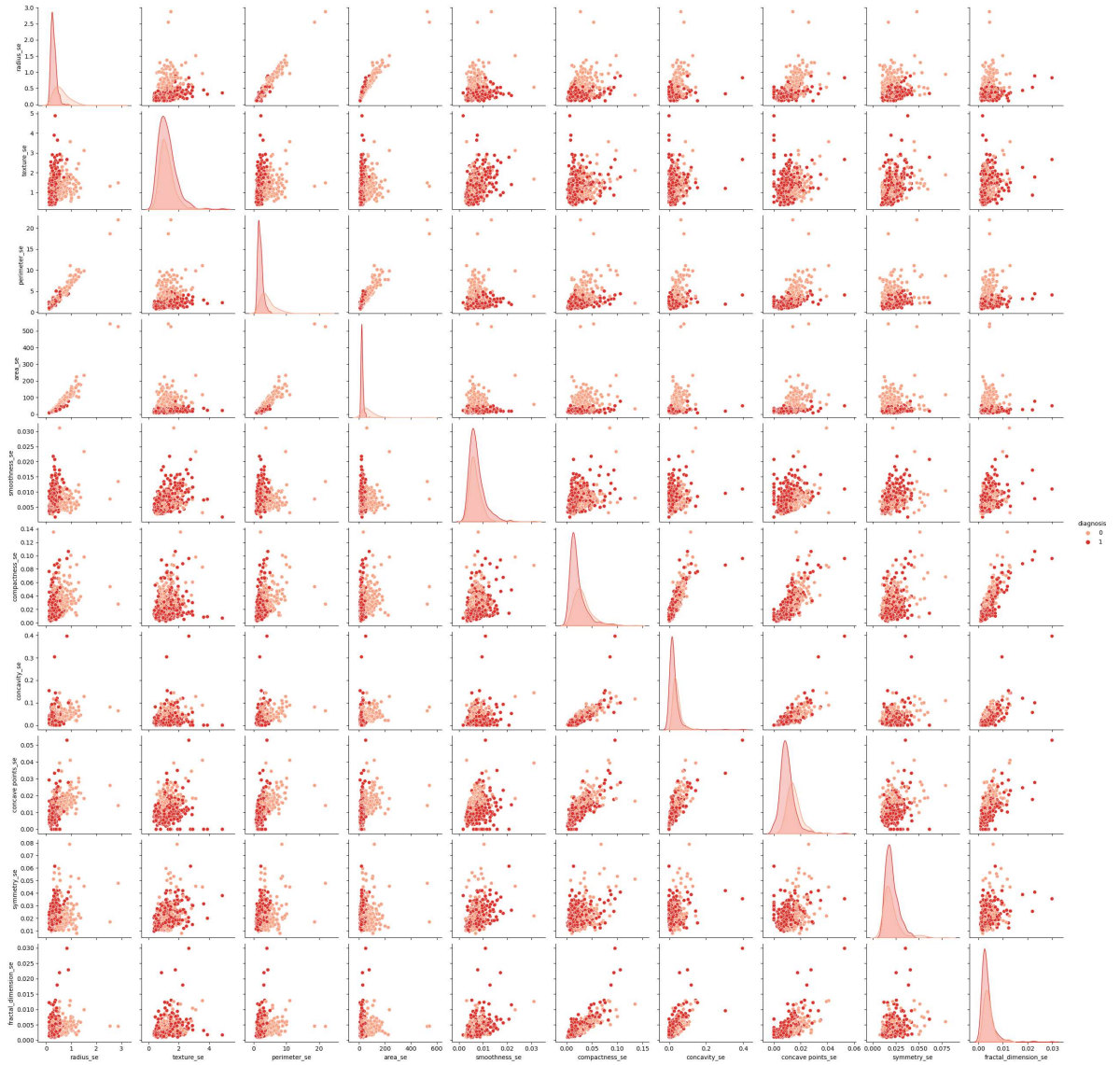
Out[20]:
```
<seaborn.axisgrid.PairGrid at 0x16ad35e2400>
```



For SE Columns

In [21]:
```
sns.pairplot(df[s_col],hue='diagnosis',palette='Reds')
```

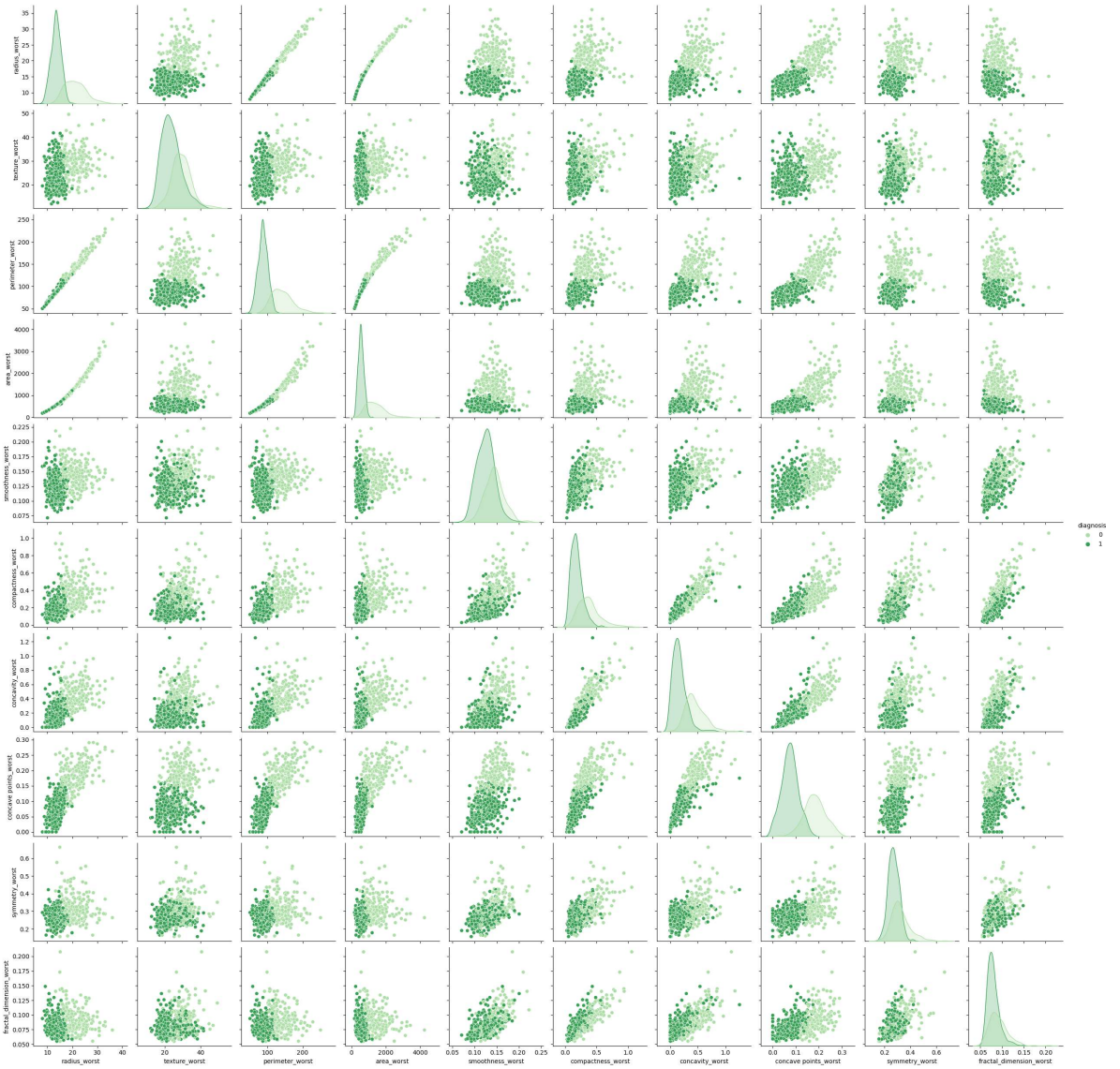
Out[21]:
```
<seaborn.axisgrid.PairGrid at 0x16adb87da90>
```



For Worst columns

In [22]:
```
sns.pairplot(df[w_col],hue='diagnosis',palette='Greens')
```

Out[22]:
```
<seaborn.axisgrid.PairGrid at 0x16ae617d9a0>
```



splitting data into train test

In [25]:
```
X=df.drop(['diagnosis'],axis=1)
y=df['diagnosis']
```

In [26]:
```
X.columns
```

Out[26]:
```
Index(['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'],
      dtype='object')
```

In [28]:
```
y
```

Out[28]:
```
0      0
1      0
2      0
3      0
4      0
      ..
564    0
565    0
566    0
567    0
568    1
Name: diagnosis, Length: 569, dtype: int64
```

In [29]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=
```

In [31]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [32]:
```
from sklearn.linear_model import LogisticRegression
lgr=LogisticRegression()
```

In [33]:
```
lgr.fit(X_train,y_train)
```

Out[33]:
```
LogisticRegression()
```

In [34]:
```
lgr.score(X_test,y_test)
```

Out[34]:
```
0.956140350877193
```

Accuracy of LogisticRegression Model is :0.956140350877193

In [35]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
```

In [36]:
```
knn.fit(X_train,y_train)
```

Out[36]:
```
KNeighborsClassifier()
```

In [37]:
```
knn.score(X_test,y_test)
```

Out[37]:
```
0.956140350877193
```

Accuracy of K Neighbors classifier Model is: 0.956140350877193

In [ ]:
```

```