In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv(r'/Users/divyanshsahai/Desktop/breast cancer.csv')
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
```

In [5]:

```
df.head()
```

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactne
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 32 columns

In [6]:

```
df.shape
```

Out[6]:

```
(569, 32)
```

In [7]:

```
df.describe()
```

Out[7]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactnes
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0

8 rows × 31 columns

In [8]:

```
df.diagnosis.unique()
```

Out[8]:

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

In [9]:

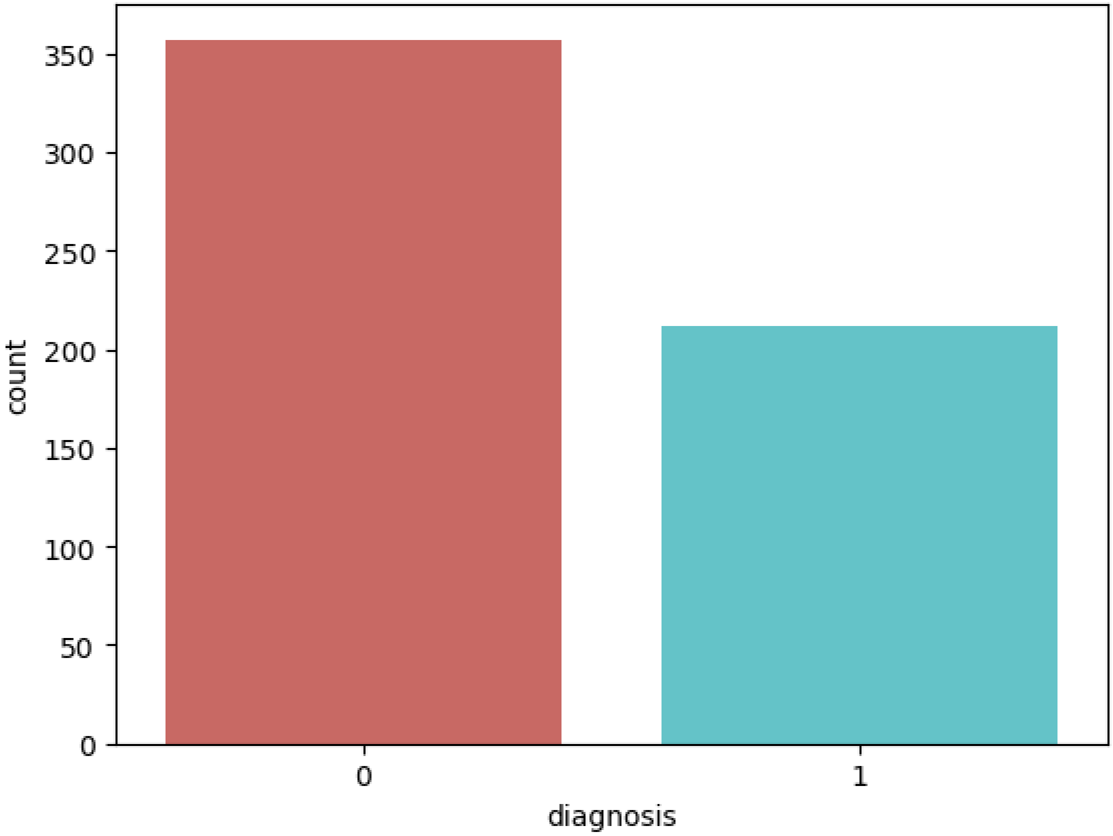
```
df['diagnosis'].value_counts()
```

Out[9]:

```
diagnosis
B    357
M    212
Name: count, dtype: int64
```

In [22]:

```
sns.countplot(x='diagnosis', data=df, palette='hls')
plt.savefig('Logistic.png')
plt.show()
```



In [12]:

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

In [13]:

```
df.head()
```

Out[13]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280

5 rows × 31 columns

In [14]:

```
df['diagnosis'] = df['diagnosis'].map({'M':1,'B':0})
df.head()
```

Out[14]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280

5 rows × 31 columns

```
In [15]: df.isnull().sum()
```

```
Out[15]: 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 [16]: df.corr()
```

Out[16]:

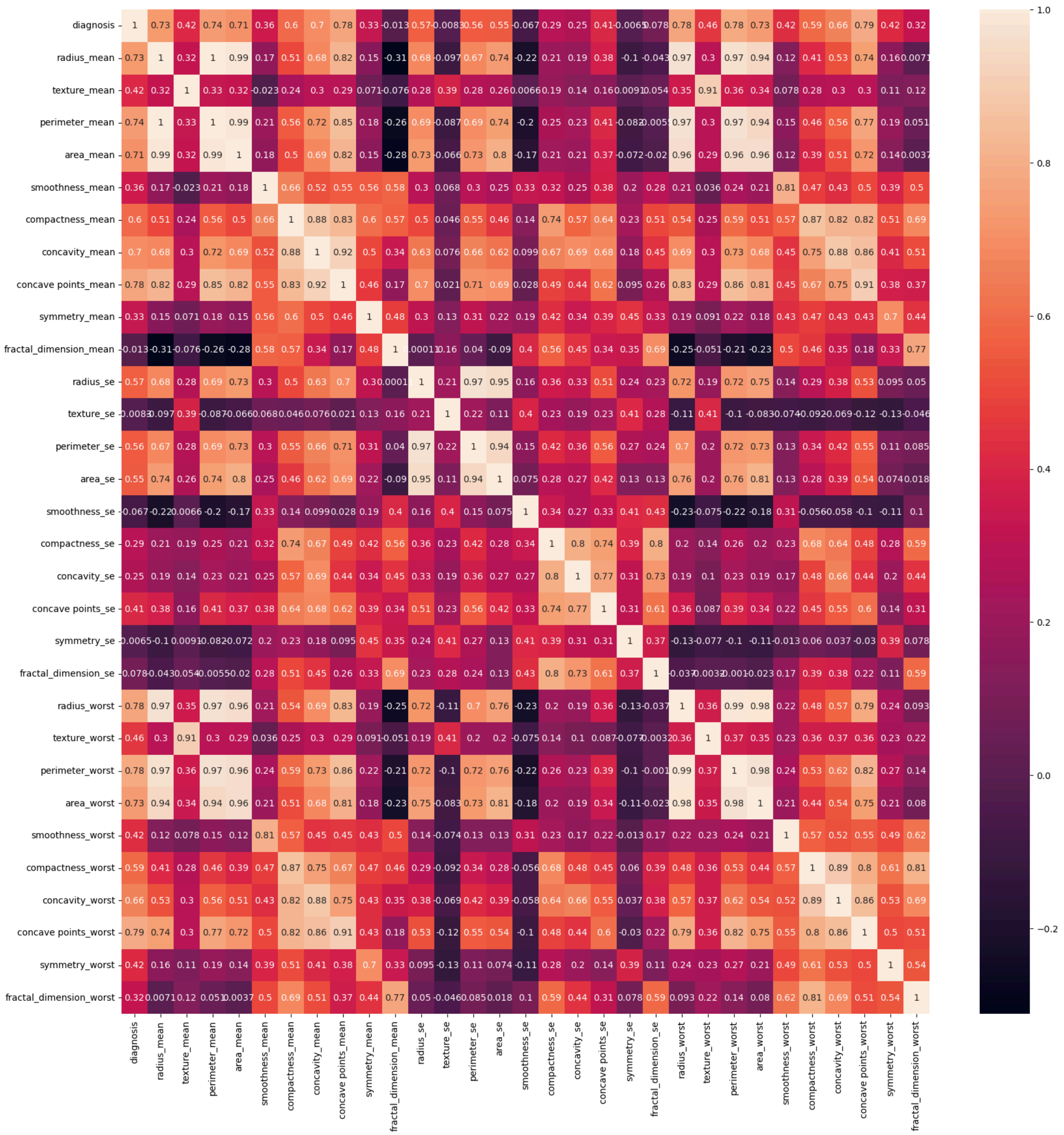
	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	0.358560
radius_mean	0.730029	1.000000	0.323782	0.997855	0.987357	0.170581
texture_mean	0.415185	0.323782	1.000000	0.329533	0.321086	-0.023389
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.986507	0.207278
area_mean	0.708984	0.987357	0.321086	0.986507	1.000000	0.177028
smoothness_mean	0.358560	0.170581	-0.023389	0.207278	0.177028	1.000000
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	0.659123
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	0.521984
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	0.553695
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	0.557775
fractal_dimension_mean	-0.012838	-0.311631	-0.076437	-0.261477	-0.283110	0.584792
radius_se	0.567134	0.679090	0.275869	0.691765	0.732562	0.301467
texture_se	-0.008303	-0.097317	0.386358	-0.086761	-0.066280	0.068406
perimeter_se	0.556141	0.674172	0.281673	0.693135	0.726628	0.296092
area_se	0.548236	0.735864	0.259845	0.744983	0.800086	0.246552
smoothness_se	-0.067016	-0.222600	0.006614	-0.202694	-0.166777	0.332375
compactness_se	0.292999	0.206000	0.191975	0.250744	0.212583	0.318943
concavity_se	0.253730	0.194204	0.143293	0.228082	0.207660	0.248396
concave points_se	0.408042	0.376169	0.163851	0.407217	0.372320	0.380676
symmetry_se	-0.006522	-0.104321	0.009127	-0.081629	-0.072497	0.200774
fractal_dimension_se	0.077972	-0.042641	0.054458	-0.005523	-0.019887	0.283607
radius_worst	0.776454	0.969539	0.352573	0.969476	0.962746	0.213120
texture_worst	0.456903	0.297008	0.912045	0.303038	0.287489	0.036072
perimeter_worst	0.782914	0.965137	0.358040	0.970387	0.959120	0.238853
area_worst	0.733825	0.941082	0.343546	0.941550	0.959213	0.206718
smoothness_worst	0.421465	0.119616	0.077503	0.150549	0.123523	0.805324
compactness_worst	0.590998	0.413463	0.277830	0.455774	0.390410	0.472468
concavity_worst	0.659610	0.526911	0.301025	0.563879	0.512606	0.434926
concave points_worst	0.793566	0.744214	0.295316	0.771241	0.722017	0.503053
symmetry_worst	0.416294	0.163953	0.105008	0.189115	0.143570	0.394309
fractal_dimension_worst	0.323872	0.007066	0.119205	0.051019	0.003738	0.499316

31 rows × 31 columns

In [17]:

```
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(), annot=True)
plt.savefig('CorrelationMatrix.png')
```

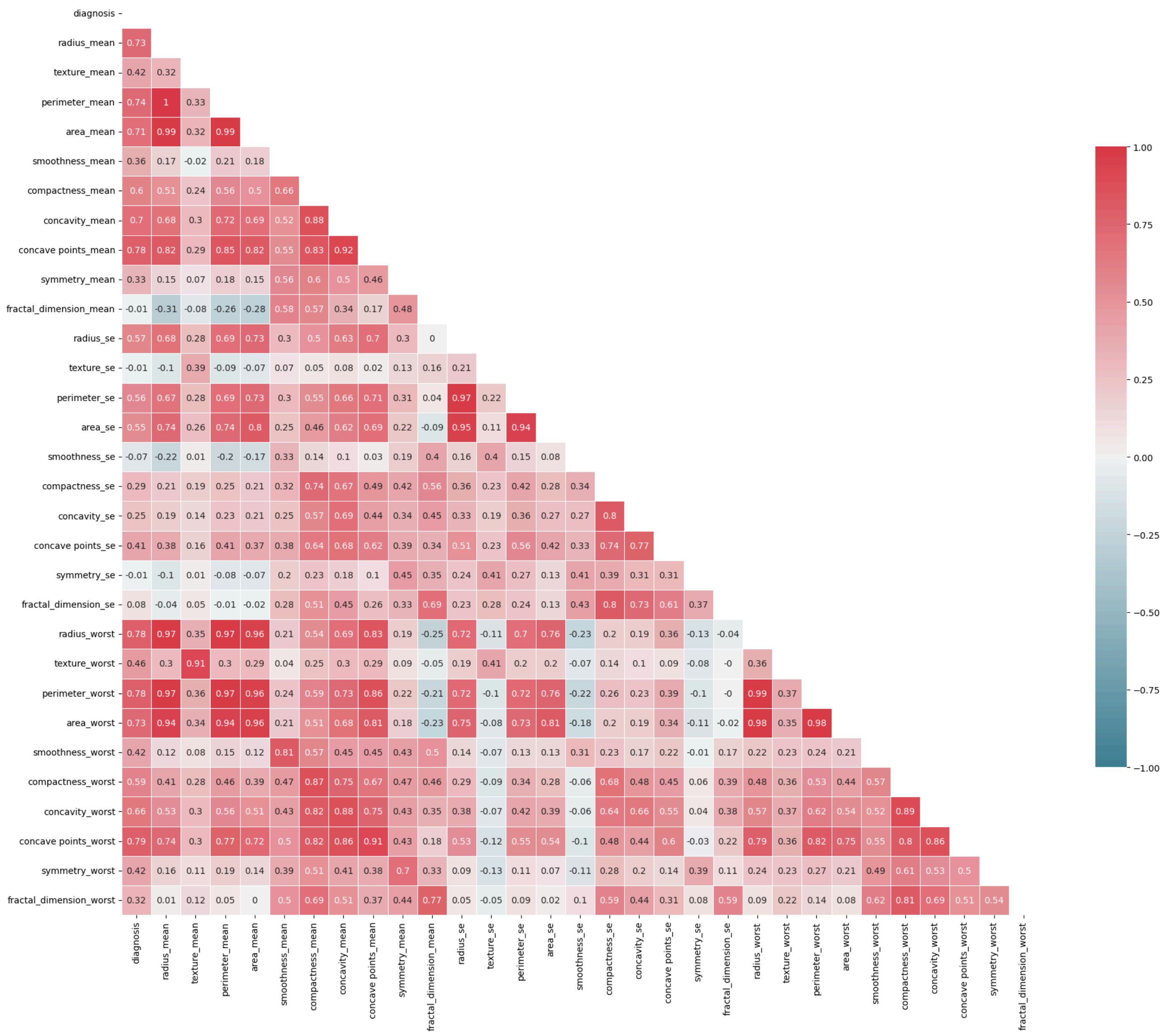
Loading [MathJax]/extensions/Safe.js



In [21]:

```
# Generate and visualize the correlation matrix
corr = df.corr().round(2)

# Mask for the upper triangle
mask = np.zeros_like(corr, dtype=bool)
mask[np.triu_indices_from(mask)] = True

# Set figure size
f, ax = plt.subplots(figsize=(20, 20))

# Define custom colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap
sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True)
```

```
plt.tight_layout()
plt.savefig('Heatmap.png')
```



	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
diagnosis																															
radius_mean	0.73																														
texture_mean	0.42	0.32																													
perimeter_mean	0.74	1	0.33																												
area_mean	0.71	0.99	0.32	0.99																											
smoothness_mean	0.36	0.17	-0.02	0.21	0.18																										
compactness_mean	0.6	0.51	0.24	0.56	0.5	0.66																									
concavity_mean	0.7	0.68	0.3	0.72	0.69	0.52	0.88																								
concave points_mean	0.78	0.82	0.29	0.85	0.82	0.55	0.83	0.92																							
symmetry_mean	0.33	0.15	0.07	0.18	0.15	0.56	0.6	0.5	0.46																						
fractal_dimension_mean	-0.01	-0.31	-0.08	-0.26	-0.28	0.58	0.57	0.34	0.17	0.48																					
radius_se	0.57	0.68	0.28	0.69	0.73	0.3	0.5	0.63	0.7	0.3	0																				
texture_se	-0.01	-0.1	0.39	-0.09	-0.07	0.07	0.05	0.08	0.02	0.13	0.16	0.21																			
perimeter_se	0.56	0.67	0.28	0.69	0.73	0.3	0.55	0.66	0.71	0.31	0.04	0.97	0.22																		
area_se	0.55	0.74	0.26	0.74	0.8	0.25	0.46	0.62	0.69	0.22	-0.09	0.95	0.11	0.94																	
smoothness_se	-0.07	-0.22	0.01	-0.2	-0.17	0.33	0.14	0.1	0.03	0.19	0.4	0.16	0.4	0.15	0.08																
compactness_se	0.29	0.21	0.19	0.25	0.21	0.32	0.74	0.67	0.49	0.42	0.56	0.36	0.23	0.42	0.28	0.34															
concavity_se	0.25	0.19	0.14	0.23	0.21	0.25	0.57	0.69	0.44	0.34	0.45	0.33	0.19	0.36	0.27	0.27	0.8														
concave points_se	0.41	0.38	0.16	0.41	0.37	0.38	0.64	0.68	0.62	0.39	0.34	0.51	0.23	0.56	0.42	0.33	0.74	0.77													
symmetry_se	-0.01	-0.1	0.01	-0.08	-0.07	0.2	0.23	0.18	0.1	0.45	0.35	0.24	0.41	0.27	0.13	0.41	0.39	0.31	0.31												
fractal_dimension_se	0.08	-0.04	0.05	-0.01	-0.02	0.28	0.51	0.45	0.26	0.33	0.69	0.23	0.28	0.24	0.13	0.43	0.8	0.73	0.61	0.37											
radius_worst	0.78	0.97	0.35	0.97	0.96	0.21	0.54	0.69	0.83	0.19	-0.25	0.72	-0.11	0.7	0.76	-0.23	0.2	0.19	0.36	-0.13	-0.04										
texture_worst	0.46	0.3	0.91	0.3	0.29	0.04	0.25	0.3	0.29	0.09	-0.05	0.19	0.41	0.2	0.2	-0.07	0.14	0.1	0.09	-0.08	-0	0.36									
perimeter_worst	0.78	0.97	0.36	0.97	0.96	0.24	0.59	0.73	0.86	0.22	-0.21	0.72	-0.1	0.72	0.76	-0.22	0.26	0.23	0.39	-0.1	-0	0.99	0.37								
area_worst	0.73	0.94	0.34	0.94	0.96	0.21	0.51	0.68	0.81	0.18	-0.23	0.75	-0.08	0.73	0.81	-0.18	0.2	0.19	0.34	-0.11	-0.02	0.98	0.35	0.98							
smoothness_worst	0.42	0.12	0.08	0.15	0.12	0.81	0.57	0.45	0.45	0.43	0.5	0.14	-0.07	0.13	0.13	0.31	0.23	0.17	0.22	-0.01	0.17	0.22	0.23	0.24	0.21						
compactness_worst	0.59	0.41	0.28	0.46	0.39	0.47	0.87	0.75	0.67	0.47	0.46	0.29	-0.09	0.34	0.28	-0.06	0.68	0.48	0.45	0.06	0.39	0.48	0.36	0.53	0.44	0.57					
concavity_worst	0.66	0.53	0.3	0.56	0.51	0.43	0.82	0.88	0.75	0.43	0.35	0.38	-0.07	0.42	0.39	-0.06	0.64	0.66	0.55	0.04	0.38	0.57	0.37	0.62	0.54	0.52	0.89				
concave points_worst	0.79	0.74	0.3	0.77	0.72	0.5	0.82	0.86	0.91	0.43	0.18	0.53	-0.12	0.55	0.54	-0.1	0.48	0.44	0.6	-0.03	0.22	0.79	0.36	0.82	0.75	0.55	0.8	0.86			
symmetry_worst	0.42	0.16	0.11	0.19	0.14	0.39	0.51	0.41	0.38	0.7	0.33	0.09	-0.13	0.11	0.07	-0.11	0.28	0.2	0.14	0.39	0.11	0.24	0.23	0.27	0.21	0.49	0.61	0.53	0.5		
fractal_dimension_worst	0.32	0.01	0.12	0.05	0	0.5	0.69	0.51	0.37	0.44	0.77	0.05	-0.05	0.09	0.02	0.1	0.59	0.44	0.31	0.08	0.59	0.09	0.22	0.14	0.08	0.62	0.81	0.69	0.51	0.54	

In [23]:

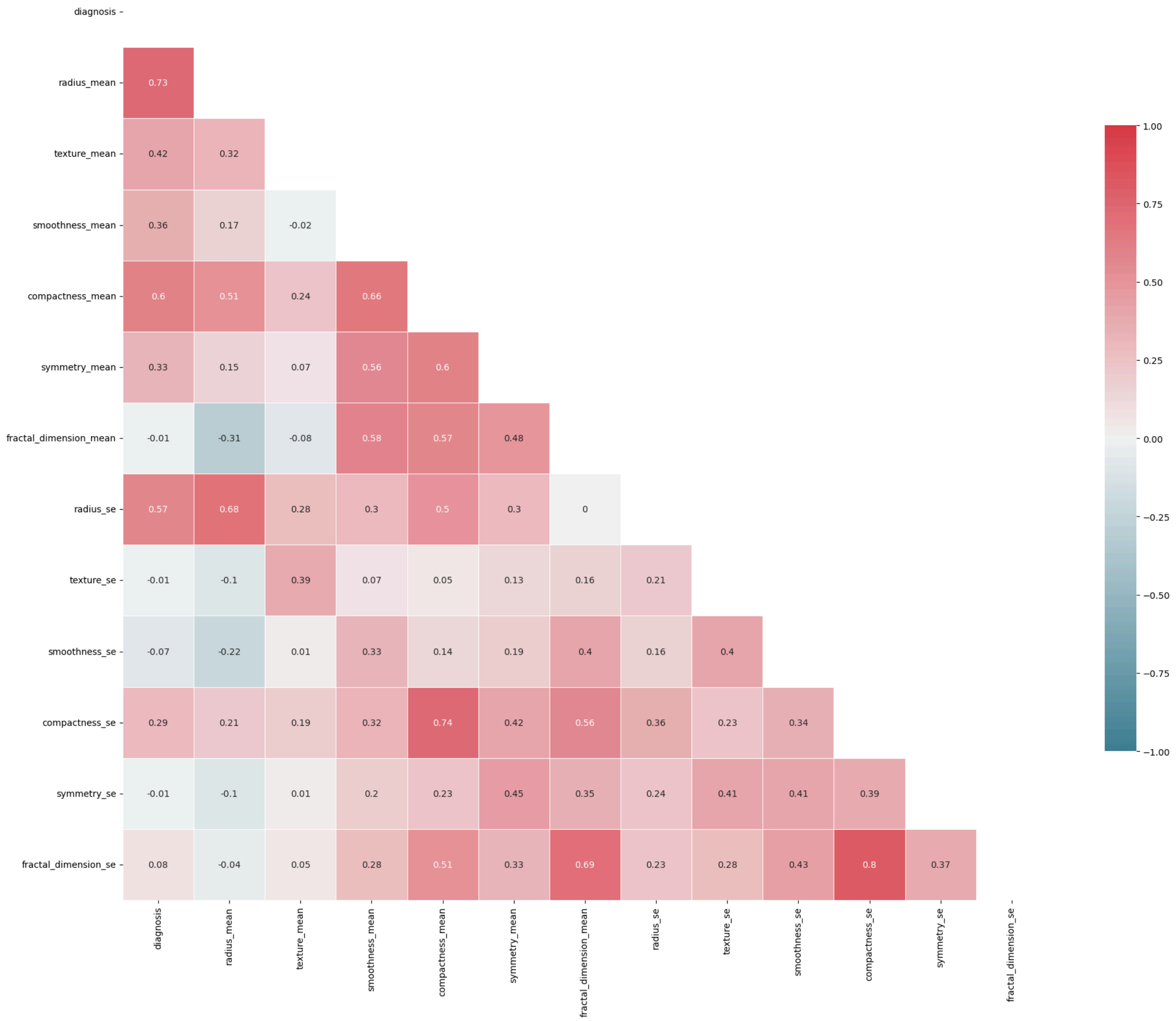
```
# first, drop all "worst" columns
cols = ['radius_worst',
        'texture_worst',
        'perimeter_worst',
        'area_worst',
        'smoothness_worst',
        'compactness_worst',
        'concavity_worst',
        'concave points_worst',
        'symmetry_worst',
        'fractal_dimension_worst']
df = df.drop(cols, axis=1)

# then, drop all columns related to the "perimeter" and "area" attributes
cols = ['perimeter_mean',
        'perimeter_se',
        'area_mean',
        'area_se']
df = df.drop(cols, axis=1)

# lastly, drop all columns related to the "concavity" and "concave points" attributes
cols = ['concavity_mean',
        'concavity_se',
        'concave points_mean',
        'concave points_se']
df = df.drop(cols, axis=1)

# verify remaining columns
df.columns
```

Out[23]:

```
Index(['diagnosis', 'radius_mean', 'texture_mean', 'smoothness_mean',
       'compactness_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'smoothness_se', 'compactness_se',
       'symmetry_se', 'fractal_dimension_se'],
      dtype='object')
```

In [25]:

```
# Draw the heatmap again, with the new correlation matrix
corr = df.corr().round(2)
mask = np.zeros_like(corr, dtype=bool)
mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize=(20, 20))
sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True)
plt.tight_layout()
plt.savefig('RevisedHeatmap.png')
```



Loading [MathJax]/extensions/Safe.js

In [28]:

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

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=0.3,random_state=40)
```

In [30]:

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()

X_train=ss.fit_transform(X_train)
X_test=ss.fit_transform(X_test)
```

In [31]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

In [32]:

```
from sklearn.linear_model import LogisticRegression
```

In [33]:

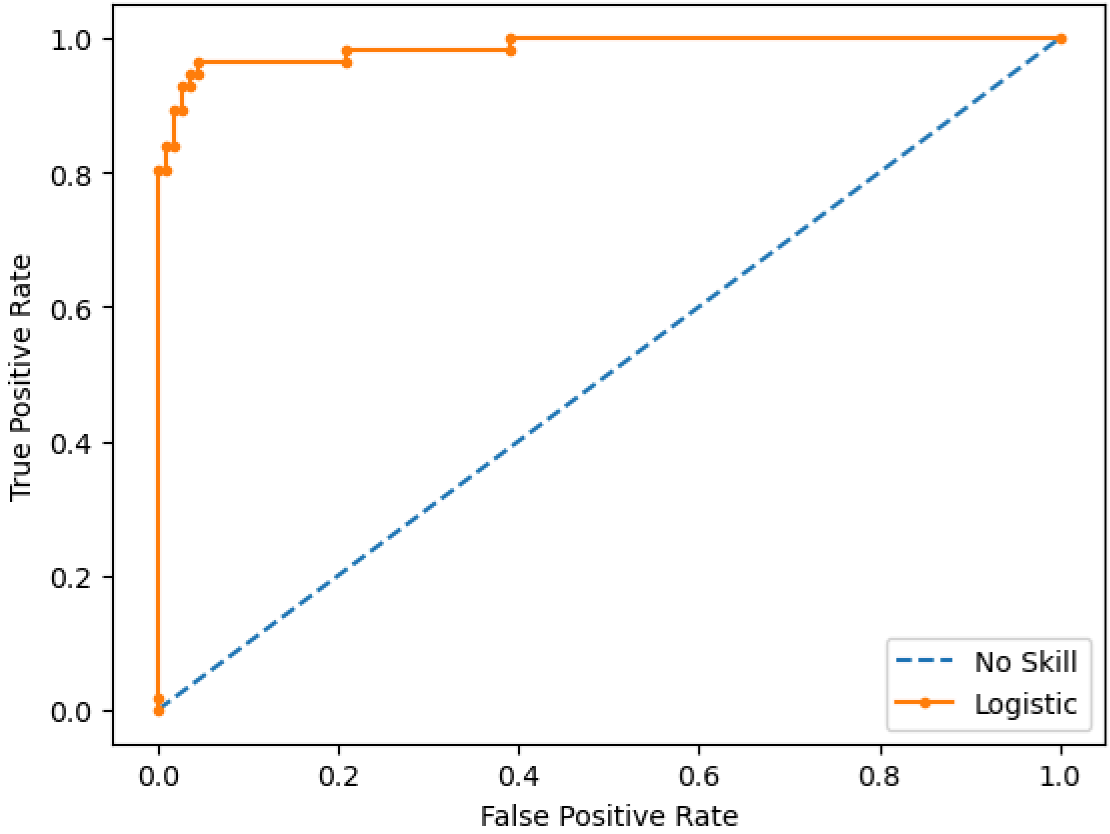
```
lr=LogisticRegression()

model1=lr.fit(X_train,y_train)
prediction1=model1.predict(X_test)
```

In [34]:

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot

# generate a no skill prediction (majority class)
ns_probs = [0 for _ in range(len(y_test))]
lr_probs = model1.predict_proba(X_test)

# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]

# calculate scores
ns_auc = roc_auc_score(y_test, ns_probs)
lr_auc = roc_auc_score(y_test, lr_probs)

# summarize scores
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))

# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(y_test, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_probs)

# plot the roc curve for the model
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')

# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')

# show the legend
pyplot.legend()

# show the plot
pyplot.show()
plt.savefig('ROCR.png')
```

Loading [MathJax]/extensions/Safe.js

```
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.986
```



```
<Figure size 640x480 with 0 Axes>
```

In [35]:

```
cm=confusion_matrix(y_test,prediction1)
cm
```

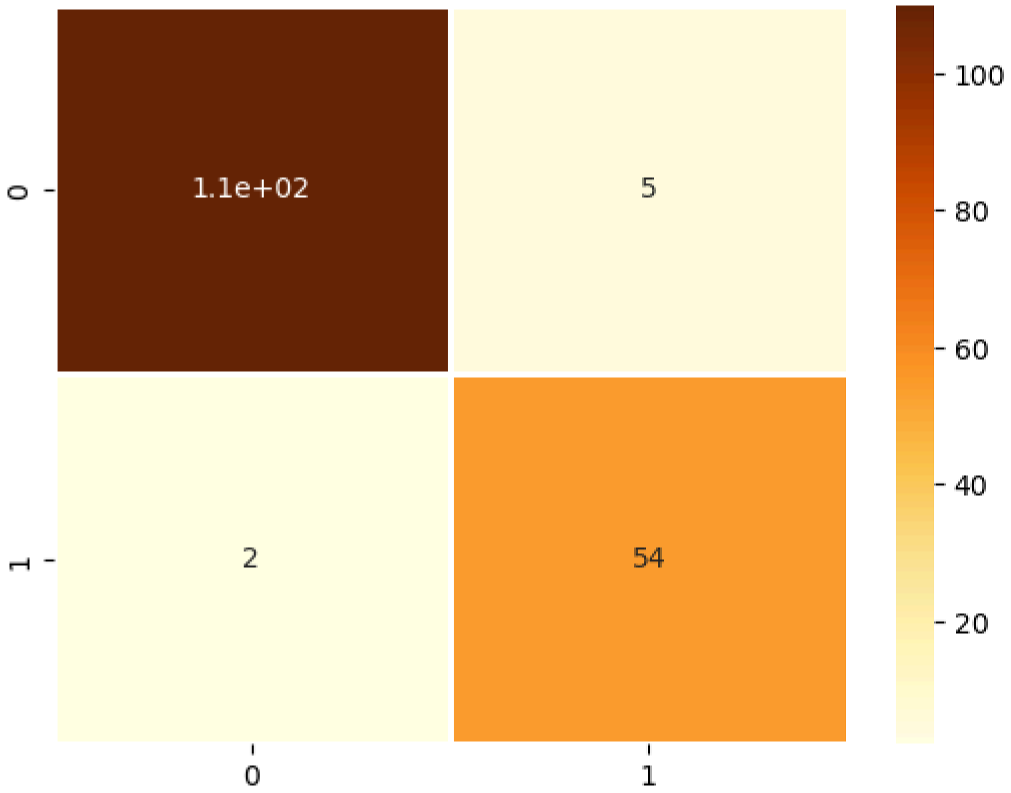
Out[35]:

```
array([[110,   5],
       [  2,  54]])
```

In [36]:

```
sns.heatmap(cm,annot=True,cmap ="YlOrBr",linecolor='White', linewidths=1)
plt.savefig('Losgistic.png')
```

Loading [MathJax]/extensions/Safe.js



In [37]:
```
accuracy_score(y_test,prediction1)
```

Out[37]:
```
0.9590643274853801
```

In [38]:
```
print(classification_report(y_test, prediction1))
```

```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       115
           1       0.92      0.96      0.94        56

    accuracy                           0.96       171
   macro avg       0.95      0.96      0.95       171
weighted avg       0.96      0.96      0.96       171
```

In [39]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from matplotlib import pyplot as plt
```

In [40]:
```
dtc=DecisionTreeClassifier()
model2=dtc.fit(X_train,y_train)
prediction2=model2.predict(X_test)
```

In [41]:
```
text_representation = tree.export_text(dtc)
print(text_representation)
```

```
|--- feature_0 <= 0.24
|   |--- feature_3 <= 0.40
|   |   |--- feature_1 <= 0.08
|   |   |   |--- feature_6 <= 0.72
|   |   |   |   |--- class: 0
|   |   |   |--- feature_6 >  0.72
|   |   |   |   |--- feature_0 <= -0.97
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_0 >  -0.97
|   |   |   |   |   |--- class: 1
|   |   |--- feature_1 >  0.08
|   |   |   |--- feature_0 <= -0.22
|   |   |   |   |--- feature_2 <= 1.41
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_2 >  1.41
|   |   |   |   |   |--- feature_11 <= -0.29
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- feature_11 >  -0.29
|   |   |   |   |   |   |--- class: 0
|   |   |   |--- feature_0 >  -0.22
|   |   |   |   |--- feature_2 <= -0.28
|   |   |   |   |   |--- feature_5 <= -0.87
|   |   |   |   |   |   |--- feature_4 <= -0.93
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- feature_4 >  -0.93
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- feature_5 >  -0.87
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_2 >  -0.28
|   |   |   |   |   |--- feature_5 <= 0.49
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- feature_5 >  0.49
|   |   |   |   |   |   |--- class: 0
|   |--- feature_3 >  0.40
|   |   |--- feature_0 <= -0.70
|   |   |   |--- class: 0
|   |   |--- feature_0 >  -0.70
|   |   |   |--- feature_1 <= -0.94
|   |   |   |   |--- class: 0
|   |   |   |--- feature_1 >  -0.94
|   |   |   |   |--- feature_4 <= 0.22
|   |   |   |   |   |--- feature_3 <= 0.47
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- feature_3 >  0.47
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_4 >  0.22
|   |   |   |   |   |--- feature_1 <= -0.68
|   |   |   |   |   |   |--- feature_2 <= 1.51
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- feature_2 >  1.51
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- feature_1 >  -0.68
|   |   |   |   |   |   |--- class: 1
|--- feature_0 >  0.24
|   |--- feature_1 <= -0.70
|   |   |--- feature_9 <= -0.16
|   |   |   |--- class: 0
|   |   |--- feature_9 >  -0.16
|   |   |   |--- feature_0 <= 0.29
|   |   |   |   |--- class: 0
|   |   |   |--- feature_0 >  0.29
|   |   |   |   |--- class: 1
|   |--- feature_1 >  -0.70
|   |   |--- feature_4 <= -1.14
|   |   |   |--- feature_11 <= -0.66
|   |   |   |   |--- class: 0
|   |   |   |--- feature_11 >  -0.66
|   |   |   |   |--- class: 1
|   |   |--- feature_4 >  -1.14
|   |   |   |--- class: 1
```

In [42]:

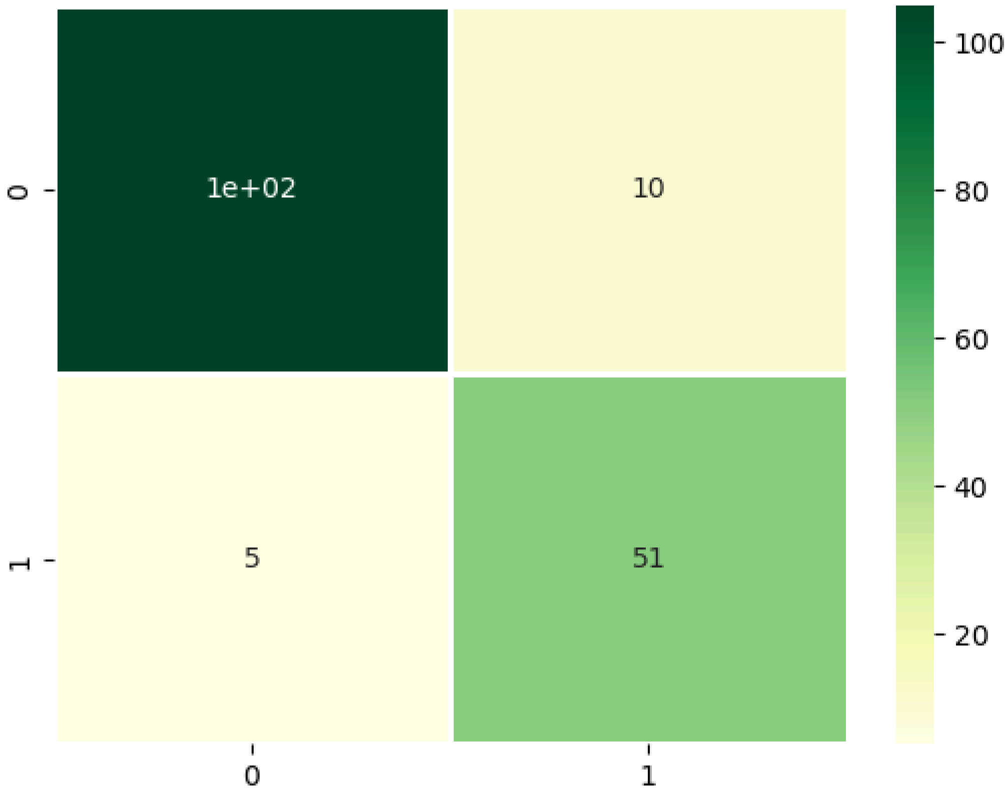
```
cm2= confusion_matrix(y_test,prediction2)
cm2
```

Out[42]:

```
array([[105,  10],
       [  5,  51]])
```

In [43]:

```
sns.heatmap(cm2,annot=True, cmap ="YlGn",linecolor='White', linewidths=1)
plt.savefig('DecisionTree.png')
```



In [44]:

```
accuracy_score(y_test,prediction2)
```

Out[44]:

```
0.9122807017543859
```

In [45]:

```
print(classification_report(y_test, prediction2))
```

```
              precision    recall  f1-score   support

           0       0.95      0.91      0.93       115
           1       0.84      0.91      0.87        56

    accuracy                           0.91       171
   macro avg       0.90      0.91      0.90       171
weighted avg       0.92      0.91      0.91       171
```

In [46]:

```
from sklearn.ensemble import RandomForestClassifier
```

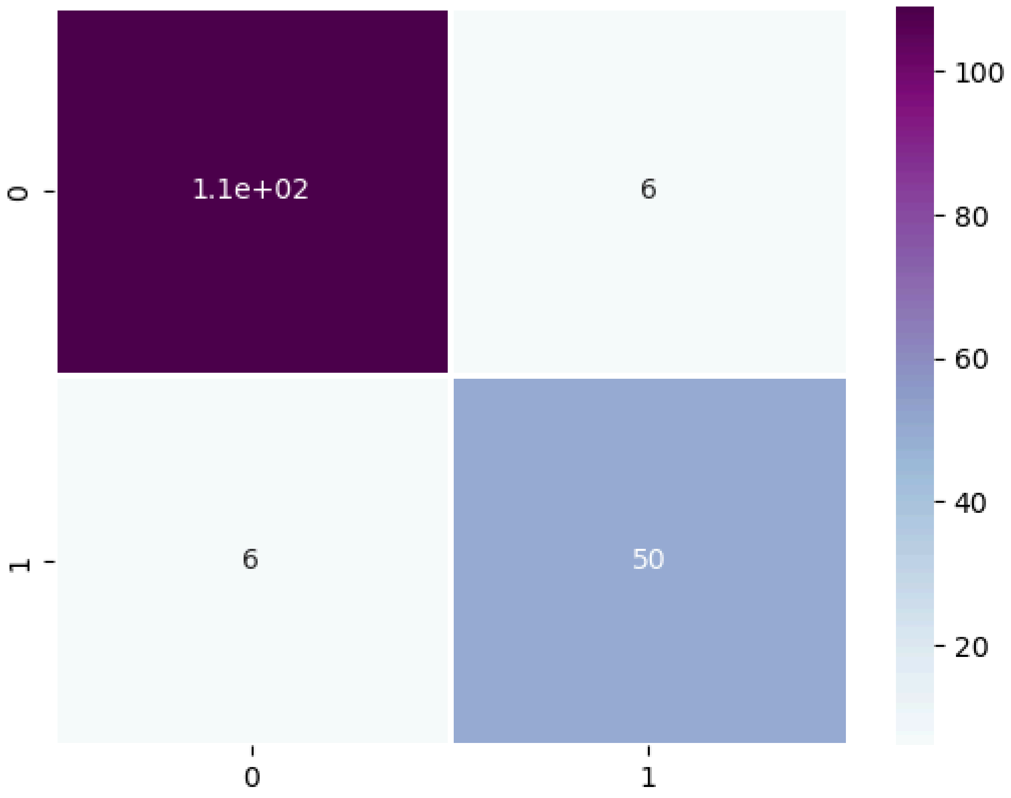
In [47]:

```
rfc=RandomForestClassifier()
model3 = rfc.fit(X_train, y_train)
= model3.predict(X_test)
```

```
cm3=confusion_matrix(y_test, prediction3)
cm3
```

Out[47]:
```
array([[109,   6],
       [  6,  50]])
```

In [48]:
```
sns.heatmap(cm3,annot=True, cmap ="BuPu",linecolor='White', linewidths=1)
plt.savefig('RandomForest.png')
```



In [49]:
```
accuracy_score(y_test, prediction3)
```

Out[49]:
```
0.9298245614035088
```

In [50]:
```
print(classification_report(y_test, prediction3))
```

```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       115
           1       0.89      0.89      0.89        56

    accuracy                           0.93       171
   macro avg       0.92      0.92      0.92       171
weighted avg       0.93      0.93      0.93       171
```

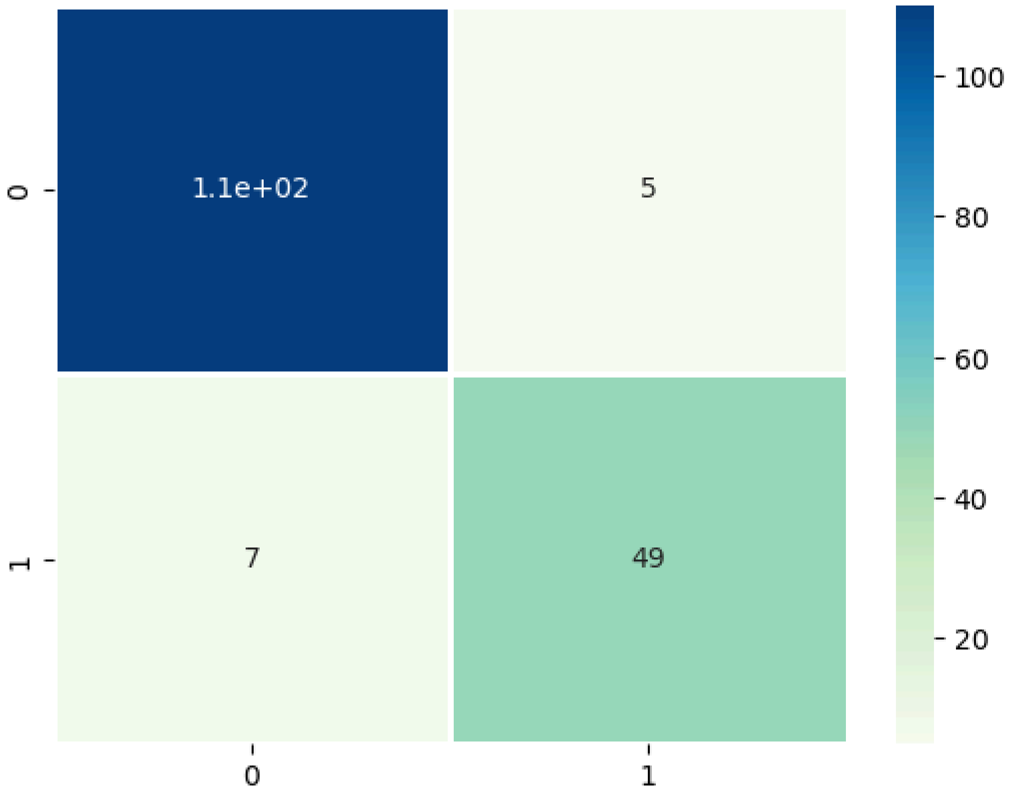
In [51]:
```
from sklearn import svm
```

In [52]:
```
model4 = svm.SVC(kernel='rbf',C=30,gamma='auto')
model4.fit(X_train,y_train)
prediction4 = model4.predict(X_test)
cm4=confusion_matrix(y_test, prediction4)
cm4
```

Out[52]:
```
array([[110,   5],
       [  7,  49]])
```

In [53]:
```
sns.heatmap(cm4,annot=True, cmap ="GnBu",linecolor='White', linewidths=1)
plt.savefig('SVM.png')
```

Loading [MathJax]/extensions/Safe.js



In [54]:

```
accuracy_score(y_test, prediction4)
```

Out[54]:

```
0.9298245614035088
```

In [55]:

```
print(classification_report(y_test, prediction4))
```

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       115
           1       0.91      0.88      0.89        56

    accuracy                           0.93       171
   macro avg       0.92      0.92      0.92       171
weighted avg       0.93      0.93      0.93       171
```

In [56]:

```
from sklearn.naive_bayes import GaussianNB
```

In [57]:

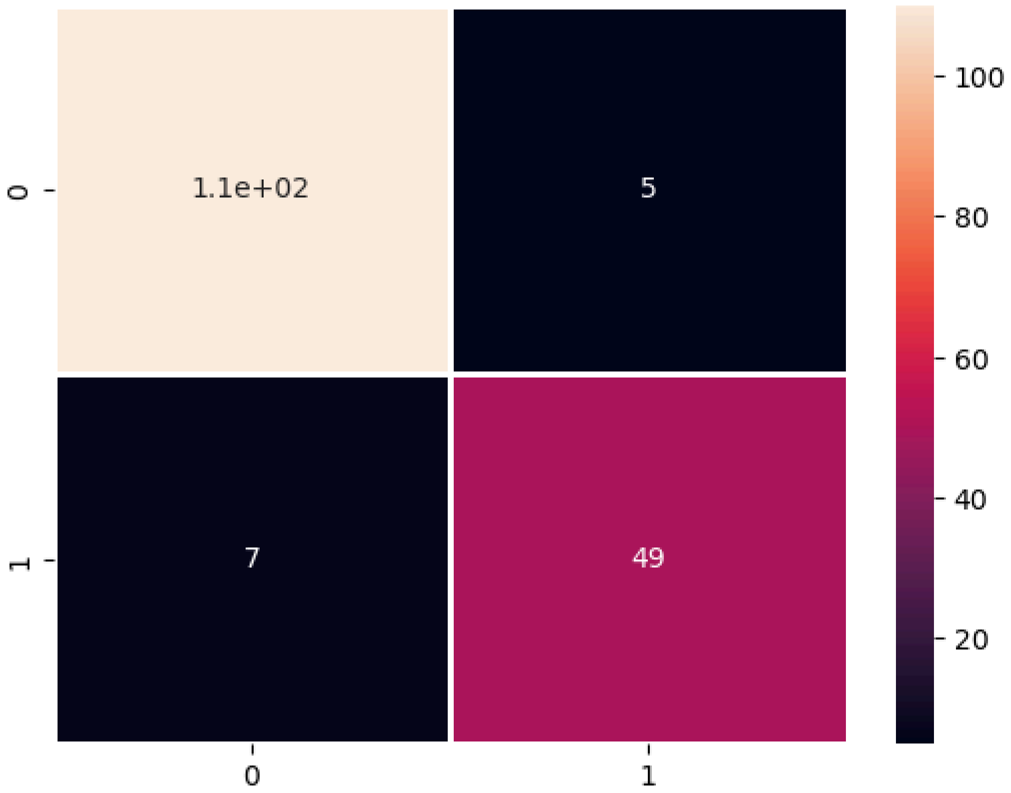
```
model5 = GaussianNB()
model5.fit(X_train,y_train)
prediction5 = model5.predict(X_test)
cm5=confusion_matrix(y_test, prediction5)
cm5
```

Out[57]:

```
array([[110,   5],
       [  7,  49]])
```

In [58]:

```
sns.heatmap(cm5,annot=True, linecolor='White', linewidths=1)
plt.savefig('NaiveBayes.png')
```



```
In [59]: accuracy_score(y_test, prediction5)

Out[59]: 0.9298245614035088
```

```
In [60]: print(classification_report(y_test, prediction5))

              precision    recall  f1-score   support

           0       0.94      0.96      0.95       115
           1       0.91      0.88      0.89        56

    accuracy                           0.93       171
   macro avg       0.92      0.92      0.92       171
weighted avg       0.93      0.93      0.93       171
```

```
In [61]: model_params = {
             'SVM': {
                 'model': svm.SVC(gamma='auto'),
                 'params' : {
                     'C': [1,10,20,30,40],
                     'kernel': ['rbf','linear']
                 }
             },
             'Random_Forest': {
                 'model': RandomForestClassifier(),
                 'params' : {
                     'n_estimators': [1,5,10,15,20]
                 }
             },
             'Logistic_Regression' : {
                 'model': LogisticRegression(solver='liblinear',multi_class='auto'),
                 'params': {
                     'C': [1,5,10,15,20]
                 }
             },
             'Naive_Bayes_Gaussian': {
                    l': GaussianNB(),
                 'params': {}
             },
             'Decision_Tree': {
                 'model': DecisionTreeClassifier(),
                 'params': {
                     'criterion': ['gini','entropy'],

                 }
             }
         }
```

In [62]:

```
from sklearn.model_selection import GridSearchCV
scores = []

for model_name, mp in model_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=5, return_train_score=False)
    clf.fit(X_train,y_train)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })

df1 = pd.DataFrame(scores,columns=['model','best_score','best_params'])
df1
```

Out[62]:

	model	best_score	best_params
0	SVM	0.927215	{'C': 10, 'kernel': 'rbf'}
1	Random_Forest	0.929684	{'n_estimators': 15}
2	Logistic_Regression	0.924620	{'C': 1}
3	Naive_Bayes_Gaussian	0.902120	{}
4	Decision_Tree	0.907025	{'criterion': 'entropy'}

In [ ]: