In [156]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [157]:

```
df = pd.read_csv(r'C:\Users\asus\Desktop\wisc_bc_data.csv')
```

In [158]:

```
df.head()
```

Out[158]:

	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	84358402	M	20.29	14.34	135.10	1297.0	

5 rows × 32 columns

In [159]:

```
df
```

Out[159]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothn
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	84358402	M	20.29	14.34	135.10	1297.0	
...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	
565	926682	M	20.13	28.25	131.20	1261.0	
566	926954	M	16.60	28.08	108.30	858.1	
567	927241	M	20.60	29.33	140.10	1265.0	
568	92751	B	7.76	24.54	47.92	181.0	

569 rows × 32 columns

In [160]:

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

Out[160]:

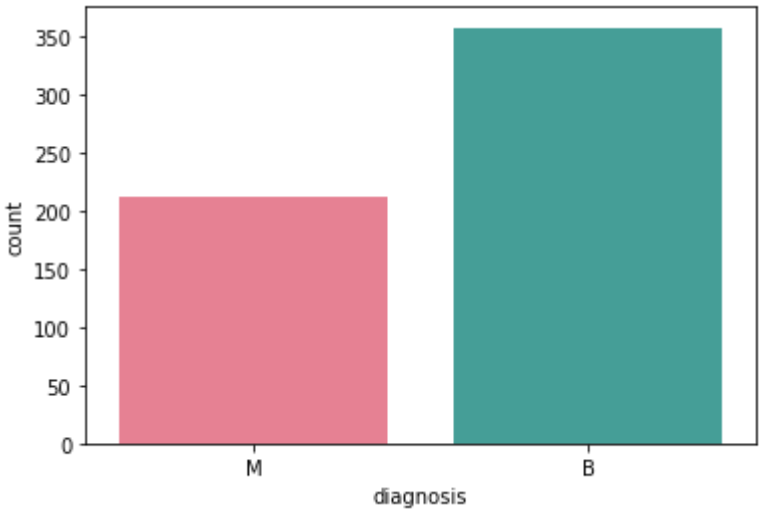
```
B    357
M    212
Name: diagnosis, dtype: int64
```

In [161]:

```
sns.countplot(df['diagnosis'], palette='husl')
```

Out[161]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21363ca6908>
```



In [162]:

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

Out[162]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	M	17.99	10.38	122.80	1001.0	0.11840
1	M	20.57	17.77	132.90	1326.0	0.08474
2	M	19.69	21.25	130.00	1203.0	0.10960
3	M	11.42	20.38	77.58	386.1	0.14250
4	M	20.29	14.34	135.10	1297.0	0.10030
...	...	...	...	...	...	...
564	M	21.56	22.39	142.00	1479.0	0.11100
565	M	20.13	28.25	131.20	1261.0	0.09780
566	M	16.60	28.08	108.30	858.1	0.08455
567	M	20.60	29.33	140.10	1265.0	0.11780
568	B	7.76	24.54	47.92	181.0	0.05263

569 rows × 31 columns

In [163]:

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

In [164]:

```
df.head()
```

Out[164]:

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

5 rows × 32 columns

In [165]:

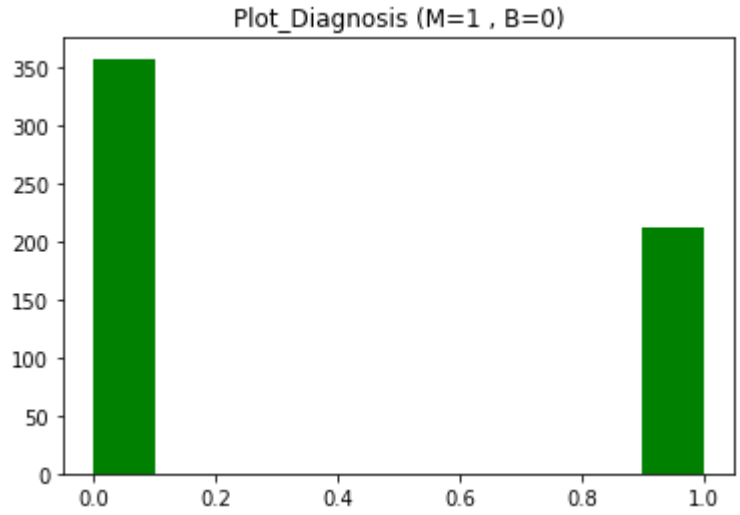
```
df.columns
```

Out[165]:

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mea
n',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_s
e',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_s
e',
       '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 [166]:

```
plt.hist(df['diagnosis'], color='g')
plt.title('Plot_Diagnosis (M=1 , B=0)')
plt.show()
```

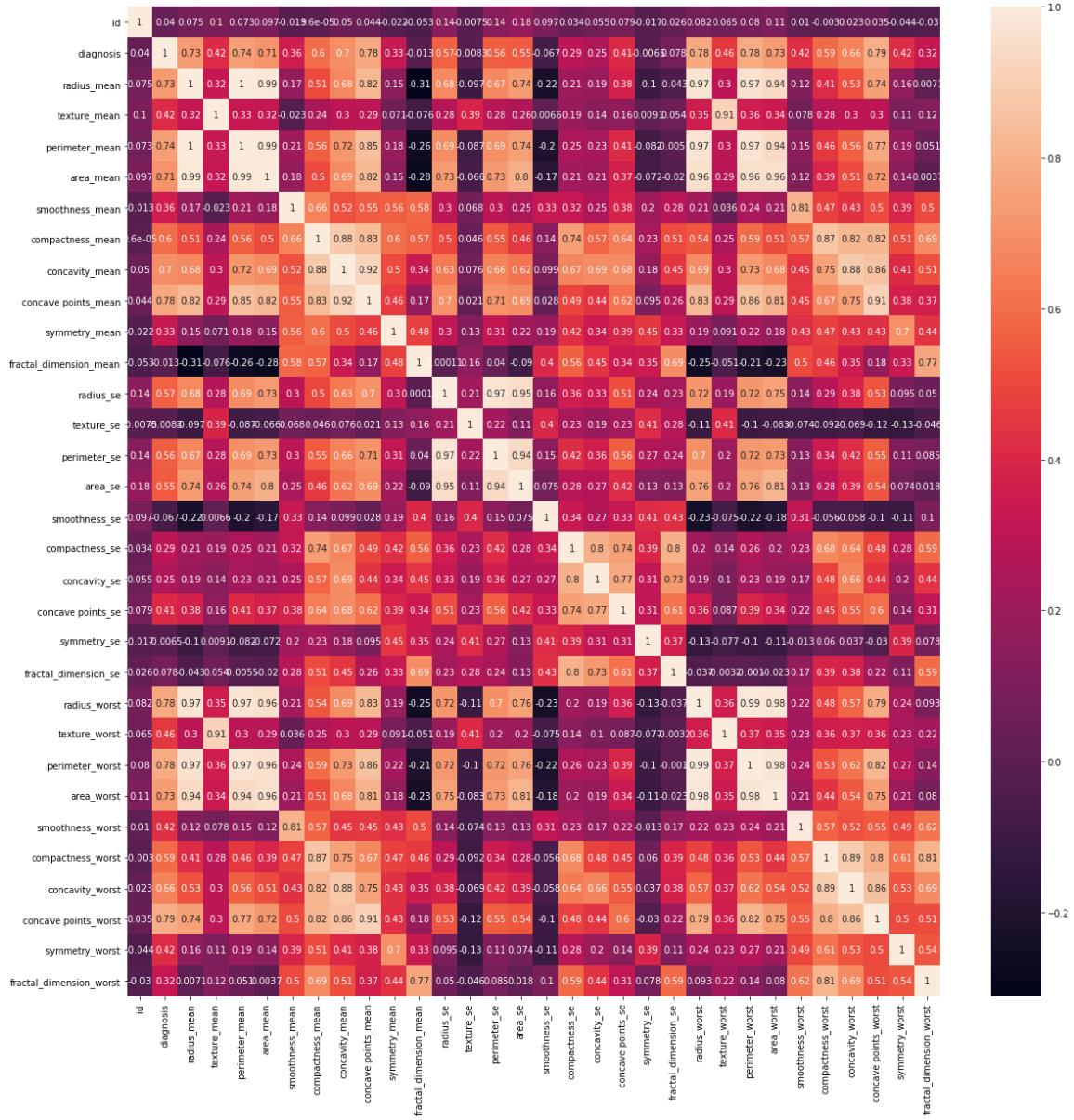


In [167]:

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

Out[167]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21361f92d08>
```



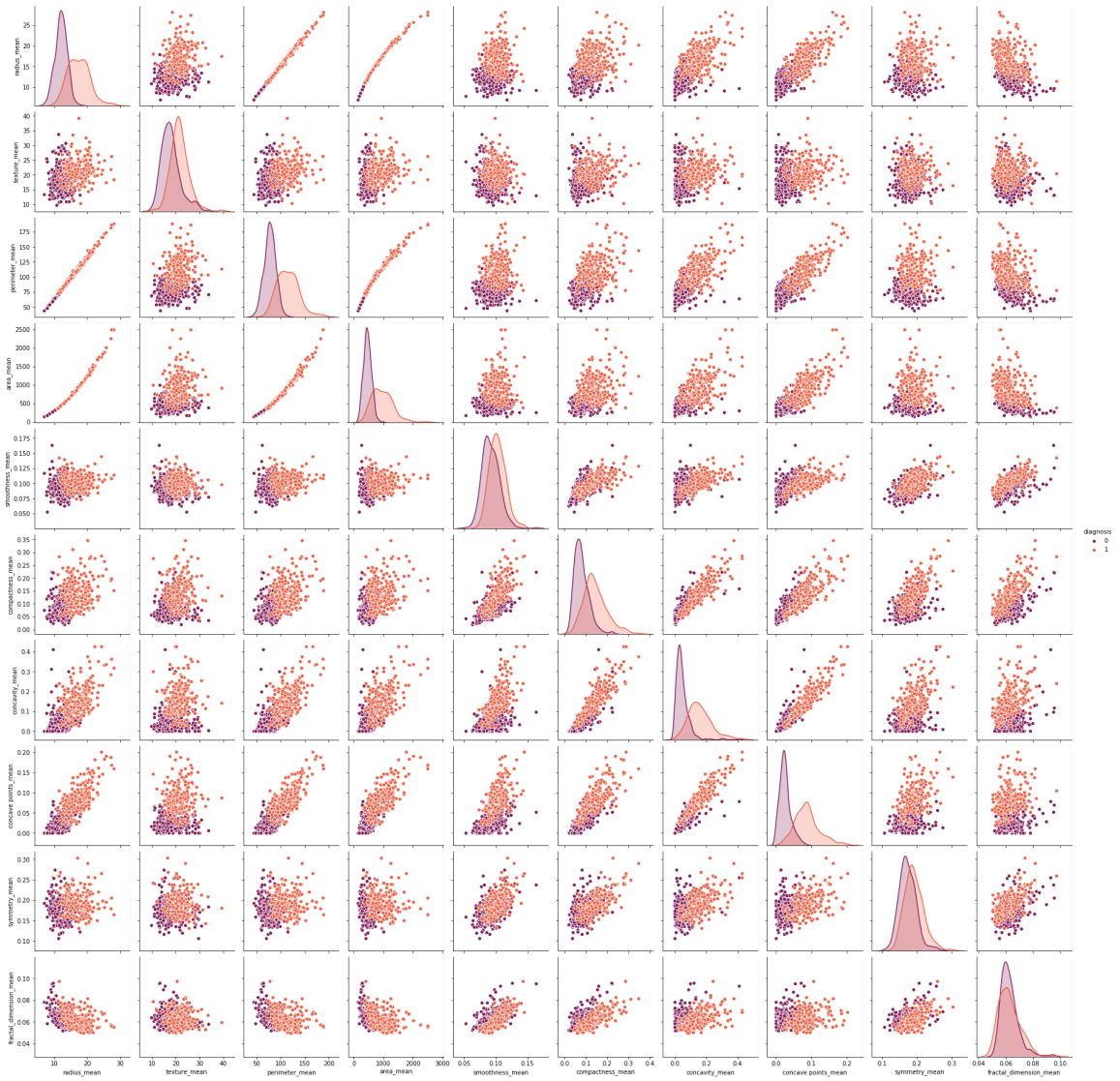
In [107]:

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

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

Out[107]:

```
<seaborn.axisgrid.PairGrid at 0x2135dfd39c8>
```



In [168]:

```
mean_features = list(df.columns[1:11])
se_features = list(df.columns[11:21])
worst_features = list(df.columns[21:31])
```

In [169]:

```
print (se_features)
```

```
['fractal_dimension_mean', 'radius_se', 'texture_se', 'perimeter_se', 'are
a_se', 'smoothness_se', 'compactness_se', 'concavity_se', 'concave points_
se', 'symmetry_se']
```

In [170]:

```
print (worst_features)
```

```
['fractal_dimension_se', 'radius_worst', 'texture_worst', 'perimeter_wors
t', 'area_worst', 'smoothness_worst', 'compactness_worst', 'concavity_wors
t', 'concave points_worst', 'symmetry_worst']
```

In [171]:

```
mean_features.append('diagnosis')
se_features.append('diagnosis')
worst_features.append('diagnosis')
```

In [172]:

```
corr = df[mean_features].corr()
```

In [173]:

```
corr
```

Out[173]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smo
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	
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	

In [174]:

```
corr = df[se_features].corr()
corr
```

Out[174]:

	fractal_dimension_mean	radius_se	texture_se	perimeter_se	area_
fractal_dimension_mean	1.000000	0.000111	0.164174	0.039830	-0.0901
radius_se	0.000111	1.000000	0.213247	0.972794	0.9518
texture_se	0.164174	0.213247	1.000000	0.223171	0.1115
perimeter_se	0.039830	0.972794	0.223171	1.000000	0.9376
area_se	-0.090170	0.951830	0.111567	0.937655	1.0000
smoothness_se	0.401964	0.164514	0.397243	0.151075	0.0751
compactness_se	0.559837	0.356065	0.231700	0.416322	0.2848
concavity_se	0.446630	0.332358	0.194998	0.362482	0.2708
concave points_se	0.341198	0.513346	0.230283	0.556264	0.4157
symmetry_se	0.345007	0.240567	0.411621	0.266487	0.1341
diagnosis	-0.012838	0.567134	-0.008303	0.556141	0.5482

In [175]:

```
corr = df[worst_features].corr()
corr
```

Out[175]:

	fractal_dimension_se	radius_worst	texture_worst	perimeter_worst	area
fractal_dimension_se	1.000000	-0.037488	-0.003195	-0.001000	-0
radius_worst	-0.037488	1.000000	0.359921	0.993708	0
texture_worst	-0.003195	0.359921	1.000000	0.365098	0
perimeter_worst	-0.001000	0.993708	0.365098	1.000000	0
area_worst	-0.022736	0.984015	0.345842	0.977578	1
smoothness_worst	0.170568	0.216574	0.225429	0.236775	0
compactness_worst	0.390159	0.475820	0.360832	0.529408	0
concavity_worst	0.379975	0.573975	0.368366	0.618344	0
concave points_worst	0.215204	0.787424	0.359755	0.816322	0
symmetry_worst	0.111094	0.243529	0.233027	0.269493	0
diagnosis	0.077972	0.776454	0.456903	0.782914	0

In [176]:

```
##Training Model
```

In [177]:

```
prediction_vars = ['perimeter_mean', 'compactness_mean', 'concavity_mean']
```

In [228]:

```
from sklearn.model_selection import train_test_split

train, test = train_test_split(df, test_size = 0.20, random_state=1)
```

In [229]:

```
train_x = train[prediction_vars]
train_y = train.diagnosis

test_x = test[prediction_vars]
test_y = test.diagnosis
```

In [230]:

```
train_x
```

Out[230]:

	perimeter_mean	compactness_mean	concavity_mean
408	117.80	0.13040	0.120100
4	135.10	0.13280	0.198000
307	56.36	0.03116	0.003681
386	78.78	0.07823	0.068390
404	78.29	0.04571	0.021090
...	...	...	...
129	130.40	0.15890	0.254500
144	68.26	0.05139	0.022510
72	114.20	0.18300	0.169200
235	89.79	0.06945	0.014620
37	82.61	0.03766	0.025620

455 rows × 3 columns

In [231]:

```
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

In [232]:

```
#model = MLPClassifier()
```

In [233]:

```
print("Random Forest")
```

```
Random Forest
```

In [234]:

```
model = RandomForestClassifier()
#model = KNeighborsClassifier()
#model = SVC()
```

In [235]:

```
model.fit(train_x, train_y)
```

Out[235]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [236]:

```
model.predict(test_x)
```

Out[236]:

```
array([1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0], dtype=int64)
```

In [237]:

```
predictions = model.predict(test_x)
```

In [238]:

```
test_y
```

Out[238]:

```
421    0
47     1
292    0
186    1
414    1
      ..
172    1
3      1
68     0
448    0
442    0
Name: diagnosis, Length: 114, dtype: int64
```

In [239]:

```
from sklearn.metrics import confusion_matrix

confusion_matrix(test_y, predictions)
```

Out[239]:

```
array([[68,  4],
       [ 9, 33]], dtype=int64)
```

In [240]:

```
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
```

In [241]:

```
precision = precision_score(test_y, predictions)
print ("The precision score is %.2f" %precision)

recall = recall_score(test_y, predictions)
print ("The recall score is %.2f" %recall)
```

```
The precision score is 0.89
The recall score is 0.79
```

In [242]:

```
F1 = 2 * (precision * recall) / (precision + recall)
print(" The F1 Score is %.2f" %F1)
```

```
 The F1 Score is 0.84
```

In [243]:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
```

In [244]:

```
accuracy = accuracy_score(test_y, predictions)
print ("The accuracy score is %.2f" %accuracy)

AUC = roc_auc_score(test_y, predictions)
print ("The AUC score is %.2f" %AUC)
```

```
The accuracy score is 0.89
The AUC score is 0.87
```

In [245]:

```
from sklearn.metrics import classification_report
```

In [246]:

```
print(classification_report(test_y, predictions))
```

```
              precision    recall  f1-score   support

           0       0.88      0.94      0.91        72
           1       0.89      0.79      0.84        42

    accuracy                           0.89       114
   macro avg       0.89      0.87      0.87       114
weighted avg       0.89      0.89      0.88       114
```

In [247]:

```
print("KNN")
```

```
KNN
```

In [248]:

```
model = KNeighborsClassifier()
```

In [249]:

```
model.fit(train_x, train_y)
```

Out[249]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [250]:

```
model.predict(test_x)
```

Out[250]:

```
array([0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0], dtype=int64)
```

In [251]:

```
predictions = model.predict(test_x)
```

In [252]:

```
confusion_matrix(test_y, predictions)
```

Out[252]:

```
array([[68,  4],
       [13, 29]], dtype=int64)
```

In [253]:

```
precision = precision_score(test_y, predictions)
print ("The precision score is %.2f" %precision)

recall = recall_score(test_y, predictions)
print ("The recall score is %.2f" %recall)
```

```
The precision score is 0.88
The recall score is 0.69
```

In [254]:

```
F1 = 2 * (precision * recall) / (precision + recall)
print(" The F1 Score is %.2f" %F1)
```

```
 The F1 Score is 0.77
```

In [255]:

```
accuracy = accuracy_score(test_y, predictions)
print ("The accuracy score is %.2f" %accuracy)

AUC = roc_auc_score(test_y, predictions)
print ("The AUC score is %.2f" %AUC)
```

```
The accuracy score is 0.85
The AUC score is 0.82
```

In [256]:

```
print(classification_report(test_y, predictions))
```

```
              precision    recall  f1-score   support

           0       0.84      0.94      0.89        72
           1       0.88      0.69      0.77        42

    accuracy                           0.85       114
   macro avg       0.86      0.82      0.83       114
weighted avg       0.85      0.85      0.85       114
```

In [257]:

```
print("SVM")
```

```
SVM
```

In [258]:

```
model = SVC()
```

In [259]:

```
model.fit(train_x, train_y)
```

Out[259]:

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [260]:

```
model.predict(test_x)
```

Out[260]:

```
array([1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0], dtype=int64)
```

In [261]:

```
predictions = model.predict(test_x)
```

In [262]:

```
confusion_matrix(test_y, predictions)
```

Out[262]:

```
array([[71,  1],
       [15, 27]], dtype=int64)
```

In [263]:

```
precision = precision_score(test_y, predictions)
print ("The precision score is %.2f" %precision)

recall = recall_score(test_y, predictions)
print ("The recall score is %.2f" %recall)
```

```
The precision score is 0.96
The recall score is 0.64
```

In [264]:

```
F1 = 2 * (precision * recall) / (precision + recall)
print(" The F1 Score is %.2f" %F1)
```

```
 The F1 Score is 0.77
```

In [265]:

```
accuracy = accuracy_score(test_y, predictions)
print ("The accuracy score is %.2f" %accuracy)

AUC = roc_auc_score(test_y, predictions)
print ("The AUC score is %.2f" %AUC)
```

```
The accuracy score is 0.86
The AUC score is 0.81
```

In [266]:

```
print(classification_report(test_y, predictions))
```

```
              precision    recall  f1-score   support

           0       0.83      0.99      0.90        72
           1       0.96      0.64      0.77        42

    accuracy                           0.86       114
   macro avg       0.89      0.81      0.84       114
weighted avg       0.88      0.86      0.85       114
```

In [267]:

```
#Naive Bayes
```

In [268]:

```
model = GaussianNB()
```

In [269]:

```
model.fit(train_x, train_y)
```

Out[269]:

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [270]:

```
model.predict(test_x)
```

Out[270]:

```
array([1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 0, 0], dtype=int64)
```

In [271]:

```
predictions = model.predict(test_x)
```

In [272]:

```
confusion_matrix(test_y, predictions)
```

Out[272]:

```
array([[67,  5],
       [ 9, 33]], dtype=int64)
```

In [273]:

```
precision = precision_score(test_y, predictions)
print ("The precision score is %.2f" %precision)

recall = recall_score(test_y, predictions)
print ("The recall score is %.2f" %recall)
```

```
The precision score is 0.87
The recall score is 0.79
```

In [274]:

```
F1 = 2 * (precision * recall) / (precision + recall)
print(" The F1 Score is %.2f" %F1)
```

```
 The F1 Score is 0.82
```

In [275]:

```
accuracy = accuracy_score(test_y, predictions)
print ("The accuracy score is %.2f" %accuracy)

AUC = roc_auc_score(test_y, predictions)
print ("The AUC score is %.2f" %AUC)
```

```
The accuracy score is 0.88
The AUC score is 0.86
```

In [276]:

```
print(classification_report(test_y, predictions))
```

```
              precision    recall  f1-score   support

           0       0.88      0.93      0.91        72
           1       0.87      0.79      0.82        42

    accuracy                           0.88       114
   macro avg       0.88      0.86      0.87       114
weighted avg       0.88      0.88      0.88       114
```

In [227]:

```
print("AYSEGUL KANDEFER")
```

```
AYSEGUL KANDEFER
```

In [ ]:

```

```