In [36]:
```
#importing the Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [37]:
```
#importing and Reading the csv file
df=pd.read_csv("D:\data science\Tumor_Detection.csv")
```

In [38]:
```
df.head()
```

Out[38]:

	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	

5 rows × 32 columns

In [39]:
```
df.columns
```

Out[39]:
```
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 [40]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 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
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

In [41]:
```
df.drop('id',axis = 1, inplace = True)
```

In [42]:
```
df.columns
```

Out[42]:
```
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 [43]:
```
type(df.columns)
```

Out[43]:
```
pandas.core.indexes.base.Index
```

In [44]:
```
l=list(df.columns)
print(l)
```

```
['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothn
ess_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_me
an', 'fractal_dimension_mean', 'radius_se', 'texture_se', 'perimeter_se', 'area_se',
'smoothness_se', 'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_s
e', 'fractal_dimension_se', 'radius_worst', 'texture_worst', 'perimeter_worst', 'are
a_worst', 'smoothness_worst', 'compactness_worst', 'concavity_worst', 'concave point
s_worst', 'symmetry_worst', 'fractal_dimension_worst']
```

In [45]:
```
#start point
features_means = l [1:11]
features_se = l [11:21]
features_worst = l [21:]
```

In [46]:
```
print(features_means)
```

```
['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean', 'c
ompactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_mean', 'fractal
_dimension_mean']
```

In [47]:
```
print(features_se)
```

```
['radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se', 'compactness
_se', 'concavity_se', 'concave points_se', 'symmetry_se', 'fractal_dimension_se']
```

In [48]:
```
print(features_worst)
```

```
['radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smoothness_wors
t', 'compactness_worst', 'concavity_worst', 'concave points_worst', 'symmetry_wors
t', 'fractal_dimension_worst']
```

In [49]:
```
df.head()
```

Out[49]:

	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

5 rows × 31 columns

In [50]:
```
df
```

Out[50]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	M	17.99	10.38	122.80	1001.0	0.1184
1	M	20.57	17.77	132.90	1326.0	0.0847
2	M	19.69	21.25	130.00	1203.0	0.1096
3	M	11.42	20.38	77.58	386.1	0.1425
4	M	20.29	14.34	135.10	1297.0	0.1003
...	...	...	...	...	...	...
564	M	21.56	22.39	142.00	1479.0	0.1110
565	M	20.13	28.25	131.20	1261.0	0.0978
566	M	16.60	28.08	108.30	858.1	0.0845
567	M	20.60	29.33	140.10	1265.0	0.1178
568	B	7.76	24.54	47.92	181.0	0.0526

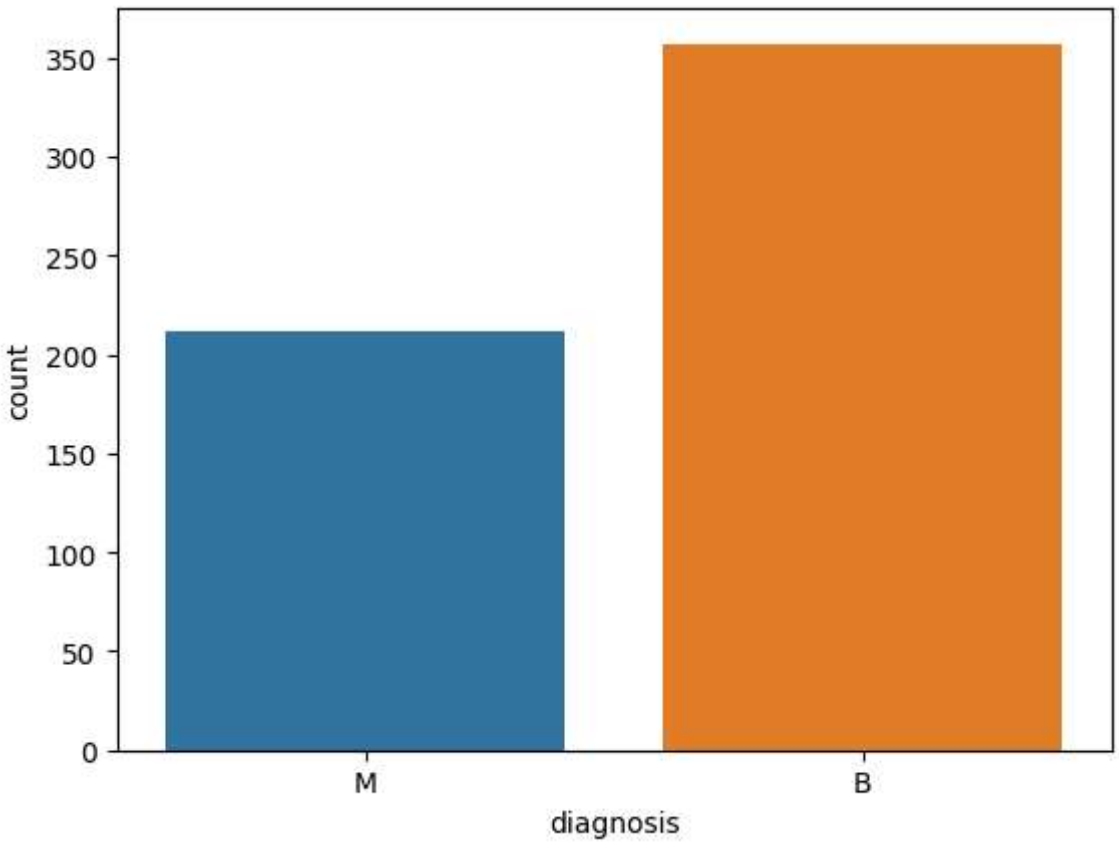
569 rows × 31 columns

In [51]:
```
df['diagnosis'].unique()
```

Out[51]:
```
array(['M', 'B'], dtype=object)
```

In [52]:
```
x = df["diagnosis"]
sns.countplot(x=x, label='count')
```

Out[52]:
```
<Axes: xlabel='diagnosis', ylabel='count'>
```



In [53]:
```
df.shape
```

Out[53]: (569, 31)

In [54]:
```
df.describe()
```

Out[54]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
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 [55]:
```
# corr = df.corr()  -----ERROR
```

In [56]:
```
# Select only numeric columns
numeric_df = df.select_dtypes(include=['number'])

# Calculate the correlation matrix
corr_matrix = numeric_df.corr()
```

In [57]:
```
corr_matrix
```

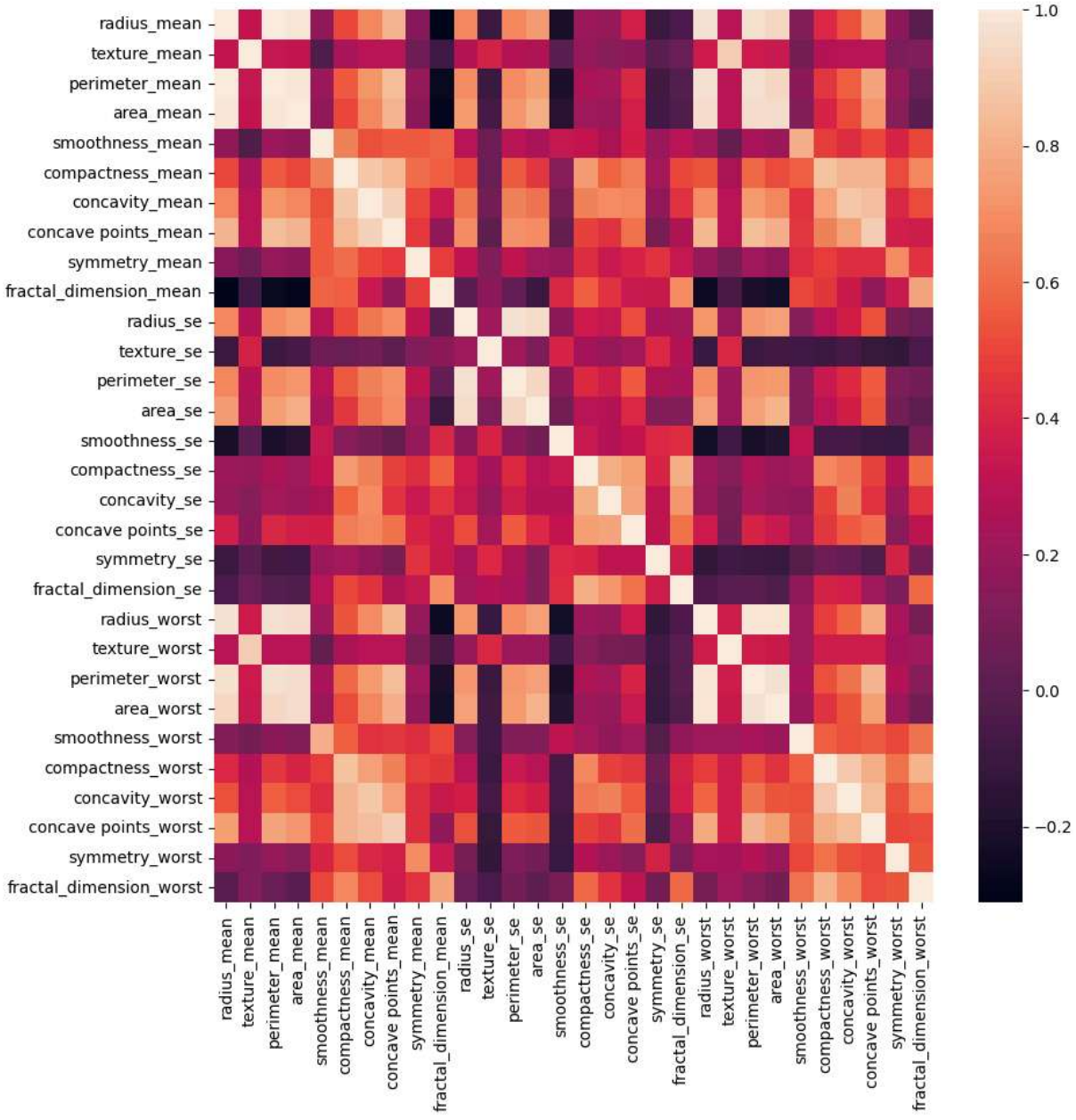
Out[57]:

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

30 rows × 30 columns

In [58]:
```
#heatmao
plt.figure(figsize=(10,10))
sns.heatmap(corr_matrix)
```

Out[58]: <Axes: >



In [59]:
```
df['diagnosis']=df['diagnosis'].map({'M':1,'B':0})
df.head()
```

Out[59]:

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

5 rows × 31 columns

In [60]:
```
df['diagnosis'].unique()
```

Out[60]:
```
array([1, 0], dtype=int64)
```

In [61]:
```
x=df.drop('diagnosis', axis=1)
x.head()
```

Out[61]:

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

5 rows × 30 columns

In [62]:
```
y=df['diagnosis']
y
```

Out[62]:
```
0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: diagnosis, Length: 569, dtype: int64
```

In [63]:
```
# Divide the dataset into train and test set
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
# from sklearn.preprocessing import standardScaler
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)
```

In [64]:
```
df.shape
```

Out[64]: (569, 31)

In [65]:
```
x_train.shape
```

Out[65]: (398, 30)

In [66]:
```
x_test.shape
```

Out[66]: (171, 30)

In [67]:
```
y_train.shape
```

Out[67]: (398,)

In [68]:
```
x_test.shape
```

Out[68]: (171, 30)

In [69]:
```
from sklearn.preprocessing import StandardScaler
ss= StandardScaler()
x_train=ss.fit_transform(x_train)
x_test=ss.transform(x_test)
x_train
```

Out[69]:
```
array([[ 0.55981589, -1.03788826,  0.492786  , ..., -0.04589802,
        -0.23757936, -0.79512129],
       [-0.0288037 , -1.48550675, -0.10572654, ..., -0.85902036,
        -1.36329253, -0.75543433],
       [-0.35065974,  1.28385008, -0.41693629, ..., -0.85435244,
        -1.02126169, -1.19813963],
       ...,
       [-0.38835459, -0.73269383, -0.42658295, ..., -0.87558397,
         1.13386471, -0.76717272],
       [-0.38255538, -0.8479895 , -0.35528153, ..., -0.21454562,
        -1.53098727, -0.73083959],
       [-0.5420336 ,  1.67495109, -0.47817163, ...,  0.45101008,
         0.48799093,  2.02097635]])
```

In [70]:
```
#appply random forest classifier
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier

rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
y_pred=rfc.predict(x_test)
print(accuracy_score(y_test,y_pred))
```

```
0.9239766081871345
```

In [ ]: