In [1]:
```
import numpy
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In [4]:
```
df=pd.read_csv("data (1).csv")
```

In [5]:
```
df.head()
```

Out[5]:

	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 × 33 columns

In [6]:
```
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 [11]:
```
df.isna().sum()
```

Out[11]:
```
id                         0
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 [8]:
```
df.shape
```

Out[8]:
```
(569, 33)
```

In [9]:
```
df=df.dropna(axis=1)
```

In [10]:
```
df.shape
```

Out[10]:
```
(569, 32)
```

In [12]:
```
df.describe()
```

Out[12]:

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

8 rows × 31 columns

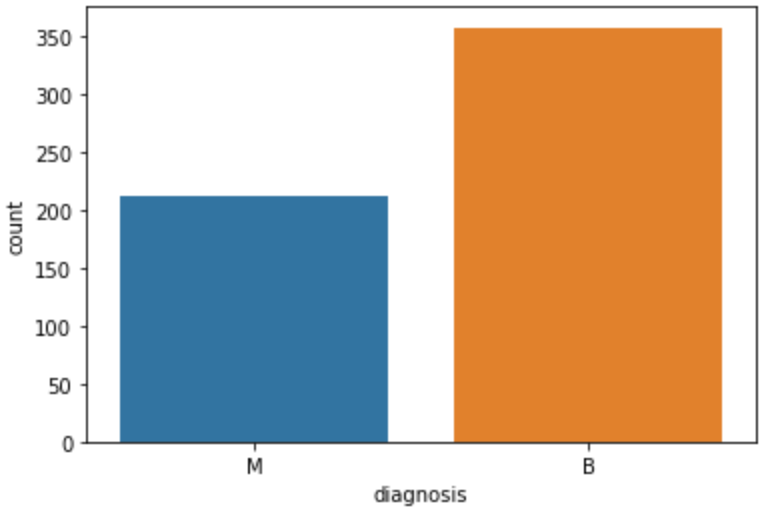
In [13]:
```
df['diagnosis'].value_counts()
```

Out[13]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

In [14]:
```
sns.countplot(df['diagnosis'],label="count")
```

```
/Users/dathkanuri/opt/anaconda3/lib/python3.9/site-packages/seaborn/_dec
orators.py:36: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `da
ta`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  warnings.warn(
```

Out[14]:
```
<AxesSubplot:xlabel='diagnosis', ylabel='count'>
```



In [18]:

```
from sklearn.preprocessing import LabelEncoder
labelencoder_Y = LabelEncoder()
df.iloc[:,1]=labelencoder_Y.fit_transform(df.iloc[:,1].values)
```

In [20]:
```
df.head(50)
```

Out[20]:

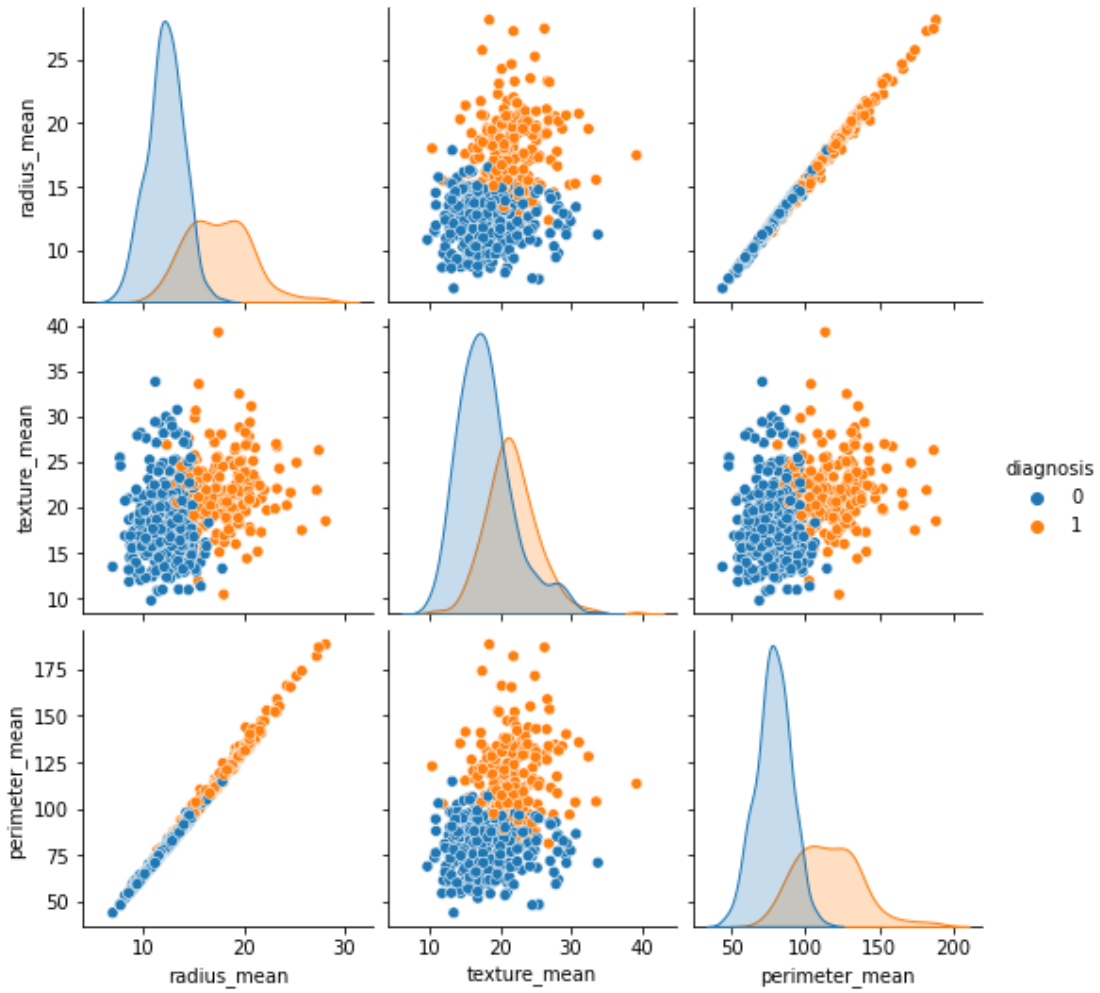
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothi
0	842302	1	17.990	10.38	122.80	1001.0	
1	842517	1	20.570	17.77	132.90	1326.0	
2	84300903	1	19.690	21.25	130.00	1203.0	
3	84348301	1	11.420	20.38	77.58	386.1	
4	84358402	1	20.290	14.34	135.10	1297.0	
5	843786	1	12.450	15.70	82.57	477.1	
6	844359	1	18.250	19.98	119.60	1040.0	
7	84458202	1	13.710	20.83	90.20	577.9	
8	844981	1	13.000	21.82	87.50	519.8	
9	84501001	1	12.460	24.04	83.97	475.9	
10	845636	1	16.020	23.24	102.70	797.8	
11	84610002	1	15.780	17.89	103.60	781.0	
12	846226	1	19.170	24.80	132.40	1123.0	
13	846381	1	15.850	23.95	103.70	782.7	
14	84667401	1	13.730	22.61	93.60	578.3	
15	84799002	1	14.540	27.54	96.73	658.8	
16	848406	1	14.680	20.13	94.74	684.5	
17	84862001	1	16.130	20.68	108.10	798.8	
18	849014	1	19.810	22.15	130.00	1260.0	
19	8510426	0	13.540	14.36	87.46	566.3	
20	8510653	0	13.080	15.71	85.63	520.0	
21	8510824	0	9.504	12.44	60.34	273.9	
22	8511133	1	15.340	14.26	102.50	704.4	
23	851509	1	21.160	23.04	137.20	1404.0	
24	852552	1	16.650	21.38	110.00	904.6	
25	852631	1	17.140	16.40	116.00	912.7	
26	852763	1	14.580	21.53	97.41	644.8	
27	852781	1	18.610	20.25	122.10	1094.0	
28	852973	1	15.300	25.27	102.40	732.4	
29	853201	1	17.570	15.05	115.00	955.1	
30	853401	1	18.630	25.11	124.80	1088.0	
31	853612	1	11.840	18.70	77.93	440.6	
32	85382601	1	17.020	23.98	112.80	899.3	
33	854002	1	19.270	26.47	127.90	1162.0	
34	854039	1	16.130	17.88	107.00	807.2	

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothi
35	854253	1	16.740	21.59	110.10	869.5	
36	854268	1	14.250	21.72	93.63	633.0	
37	854941	0	13.030	18.42	82.61	523.8	
38	855133	1	14.990	25.20	95.54	698.8	
39	855138	1	13.480	20.82	88.40	559.2	
40	855167	1	13.440	21.58	86.18	563.0	
41	855563	1	10.950	21.35	71.90	371.1	
42	855625	1	19.070	24.81	128.30	1104.0	
43	856106	1	13.280	20.28	87.32	545.2	
44	85638502	1	13.170	21.81	85.42	531.5	
45	857010	1	18.650	17.60	123.70	1076.0	
46	85713702	0	8.196	16.84	51.71	201.9	
47	85715	1	13.170	18.66	85.98	534.6	
48	857155	0	12.050	14.63	78.04	449.3	
49	857156	0	13.490	22.30	86.91	561.0	

50 rows × 32 columns

In [21]:
```
sns.pairplot(df.iloc[:,1:5],hue="diagnosis")
```

Out[21]: <seaborn.axisgrid.PairGrid at 0x7fca856c6be0>



In [22]:
```
df.iloc[:,1:32].corr()
```

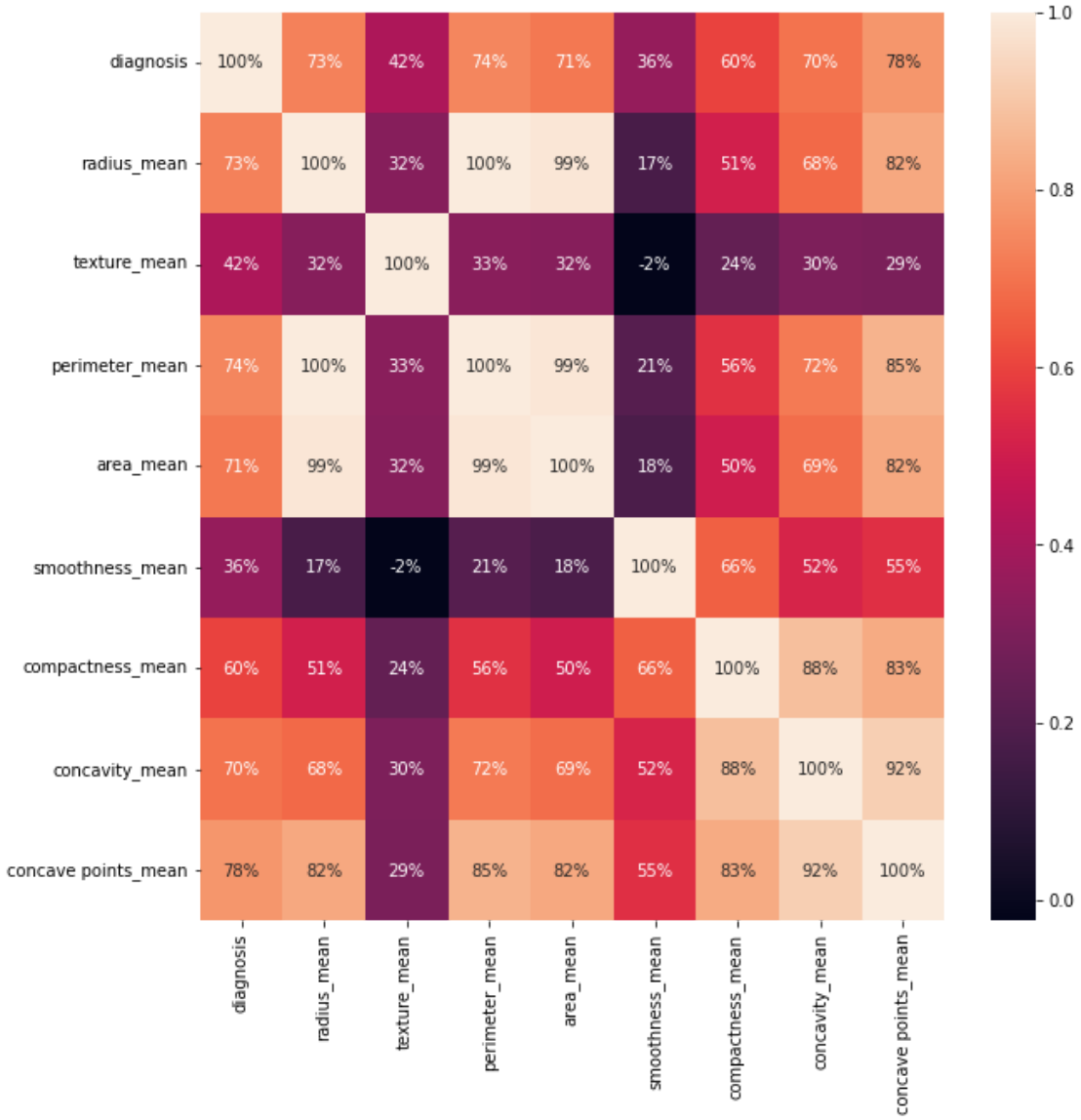
Out[22]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
diagnosis	1.000000	0.730029	0.415185	0.742636	0.70898
radius_mean	0.730029	1.000000	0.323782	0.997855	0.98735
texture_mean	0.415185	0.323782	1.000000	0.329533	0.32108
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.98650
area_mean	0.708984	0.987357	0.321086	0.986507	1.00000
smoothness_mean	0.358560	0.170581	-0.023389	0.207278	0.17702
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.49850
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.68598
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.82326
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.15129
fractal_dimension_mean	-0.012838	-0.311631	-0.076437	-0.261477	-0.28311
radius_se	0.567134	0.679090	0.275869	0.691765	0.73256
texture_se	-0.008303	-0.097317	0.386358	-0.086761	-0.06628
perimeter_se	0.556141	0.674172	0.281673	0.693135	0.72662
area_se	0.548236	0.735864	0.259845	0.744983	0.80008
smoothness_se	-0.067016	-0.222600	0.006614	-0.202694	-0.16677
compactness_se	0.292999	0.206000	0.191975	0.250744	0.21258
concavity_se	0.253730	0.194204	0.143293	0.228082	0.20766
concave points_se	0.408042	0.376169	0.163851	0.407217	0.37232
symmetry_se	-0.006522	-0.104321	0.009127	-0.081629	-0.07249
fractal_dimension_se	0.077972	-0.042641	0.054458	-0.005523	-0.01988
radius_worst	0.776454	0.969539	0.352573	0.969476	0.96274
texture_worst	0.456903	0.297008	0.912045	0.303038	0.28748
perimeter_worst	0.782914	0.965137	0.358040	0.970387	0.95912
area_worst	0.733825	0.941082	0.343546	0.941550	0.95921
smoothness_worst	0.421465	0.119616	0.077503	0.150549	0.12352
compactness_worst	0.590998	0.413463	0.277830	0.455774	0.39041
concavity_worst	0.659610	0.526911	0.301025	0.563879	0.51260
concave points_worst	0.793566	0.744214	0.295316	0.771241	0.72201
symmetry_worst	0.416294	0.163953	0.105008	0.189115	0.14357
fractal_dimension_worst	0.323872	0.007066	0.119205	0.051019	0.00373

31 rows × 31 columns

In [23]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.iloc[:,1:10].corr(),annot=True,fmt=".0%")
```

Out[23]: <AxesSubplot:>



In [24]:
```
X=df.iloc[:,2:31].values
Y=df.iloc[:,1].values
```

In [25]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20,random_s
```

In [26]:
```
from sklearn.preprocessing import StandardScaler
X_train=StandardScaler().fit_transform(X_train)
X_test=StandardScaler().fit_transform(X_test)
```

In [27]:
```
def models(X_train,Y_train):
        #logistic regression
        from sklearn.linear_model import LogisticRegression
        log=LogisticRegression(random_state=0)
        log.fit(X_train,Y_train)


        #Decision Tree
        from sklearn.tree import DecisionTreeClassifier
        tree=DecisionTreeClassifier(random_state=0,criterion="entropy")
        tree.fit(X_train,Y_train)

        #Random Forest
        from sklearn.ensemble import RandomForestClassifier
        forest=RandomForestClassifier(random_state=0,criterion="entropy",r
        forest.fit(X_train,Y_train)

        print('[0]logistic regression accuracy:',log.score(X_train,Y_trair
        print('[1]Decision tree accuracy:',tree.score(X_train,Y_train))
        print('[2]Random forest accuracy:',forest.score(X_train,Y_train))

        return log,tree,forest
```

In [28]:
```
model=models(X_train,Y_train)
```

```
[0]logistic regression accuracy: 0.9912087912087912
[1]Decision tree accuracy: 1.0
[2]Random forest accuracy: 0.9978021978021978
```

In [29]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

for i in range(len(model)):
    print("Model",i)
    print(classification_report(Y_test,model[i].predict(X_test)))
    print('Accuracy : ',accuracy_score(Y_test,model[i].predict(X_test)))
```

```
Model 0
              precision    recall  f1-score   support

           0       0.96      0.99      0.97        67
           1       0.98      0.94      0.96        47

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114

Accuracy :  0.9649122807017544
Model 1
              precision    recall  f1-score   support

           0       0.94      0.96      0.95        67
           1       0.93      0.91      0.92        47

    accuracy                           0.94       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.94      0.94      0.94       114

Accuracy :  0.9385964912280702
Model 2
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        67
           1       1.00      0.94      0.97        47

    accuracy                           0.97       114
   macro avg       0.98      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114

Accuracy :  0.9736842105263158
```

In [30]:
```
pred=model[2].predict(X_test)
print('Predicted values:')
print(pred)
print('Actual values:')
print(Y_test)
```

```
Predicted values:
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1
0
 1 0 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0
0
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1
0
 1 1 0]
Actual values:
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1
0
 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0
1
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1
0
 1 1 0]
```

In [31]:
```
from joblib import dump
dump(model[2],"Cancer_prediction.joblib")
```

Out[31]:
```
['Cancer_prediction.joblib']
```

In [ ]:
```

```