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

In [2]:
```
#importing  data from the files
df = pd.read_csv('C:/Users/royalgifts/Downloads/cancer.csv')
```

In [3]:
```
df.head()
```

Out[3]:

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

5 rows × 33 columns

In [4]:
```
# return the size of dataset
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]:
```
df.isna().sum()
```

Out[6]:
```
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
Unnamed: 32                569
dtype: int64
```

In [7]:
```
# remove the column
df = df.dropna(axis = 1)
```

In [8]:
```
# describe the dataset
df.describe()
```

Out[8]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_r
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.00
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.10
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.05
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.01

25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.06
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.09
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.13
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.34

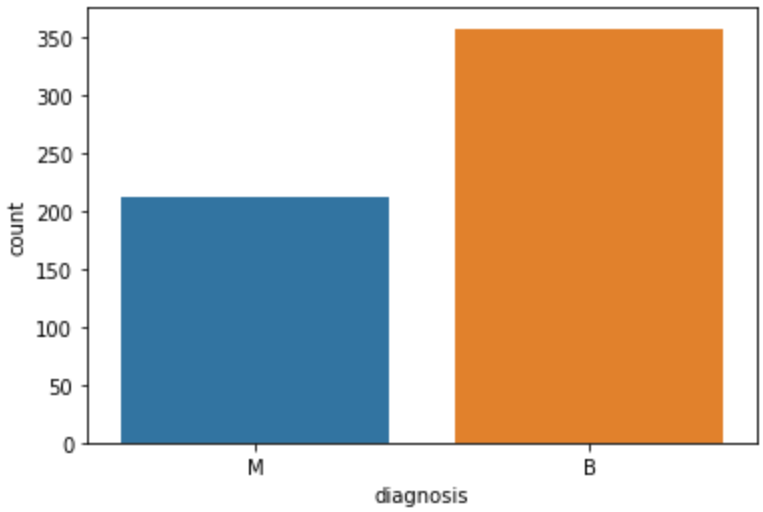
8 rows × 31 columns

In [9]:
```
# Get the count of malignant<M> and Benign<B> cells
df['diagnosis'].value_counts()
```

Out[9]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

In [10]:
```
sns.countplot(df['diagnosis'],label='count')
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x815d5d1708>
```



In [11]:
```
# label encoding(convert the value of M and B into 1 and 0)
from sklearn.preprocessing import LabelEncoder
labelencoder_Y = LabelEncoder()
df.iloc[:,1]=labelencoder_Y.fit_transform(df.iloc[:,1].values)
```

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

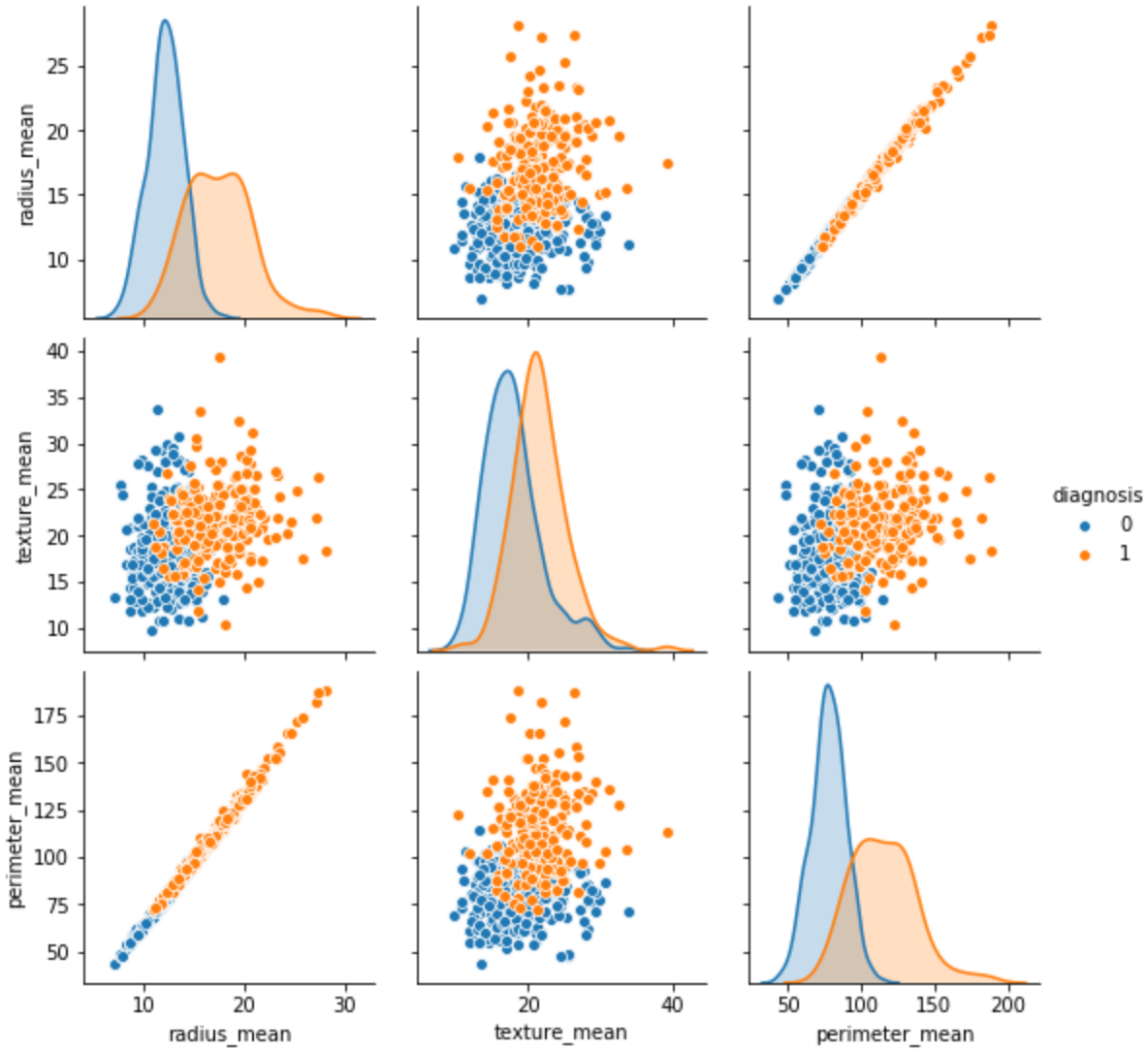
Out[12]:

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

5 rows × 32 columns

In [13]:
```
sns.pairplot(df.iloc[:,1:5],hue='diagnosis')
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x815a4b6988>
```



In [14]:
```
# get the correlation
df.iloc[:,1:32].corr().head()
```

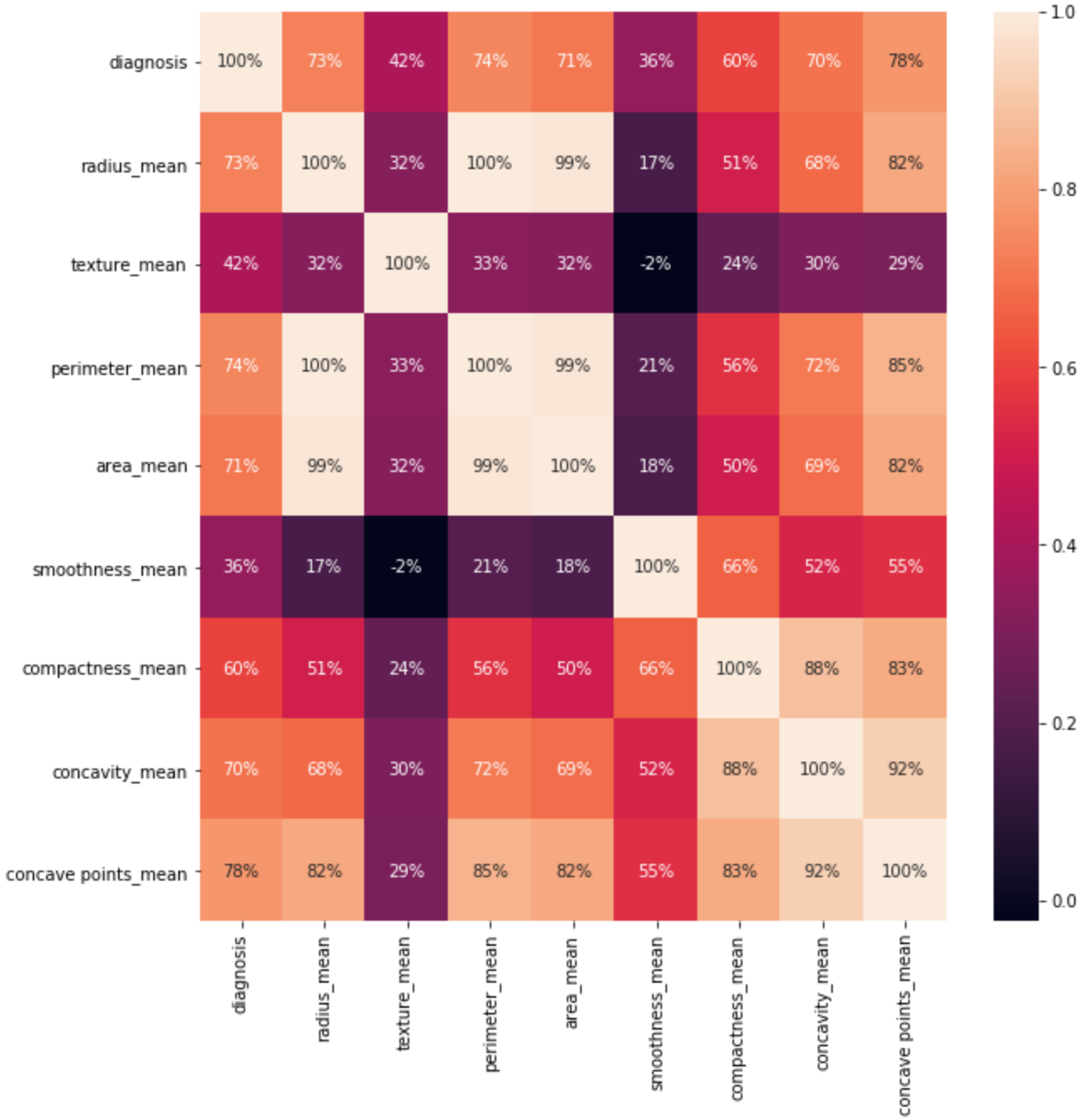
Out[14]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactr
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	

5 rows × 31 columns

In [15]:
```
# visualize the correlation
plt.figure(figsize=(10,10))
sns.heatmap(df.iloc[:,1:10].corr(),annot=True,fmt='.0%')
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x815c3c8388>
```



In [16]:
```
# split the dataset into dependent(X) and Independent(Y) datasets
X=df.iloc[:,2:31].values
Y=df.iloc[:,1].values
```

In [17]:
```
# splitting the data into training and test dataset
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_state=0)
```

In [18]:
```
# feature scaling
from sklearn.preprocessing import StandardScaler
X_train=StandardScaler().fit_transform(X_train)
X_test=StandardScaler().fit_transform(X_test)
```

In [22]:
```
# models/ Algorithms

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',n_estimators=10)
    forest.fit(X_train,Y_train)

    print('[0]logistic regression accuracy:',log.score(X_train,Y_train))
    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 [23]:
```
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 [26]:
```
# testing the models/result

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 [27]:
```
# prediction of random forest
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 [28]:
```
from joblib import dump
dump(model[2],'Breast_Cancer_prediction.joblib')
```

Out[28]:
```
['Breast_Cancer_prediction.joblib']
```

In [ ]: