Importing the required libraries

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

Loading the dataset

In [2]:
```
data = pd.read_csv('breast-cancer.csv')
data.head()
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
0	842302	M	17.99	10.38	122.80	1001.0	0.1184
1	842517	M	20.57	17.77	132.90	1326.0	0.0847
2	84300903	M	19.69	21.25	130.00	1203.0	0.1096
3	84348301	M	11.42	20.38	77.58	386.1	0.1425
4	84358402	M	20.29	14.34	135.10	1297.0	0.1003

5 rows × 32 columns

In [3]:
```
data.shape
```

Out[3]:
```
(569, 32)
```

In [4]:
```
data.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 [5]:
```
data.isnull().sum()
```

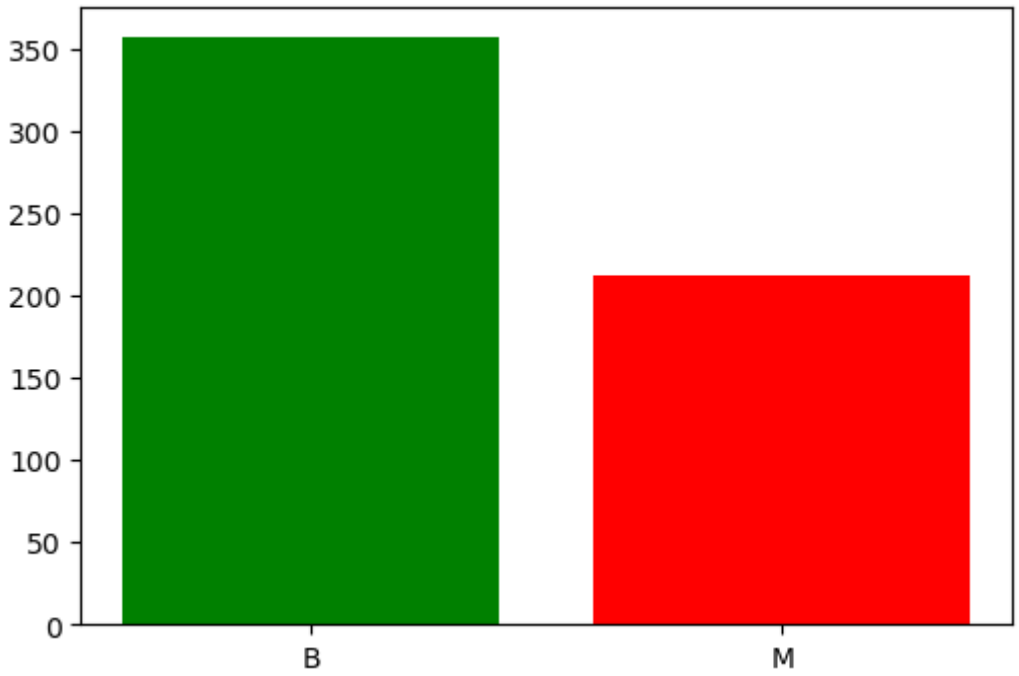
Out[5]:
```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
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 [6]:
```
data['diagnosis'].value_counts()
```

Out[6]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

EDA

In [7]:
```
plt.figure(figsize=(6,4))
plt.bar(data['diagnosis'].value_counts().keys(),data['diagnosis'].value_counts(),co
plt.show()
```



In [8]:
```
data.dtypes
```

Out[8]:
```
id                         int64
diagnosis                 object
radius_mean              float64
texture_mean             float64
perimeter_mean           float64
area_mean                float64
smoothness_mean          float64
compactness_mean         float64
concavity_mean           float64
concave points_mean      float64
symmetry_mean            float64
fractal_dimension_mean   float64
radius_se                float64
texture_se               float64
perimeter_se             float64
area_se                  float64
smoothness_se            float64
compactness_se           float64
concavity_se             float64
concave points_se        float64
symmetry_se              float64
fractal_dimension_se     float64
radius_worst             float64
texture_worst            float64
perimeter_worst          float64
area_worst               float64
smoothness_worst         float64
compactness_worst        float64
concavity_worst          float64
concave points_worst     float64
symmetry_worst           float64
fractal_dimension_worst  float64
dtype: object
```

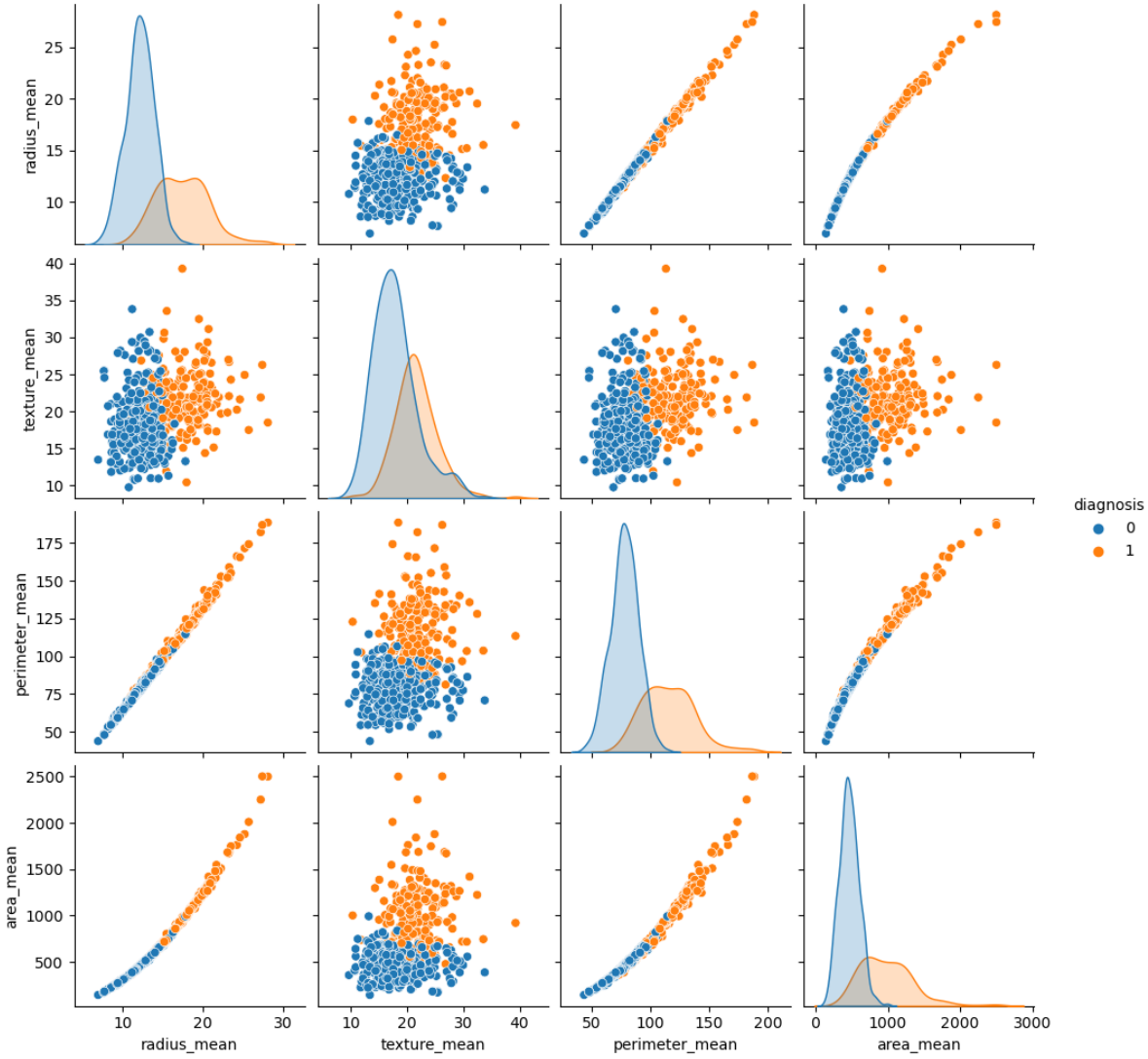
In [9]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
data.iloc[:,1]=encoder.fit_transform(data.iloc[:,1].values)
```

In [10]:
```
data.iloc[:,1]
```

Out[10]:
```
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: int32
```

In [11]:
```
sns.pairplot(data.iloc[:,1:6],hue='diagnosis')
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x26c29435130>
```



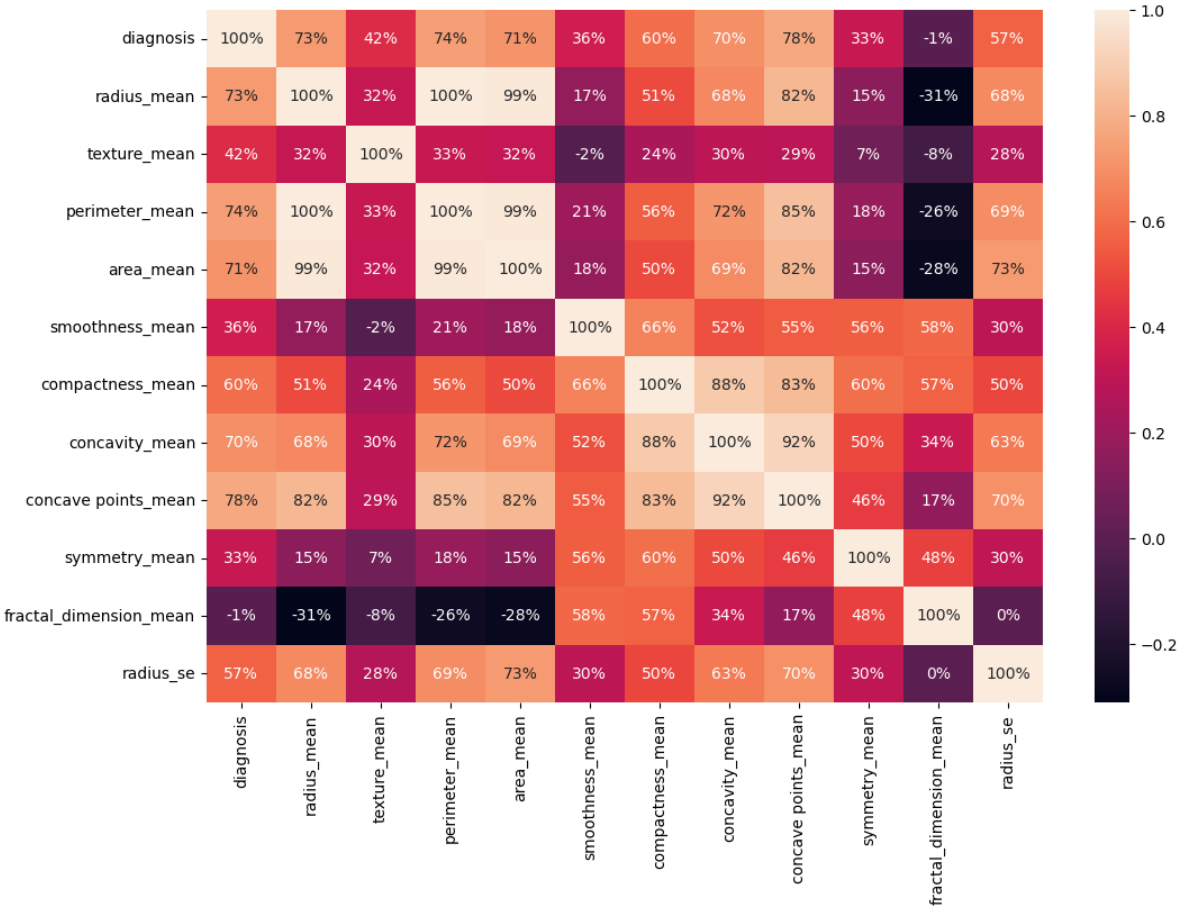
In [12]:
```
data.iloc[:,1:14].corr()
```

Out[12]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoo
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	
fractal_dimension_mean	-0.012838	-0.311631	-0.076437	-0.261477	-0.283110	
radius_se	0.567134	0.679090	0.275869	0.691765	0.732562	
texture_se	-0.008303	-0.097317	0.386358	-0.086761	-0.066280	

In [13]:
```
plt.figure(figsize=(12,8))
sns.heatmap(data.iloc[:,1:13].corr(),annot=True,fmt='.0%')
```

Out[13]:
```
<AxesSubplot:>
```



Machine Learning Model Training

In [14]:
```
X = data.iloc[:,2:31].values
y = data.iloc[:,1].values
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=0)
```

In [15]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train)
model.score(X_train,y_train)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: C
onvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[15]:
```
0.9522613065326633
```

In [16]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train,y_train)
model.score(X_train,y_train)
```

Out[16]:
```
1.0
```

In [17]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train,y_train)
model.score(X_train,y_train)
```

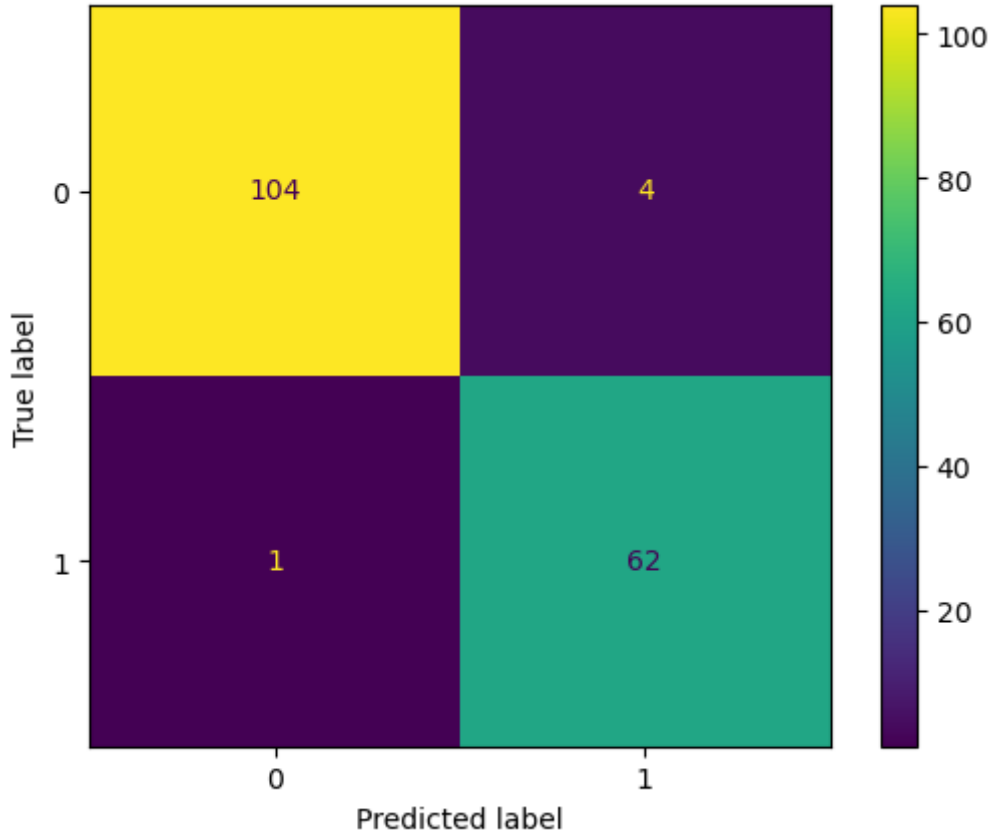
Out[17]:
```
1.0
```

Model Evaluation

In [18]:
```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(model, X_test, y_test)
plt.show()
```

```
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureW
arning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_mat
rix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class method
s: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimato
r.
  warnings.warn(msg, category=FutureWarning)
```



In [19]:
```
#import classification_report
from sklearn.metrics import classification_report
```

In [20]:
```
classifier = RandomForestClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       108
           1       0.94      0.97      0.95        63

    accuracy                           0.96       171
   macro avg       0.96      0.97      0.96       171
weighted avg       0.97      0.96      0.97       171
```

In [21]:
```
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.97      0.89      0.93       108
           1       0.83      0.95      0.89        63

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

Decision Tree Classifier and Random Forest Classifier give high accuracy in breast cancer analysis.