
predict
breast cancer

Breast Cancer Prediction

Imorting Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

Data Understanding

```
cancer_data = pd.read_csv(r"C:\Users\user\Desktop\cancer.csv")

cancer_data
```

```
           id diagnosis  radius_mean  texture_mean  perimeter_mean
area_mean  \
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
```

```
     smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
0            0.11840           0.27760         0.30010
0.14710
1            0.08474           0.07864         0.08690
0.07017
2            0.10960           0.15990         0.19740
0.12790
3            0.14250           0.28390         0.24140
0.10520
4            0.10030           0.13280         0.19800
0.10430
..               ...               ...             ...
...
564          0.11100           0.11590         0.24390
0.13890
565          0.09780           0.10340         0.14400
0.09791
566          0.08455           0.10230         0.09251
0.05302
567          0.11780           0.27700         0.35140
0.15200
568          0.05263           0.04362         0.00000
0.00000

     ...  texture_worst  perimeter_worst  area_worst  smoothness_worst
\
0    ...          17.33           184.60      2019.0           0.16220

1    ...          23.41           158.80      1956.0           0.12380

2    ...          25.53           152.50      1709.0           0.14440

3    ...          26.50            98.87       567.7           0.20980

4    ...          16.67           152.20      1575.0           0.13740

..   ...            ...              ...         ...               ...

564  ...          26.40           166.10      2027.0           0.14100

565  ...          38.25           155.00      1731.0           0.11660

566  ...          34.12           126.70      1124.0           0.11390

567  ...          39.42           184.60      1821.0           0.16500

568  ...          30.37            59.16       268.6           0.08996
```

```
     compactness_worst  concavity_worst  concave points_worst
symmetry_worst  \
0              0.66560           0.7119                0.2654
0.4601
1              0.18660           0.2416                0.1860
0.2750
2              0.42450           0.4504                0.2430
0.3613
3              0.86630           0.6869                0.2575
0.6638
4              0.20500           0.4000                0.1625
0.2364
..                 ...              ...                   ...
...
564            0.21130           0.4107                0.2216
0.2060
565            0.19220           0.3215                0.1628
0.2572
566            0.30940           0.3403                0.1418
0.2218
567            0.86810           0.9387                0.2650
0.4087
568            0.06444           0.0000                0.0000
0.2871

     fractal_dimension_worst  Unnamed: 32
0                    0.11890          NaN
1                    0.08902          NaN
2                    0.08758          NaN
3                    0.17300          NaN
4                    0.07678          NaN
..                       ...          ...
564                  0.07115          NaN
565                  0.06637          NaN
566                  0.07820          NaN
567                  0.12400          NaN
568                  0.07039          NaN

[569 rows x 33 columns]
```

```
# getting the first 5 rows values
cancer_data.head()
```

```
         id diagnosis  radius_mean  texture_mean  perimeter_mean
area_mean  \
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

   smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
0          0.11840           0.27760          0.3001
0.14710
1          0.08474           0.07864          0.0869
0.07017
2          0.10960           0.15990          0.1974
0.12790
3          0.14250           0.28390          0.2414
0.10520
4          0.10030           0.13280          0.1980
0.10430

   ...  texture_worst  perimeter_worst  area_worst
smoothness_worst  \
0  ...          17.33           184.60      2019.0            0.1622

1  ...          23.41           158.80      1956.0            0.1238

2  ...          25.53           152.50      1709.0            0.1444

3  ...          26.50            98.87       567.7            0.2098

4  ...          16.67           152.20      1575.0            0.1374


   compactness_worst  concavity_worst  concave points_worst
symmetry_worst  \
0             0.6656           0.7119                0.2654
0.4601
1             0.1866           0.2416                0.1860
0.2750
2             0.4245           0.4504                0.2430
0.3613
3             0.8663           0.6869                0.2575
0.6638
4             0.2050           0.4000                0.1625
0.2364

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
```

```
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
```

```
# getting the last 5 rows values
cancer_data.tail()
```

```
         id diagnosis  radius_mean  texture_mean  perimeter_mean
area_mean  \
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

     smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
564          0.11100           0.11590         0.24390
0.13890
565          0.09780           0.10340         0.14400
0.09791
566          0.08455           0.10230         0.09251
0.05302
567          0.11780           0.27700         0.35140
0.15200
568          0.05263           0.04362         0.00000
0.00000

     ...  texture_worst  perimeter_worst  area_worst  smoothness_worst
\
564  ...          26.40           166.10      2027.0           0.14100

565  ...          38.25           155.00      1731.0           0.11660

566  ...          34.12           126.70      1124.0           0.11390

567  ...          39.42           184.60      1821.0           0.16500

568  ...          30.37            59.16       268.6           0.08996


     compactness_worst  concavity_worst  concave points_worst
symmetry_worst  \
```

```
564          0.21130          0.4107                0.2216
0.2060
565          0.19220          0.3215                0.1628
0.2572
566          0.30940          0.3403                0.1418
0.2218
567          0.86810          0.9387                0.2650
0.4087
568          0.06444          0.0000                0.0000
0.2871

     fractal_dimension_worst  Unnamed: 32
564                  0.07115          NaN
565                  0.06637          NaN
566                  0.07820          NaN
567                  0.12400          NaN
568                  0.07039          NaN

[5 rows x 33 columns]
```

```
# getting the information about the whole datasets
cancer_data.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
```

```
# to check the null values in each columns
cancer_data.isnull().sum()
```

```
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
```

```
# getting the informarion about the shape of the dataset
cancer_data.shape
```

```
(569, 33)
```

```
# remove the last column
cancer_data = cancer_data.dropna(axis=1)
```

```
cancer_data.shape
```

```
(569, 32)
```

```
# describe the dataset
cancer_data.describe()
```

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

       smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
count       569.000000        569.000000      569.000000
569.000000
mean          0.096360          0.104341        0.088799
0.048919
std           0.014064          0.052813        0.079720
0.038803
min           0.052630          0.019380        0.000000
0.000000
25%           0.086370          0.064920        0.029560
0.020310
50%           0.095870          0.092630        0.061540
0.033500
75%           0.105300          0.130400        0.130700
0.074000
max           0.163400          0.345400        0.426800
```

```
0.201200

       symmetry_mean  ...  radius_worst  texture_worst
perimeter_worst  \
count     569.000000  ...    569.000000     569.000000
569.000000
mean        0.181162  ...     16.269190      25.677223
107.261213
std         0.027414  ...      4.833242       6.146258
33.602542
min         0.106000  ...      7.930000      12.020000
50.410000
25%         0.161900  ...     13.010000      21.080000
84.110000
50%         0.179200  ...     14.970000      25.410000
97.660000
75%         0.195700  ...     18.790000      29.720000
125.400000
max         0.304000  ...     36.040000      49.540000
251.200000

        area_worst  smoothness_worst  compactness_worst
concavity_worst  \
count   569.000000        569.000000         569.000000
569.000000
mean    880.583128          0.132369           0.254265
0.272188
std     569.356993          0.022832           0.157336
0.208624
min     185.200000          0.071170           0.027290
0.000000
25%     515.300000          0.116600           0.147200
0.114500
50%     686.500000          0.131300           0.211900
0.226700
75%    1084.000000          0.146000           0.339100
0.382900
max    4254.000000          0.222600           1.058000
1.252000

       concave points_worst  symmetry_worst  fractal_dimension_worst
count            569.000000      569.000000               569.000000
mean               0.114606        0.290076                 0.083946
std                0.065732        0.061867                 0.018061
min                0.000000        0.156500                 0.055040
25%                0.064930        0.250400                 0.071460
50%                0.099930        0.282200                 0.080040
75%                0.161400        0.317900                 0.092080
max                0.291000        0.663800                 0.207500
```

```
[8 rows x 31 columns]
```

Data Preparation

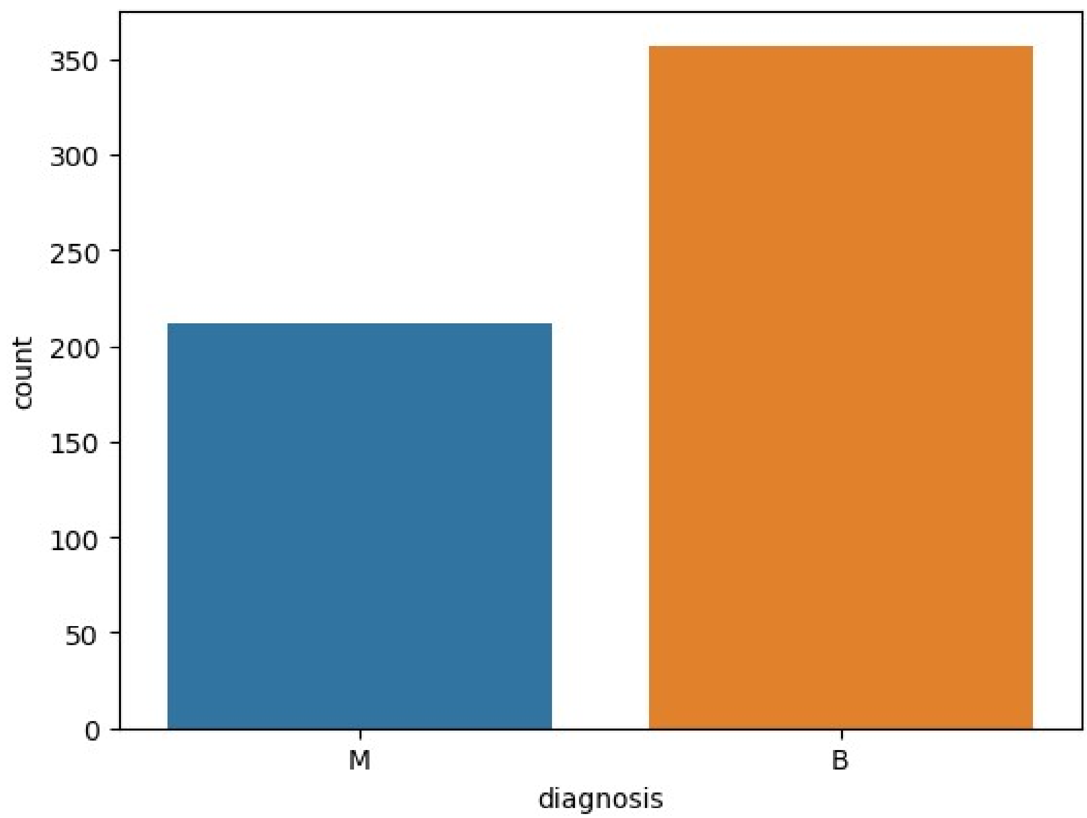
```
# checking the count of Malignant(M) and Benign(B)
cancer_data['diagnosis'].value_counts()
```

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

```
# Plot the bar graph
sns.countplot(cancer_data['diagnosis'], label="counts")
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:xlabel='diagnosis', ylabel='count'>
```



```
# converting the M and B values into 1 and 0 by the LabelEncoder
labelencoder = LabelEncoder()
cancer_data.iloc[:,1] =
labelencoder.fit_transform(cancer_data.iloc[:,1])
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_14616\3778596548.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  cancer_data.iloc[:,1] =
labelencoder.fit_transform(cancer_data.iloc[:,1])
```

```
cancer_data.tail()
```

```
         id  diagnosis  radius_mean  texture_mean  perimeter_mean
area_mean  \
564  926424          1        21.56         22.39          142.00
1479.0
565  926682          1        20.13         28.25          131.20
1261.0
566  926954          1        16.60         28.08          108.30
858.1
567  927241          1        20.60         29.33          140.10
1265.0
568   92751          0         7.76         24.54           47.92
181.0

     smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
564          0.11100           0.11590         0.24390
0.13890
565          0.09780           0.10340         0.14400
0.09791
566          0.08455           0.10230         0.09251
0.05302
567          0.11780           0.27700         0.35140
0.15200
568          0.05263           0.04362         0.00000
0.00000

     ...  radius_worst  texture_worst  perimeter_worst  area_worst  \
564  ...        25.450          26.40           166.10      2027.0
565  ...        23.690          38.25           155.00      1731.0
566  ...        18.980          34.12           126.70      1124.0
567  ...        25.740          39.42           184.60      1821.0
568  ...         9.456          30.37            59.16       268.6
```

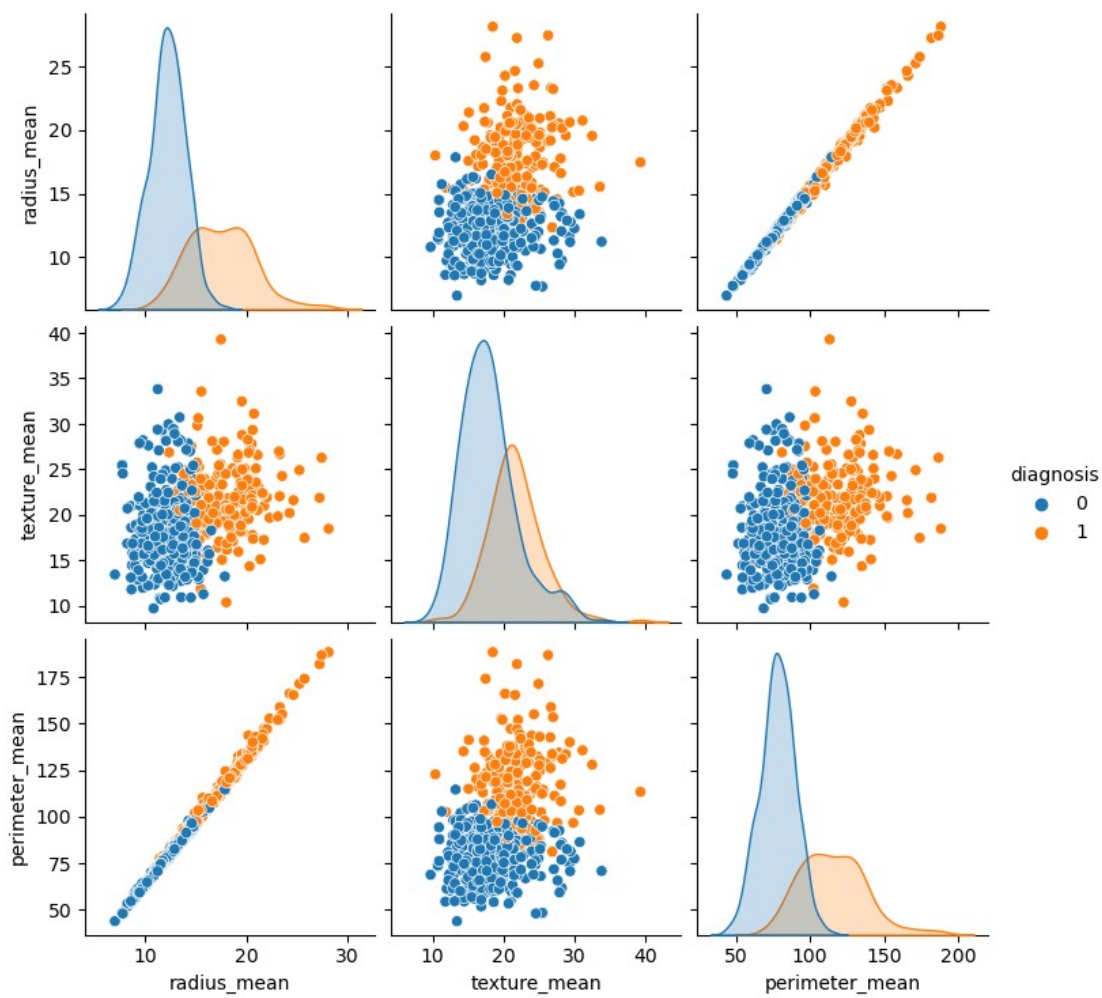
```
     smoothness_worst  compactness_worst  concavity_worst  \
564           0.14100            0.21130           0.4107   
565           0.11660            0.19220           0.3215   
566           0.11390            0.30940           0.3403   
567           0.16500            0.86810           0.9387   
568           0.08996            0.06444           0.0000   

     concave points_worst  symmetry_worst  fractal_dimension_worst  
564                0.2216          0.2060                  0.07115  
565                0.1628          0.2572                  0.06637  
566                0.1418          0.2218                  0.07820  
567                0.2650          0.4087                  0.12400  
568                0.0000          0.2871                  0.07039  

[5 rows x 32 columns]
```

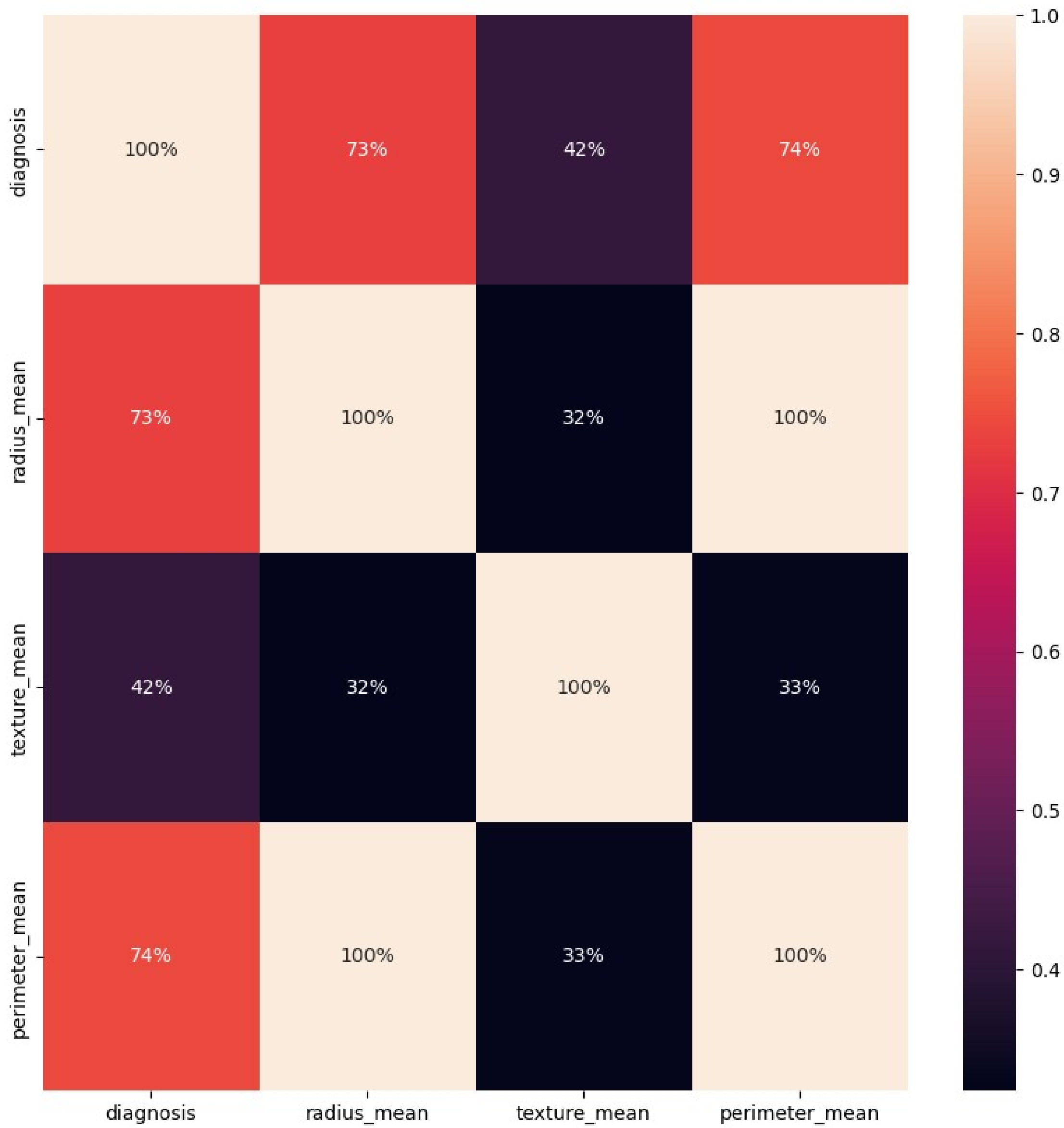
```
sns.pairplot(cancer_data.iloc[:,1:5], hue='diagnosis')
```

```
<seaborn.axisgrid.PairGrid at 0x243b22e7400>
```



```python
# Correlation
plt.figure(figsize = (10, 10))
sns.heatmap(cancer_data.iloc[:,1:5].corr(), annot = True, fmt = ".0%")
```

```
<AxesSubplot:>
```



Now spliting the dataset into Features and Targets

```python
X = cancer_data.drop(columns = 'diagnosis', axis = 1)
Y = cancer_data['diagnosis']
```

```python
print(X)
```

```
           id  radius_mean  texture_mean  perimeter_mean  area_mean  \
0      842302        17.99         10.38          122.80     1001.0
```

```
1      842517        20.57         17.77          132.90     1326.0
2    84300903        19.69         21.25          130.00     1203.0
3    84348301        11.42         20.38           77.58      386.1
4    84358402        20.29         14.34          135.10     1297.0
..        ...          ...           ...             ...        ...
564    926424        21.56         22.39          142.00     1479.0
565    926682        20.13         28.25          131.20     1261.0
566    926954        16.60         28.08          108.30      858.1
567    927241        20.60         29.33          140.10     1265.0
568     92751         7.76         24.54           47.92      181.0

     smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
0            0.11840           0.27760         0.30010
0.14710
1            0.08474           0.07864         0.08690
0.07017
2            0.10960           0.15990         0.19740
0.12790
3            0.14250           0.28390         0.24140
0.10520
4            0.10030           0.13280         0.19800
0.10430
..               ...               ...             ...
...
564          0.11100           0.11590         0.24390
0.13890
565          0.09780           0.10340         0.14400
0.09791
566          0.08455           0.10230         0.09251
0.05302
567          0.11780           0.27700         0.35140
0.15200
568          0.05263           0.04362         0.00000
0.00000

     symmetry_mean  ...  radius_worst  texture_worst  perimeter_worst
\
0           0.2419  ...        25.380          17.33           184.60

1           0.1812  ...        24.990          23.41           158.80

2           0.2069  ...        23.570          25.53           152.50

3           0.2597  ...        14.910          26.50            98.87

4           0.1809  ...        22.540          16.67           152.20

..             ...  ...           ...            ...              ...
```

564	0.1726	...	25.450	26.40	166.10
565	0.1752	...	23.690	38.25	155.00
566	0.1590	...	18.980	34.12	126.70
567	0.2397	...	25.740	39.42	184.60
568	0.1587	...	9.456	30.37	59.16

\	area_worst	smoothness_worst	compactness_worst	concavity_worst
0	2019.0	0.16220	0.66560	0.7119
1	1956.0	0.12380	0.18660	0.2416
2	1709.0	0.14440	0.42450	0.4504
3	567.7	0.20980	0.86630	0.6869
4	1575.0	0.13740	0.20500	0.4000
..	...	...	...	...
564	2027.0	0.14100	0.21130	0.4107
565	1731.0	0.11660	0.19220	0.3215
566	1124.0	0.11390	0.30940	0.3403
567	1821.0	0.16500	0.86810	0.9387
568	268.6	0.08996	0.06444	0.0000

	concave points_worst	symmetry_worst	fractal_dimension_worst
0	0.2654	0.4601	0.11890
1	0.1860	0.2750	0.08902
2	0.2430	0.3613	0.08758
3	0.2575	0.6638	0.17300
4	0.1625	0.2364	0.07678
..	...	...	...
564	0.2216	0.2060	0.07115
565	0.1628	0.2572	0.06637
566	0.1418	0.2218	0.07820
567	0.2650	0.4087	0.12400

```
568                  0.0000          0.2871                      0.07039

[569 rows x 31 columns]
```

```
print(Y)
```

```
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
```

Now spliting the dataset into trainig and testing dataset

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=0, stratify=Y)

# feature Scaling
X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)

# function for models
def models(X_train, Y_train):

    # Logistic Regression
    log = LogisticRegression(random_state=0)
    log.fit(X_train, Y_train)

    # Decision tree classifier
    tree = DecisionTreeClassifier(random_state=0, criterion="entropy")
    tree.fit(X_train, Y_train)

    # Random Forest
    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

model = models(X_train, Y_train)

[0] Logistic Regression Accuracy:  0.9934065934065934
[1] Decision tree Accuracy:  1.0
[2] Random forest Accuracy:  0.9956043956043956

#  Testing the Models

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        72
           1       0.97      0.93      0.95        42

    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       0.97      0.96      0.97        72
           1       0.93      0.95      0.94        42

    accuracy                           0.96       114
   macro avg       0.95      0.96      0.95       114
weighted avg       0.96      0.96      0.96       114

Accuracy:  0.956140350877193
Model 2
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        72
           1       0.95      0.95      0.95        42

    accuracy                           0.96       114
   macro avg       0.96      0.96      0.96       114
weighted avg       0.96      0.96      0.96       114
```

```
Accuracy:  0.9649122807017544
```

As we can see that the Random Forest classifier and Logistic Regression Accuracy are same so we can use any model for prediction

we will take Random Forest Classifier model

```
pred = model[2].predict(X_test)
print("Predicted values")
print(pred)
print("Actual values")
Y_test = np.asarray(Y_test)
print(Y_test)
```

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