In [13]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew
```

In [15]:

```
df=pd.read_csv("breast_cancer.csv")
```

In [16]:

```
df.head()
```

Out[16]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	..
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	..
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	..
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	..
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	..
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	..

5 rows × 31 columns

In [17]:

```
df.tail()
```

Out[17]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884

5 rows × 31 columns

In [18]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  outcome                  569 non-null    int64  
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

In [19]:
```
df.describe()
```

Out[19]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200

8 rows × 31 columns

In [20]:
```
df.isnull()
```

Out[20]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
564	False	False	False	False	False	False	False	False	False	False
565	False	False	False	False	False	False	False	False	False	False
566	False	False	False	False	False	False	False	False	False	False
567	False	False	False	False	False	False	False	False	False	False
568	False	False	False	False	False	False	False	False	False	False

569 rows × 31 columns

In [21]:

```
df.isnull().sum()
```

Out[21]:

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
outcome                    0
dtype: int64
```

In [23]:

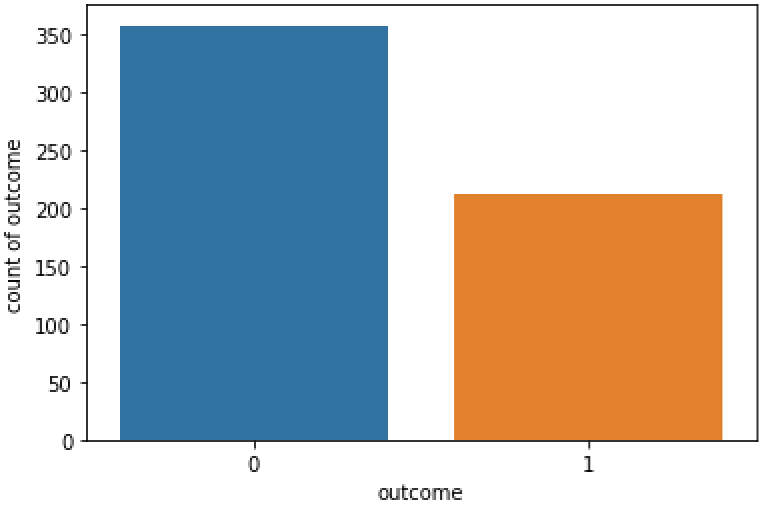
```
df["outcome"].value_counts()
```

Out[23]:

```
0    357
1    212
Name: outcome, dtype: int64
```

In [24]:
```
sns.countplot(df["outcome"])
plt.xlabel("outcome")
plt.ylabel("count of outcome")
plt.show()
```

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



In [26]:
```
X=df.iloc[:, :-1]
y=df.iloc[:, -1]
```

In [27]:
```
X.shape
```

Out[27]:
```
(569, 30)
```

In [28]:
```
y.shape
```

Out[28]:
```
(569,)
```

In [38]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=99)
```

In [39]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error,r2_score
forest_model=RandomForestRegressor(n_estimators=750,max_depth=4,max_leaf_nodes=500,rando
```

In [40]:
```
forest_model.fit(X_train, y_train)
```

Out[40]:
```
RandomForestRegressor(max_depth=4, max_leaf_nodes=500, n_estimators=750,
                      random_state=1)
```

In [43]:
```
forest_model.feature_importances_
```

Out[43]:
```
array([0.00268249, 0.01581475, 0.00211393, 0.01592123, 0.00323533,
       0.00130309, 0.00458619, 0.08535125, 0.00101596, 0.00209099,
       0.0052539 , 0.00149769, 0.00382338, 0.02354548, 0.00100886,
       0.0017764 , 0.00406216, 0.00228771, 0.0011848 , 0.00530255,
       0.02044271, 0.01611963, 0.25185632, 0.07679379, 0.00688009,
       0.0023232 , 0.00808895, 0.42939713, 0.00254625, 0.00169379])
```

df.columns

In [44]:
```
df.columns
```

```
Out[44]: Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
               'mean smoothness', 'mean compactness', 'mean concavity',
               'mean concave points', 'mean symmetry', 'mean fractal dimension',
               'radius error', 'texture error', 'perimeter error', 'area error',
               'smoothness error', 'compactness error', 'concavity error',
               'concave points error', 'symmetry error', 'fractal dimension error',
               'worst radius', 'worst texture', 'worst perimeter', 'worst area',
               'worst smoothness', 'worst compactness', 'worst concavity',
               'worst concave points', 'worst symmetry', 'worst fractal dimension',
               'outcome'],
              dtype='object')
```

```
In [49]: power_preds=forest_model.predict(X_test)
```

```
In [51]: power_preds
```

```
Out[51]: array([0.01174317, 0.00507125, 0.0405616 , 0.00507125, 0.00515642,
                0.99894507, 0.75169602, 0.9989984 , 0.00537539, 0.01106544,
                0.9989984 , 0.06463776, 0.87146066, 0.00687874, 0.00507125,
                0.96111096, 0.9989984 , 0.29701253, 0.02709731, 0.0058881 ,
                0.28437355, 0.00507125, 0.00607125, 0.00507125, 0.9989984 ,
                0.00578553, 0.00507125, 0.00507125, 0.00507125, 0.00507125,
                0.9989984 , 0.00507125, 0.0088516 , 0.00607125, 0.00576466,
                0.02980223, 0.00507125, 0.01346395, 0.00507125, 0.88928831,
                0.02108799, 0.00517381, 0.06143323, 0.85816726, 0.00520633,
                0.04370317, 0.02119105, 0.00510377, 0.99633173, 0.01442264,
                0.35763056, 0.97624259, 0.56402574, 0.00507125, 0.99638228,
                0.00507125, 0.00592062, 0.00507125, 0.00507125, 0.00507125,
                0.9140906 , 0.9979984 , 0.9989984 , 0.00507125, 0.99533173,
                0.00507125, 0.00507125, 0.00684142, 0.0050654 , 0.9989984 ,
                0.07283112, 0.9989984 , 0.96468437, 0.07879092, 0.97826305,
                0.9989984 , 0.18051927, 0.9989984 , 0.96948153, 0.04340454,
                0.00507125, 0.00507125, 0.00507125, 0.08939125, 0.9989984 ,
                0.00510377, 0.00915486, 0.00510377, 0.33110267, 0.00507125,
                0.9989984 , 0.02788602, 0.00507125, 0.09251526, 0.67496218,
                0.9989984 , 0.9989984 , 0.32203922, 0.00507125, 0.91574758,
                0.00607125, 0.00507125, 0.1387028 , 0.41724875, 0.14574524,
                0.00507125, 0.60220517, 0.00507125, 0.00507125, 0.00507125,
                0.47925458, 0.94709742, 0.00520633, 0.00520633, 0.00640458,
                0.00510377, 0.92696388, 0.00520633, 0.00694294, 0.02653579,
                0.00507125, 0.0427797 , 0.99766507, 0.0547102 , 0.22950025,
                0.00507125, 0.82774075, 0.00607125, 0.00641641, 0.99633173,
                0.00507125, 0.90658357, 0.00507125, 0.00517381, 0.07333024,
                0.01211246, 0.0050654 , 0.99639234, 0.01590958, 0.01153497,
                0.00507125, 0.00517381, 0.00507125])
```

```
In [70]: from sklearn.metrics import mean_absolute_error, mean_squared_error, explained_variance_
```

```
In [74]: forest_model.score(X_train,y_train)
```

```
Out[74]: 0.9543176382164082
```

```
In [76]: ytrain_preds=forest_model.predict(X_train)
```

```
In [77]: mean_absolute_error(y_train,ytrain_preds)
```

```
Out[77]: 0.041683460079334614
```

mean_squared_error(y_train,ytrain_preds)

```
In [78]: mean_squared_error(y_train,ytrain_preds)
```

```
Out[78]: 0.010910843798342484
```

```
In [ ]:
```