In [3]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [4]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:

```
df=pd.read_csv("https://raw.githubusercontent.com/ingledarshan/AIML-B2/main/data.csv")
```

In [6]:

```
df
```

Out[6]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.
...	...	...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.

569 rows × 33 columns

In [7]:

```
df.head()
```

Out[7]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1

5 rows × 33 columns

In [8]:

```
df.tail()
```

Out[8]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00

5 rows × 33 columns

In [9]:

```
df.columns
```

Out[9]:

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

In [10]:

```
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
 32  Unnamed: 32               0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 144.5+ KB
```

In [11]:

```
df['Unnamed: 32']
```

Out[11]:

```
0     NaN
1     NaN
2     NaN
3     NaN
4     NaN
       ..
564   NaN
565   NaN
566   NaN
567   NaN
568   NaN
Name: Unnamed: 32, Length: 569, dtype: float64
```

In [12]:

```
df=df.drop("Unnamed: 32",axis=1)
```

In [13]:

```
df.head()
```

Out[13]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1

5 rows × 32 columns

In [14]:

```
df.columns
```

Out[14]:

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

In [15]:

```
df=df.drop("id",axis=1)
#df.drop('id',axis=1,inplace=True)
```

In [16]:

```
df
```

Out[16]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poi
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	
...	...	...	...	...	...	...	...	...	
564	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	
565	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	
566	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	
567	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	
568	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	

569 rows × 31 columns

In [17]:

```
df.columns
```

Out[17]:

```
Index(['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

In [18]:

```
type(df.columns)
```

Out[18]:

```
pandas.core.indexes.base.Index
```

In [19]:

```
l=list(df.columns)
print(l)
```

```
['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean',
'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_mean',
'fractal_dimension_mean', 'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se', '
compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se', 'fractal_dimension_se',
'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smoothness_worst',
'compactness_worst', 'concavity_worst', 'concave points_worst', 'symmetry_worst',
'fractal_dimension_worst']
```

In [20]:

```
features_mean=l[1:11]

features_se=l[11:21]

features_worst=l[21:]
```

In [21]:

```
print(features_mean)
```

```
['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean',
'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_mean',
'fractal_dimension_mean']
```

In [22]:

```
df.head(2)
```

Out[22]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	c points
0	M	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	
1	M	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	

2 rows × 31 columns

In [23]:

```
df['diagnosis'].unique()
#M=Maligant cancer,B=Benign cancer
```

Out[23]:

```
array(['M', 'B'], dtype=object)
```

In [24]:

```
df['diagnosis'].value_counts()
```

Out[24]:

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

In [25]:

```
df.shape
#rows,columns
```

Out[25]:

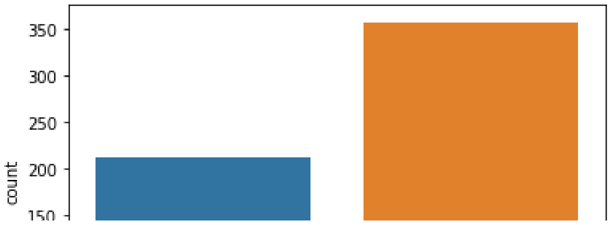
```
(569, 31)
```

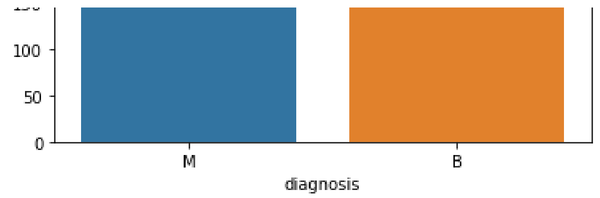
In [26]:

```
sns.countplot(df['diagnosis'],label="Count")
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x9c24bf0>
```





explore the data

In [27]:

```
df.describe()
#summary of all the numeric columns
```

Out[27]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concav points_mea
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.00000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.04891
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.03880
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.00000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.02031
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.03350
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.07400
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.20120

8 rows × 30 columns

In [28]:

```
len(df.columns)
```

Out[28]:

```
31
```

In [29]:

```
#correlation plot(+ve,_ve,No)[-1,+1]
```

In [30]:

```
corr=df.corr()
corr
```

Out[30]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
radius_mean	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676
texture_mean	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302
perimeter_mean	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716
area_mean	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685
smoothness_mean	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521
compactness_mean	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883
concavity_mean	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000
concave points_mean	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921
symmetry_mean	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
symmetry_mean	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.50
fractal_dimension_mean	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.33
radius_se	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631
texture_se	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.07
perimeter_se	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.66
area_se	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653	0.61
smoothness_se	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.09
compactness_se	0.206000	0.191975	0.250744	0.212583	0.318943	0.738722	0.67
concavity_se	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691
concave points_se	0.376169	0.163851	0.407217	0.372320	0.380676	0.642262	0.68
symmetry_se	-0.104321	0.009127	-0.081629	-0.072497	0.200774	0.229977	0.17
fractal_dimension_se	-0.042641	0.054458	-0.005523	-0.019887	0.283607	0.507318	0.44
radius_worst	0.969539	0.352573	0.969476	0.962746	0.213120	0.535315	0.68
texture_worst	0.297008	0.912045	0.303038	0.287489	0.036072	0.248133	0.29
perimeter_worst	0.965137	0.358040	0.970387	0.959120	0.238853	0.590210	0.72
area_worst	0.941082	0.343546	0.941550	0.959213	0.206718	0.509604	0.67
smoothness_worst	0.119616	0.077503	0.150549	0.123523	0.805324	0.565541	0.44
compactness_worst	0.413463	0.277830	0.455774	0.390410	0.472468	0.865809	0.75
concavity_worst	0.526911	0.301025	0.563879	0.512606	0.434926	0.816275	0.88
concave points_worst	0.744214	0.295316	0.771241	0.722017	0.503053	0.815573	0.861
symmetry_worst	0.163953	0.105008	0.189115	0.143570	0.394309	0.510223	0.40
fractal_dimension_worst	0.007066	0.119205	0.051019	0.003738	0.499316	0.687382	0.51

30 rows × 30 columns

In [31]:

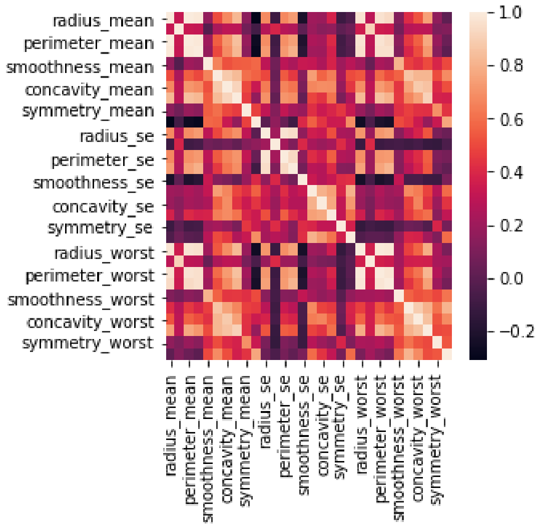
```
corr.shape
```

Out[31]:

```
(30, 30)
```

In [32]:

```
plt.figure(figsize=(4,4))
sns.heatmap(corr);
```



In [33]:

```
df.head()
```

Out[33]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	c points
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 31 columns

In [34]:

```
df['diagnosis']=df['diagnosis'].map({'M':1,'B':0})
```

In [35]:

```
df.head()
```

Out[35]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	c points
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 31 columns

In [36]:

```
df['diagnosis'].unique()
```

Out[36]:

```
array([1, 0], dtype=int64)
```

In [37]:

```
x = df.drop('diagnosis', axis=1)
x.head()
```

Out[37]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	sy
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

5 rows × 30 columns

In [38]:

```
y=df['diagnosis']
```

```
y=df['diagnosis']
y.head()
```

Out[38]:

```
0    1
1    1
2    1
3    1
4    1
Name: diagnosis, dtype: int64
```

In [39]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [40]:

```
x_train.shape
```

Out[40]:

```
(398, 30)
```

In [41]:

```
x_test.shape
```

Out[41]:

```
(171, 30)
```

In [42]:

```
y_train.shape
```

Out[42]:

```
(398,)
```

In [43]:

```
y_test.shape
```

Out[43]:

```
(171,)
```

In [44]:

```
x_train.head(1)
```

Out[44]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
465	13.24	20.13	86.87	542.9	0.08284	0.1223	0.101	0.02833

1 rows × 30 columns

In [45]:

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x_train=ss.fit_transform(x_train)
x_test=ss.transform(x_test)
```

In [46]:

```
x_train
```

Out[46]:

```
array([[-0.22629234,  0.18547336, -0.1853857 , ...,  0.3250308 ,
        -0.09925287,  2.2706719 ],
       [-0.29517323,  1.33784151, -0.36281398, ..., -0.85545553,
        -0.98919743, -1.19725023],
       [ 0.48834693, -1.85421826,  0.47809611, ...,  0.4716463 ,
        -0.57477055, -0.13992279],
       ...,
       [ 1.42684911, -0.09570446,  1.33608264, ...,  0.11050916,
        -0.10585728, -1.0573921 ],
       [ 0.12959228,  0.49200329,  0.12199005, ...,  0.12131241,
         0.18969017, -0.88676519],
       [ 2.10704794, -0.97380899,  2.08161462, ...,  1.26491336,
        -0.28912972,  0.13140198]])
```

Machine learning models

1.logistic Regression

In [47]:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train, y_train)
```

Out[47]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [48]:

```
y_pred=lr.predict(x_test)
```

In [49]:

```
y_pred
```

Out[49]:

```
array([1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1], dtype=int64)
```

In [50]:

```
y_test
```

Out[50]:

```
75     1
561    0
454    0
241    0
402    0
```

```
       ..
39     1
533    1
382    0
108    1
274    1
Name: diagnosis, Length: 171, dtype: int64
```

In [51]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

```
0.9766081871345029
```

In [52]:

```
lr_acc = accuracy_score(y_test, y_pred)
print(lr_acc)
```

```
0.9766081871345029
```

In [53]:

```
results = pd.DataFrame()
results
```

Out[53]:

—

In [57]:

```
tempResults = pd.DataFrame({'Algorithm':['LogisticRegressionMethod'], 'Accuracy':[lr_acc]})
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[57]:

	Algorithm	Accuracy
0	LogisticRegressionMethod	0.976608

Decision Tree Classifier

In [58]:

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(x_train, y_train)
```

Out[58]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [59]:

```
y_pred = dtc.predict(x_test)
y_pred
```

Out[59]:

```
array([1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1], dtype=int64)
```

In [60]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

```
0.9415204678362573
```

In [61]:

```
dtc_acc = accuracy_score(y_test, y_pred)
print(dtc_acc)
```

```
0.9415204678362573
```

In [62]:

```
tempResults = pd.DataFrame({'Algorithm':['DecisionTreeClassifierMethod'], 'Accuracy':[dtc_acc]})
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[62]:

	Algorithm	Accuracy
0	LogisticRegressionMethod	0.976608
0	DecisionTreeClassifierMethod	0.941520

In [63]:

```
#Random Forest Classifier
```

In [64]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
```

Out[64]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [65]:

```
y_pred = rfc.predict(x_test)
y_pred
```

Out[65]:

```
array([1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1], dtype=int64)
```

In [66]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

```
0.9649122807017544
```

In [67]:

```
rfc_acc = accuracy_score(y_test, y_pred)
print(rfc_acc)
```

```
0.9649122807017544
```

In [68]:

```
tempResults = pd.DataFrame({'Algorithm':['RandomForestClassifierMethod'], 'Accuracy':[rfc_acc]})
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[68]:

	Algorithm	Accuracy
0	LogisticRegressionMethod	0.976608
0	DecisionTreeClassifierMethod	0.941520
0	RandomForestClassifierMethod	0.964912

In [69]:

```
#Support Vector Classifier
```

In [70]:

```
from sklearn import svm
svc = svm.SVC()
svc.fit(x_train,y_train)
```

Out[70]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [71]:

```
y_pred = svc.predict(x_test)
y_pred
```

Out[71]:

```
array([1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1], dtype=int64)
```

In [72]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

```
0.9766081871345029
```

In [73]:

```
svc_acc = accuracy_score(y_test, y_pred)
print(svc_acc)
```

```
0.9766081871345029
```

In [74]:

```
tempResults = pd.DataFrame({'Algorithm':['SupportVectorClassifierMethod'], 'Accuracy':[svc_acc]})
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[74]:

	Algorithm	Accuracy
0	LogisticRegressionMethod	0.976608
0	DecisionTreeClassifierMethod	0.941520
0	RandomForestClassifierMethod	0.964912
0	SupportVectorClassifierMethod	0.976608

In [ ]:

In [ ]: