RAMCO INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MINI PROJECT

PROJECT TITLE: Breast Cancer Detection Using Classification Algorithms

NAME:VENKATESH M

REG NO:953622243114

About the Notebook:

In this notebook, I tried to use different ways of machine learning in diagnosing breast cancer. First, let's find out what breast cancer is, then let's start examining the data. I hope you enjoy reading it.

Breast cancer is a disease in which cells in the breast grow out of control. There are different kinds of breast cancer. The kind of breast cancer depends on which cells in the breast turn into cancer.

PROGRAM:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns # data visualization library
import matplotlib.pyplot as plt

#Machine Learning Libraries
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

#import warnings library
```

```
import warnings
# ignore all warnings
warnings.filterwarnings('ignore')
data = pd.read_csv("breast-cancer-wisconsin-data.csv")
data.isna().sum()
```

OUTPUT:

```
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

PROGRAM:

```
data.drop(["id","Unnamed: 32"],axis=1,inplace=True)
data.isna().sum()
```

OUTPUT:

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

PROGRAM:

```
data.head()
```

OUTPUT:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
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

PROGRAM:

```
col = data.columns       # .columns gives columns names in data
print(col)
```

OUTPUT:

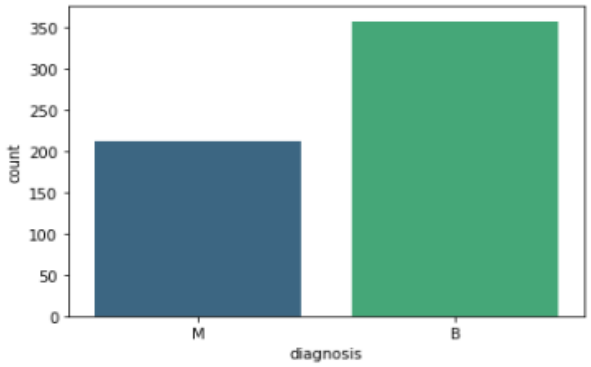
```
Index(['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       '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')
```

PROGRAM:

```
n = data.diagnosis
B, M = n.value_counts()
ax = sns.countplot(n,label="Count",palette="viridis")
print('Number of Benign: ',B)
print('Number of Malignant : ',M)
```

OUTPUT:

```
Number of Benign:  357
Number of Malignant :  212
```



PROGRAM:

```
m = data.drop("diagnosis",axis=1)
m.describe()
```

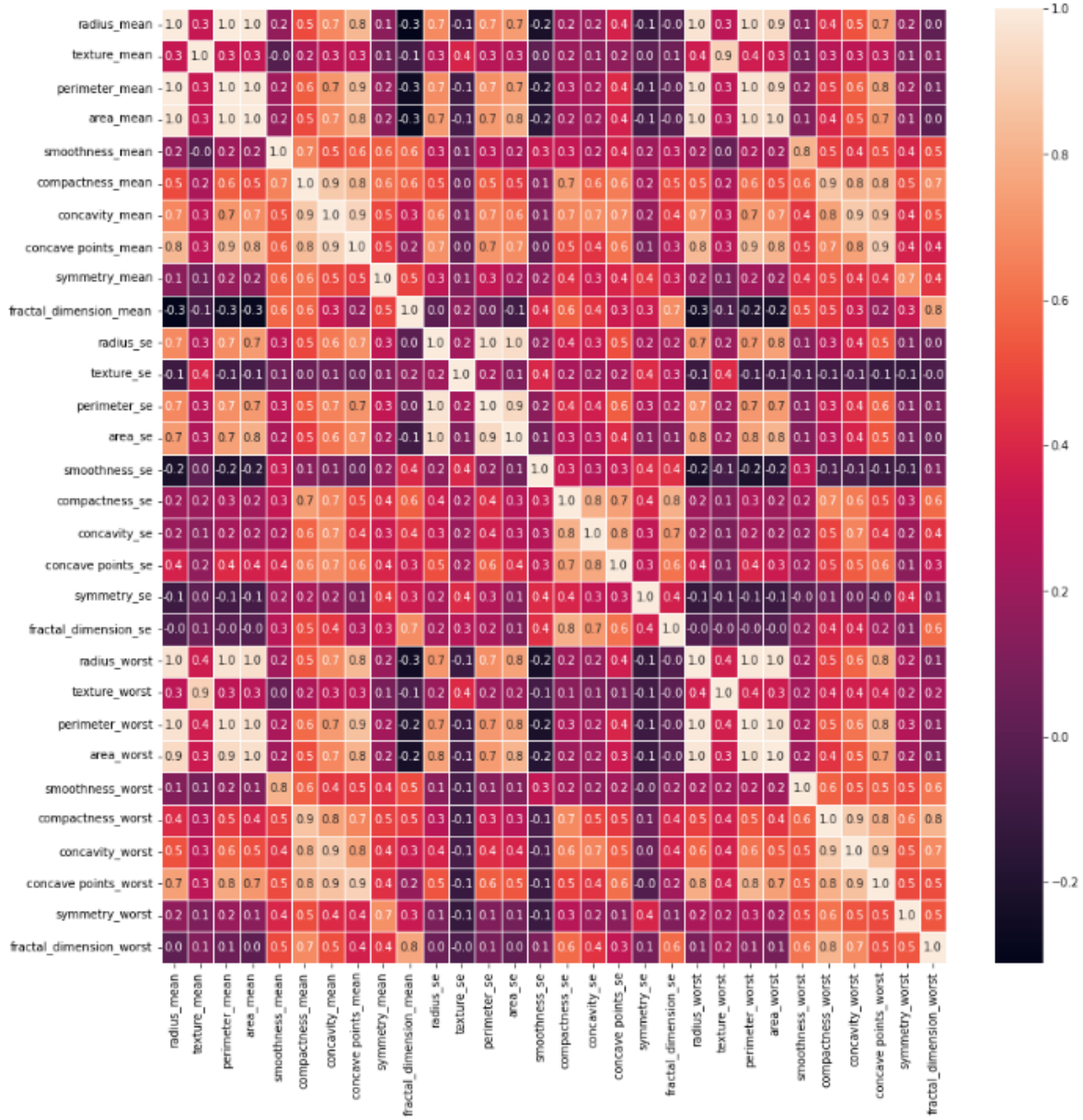
OUTPUT:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	co po
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	56
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.0
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.0
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.0
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.0
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.0
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.0
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.2

PROGRAM:

```
f,ax = plt.subplots(figsize=(15, 15))
sns.heatmap(m.corr(), annot=True,linewidths=.5, fmt= '.1f',ax=ax)
```

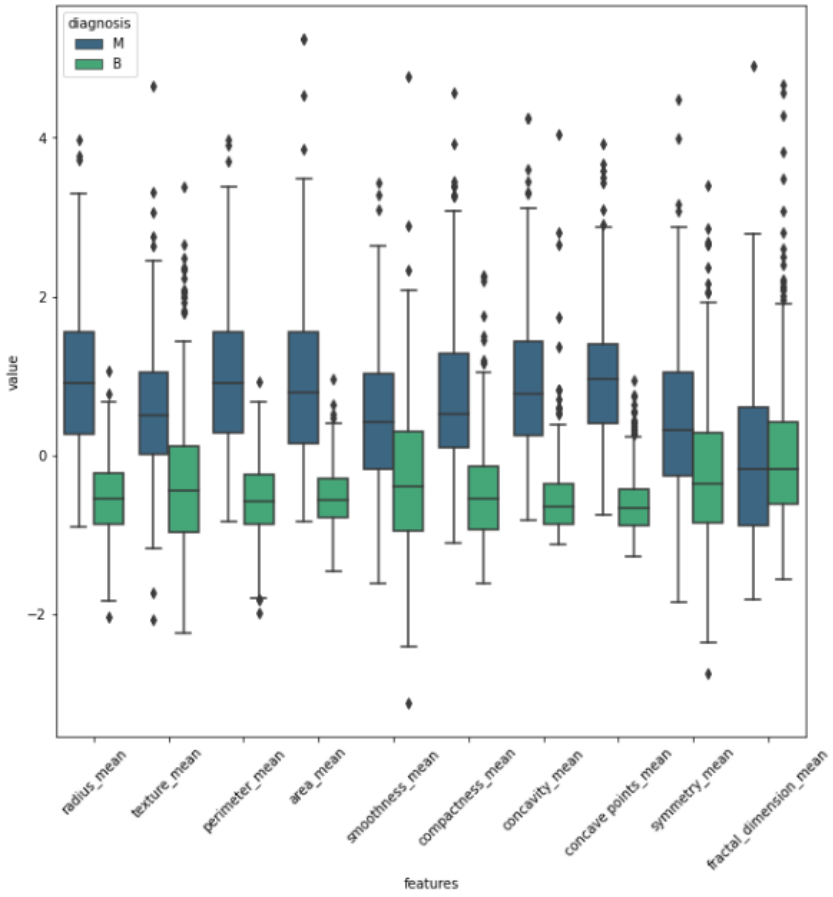
OUTPUT:



PROGRAM:

```
data_dia = n

data_m = m

data_n_2 = (data_m - data_m.mean()) / (data_m.std())              # standardization

data_p = pd.concat([n,data_n_2.iloc[:,0:10]],axis=1)

data_p = pd.melt(data_p,id_vars="diagnosis",
                    var_name="features",
                    value_name='value')

plt.figure(figsize=(10,10))

sns.boxplot(x="features", y="value", hue="diagnosis",
data=data_p,palette="viridis")

plt.xticks(rotation=45)
```

OUTPUT:



PROGRAM:

```
data.diagnosis = [1 if each == "M" else 0 for each in data.diagnosis]
data.head()
```

OUTPUT:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
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

PROGRAM:

```
y = data.diagnosis.values
x_data = data.drop(["diagnosis"],axis=1)
x = (x_data - np.min(x_data))/(np.max(x_data)-np.min(x_data))
x.head()
```

OUTPUT:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_me
0	0.521037	0.022658	0.545989	0.363733	0.593753	0.792037	0.703140	0.731113
1	0.643144	0.272574	0.615783	0.501591	0.289880	0.181768	0.203608	0.348757
2	0.601496	0.390260	0.595743	0.449417	0.514309	0.431017	0.462512	0.635686
3	0.210090	0.360839	0.233501	0.102906	0.811321	0.811361	0.565604	0.522863
4	0.629893	0.156578	0.630986	0.489290	0.430351	0.347893	0.463918	0.518390

PROGRAM:

Logistic Regression Classifier:

```
lr = LogisticRegression(random_state = 1) #We are building our model
lr.fit(x_train,y_train) #We are training our model
print("Print accuracy of Logistic Regression Classifier:
{}".format(lr.score(x_test,y_test)))
lr_acc_score = lr.score(x_test,y_test)
```

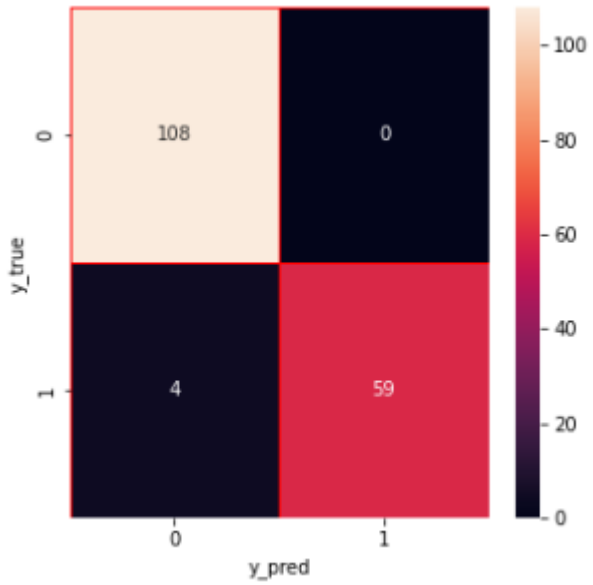
OUTPUT:

Print accuracy of Logistic Regression Classifier: 0.9766081871345029

PROGRAM:

```
y_pred = lr.predict(x_test)

y_true = y_test

cm = confusion_matrix(y_true, y_pred)

#visualize

f, ax = plt.subplots(figsize=(5,5))

sns.heatmap(cm,annot = True, linewidths=0.5,linecolor="red",fmt = ".0f",ax=ax)

plt.xlabel("y_pred")

plt.ylabel("y_true")

plt.show()
```

OUTPUT:



PROGRAM:

K Neighbors Classifier:

```
knn = KNeighborsClassifier(n_neighbors=10) #We are building our model
knn.fit(x_train,y_train) #We are training our model
print("Print accuracy of K Neighbors Classifier algo:
{}".format(knn.score(x_test,y_test)))
knn_acc_score = knn.score(x_test,y_test)
```

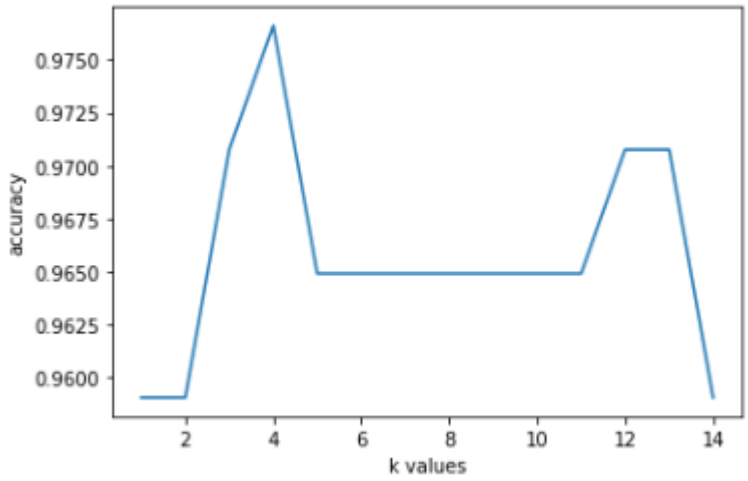
OUTPUT:

Print accuracy of K Neighbors Classifier algo: 0.9649122807017544

PROGRAM:

```
score_list = []
for each in range(1,15):
    knn2 = KNeighborsClassifier(n_neighbors = each)
    knn2.fit(x_train,y_train)
    score_list.append(knn2.score(x_test,y_test))
    #visualize
plt.plot(range(1,15),score_list)
plt.xlabel("k values")
plt.ylabel("accuracy")
plt.savefig('plot')
plt.show()
```

OUTPUT:



PROGRAM:

Naive Bayes:

```
nb = GaussianNB()       #We are building our model
nb.fit(x_train,y_train) #We are training our model
print("Print accuracy of naive bayes algo: {}".format(nb.score(x_test,y_test)))
nb_acc_score = nb.score(x_test,y_test)
```

OUTPUT:

Print accuracy of naive bayes algo: 0.935672514619883

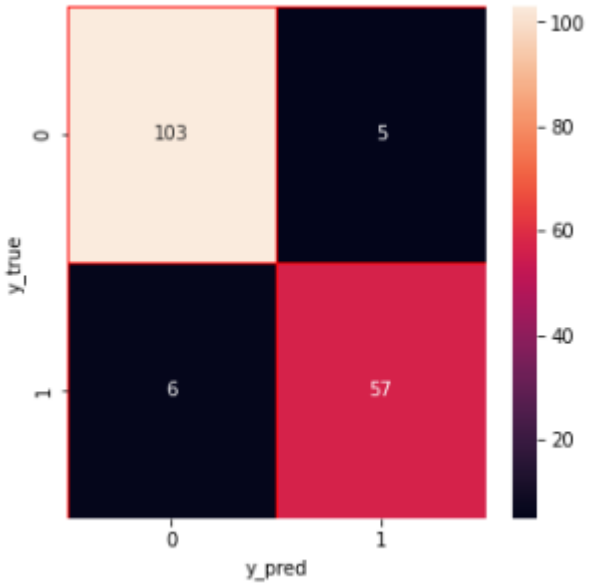
PROGRAM:

```
y_pred = nb.predict(x_test)
y_true = y_test


cm = confusion_matrix(y_true, y_pred)
#visualize
f, ax = plt.subplots(figsize=(5,5))
sns.heatmap(cm,annot = True, linewidths=0.5,linecolor="red",fmt =
".0f",ax=ax)
plt.xlabel("y_pred")
```

plt.ylabel("y_true")

plt.show()

OUTPUT:



PROGRAM:

SVM:

svm = SVC() #We are building our model

svm.fit(x_train,y_train) #We are training our model

print("Print accuracy of svm algo: ",svm.score(x_test,y_test))

svm_acc_score = svm.score(x_test,y_test)

OUTPUT:

Print accuracy of svm algo: 0.9824561403508771

PROGRAM:

y_pred = svm.predict(x_test)

y_true = y_test

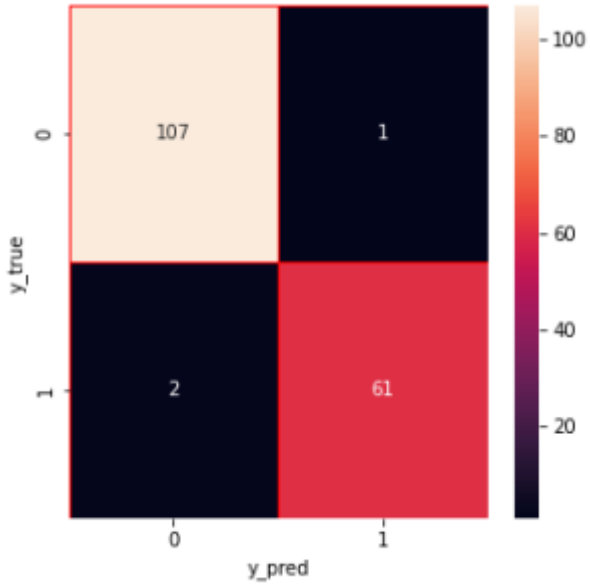
cm = confusion_matrix(y_true, y_pred)

#visualize

f, ax = plt.subplots(figsize=(5,5))

```
sns.heatmap(cm,annot = True, linewidths=0.5,linecolor="red",fmt =
".0f",ax=ax)

plt.xlabel("y_pred")

plt.ylabel("y_true")

plt.show()
```

OUTPUT:



PROGRAM:

Decision Tree Classifier:

```
dt = DecisionTreeClassifier(random_state = 1)   #We are building our model

dt.fit(x_train,y_train)       #We are training our model

print("Print accuracy of Decision Tree Classifier algo: ",dt.score(x_test,y_test))

dt_acc_score = dt.score(x_test,y_test)
```

OUTPUT:

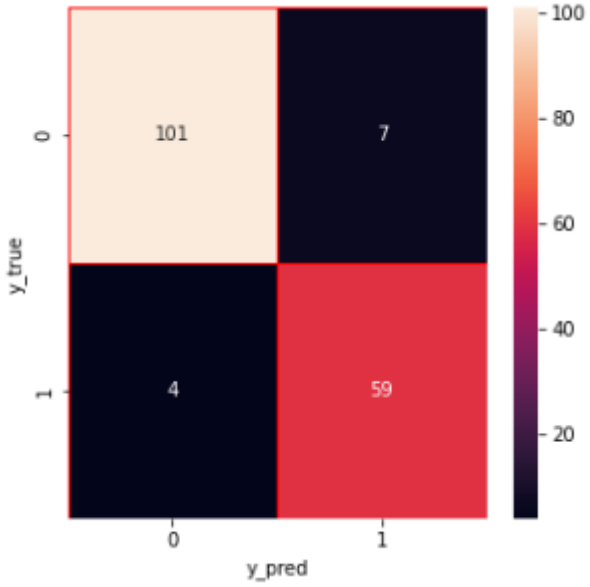
Print accuracy of Decision Tree Classifier algo: 0.935672514619883

PROGRAM:

```
y_pred = dt.predict(x_test)

y_true = y_test
```

```
cm = confusion_matrix(y_true, y_pred)
#visualize
f, ax = plt.subplots(figsize=(5,5))
sns.heatmap(cm,annot = True, linewidths=0.5,linecolor="red",fmt = ".0f",ax=ax)
plt.xlabel("y_pred")
plt.ylabel("y_true")
plt.show()
```

OUTPUT:



PROGRAM:

Random Forest Classifier:

```
rf = RandomForestClassifier(n_estimators=10,random_state=1)
rf.fit(x_train,y_train)
print("Print accuracy of Random Forest Classifier algo: ",rf.score(x_test,y_test))
rf_acc_score = rf.score(x_test,y_test)
```

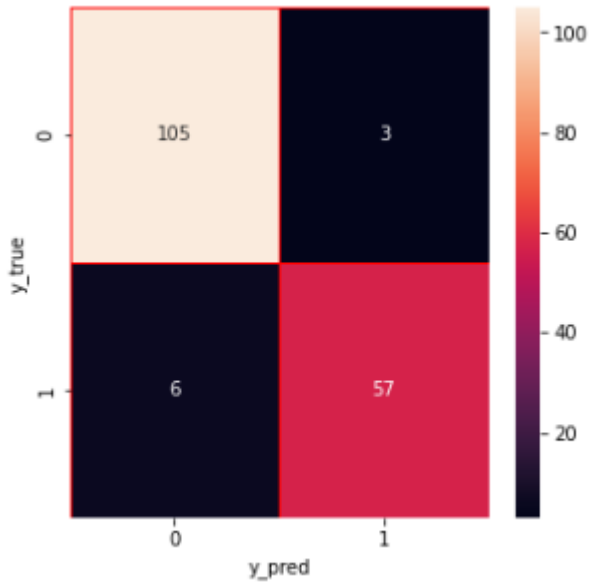
OUTPUT:

Print accuracy of Random Forest Classifier algo: 0.9473684210526315

PROGRAM:

```
y_pred = rf.predict(x_test)

y_true = y_test


cm = confusion_matrix(y_true, y_pred)

#visualize

f, ax = plt.subplots(figsize=(5,5))

sns.heatmap(cm,annot = True, linewidths=0.5,linecolor="red",fmt =
".0f",ax=ax)

plt.xlabel("y_pred")

plt.ylabel("y_true")

plt.show()
```

OUTPUT:



Conclusion:

In short, we conclude that the diagnosis of breast cancer can best be made with the SVM algorithm, which makes accurate predictions with a rate of 98.2 percent.