Breast Cancer Prediction using various regressors and classifiers

The dataset is sourced from : https://archive.ics.uci.edu/dataset/15/breast+cancer+wisconsin+original

Class Labels 2 = benign, 4 = malignant

Our aim is to take into considerations, all the independent attributes wich forms the independent matrix of features X and predict the outcome into two classes of dependent variables namely: *benign* and *maligant*

Importing the libraries

```
import numpy as np #for operation over numerical arrays
import pandas as pd #for data processing and storage
import matplotlib.pyplot as plt #for data visualisation
import seaborn as sns #for advanced data visualisation
```

```
from sklearn.model_selection import train_test_split #for splitting the mdataset
from sklearn.preprocessing import StandardScaler #for performing feature scaling
from sklearn.linear_model import LogisticRegression #using sigmoid function to cl
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score , r2_score , mean_squared_error,precis
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

Importing the dataset

```
data=pd.read_csv("/content/breast_cancer.csv")
data.head()
```

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Ch
0	1000025	5	1	1	1	2	1	
1	1002945	5	4	4	5	7	10	
2	1015425	3	1	1	1	2	2	
3	1016277	6	8	8	1	3	4	
4	1017023	4	1	1	3	2	1	

Next steps: Generate code with data View recommended plots

Data Preprocessing

```
X=data.iloc[:,1:-1].values
y=data.iloc[:,-1].values
#splitting into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, randc
#Performing feature scaling
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

Performing Machine Learning Classification

- Logistic Regression

```
log_regression=LogisticRegression(random_state=42)
log_regression.fit(X_train,y_train)
```

```
▾        LogisticRegression
LogisticRegression(random_state=42)
```

- Decision Tree Classifier

```
des_tree=DecisionTreeClassifier(random_state=42)
des_tree.fit(X_train,y_train)
```

```
▾        DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

- SVM (Support Vector Machine Classifiers)

```
svm_class=SVC(kernel='rbf',random_state=42)
svm_class.fit(X_train,y_train)
```

```
▾       SVC
SVC(random_state=42)
```

- Random Forest Classifier

```
rand_for=RandomForestClassifier(random_state=42)
rand_for.fit(X_train,y_train)
```

```
▾       RandomForestClassifier
RandomForestClassifier(random_state=42)
```

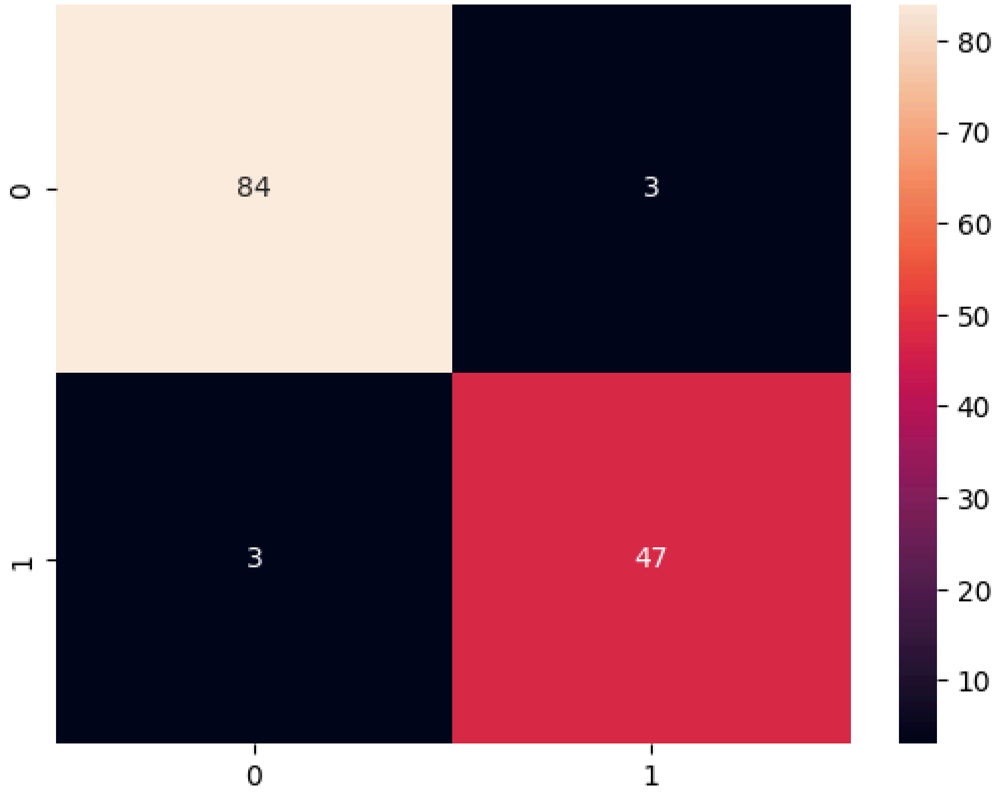
Predicting the results of each ML models

- Logistic Regression

```
y_pred=log_regression.predict(X_test)
ac=accuracy_score(y_test,y_pred)
#pres=precision_score(y_test,y_pred)
#rec=recall_score(y_test,y_pred)
r2=r2_score(y_test,y_pred)
#f1=f1_score(y_test,y_pred)
print("Accuracy score of Logistic Regression is = ",ac)
#print("Precision score of Logistic Regression is = ",pres)
#print("Recall score of Logistic Regression is = ",rec)
print("R squared score of Logistic Regression is = ",r2)
#print("f1 score of Logistic Regression is = ",f1,"\n")
```

```
cm=confusion_matrix(y_test,y_pred)
print("The confusion matrix is : \n",cm)
sns.heatmap(cm,annot=True)
plt.show()
```

```
Accuracy score of Logistic Regression is =  0.9562043795620438
R squared score of Logistic Regression is =  0.8110344827586207
The confusion matrix is :
 [[84  3]
 [ 3 47]]
```

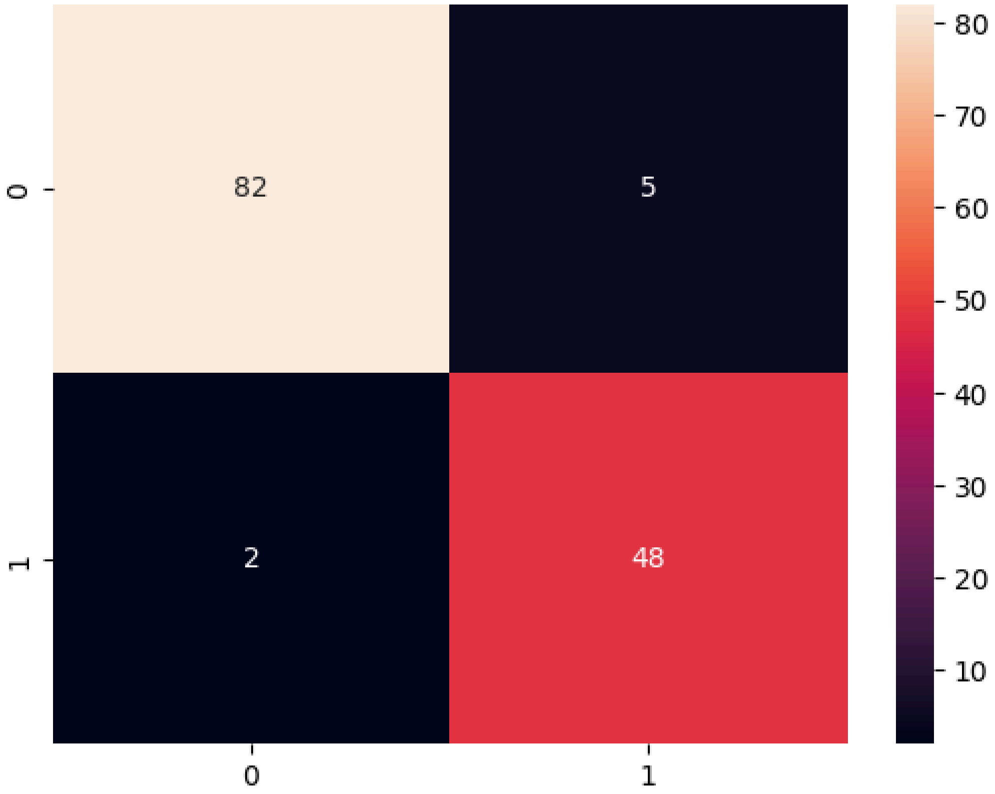


- Decision tree classifier

```
y_pred=des_tree.predict(X_test)
ac=accuracy_score(y_test,y_pred)
#pres=precision_score(y_test,y_pred)
#rec=recall_score(y_test,y_pred)
r2=r2_score(y_test,y_pred)
#f1=f1_score(y_test,y_pred)
print("Accuracy score of Decision Tree Classifier is = ",ac)
#print("Precision score of Decision Tree Classifier is = ",pres)
#print("Recall score of Decision Tree Classifier is = ",rec)
print("R squared score of Decision Tree Classifier is = ",r2)
#print("f1 score of Decision Tree Classifier is = ",f1,"\n")
```

```
cm=confusion_matrix(y_test,y_pred)
print("The confusion matrix is : \n",cm)
sns.heatmap(cm,annot=True)
plt.show()
```

```
Accuracy score of Decision Tree Classifier is =  0.948905109489051
R squared score of Decision Tree Classifier is =  0.7795402298850574
The confusion matrix is :
 [[82  5]
 [ 2 48]]
```



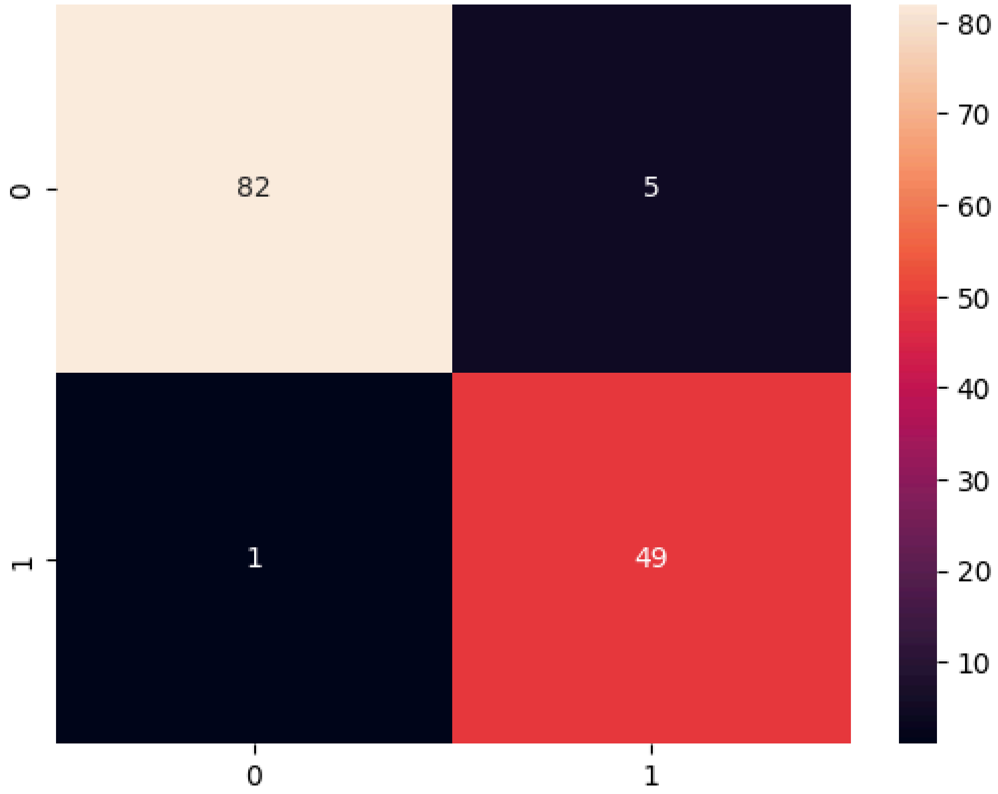
- Svm classifier

```
y_pred=svm_class.predict(X_test)
ac=accuracy_score(y_test,y_pred)
#pres=precision_score(y_test,y_pred)
#rec=recall_score(y_test,y_pred)
r2=r2_score(y_test,y_pred)
#f1=f1_score(y_test,y_pred)
print("Accuracy score of Support Vector Machine is = ",ac)
#print("Precision score of Support Vector Machine is = ",pres)
#print("Recall score of Support Vector machine is = ",rec)
print("R squared score of Support Vector machine is = ",r2)
#print("f1 score of Support Vector Machine is = ",f1,"\n")



cm=confusion_matrix(y_test,y_pred)
print("The confusion matrix is : \n",cm)
sns.heatmap(cm,annot=True)
plt.show()
```

```
Accuracy score of Support Vector Machine is =  0.9562043795620438
R squared score of Support Vector machine is =  0.8110344827586207
The confusion matrix is :
 [[82  5]
 [ 1 49]]
```



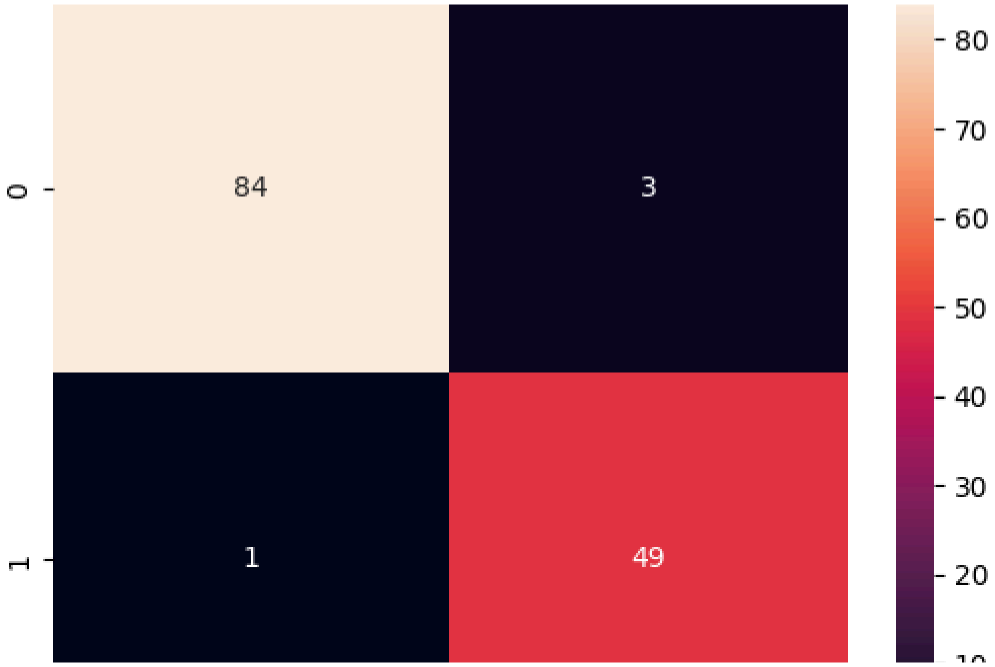
- Random Forest classification

```
y_pred=rand_for.predict(X_test)
ac=accuracy_score(y_test,y_pred)
#pres=precision_score(y_test,y_pred)
#rec=recall_score(y_test,y_pred)
r2=r2_score(y_test,y_pred)
#f1=f1_score(y_test,y_pred)
print("Accuracy score of Random Forest is = ",ac)
#print("Precision score of Random Forest is = ",pres)
#print("Recall score of Random Forest is = ",rec)
print("R squared score of Random Forest is = ",r2)
#print("f1 score of Random Forest is = ",f1,"\n")



cm=confusion_matrix(y_test,y_pred)
print("The confusion matrix is : \n",cm)
sns.heatmap(cm,annot=True)
plt.show()
```

```
Accuracy score of Random Forest is =  0.9708029197080292
R squared score of Random Forest is =  0.8740229885057471
The confusion matrix is :
 [[84  3]
 [ 1 49]]
```



Implementing K-fold cross validation