Machine Learning Project-1

Name: Amandeep Singh

SRN: PES1UG19EC034

Section: 6A

Problem statement:

Given the features and the target, build a Machine Learning Classification model that can classify from a given set of features, if the cancer is Benign or Malignant. You can use any classification algorithm. Please make sure you answer the following are in your report with the code:

1. **Briefly explain the algorithm.**
    a. First, the dataset is loaded from sklearn.datasets or can be downloaded as well. We converted this file into a Pandas dataframe for convenience.
    b. We display the number of Malignant, Benign and total number of diagnosis.
    c. The dataset is split into two parts with one of them holding the features and the other having the target values, 0-Malignant type and 1-Benign typer of cancer classification.
    d. These parts are further split into training sets which are 66.67% and the test sets being 33.33%, along with shuffling. We split them using the train_test_split function from sklearn.model_selection.
    e. Used 4 models for classification:
        i. Gaussian Naïve Bayes
        ii. K Neighbours Classifier
        iii. Logistic Regression
        iv. Forest Random Classifier
    f. We display the accuracy of the models for the specific sets.
    g. The sklearn.metrics method helps in displaying the precision,recall,f1-score support and the confusion matrix.

2. **What was the accuracy?**
    a. Naïve Bayes Model:
        i. Training set: 0.92
        ii. Test set: 0.88
    b. Forest Random classifier:

i. Training set for 10 nodes , 5 nodes, 13 nodes: 0.997 , 0.9973 , 1.0 respectively
ii. Test set for 10 nodes,5 nodes,13 nodes:0.914 , 0.914 , 0.904 respectively

c. Tuned KNN model:
   i. Training set: 1.0
   ii. Test Set: 0.877
d. Logistic Regression:
   i. Training Set: 0.9238
   ii. Test Set: 0.8936

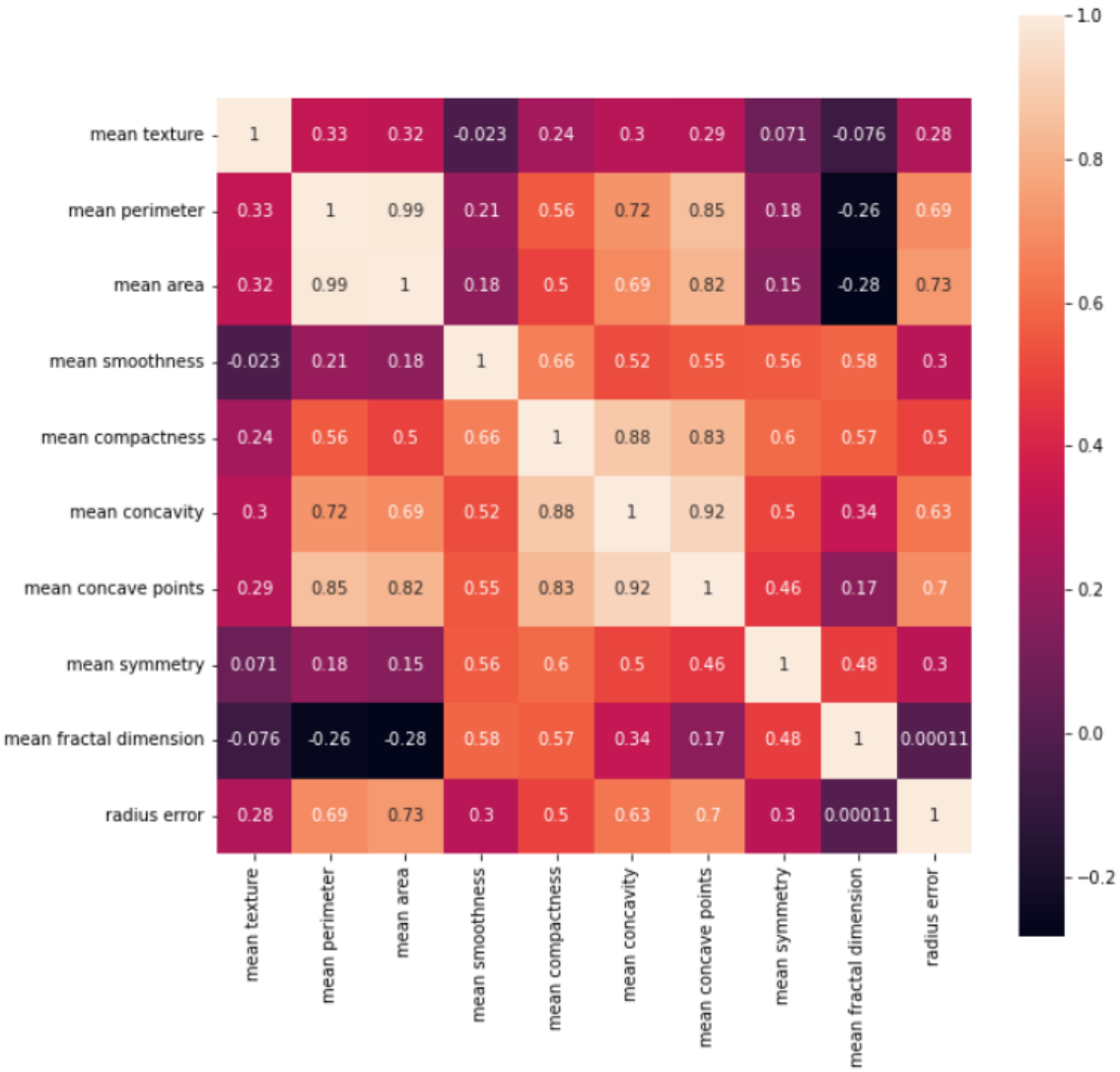
3. **Build and Understand items of Classification Report using scikit learn.**
   a. Sklearn.metrics is used to build and present the classification report.
   b. The parameters passed to the classification_report is y_test (Actual Target values) and the predicted values from the model.

4. **Observe and note changes in accuracy as you vary parameters. Ex. Number of Nodes in Random Forest classifier.**
   a. When varying the number of nodes for a Forest Random Classifier, we observe that there isn't a big difference in the accuracy when the nodes are changed from 10 to 5 but we see an increase in the training set accuracy and a decrease in the test set accuracy when nodes are changed from 10 to 13.

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

The seaborn library allows us to perform these tasks.

It is necessary to remove correlated data prior to this. It can be done using the ".corr()" function of the pandas library


mean texture 1 0.33 0.32 -0.023 0.24 0.3 0.29 0.071 -0.076 0.28
mean perimeter 0.33 1 0.99 0.21 0.56 0.72 0.85 0.18 -0.26 0.69
mean area 0.32 0.99 1 0.18 0.5 0.69 0.82 0.15 -0.28 0.73
mean smoothness -0.023 0.21 0.18 1 0.66 0.52 0.55 0.56 0.58 0.3
mean compactness 0.24 0.56 0.5 0.66 1 0.88 0.83 0.6 0.57 0.5
mean concavity 0.3 0.72 0.69 0.52 0.88 1 0.92 0.5 0.34 0.63
mean concave points 0.29 0.85 0.82 0.55 0.83 0.92 1 0.46 0.17 0.7
mean symmetry 0.071 0.18 0.15 0.56 0.6 0.5 0.46 1 0.48 0.3
mean fractal dimension -0.076 -0.26 -0.28 0.58 0.57 0.34 0.17 0.48 1 0.00011
radius error 0.28 0.69 0.73 0.3 0.5 0.63 0.7 0.3 0.00011 1
mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Code:

In [3]:

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

from random import random
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_breast_cancer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression as LR
from sklearn.ensemble import RandomForestClassifier

data = load_breast_cancer(as_frame=True)
df = data.frame

print("Total diagnosis are : ",df.shape[0])
print("Malignant (0) : ",df.target.value_counts()[0])
print("Benign (1) : ",df.target.value_counts()[1])

featureMeans = list(df.columns[1:11])
x = df.loc[:,featureMeans]
y = df.loc[:,'target']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33,random_state=49)

def NaiveBayes(x_train,x_test,y_train,y_test):
    clf = GaussianNB().fit(x_train,y_train)
    prediction = clf.predict(x_test)

    print("\n\nNaive Bayes Model:")
    print("Training set accuracy",clf.score(x_train,y_train))
    print("Test Set accuracy",clf.score(x_test,y_test))
    print("Classification report for classifier %s :\n%s" %(clf,metrics.classification_report(y_test,prediction)))
    print("Confusion Matrix:\n%s" % (metrics.confusion_matrix(y_test,prediction)))

def tuned_KNN(x_train,x_test,y_train,y_test):
    clf = KNeighborsClassifier(weights='distance',algorithm='ball_tree').fit(x_train,y_train)
    prediction = clf.predict(x_test)

    print("\n\nTuned KNN Model:")
    print("Training set accuracy",clf.score(x_train,y_train))
    print("Test set accuracy",clf.score(x_test,y_test))
    print("Classification report for classifier %s :\n%s" %(clf,metrics.classification_report(y_test,prediction)))
    print("Confusion Matrix:\n%s" % (metrics.confusion_matrix(y_test,prediction)))

def LogisticReg(x_train,x_test,y_train,y_test):
    lr = LR().fit(x_train,y_train)
    prediction = lr.predict(x_test)

    print("\n\nLogistic Regression Model:")
    print("Training set accuracy",lr.score(x_train,y_train))
    print("Test set accuracy",lr.score(x_test,y_test))
    print("Classification report for classifier %s :\n%s" %(lr,metrics.classification_report(y_test,prediction)))
    print("Confusion Matrix:\n%s" % (metrics.confusion_matrix(y_test,prediction)))

def ForestRandom(x_train,x_test,y_test,y_train):
    classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
    classifier.fit(x_train, y_train)
    prediction=classifier.predict(x_test)
    print("\n\nForest Random Classifier for 10 nodes")
    print("Training set accuracy",classifier.score(x_train,y_train))
    print("Test Set accuracy",classifier.score(x_test,y_test))
    print("Classification report for classifier %s :\n%s" %(classifier,metrics.classification_report(y_test,prediction)))
    print("Confusion Matrix:\n%s" % (metrics.confusion_matrix(y_test,prediction)))

    classifier = RandomForestClassifier(n_estimators = 5, criterion = 'entropy', random_state = 0)
    classifier.fit(x_train, y_train)
    prediction=classifier.predict(x_test)
    print("\n\nForest Random Classifier for 5 nodes")
    print("Training set accuracy",classifier.score(x_train,y_train))
    print("Test Set accuracy",classifier.score(x_test,y_test))
    print("Classification report for classifier %s :\n%s" %(classifier,metrics.classification_report(y_test,prediction)))
    print("Confusion Matrix:\n%s" % (metrics.confusion_matrix(y_test,prediction)))

    classifier = RandomForestClassifier(n_estimators = 13, criterion = 'entropy', random_state = 0)
    classifier.fit(x_train, y_train)
    prediction=classifier.predict(x_test)
    print("\n\nForest Random Classifier for 13 nodes")
```

```
    print("Training set accuracy",classifier.score(x_train,y_train))
    print("Test Set accuracy",classifier.score(x_test,y_test))
    print("Classification report for classifier %s :\n%s" %(classifier,metrics.classification_report(y_test,prediction)))
    print("Confusion Matrix:\n%s" % (metrics.confusion_matrix(y_test,prediction)))

NaiveBayes(x_train,x_test,y_train,y_test)
ForestRandom(x_train,x_test,y_test,y_train)
tuned_KNN(x_train,x_test,y_train,y_test)
LogisticReg(x_train,x_test,y_train,y_test)

plt.figure(figsize=(10,10))
sns.heatmap(df[featureMeans].corr(),annot=True,square=True)
plt.show()
```

Outputs/Results:

```
Total diagnosis are :  569
Malignant (0) :  212
Benign (1) :  357


Naive Bayes Model:
Training set accuracy 0.9238845144356955
Test Set accuracy 0.8829787234042553
Classification report for classifier GaussianNB() :
              precision    recall  f1-score   support

           0       0.86      0.83      0.84        71
           1       0.90      0.91      0.91       117

    accuracy                           0.88       188
   macro avg       0.88      0.87      0.87       188
weighted avg       0.88      0.88      0.88       188

Confusion Matrix:
[[ 59  12]
 [ 10 107]]


Forest Random Classifier for 10 nodes
Training set accuracy 0.9973753280839895
Test Set accuracy 0.9148936170212766
Classification report for classifier RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0) :
              precision    recall  f1-score   support

           0       0.89      0.89      0.89        71
           1       0.93      0.93      0.93       117

    accuracy                           0.91       188
   macro avg       0.91      0.91      0.91       188
weighted avg       0.91      0.91      0.91       188

Confusion Matrix:
[[ 63   8]
 [  8 109]]
```

```
Forest Random Classifier for 5 nodes
Training set accuracy 0.9973753280839895
Test Set accuracy 0.9148936170212766
Classification report for classifier RandomForestClassifier(criterion='entropy', n_estimators=5, random_state=0) :
              precision    recall  f1-score   support

           0       0.91      0.86      0.88        71
           1       0.92      0.95      0.93       117

    accuracy                           0.91       188
   macro avg       0.91      0.90      0.91       188
weighted avg       0.91      0.91      0.91       188

Confusion Matrix:
[[ 61  10]
 [  6 111]]


Forest Random Classifier for 13 nodes
Training set accuracy 1.0
Test Set accuracy 0.9042553191489362
Classification report for classifier RandomForestClassifier(criterion='entropy', n_estimators=13, random_state=0) :
              precision    recall  f1-score   support

           0       0.87      0.87      0.87        71
           1       0.92      0.92      0.92       117

    accuracy                           0.90       188
   macro avg       0.90      0.90      0.90       188
weighted avg       0.90      0.90      0.90       188

Confusion Matrix:
[[ 62   9]
 [  9 108]]


Tuned KNN Model:
Training set accuracy 1.0
Test set accuracy 0.8776595744680851
Classification report for classifier KNeighborsClassifier(algorithm='ball_tree', weights='distance') :
              precision    recall  f1-score   support
```

```
Tuned KNN Model:
Training set accuracy 1.0
Test set accuracy 0.8776595744680851
Classification report for classifier KNeighborsClassifier(algorithm='ball_tree', weights='distance') :
              precision    recall  f1-score   support

           0       0.88      0.79      0.83        71
           1       0.88      0.93      0.90       117

    accuracy                           0.88       188
   macro avg       0.88      0.86      0.87       188
weighted avg       0.88      0.88      0.88       188

Confusion Matrix:
[[ 56  15]
 [  8 109]]


Logistic Regression Model:
Training set accuracy 0.9238845144356955
Test set accuracy 0.8936170212765957
Classification report for classifier LogisticRegression() :
              precision    recall  f1-score   support

           0       0.88      0.83      0.86        71
           1       0.90      0.93      0.92       117

    accuracy                           0.89       188
   macro avg       0.89      0.88      0.89       188
weighted avg       0.89      0.89      0.89       188

Confusion Matrix:
[[ 59  12]
 [  8 109]]
```

As we can see here, the Forest Random Forest Classifier turns out to be the better with 92% accuracy in the test cases and 99% accuracy on the training set