Pooja Agarwal
1905330

TNT Sessional Exam

Q1: Implement the following algoithms on your dataset 'Iris Form Docker.csv'
With the various performance metrices: (10 marks). Divide dataset into train data and test data in 80% and 20% ratio.

TABLE:

	Sensitivity/ Recall	Specificity	F-Score	Precision	Accuracy
Logistic Regression	**100.0**	**1.0**	**100.0**	**100.0**	**100.0**
Decision Tree	**100.0**	**1.0**	**100.0**	**100.0**	**100.0**
Gradient Boost	**100.0**	**1.0**	**100.0**	**100.0**	**100.0**
Random Forest	**100.0**	**1.0**	**100.0**	**100.0**	**100.0**

CODE:

importing necessary libraries pandas for data frame matplotlib for barplot sklearn for regression models

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score

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

importing data frame from csv file using read_csv function

In [2]:
```
df = pd.read_csv('iris-write-from-docker.csv', encoding = "ISO-8859-1")
df
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [3]:
```
le = preprocessing.LabelEncoder()
df['class'] = le.fit_transform(df['class'])
```

In [4]:
```
df['class'].value_counts()
```

Out[4]:
```
0    50
1    50
2    50
Name: class, dtype: int64
```

In [5]:
```
corr = df.corr()
corr
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	class
sepal_length	1.000000	-0.109369	0.871754	0.817954	0.782561
sepal_width	-0.109369	1.000000	-0.420516	-0.356544	-0.419446
petal_length	0.871754	-0.420516	1.000000	0.962757	0.949043
petal_width	0.817954	-0.356544	0.962757	1.000000	0.956464
class	0.782561	-0.419446	0.949043	0.956464	1.000000

Executing Test Train split with class as the target column

In [6]:
```
x = df.drop(columns=['class'])
y = df['class']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[6]: ((120, 4), (120,), (30, 4), (30,))

Resulting lists

In [7]:
```
algos = []
accuracy = []
recall = []
precision = []
f1Score = []
specificity = []
```

Pooja Agarwal
1905330

Logistic Regression

In [8]:
```
algo = "Logistic Regression"
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred, average='weighted') * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred, average='weighted') * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred, average='weighted') * 100
print('F score:', f1s)

algos.append(algo)
accuracy.append(acc)
precision.append(pre)
f1Score.append(f1s)
```

```
Logistic Regression
[[ 7  0  0]
 [ 0 12  0]
 [ 0  0 11]]


Accuracy: 100.0
Recall: 100.0
Precision: 100.0
F score: 100.0
```

Pooja Agarwal
1905330

Decision Tree

In [9]:
```
algo = "Decision Tree"
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred, average='weighted') * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred, average='weighted') * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred, average='weighted') * 100
print('F score:', f1s)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)
```

```
Decision Tree
[[ 7  0  0]
 [ 0 12  0]
 [ 0  0 11]]


Accuracy: 100.0
Recall: 100.0
Precision: 100.0
F score: 100.0
```

Gradient Boost

In [10]:
```
algo = "GradientBoost"
model = GradientBoostingClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred, average='weighted') * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred, average='weighted') * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred, average='weighted') * 100
print('F score:', f1s)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)
```

```
GradientBoost
[[ 7  0  0]
 [ 0 12  0]
 [ 0  0 11]]


Accuracy: 100.0
Recall: 100.0
Precision: 100.0
F score: 100.0
```

Random Forest

In [11]:
```
algo = "Random forest"
model = RandomForestClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred, average='weighted') * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred, average='weighted') * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred, average='weighted') * 100
print('F score:', f1s)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)
```

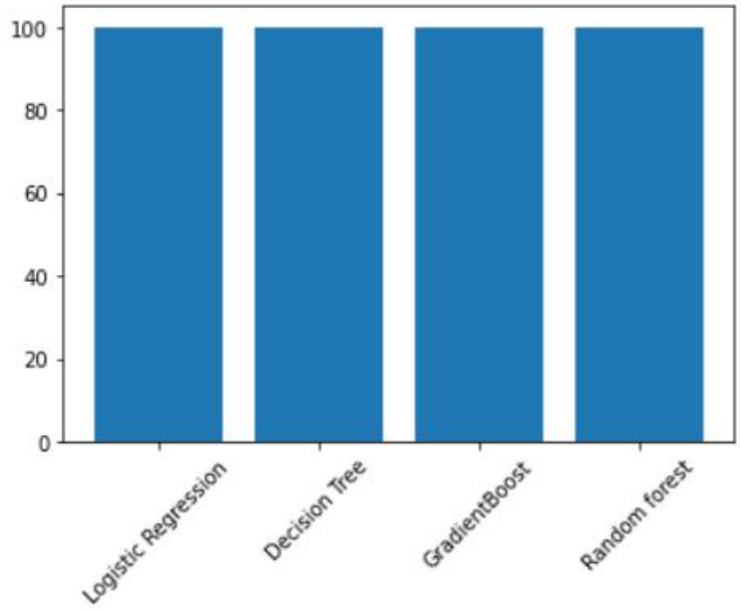
```
Random forest
[[ 7  0  0]
 [ 0 12  0]
 [ 0  0 11]]


Accuracy: 100.0
Recall: 100.0
Precision: 100.0
F score: 100.0
```

Pooja Agarwal
1905330

Q2: Draw the graph(barplot) for each algorithm,keep on y-axis the accuracy and thus compare the accuacy in each case.(2 marks)

CODE:

In [12]:
```
plt.bar(algos, accuracy)
plt.xticks(rotation = 45)
plt.show()
```



Pooja Agarwal
1905330

Q4: Write the python program using loop to print sum of first 10 natural numbers (2 marks).

CODE:

```
In [1]: sum = 0
        for x in range(1, 11):
            sum = sum + x

        print("Sum of first 10 Natural Numbers is:", sum)
```

OUTPUT:

```
Sum of first 10 Natural Numbers is: 55
```