# Ramdom Forest(Classification)

## Importing the Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
```

## Loading the Data set

In [2]:
```
df=sns.load_dataset('penguins')
df.head()
```

Out[2]:

| | species | island | bill_length_mm | bill_depth_mm | flipper_length_mm | body_mass_g | sex |
|---|---|---|---|---|---|---|---|
| **0** | Adelie | Torgersen | 39.1 | 18.7 | 181.0 | 3750.0 | Male |
| **1** | Adelie | Torgersen | 39.5 | 17.4 | 186.0 | 3800.0 | Female |
| **2** | Adelie | Torgersen | 40.3 | 18.0 | 195.0 | 3250.0 | Female |
| **3** | Adelie | Torgersen | NaN | NaN | NaN | NaN | NaN |
| **4** | Adelie | Torgersen | 36.7 | 19.3 | 193.0 | 3450.0 | Female |

In [3]:
```
df.shape
```

Out[3]:
```
(344, 7)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
species               0
island                0
bill_length_mm        2
bill_depth_mm         2
flipper_length_mm     2
body_mass_g           2
sex                  11
dtype: int64
```

# Drop the null values

In [6]:
```
df.dropna(inplace=True)
```

## checking dataset

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
species              0
island               0
bill_length_mm       0
bill_depth_mm        0
flipper_length_mm    0
body_mass_g          0
sex                  0
dtype: int64
```

## Feature Engineering

label encoding transformation categorical data into numeric

In [8]:
```
df.sex.unique()
```

Out[8]:
```
array(['Male', 'Female'], dtype=object)
```

In [9]:
```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
df['sex']=encoder.fit_transform(df['sex'])
```

In [10]:
```
df.head()
```

Out[10]:

| | species | island | bill_length_mm | bill_depth_mm | flipper_length_mm | body_mass_g | sex |
|---|---|---|---|---|---|---|---|
| **0** | Adelie | Torgersen | 39.1 | 18.7 | 181.0 | 3750.0 | 1 |
| **1** | Adelie | Torgersen | 39.5 | 17.4 | 186.0 | 3800.0 | 0 |
| **2** | Adelie | Torgersen | 40.3 | 18.0 | 195.0 | 3250.0 | 0 |
| **4** | Adelie | Torgersen | 36.7 | 19.3 | 193.0 | 3450.0 | 0 |
| **5** | Adelie | Torgersen | 39.3 | 20.6 | 190.0 | 3650.0 | 1 |

In [11]:
```
df.island.unique()
```

Out[11]:
```
array(['Torgersen', 'Biscoe', 'Dream'], dtype=object)
```

In [12]:
```
df['island']=encoder.fit_transform(df['island'])
```

In [13]:
```
df.head()
```

Out[13]:

| | species | island | bill_length_mm | bill_depth_mm | flipper_length_mm | body_mass_g | sex |
|---|---|---|---|---|---|---|---|
| **0** | Adelie | 2 | 39.1 | 18.7 | 181.0 | 3750.0 | 1 |
| **1** | Adelie | 2 | 39.5 | 17.4 | 186.0 | 3800.0 | 0 |
| **2** | Adelie | 2 | 40.3 | 18.0 | 195.0 | 3250.0 | 0 |
| **4** | Adelie | 2 | 36.7 | 19.3 | 193.0 | 3450.0 | 0 |
| **5** | Adelie | 2 | 39.3 | 20.6 | 190.0 | 3650.0 | 1 |

In [14]:
```
x=df.drop('species',axis=1)
```

In [15]:
```
y=df['species']
```

In [16]:
```
x.head()
```

Out[16]:

| | island | bill_length_mm | bill_depth_mm | flipper_length_mm | body_mass_g | sex |
|---|---|---|---|---|---|---|
| **0** | 2 | 39.1 | 18.7 | 181.0 | 3750.0 | 1 |
| **1** | 2 | 39.5 | 17.4 | 186.0 | 3800.0 | 0 |
| **2** | 2 | 40.3 | 18.0 | 195.0 | 3250.0 | 0 |
| **4** | 2 | 36.7 | 19.3 | 193.0 | 3450.0 | 0 |
| **5** | 2 | 39.3 | 20.6 | 190.0 | 3650.0 | 1 |

In [17]:
```
y.head()
```

Out[17]:
```
0    Adelie
1    Adelie
2    Adelie
4    Adelie
5    Adelie
Name: species, dtype: object
```

In [18]:
```
y.unique()
```

Out[18]:
```
array(['Adelie', 'Chinstrap', 'Gentoo'], dtype=object)
```

In [19]:
```
y=y.map({'Adelie':0, 'Chinstrap':1, 'Gentoo':2})
y
```

Out[19]:
```
0      0
1      0
2      0
4      0
5      0
      ..
338    2
340    2
341    2
342    2
343    2
Name: species, Length: 333, dtype: int64
```

In [20]:
```
y.isnull().sum()
```

Out[20]:
```
0
```

# Train Test Data Splitting

In [21]:
```
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,random_state=0)
```

In [22]:
```
print('x_train',x_train.shape)
print('y_train',y_train.shape)
print('x_test',x_test.shape)
print('y_test',y_test.shape)
```

```
x_train (233, 6)
y_train (233,)
x_test (100, 6)
y_test (100,)
```

## Training Random Forest Classification on Training Set

In [23]:
```
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(n_estimators=5,criterion='entropy',random_state=0)
classifier.fit(x_train,y_train)
```

Out[23]:
```
RandomForestClassifier(criterion='entropy', n_estimators=5, random_state=0)
```

# Predictions the Test Results

In [24]:
```
y_pred=classifier.predict(x_test)
y_pred
```

Out[24]:
```
array([0, 0, 2, 0, 0, 0, 1, 2, 2, 1, 2, 0, 0, 1, 0, 0, 2, 0, 1, 0, 0, 0,
       2, 2, 2, 2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 1, 0, 1, 0, 2, 2, 0, 0,
       0, 0, 0, 0, 2, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 2, 0, 1, 0, 2, 0, 0,
       2, 2, 1, 2, 2, 1, 2, 1, 0, 2, 0, 2, 0, 2, 1, 2, 2, 2, 1, 2, 1, 0,
       0, 2, 2, 0, 2, 0, 2, 0, 2, 0, 2, 2], dtype=int64)
```

## confusion matrix

In [25]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report,accuracy_score
```

In [26]:
```
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[48  0  0]
 [ 2 14  0]
 [ 0  0 36]]
```

In [27]:
```
accuracy_score(y_test,y_pred)
```

Out[27]:
```
0.98
```

In [28]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        48
           1       1.00      0.88      0.93        16
           2       1.00      1.00      1.00        36

    accuracy                           0.98       100
   macro avg       0.99      0.96      0.97       100
weighted avg       0.98      0.98      0.98       100
```

## Try with Different number of trees and gini criteria

In [29]:
```
from sklearn.ensemble import RandomForestClassifier
classifier_1=RandomForestClassifier(n_estimators=5,criterion='gini',random_state=0)
```

```
classifier_1.fit(x_train,y_train)
```

Out[29]:

RandomForestClassifier

RandomForestClassifier(n_estimators=5, random_state=0)

In [30]:
```
y1_pred=classifier_1.predict(x_test)
y1_pred
```

Out[30]:
```
array([0, 0, 2, 0, 0, 0, 1, 2, 2, 1, 2, 0, 0, 1, 0, 0, 2, 0, 1, 0, 0, 0,
       2, 2, 2, 2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 1, 0, 1, 0, 2, 2, 0, 0,
       0, 0, 0, 0, 2, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 2, 0, 1, 0, 2, 0, 0,
       2, 2, 1, 2, 2, 1, 2, 1, 0, 2, 0, 2, 0, 2, 1, 2, 2, 2, 1, 2, 1, 0,
       0, 2, 2, 0, 2, 0, 2, 0, 2, 0, 2, 2], dtype=int64)
```

In [31]:
```
accuracy_score(y_test,y1_pred)
```

Out[31]:
```
0.98
```

In [ ]:
```
!jupyter nbconvert --to webpdf --allow-chromium-download SVM_KERNALS.ipynb
```

In [ ]:

In [ ]: