RANDOM_FOREST_CLASSIFICATION

```
import pandas as pd
import numpy as np
import seaborn as sns
```

Loading the Data set

```
df = sns.load_dataset('penguins')
df.head()
```

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

Next steps: Generate code with df View recommended plots

```
df.shape
```

```
(344, 7)
```

```
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
```

```
df.isnull().sum()
```

```
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

```
df.dropna(inplace=True)
```

```
df.isnull().sum()  #checking
```

```
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 to convert all categorical values into numerical

ONE HOT ENCODING

firstly will apply to SEX column

```
df.sex.unique()
```

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

```
pd.get_dummies(df['sex'], dtype=int).head()
```

	Female	Male
0	0	1
1	1	0
2	1	0
4	1	0
5	0	1

```
sex = pd.get_dummies(df['sex'], drop_first = True, dtype=int).head() #to avoid colinearity
```

applying to ISLAND

```
df.island.unique()
```

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

```
pd.get_dummies(df['island'], dtype = int).head()
```

	Biscoe	Dream	Torgersen
0	0	0	1
1	0	0	1
2	0	0	1
4	0	0	1
5	0	0	1

```
island = pd.get_dummies(df['island'],dtype = int, drop_first=True).head()
```

Combining the above two data frames to the original df

```
new_data = pd.concat([df, island, sex], axis =1)
```

```
new_data.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	Dream	Torgersen	Male
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male	0.0	1.0	1.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female	0.0	1.0	0.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female	0.0	1.0	0.0
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female	0.0	1.0	0.0
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	Male	0.0	1.0	1.0

Next steps: Generate code with new_data View recommended plots

Drop repeated columns

```
new_data.drop(columns=['sex', 'island'], axis =1, inplace =True)
```

```
new_data.head()
```

	species	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	Dream	Torgersen	Male
0	Adelie	39.1	18.7	181.0	3750.0	0.0	1.0	1.0
1	Adelie	39.5	17.4	186.0	3800.0	0.0	1.0	0.0
2	Adelie	40.3	18.0	195.0	3250.0	0.0	1.0	0.0
4	Adelie	36.7	19.3	193.0	3450.0	0.0	1.0	0.0
5	Adelie	39.3	20.6	190.0	3650.0	0.0	1.0	1.0

Next steps: Generate code with new_data View recommended plots

```
new_data.rename(columns = {'Male':'Sex'}, inplace = True)
```

```
new_data.head()
```

	species	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	Dream	Torgersen	Sex
0	Adelie	39.1	18.7	181.0	3750.0	0.0	1.0	1.0
1	Adelie	39.5	17.4	186.0	3800.0	0.0	1.0	0.0
2	Adelie	40.3	18.0	195.0	3250.0	0.0	1.0	0.0
4	Adelie	36.7	19.3	193.0	3450.0	0.0	1.0	0.0
5	Adelie	39.3	20.6	190.0	3650.0	0.0	1.0	1.0

Next steps: Generate code with new_data View recommended plots

Creating separate target variable

```
Y = new_data.species
Y.head()
```

```
0    Adelie
1    Adelie
2    Adelie
4    Adelie
5    Adelie
Name: species, dtype: object
```

```
Y.unique()
```

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

```
Y = Y.map({ 'Adelie':0, 'Chinstrap':1, 'Gentoo':2})
```

```
Y.head()
```

```
0    0
1    0
2    0
4    0
5    0
Name: species, dtype: int64
```

Dropping th Target Variable Species Y to create input variable X

```
new_data.drop('species', inplace = True, axis =1)
new_data.head()
```

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	Dream	Torgersen	Sex
0	39.1	18.7	181.0	3750.0	0.0	1.0	1.0
1	39.5	17.4	186.0	3800.0	0.0	1.0	0.0
2	40.3	18.0	195.0	3250.0	0.0	1.0	0.0
4	36.7	19.3	193.0	3450.0	0.0	1.0	0.0
5	39.3	20.6	190.0	3650.0	0.0	1.0	1.0

Next steps: Generate code with new_data View recommended plots

```
X= new_data
```

```
X.isnull( ).sum()
```

```
bill_length_mm         0
bill_depth_mm          0
flipper_length_mm      0
body_mass_g            0
Dream                328
Torgersen            328
Sex                  328
dtype: int64
```

```
X['Dream'].fillna(0, inplace = True)
```

```
X['Torgersen'].fillna(0, inplace = True)
```

```
X['Sex'].fillna(0, inplace = True)
```

```
X.isnull( ).sum()
```

```
bill_length_mm       0
bill_depth_mm        0
flipper_length_mm    0
body_mass_g          0
Dream                0
Torgersen            0
Sex                  0
dtype: int64
```

Everything's in numerical and we have X and Y .. Next step Splitting Dataset into Training and Test Data

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =0.2, random_state=0)
```

Training RANDOM FOREST CLASSIFICATION on training data set

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators =5, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, Y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=5, random_state=0)
```

Predicting

```
y_pred = classifier.predict(X_test)
y_pred
```

```
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, 1, 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])
```

Metrics

```
from sklearn.metrics import confusion_matrix, accuracy_score
```

```
cm = confusion_matrix(Y_test, y_pred)
cm
```

```
array([[39,  0,  0],
       [ 1,  9,  0],
       [ 0,  0, 18]])
```

```
accuracy_score(Y_test, y_pred)
```

```
0.9850746268656716
```

Start coding or generate with AI.