UNDERSTANDING RANDOM FOREST CLASSIFIER WORKFLOW AND HOW IT IS RELIABLE

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
```

We look into the keys of this json file to have a better understanding of the data and then we will use the relevant keys to generate a data frame

In [2]:
```
iris = load_iris()

iris.keys()
```

Out[2]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
'filename', 'data_module'])
```

After visiting the data we have some idea of what key values we are about to use to make the test and train data for this model. So we will create an initial version of data frame use DataFrame functions in pandas with parems as iris.data as the data and columns as iris.feature names for which iris.data is listed

In [3]:
```
df = pd.DataFrame(iris.data, columns= iris.feature_names)

df.head(10)
```

Out[3]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1

Now we would like to know what are the target names for a flower with certain sepal width, petal len and petal width, For that we would create a categorical data and we will map the code in target key and string categories in target_names key and we will map them to create a memory efficient categorical data.

In [4]:
```
df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)
df.head(10)
```

Out[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

Just to divide the data into 20% 80% split. Just so we could use the 8-% data for training and 20% for testing. It is an alternative to train_test_split and provide a better understanding of how splitiing of training and testing data works for the larger dataset.

In [5]:
```
df['divide'] = np.random.uniform(0,1, len(df)) >=0.80

df.head(10)
```

Out[5]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species	divide
0	5.1	3.5	1.4	0.2	setosa	False
1	4.9	3.0	1.4	0.2	setosa	False
2	4.7	3.2	1.3	0.2	setosa	False
3	4.6	3.1	1.5	0.2	setosa	False
4	5.0	3.6	1.4	0.2	setosa	False
5	5.4	3.9	1.7	0.4	setosa	False
6	4.6	3.4	1.4	0.3	setosa	True
7	5.0	3.4	1.5	0.2	setosa	False
8	4.4	2.9	1.4	0.2	setosa	False
9	4.9	3.1	1.5	0.1	setosa	False

Now we will asignt the splitted data to 2 data sets as test and train

In [6]:
```
test, train = df[df['divide'] == True], df[df['divide']== False]
print(len(test), len(train))
```

```
23 127
```

In [7]:
```
train.head(5)
```

Out[7]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species	divide
0	5.1	3.5	1.4	0.2	setosa	False
1	4.9	3.0	1.4	0.2	setosa	False
2	4.7	3.2	1.3	0.2	setosa	False
3	4.6	3.1	1.5	0.2	setosa	False
4	5.0	3.6	1.4	0.2	setosa	False

Selecting the First 4 columns for training purpose

In [8]:
```
features = df.columns[:4]
# Converting each specie name to digit so that the computer could differentiate
Y= pd.factorize(train['species'])[0]
Y
```

Out[8]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [9]:
```
model_1 = RandomForestClassifier( random_state = 0)
model_1.fit(train[features], Y)
```

Out[9]:
```
RandomForestClassifier(random_state=0)
```

In [10]:
```
prediction = model_1.predict(test[features])

prediction
```

Out[10]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 2, 2, 2,
       2])
```

Testing the Prediction Probability of this prediction

In [11]:
```
model_1.predict_proba(test[features])[:20]
```

Out[11]:
```
array([[1.  , 0.  , 0.  ],
       [1.  , 0.  , 0.  ],
       [0.98, 0.02, 0.  ],
       [1.  , 0.  , 0.  ],
       [1.  , 0.  , 0.  ],
       [1.  , 0.  , 0.  ],
       [0.97, 0.03, 0.  ],
       [0.91, 0.09, 0.  ],
       [1.  , 0.  , 0.  ],
       [0.  , 1.  , 0.  ],
       [0.  , 1.  , 0.  ],
       [0.  , 1.  , 0.  ],
       [0.  , 1.  , 0.  ],
       [0.  , 0.98, 0.02],
       [0.  , 1.  , 0.  ],
       [0.  , 0.01, 0.99],
       [0.  , 0.01, 0.99],
       [0.  , 0.  , 1.  ],
       [0.  , 0.86, 0.14],
       [0.  , 0.1 , 0.9 ]])
```

In [12]:
```
#mapping names for each predicition target to the name of the flowers

preds = iris.target_names[model_1.predict(test[features])]
preds
```

Out[12]:
```
array(['setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
       'setosa', 'setosa', 'setosa', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'virginica', 'virginica', 'virginica', 'versicolor', 'virginica',
       'virginica', 'virginica', 'virginica'], dtype='<U10')
```

In [13]:
```
# Creating a Confussion Matrix to better explain how the Random Tree Forest Work

table = pd.crosstab(test['species'], preds, rownames = ["Actual Species"], colna
table
```

Out[13]:

Actual Species \ Predicted Species	setosa	versicolor	virginica
setosa	9	0	0
versicolor	0	6	0
virginica	0	1	7

Checking the Accuracy of Our prediction model.

In [14]:
```
accuracy =accuracy_score(test['species'], preds)
print(accuracy*100)
```

```
95.65217391304348
```