Random Forest

In [1]:
```
from hyperopt import hp, fmin, tpe, rand, STATUS_OK, Trials
import pandas as pd
import numpy as np
from time import time
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PowerTransformer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold, StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report, confusion_matrix
```

In [2]:
```
df = pd.read_excel("Dry_Bean_Dataset.xlsx")
df = df.drop_duplicates()
X = df.iloc[:,:16]
y = df.iloc[:,16:]
y = y.reset_index().drop(columns = "index")
scaler = StandardScaler()
X = pd.DataFrame(scaler.fit_transform(X), columns = X.columns)
pt = PowerTransformer(method = "yeo-johnson")
X = pd.DataFrame(pt.fit_transform(X), columns = X.columns)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 101)
```

In [3]:
```
best_score=1.0
def objective(space):

    global best_score
    model = RandomForestClassifier(**space, n_jobs = -1)
    kfold = StratifiedKFold(n_splits=5, random_state=101, shuffle=True)
    score = 1-cross_val_score(model, X_train, np.ravel(y_train), cv=kfold, scoring="accuracy", verbose=False).n

    if (score < best_score):
        best_score=score

    return score
```

In [4]:
```
space = {
        "max_depth": hp.choice("max_depth", np.arange(2,101,1)),
    "min_samples_leaf": hp.choice("min_samples_leaf", np.arange(1,201,1)),
    "n_estimators": hp.choice("n_estimators", np.arange(10,201,1))
      }
```

In [5]:
```
n_iter_hopt = 1000
trials = Trials() # Initialize an empty trials database for further saving/loading ran iteractions

start = time()

best = fmin(objective,
            space = space,
            algo = tpe.suggest,
            max_evals = n_iter_hopt,
            trials = trials,
            rstate = np.random.default_rng(101))

elapsed_time_hopt = time() - start
```

```
100%|██████████████████████████████████████████████| 1000/1000 [15:48<00:00,  1.05trial/s, best loss: 0.0765188479
4199582]
```

In [6]:
```
print("\nHyperopt search took %.2f seconds for %d candidates. Accuracy reached: %.3f\nOptimal parameters found:
```

```
Hyperopt search took 948.17 seconds for 1000 candidates. Accuracy reached: 92.348
Optimal parameters found: {'max_depth': 97, 'min_samples_leaf': 2, 'n_estimators': 143}
```

In [13]:
```
rf = RandomForestClassifier(max_depth = 99, min_samples_leaf = 3, n_estimators = 153)
rf.fit(X_train, np.ravel(y_train))
y_pred = rf.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=y.Class.unique()))
```

```
[[241   1  12   0   0   2   4]
 [  0 115   0   0   0   0   0]
 [ 10   0 323   0   1   1   4]
 [  0   0   0 669   0   8  36]
 [  2   0   3   3 331   0   9]
 [  0   0   0  11   0 387  11]
 [  3   0   3  49   7   6 457]]
              precision    recall  f1-score   support

       SEKER       0.94      0.93      0.93       260
    BARBUNYA       0.99      1.00      1.00       115
      BOMBAY       0.95      0.95      0.95       339
        CALI       0.91      0.94      0.93       713
       HOROZ       0.98      0.95      0.96       348
        SIRA       0.96      0.95      0.95       409
    DERMASON       0.88      0.87      0.87       525

    accuracy                           0.93      2709
   macro avg       0.94      0.94      0.94      2709
weighted avg       0.93      0.93      0.93      2709
```

In [14]:
```
metrics.accuracy_score(y_test, y_pred)
```

Out[14]:
```
0.9313399778516057
```