XGBoost Classifier

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import PowerTransformer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegressionCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
```

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)
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.fit_transform(y_test)
```

```
C:\Users\matth\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vect
or y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
```

In [3]:
```
XGB = xgb.XGBClassifier(objective="multi:softprob", random_state=101)
XGB.fit(X_train, y_train)
```

Out[3]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_threshold=64, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, objective='multi:softprob',
              predictor='auto', ...)
```

In [4]:
```
y_pred = XGB.predict(X_test)
```

In [5]:
```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=y.Class.unique()))
```

```
[[241   1   9   0   0   5   4]
 [  0 115   0   0   0   0   0]
 [ 11   0 323   0   0   1   4]
 [  0   0   0 661   3   6  43]
 [  1   0   6   3 331   0   7]
 [  3   0   0  13   0 385   8]
 [  1   0   2  53  13   6 450]]
              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      0.93      0.92       713
       HOROZ       0.95      0.95      0.95       348
        SIRA       0.96      0.94      0.95       409
    DERMASON       0.87      0.86      0.86       525

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

In [6]:
```
metrics.accuracy_score(y_test, y_pred)
```

Out[6]:
```
0.9250645994832042
```