Machine Learning

Olivetti Face Data – XGBoost Classifier

Download the Olivetti faces dataset.
Visit https://scikit-learn.org/0.19/datasets/olivetti_faces.html
There are 40 classes (corresponding to 40 people), each class having 10 faces of the individual; so there are a total of 400 images.
Here each face is viewed as an image of size 64 × 64 (= 4096) pixels; each pixel having values 0 to 255 which are ultimately converted into floating numbers in the range [0,1].

Split the dataset into train
and test parts. Do this splitting randomly 10 times and report the average accuracy.
You may vary the test and train dataset sizes.

Build a XGBoost Classifier using the training data. Tune the parameters corresponding to pruning the decision tree. Use the best decision tree to classify the test dataset and obtain the accuracy. Use misclassification impurity also.

CODE:

Please find the code attached for XGBoost Classifer :

- XGBoost Classifer from xgboost platform is used on a train size of 320 samples (80%) of the olivetti dataset that contains a total of 400 samples.
- Using a library function called **GridSearchCV** of sklearn's model selection package, best parameters from the listed hyper parameters of the model are found.
- Now using the best estimator, the **XGBClassifier** is run 10 times on randomly split data using randomly selected test and train sizes from range 0.2 to 0.35.
- Accuracy is found after fitting this best decision tree and average accuracy is computed for 10 iterations
- **Feature importance is also computed**

RESULT:

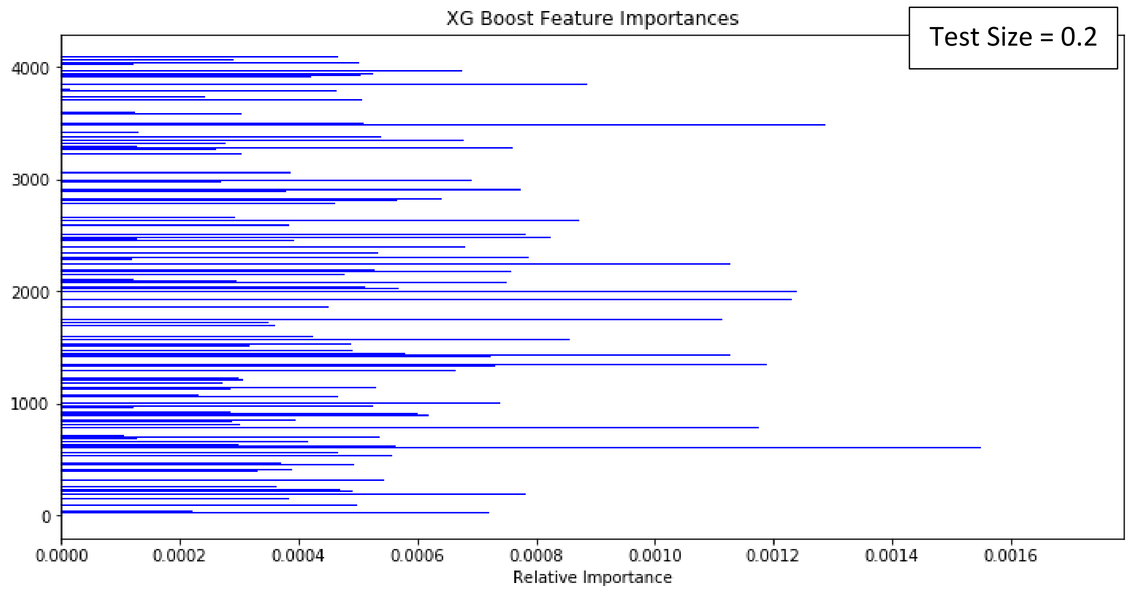
Result is captured below for showing the tuned hyper parameters from GridSearchCV and related accuracies for the 10 iterations.

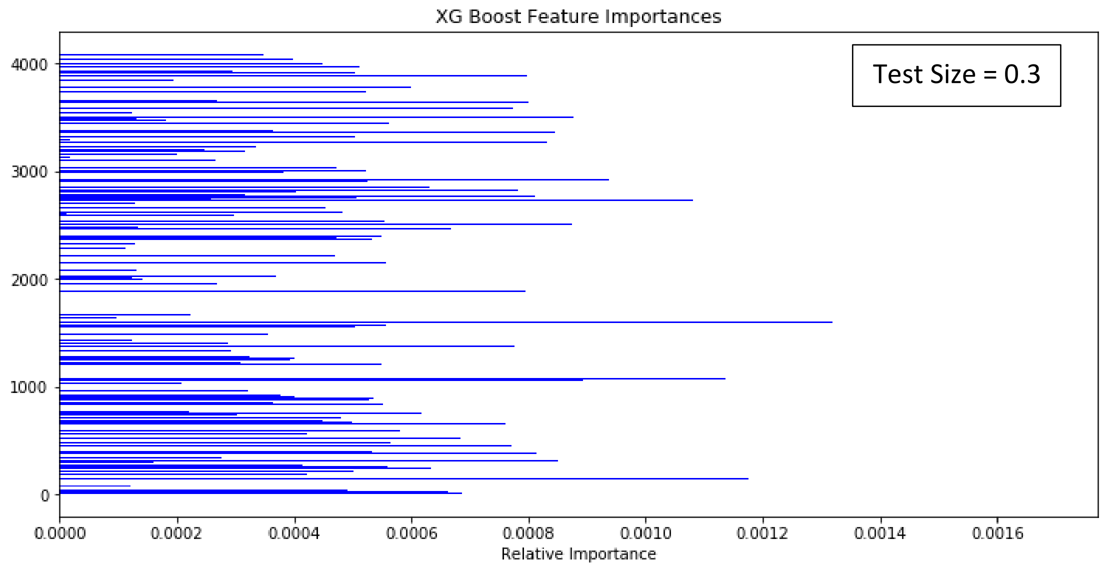
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
    colsample_bynode=1, colsample_bytree=0.7, gamma=0, gpu_id=-1,
    importance_type='gain', interaction_constraints='',
    learning_rate=0.1, max_delta_step=0, max_depth=10,
    min_child_weight=3, missing=nan, monotone_constraints='()',
    n_estimators=400, n_jobs=4, num_parallel_tree=1,
    objective='multi:softprob', random_state=0, reg_alpha=0,
    reg_lambda=1, scale_pos_weight=None, subsample=0.5,
    tree_method='exact', use_label_encoder=True, validate_parameters=1,
    verbosity=None)

Test Size Ratio: 0.25
Accuracy : 0.91
Test Size Ratio: 0.3
```

```
            Accuracy : 0.9583333333333334
            Test Size Ratio: 0.3
            Accuracy : 0.9416666666666667
            Test Size Ratio: 0.25
            Accuracy : 0.95
            Test Size Ratio: 0.3
            Accuracy : 0.9666666666666667
            Test Size Ratio: 0.2
            Accuracy : 0.975
            Test Size Ratio: 0.3
            Accuracy : 0.95
            Test Size Ratio: 0.2
            Accuracy : 0.9
            Test Size Ratio: 0.35
            Accuracy : 0.9214285714285714
            Test Size Ratio: 0.35
            Accuracy : 0.9
avg accuracy =  0.9373095238095239
```

PLOT:





INFERENCE/ANALYSIS:

- XGBoost provides **a wrapper class** to allow models to be treated like classifiers or regressors in the scikit-learn framework.
- We first need to hyperparameter tune the XGboost classifier .
- **In the task following parameters were considered for tuning**

```
'learning_rate': [0.1],
 'max_depth': [10, 20, 50],
   'min_child_weight': [1, 2, 3],
   'subsample': [0.5, 0.7],
   'colsample_bytree': [0.5, 0.7],
   'n_estimators' : [100, 200, 400],
```

- With GridsearchCV based cross validation for hyperparameter tuning, the result shows that the best values are as shown in RESULT section

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=0.7, gamma=0, gpu_id=-1,
       importance_type='gain', interaction_constraints='',
       learning_rate=0.1, max_delta_step=0, max_depth=10,
       min_child_weight=3, missing=nan, monotone_constraints='()',
       n_estimators=400, n_jobs=4, num_parallel_tree=1,
       objective='multi:softprob', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=None, subsample=0.5,
       tree_method='exact', use_label_encoder=True, validate_parameters=1,
       verbosity=None)
```

- **Average accuracy with these parameters for running the XGB classifier is** avg accuracy = 0.9373095238095239 for 10 iterations of randomly generated test and train samples.

Decision tree did not give good accuracy for high dimensional featured dataset like an olivetti dataset with 4096 features due to risk of overfitting.
In order to solve this limitation of decision tree, classifiers like Random forest are preferred that use multiple decision tree and XGboost classifier which also uses multiple decision trees and a host of other parameters outperforms the simple decision tree despite best parameterization.

RESOURCES USED FOR THE ASSIGNMENT:

• **Environment:** Anaconda, Jupyter notebook
• **Software :** Python **Python libraries/modules:** Pandas, Numpy, SkLearn ,XGboost etc