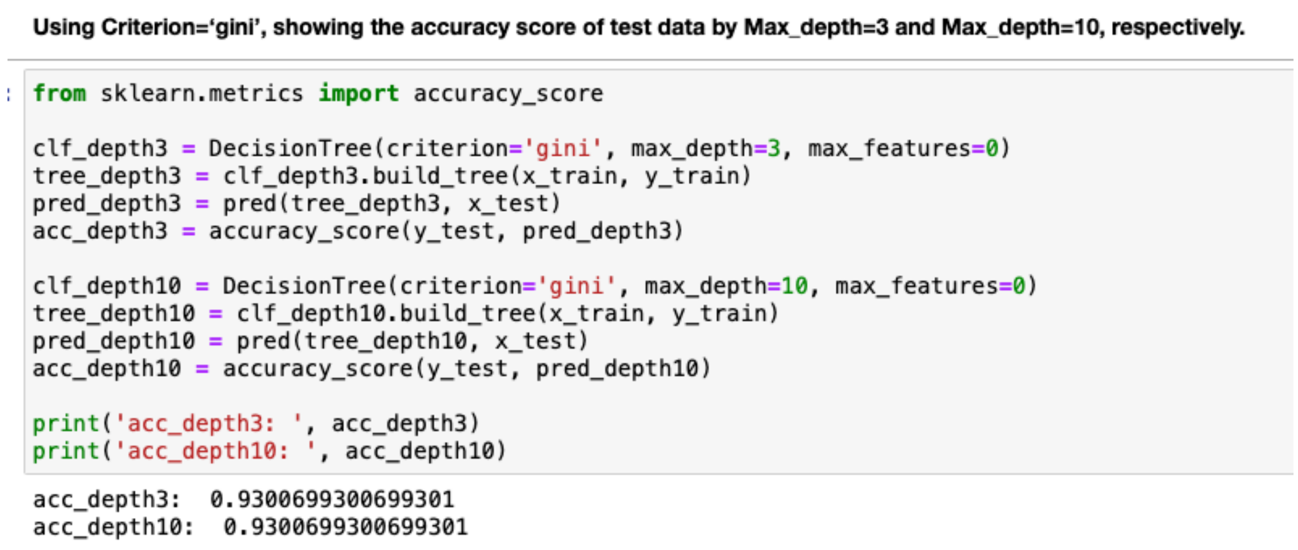
0616066_HW2 report

Experiment result

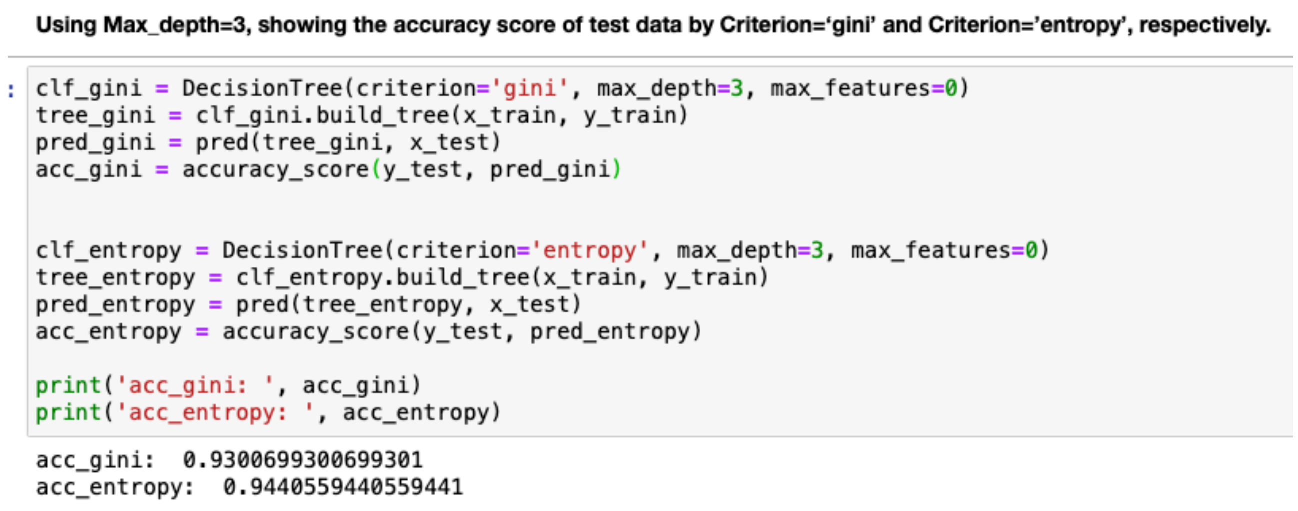
Compare different Criterion(gini and entropy):


Using Criterion='gini', showing the accuracy score of test data by Max_depth=3 and Max_depth=10, respectively.

```
from sklearn.metrics import accuracy_score

clf_depth3 = DecisionTree(criterion='gini', max_depth=3, max_features=0)
tree_depth3 = clf_depth3.build_tree(x_train, y_train)
pred_depth3 = pred(tree_depth3, x_test)
acc_depth3 = accuracy_score(y_test, pred_depth3)

clf_depth10 = DecisionTree(criterion='gini', max_depth=10, max_features=0)
tree_depth10 = clf_depth10.build_tree(x_train, y_train)
pred_depth10 = pred(tree_depth10, x_test)
acc_depth10 = accuracy_score(y_test, pred_depth10)

print('acc_depth3: ', acc_depth3)
print('acc_depth10: ', acc_depth10)
```

```
acc_depth3:  0.9300699300699301
acc_depth10:  0.9300699300699301
```

Compare different tree depth(3 and 10):


Using Max_depth=3, showing the accuracy score of test data by Criterion='gini' and Criterion='entropy', respectively.

```
clf_gini = DecisionTree(criterion='gini', max_depth=3, max_features=0)
tree_gini = clf_gini.build_tree(x_train, y_train)
pred_gini = pred(tree_gini, x_test)
acc_gini = accuracy_score(y_test, pred_gini)


clf_entropy = DecisionTree(criterion='entropy', max_depth=3, max_features=0)
tree_entropy = clf_entropy.build_tree(x_train, y_train)
pred_entropy = pred(tree_entropy, x_test)
acc_entropy = accuracy_score(y_test, pred_entropy)

print('acc_gini: ', acc_gini)
print('acc_entropy: ', acc_entropy)
```

```
acc_gini:  0.9300699300699301
acc_entropy:  0.9440559440559441
```

Compare different number of tree in random forest(10 and 100):

Using Criterion='gini', Max_depth=None, N_estimators=10, showing the accuracy score of test data by Max_features=sqrt(n_features) and Max_features=n_features, respectively.

```python
clf_random_features = RandomForest(n_estimators=10, max_features=np.sqrt(x_train.shape[1]))
clf_random_features_forest = clf_random_features.build_forest(x_train, y_train)
clf_random_features_forest_prediction = []
for tree in clf_random_features_forest:
    clf_random_features_forest_prediction.append(pred(tree, x_test))
clf_random_features_forest_prediction = pd.DataFrame(clf_random_features_forest_prediction)
clf_random_features_prediction = []
for cols in range(np.shape(clf_random_features_forest_prediction)[1]):
    col = clf_random_features_forest_prediction.iloc[:,cols]
    unique, counts = np.unique(col, return_counts=True)
    unique=list(unique)
    counts=list(counts)
    clf_random_features_prediction.append(unique[counts.index(max(counts))])
acc_clf_random_features = accuracy_score(y_test, clf_random_features_prediction)

print('random_features acc: ', acc_clf_random_features)


clf_all_features = RandomForest(n_estimators=10, max_features=x_train.shape[1])
clf_all_features_forest = clf_all_features.build_forest(x_train, y_train)
clf_all_features_forest_prediction = []
for tree in clf_all_features_forest:
    clf_all_features_forest_prediction.append(pred(tree, x_test))
clf_all_features_forest_prediction = pd.DataFrame(clf_all_features_forest_prediction)
clf_all_features_prediction = []
for cols in range(np.shape(clf_all_features_forest_prediction)[1]):
    col = clf_all_features_forest_prediction.iloc[:,cols]
    unique, counts = np.unique(col, return_counts=True)
    unique=list(unique)
    counts=list(counts)
    clf_all_features_prediction.append(unique[counts.index(max(counts))])
acc_clf_all_features = accuracy_score(y_test, clf_all_features_prediction)

print('all_features acc: ', acc_clf_all_features)
```

```
random_features acc:  0.958041958041958
all_features acc:  0.9440559440559441
```

Compare different max_featurein random forest

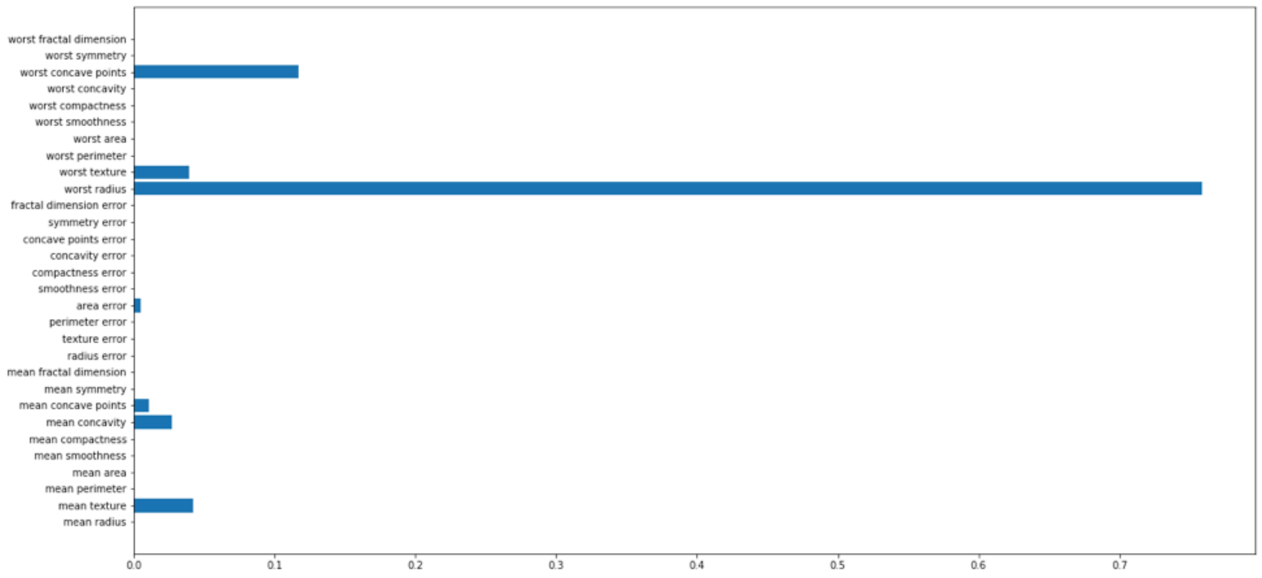
max_feature:The number of random select features to consider when looking for the best split

Using Criterion='gini', Max_depth=None, Max_features=sqrt(n_features), showing the accuracy score of test data by n_estimators=10 and n_estimators=100, respectively.

```python
clf_10tree = RandomForest(n_estimators=10, max_features=np.sqrt(x_train.shape[1]))
clf_10forest = clf_10tree.build_forest(x_train, y_train)
clf_10forest_prediction = []
for tree in clf_10forest:
    clf_10forest_prediction.append(pred(tree, x_test))
clf_10forest_prediction = pd.DataFrame(clf_10forest_prediction)
clf_10tree_prediction = []
for cols in range(np.shape(clf_10forest_prediction)[1]):
    col = clf_10forest_prediction.iloc[:,cols]
    unique, counts = np.unique(col, return_counts=True)
    unique=list(unique)
    counts=list(counts)
    clf_10tree_prediction.append(unique[counts.index(max(counts))])
acc_clf_10forest = accuracy_score(y_test, clf_10tree_prediction)

print('10_forest acc: ',acc_clf_10forest)

clf_100tree = RandomForest(n_estimators=100, max_features=np.sqrt(x_train.shape[1]))
clf_100forest = clf_100tree.build_forest(x_train, y_train)
clf_100forest_prediction = []
for tree in clf_100forest:
    clf_100forest_prediction.append(pred(tree, x_test))
clf_100forest_prediction = pd.DataFrame(clf_100forest_prediction)
clf_100tree_prediction = []
for cols in range(np.shape(clf_100forest_prediction)[1]):
    col = clf_100forest_prediction.iloc[:,cols]
    unique, counts = np.unique(col, return_counts=True)
    unique=list(unique)
    counts=list(counts)
    clf_100tree_prediction.append(unique[counts.index(max(counts))])
acc_clf_100forest = accuracy_score(y_test, clf_100tree_prediction)

print('100_forest acc: ',acc_clf_100forest)
```

```
10_forest acc:  0.9440559440559441
100_forest acc:  0.9230769230769231
```

Plot the feature importance(count feature used for splitting data)



Code:https://github.com/ryanlu2240/NCTU_AI_2021_spring/blob/master/HW2/0616066_HW2.ipynb