Random_forests

December 10, 2018

1 Random Forests

```
In [38]: import pandas as pd
         import matplotlib.pyplot as plt
         import numpy as np

         from IPython.display import set_matplotlib_formats
         set_matplotlib_formats('svg')

         from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import accuracy_score, balanced_accuracy_score, make_scorer
         from sklearn.metrics import roc_auc_score, roc_curve
         from sklearn.model_selection import train_test_split, RandomizedSearchCV
         from sklearn.model_selection import cross_val_score
```

```
In [39]: # Get features
         samples = pd.read_csv("samples.csv", index_col=0)
         microbes = pd.read_csv("microbes.csv", index_col=0)
         metabolites = pd.read_csv("metabolites.csv", index_col=0)
         combined_features = pd.concat([microbes, metabolites], axis=1)
         # Label vector
         labels = samples.case
```

1.1 Metabolites

```
In [64]: # We'll use 80% for training, and 20% for testing
         X_train, X_test, labels_train, labels_test = train_test_split(
             metabolites, labels, test_size=0.2, random_state=42)
```

Random forest

```
In [65]: rf = RandomForestClassifier(n_estimators=10, random_state=42)
         #scores = cross_val_score(rf, X_train, y_train, cv=5)
         #print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
         scores = cross_val_score(rf, X_train, labels_train, cv=5,
                                  scoring=make_scorer(roc_auc_score))
         print("AUC: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
AUC: 0.63 (+/- 0.20)
```

```
In [66]: # Train the best model with the entire training set
         rf.fit(X_train, labels_train)
         microbes_pred = rf.predict(X_test)
         balanced_accuracy = balanced_accuracy_score(labels_test, microbes_pred)
         accuracy = accuracy_score(labels_test, microbes_pred)
         auc = roc_auc_score(y_true=labels_test, y_score=[j for i, j in rf.predict_proba(X_tes
         print("Accuracy_score:", accuracy)
         print("Balanced accuracy score:", balanced_accuracy)
         print("AUC score:", auc)
```

```
Accuracy_score: 0.851851851852
Balanced accuracy score: 0.75
AUC score: 0.799342105263
```

1.2 Generate hyperparameter grid

```
In [67]: # Number of trees in random forest
         n_estimators = [int(x) for x in np.linspace(start = 10, stop = 2000, num = 10)]
         # Number of features to consider at every split
         max_features = ['auto', 'sqrt']
         # Maximum number of levels in tree
         max_depth = [int(x) for x in np.linspace(2, 110, num = 11)]
         max_depth.append(None)
         # Minimum number of samples required to split a node
         min_samples_split = [2, 5, 10]
         # Minimum number of samples required at each leaf node
         min_samples_leaf = [1, 2, 4]
         # Method of selecting samples for training each tree
         bootstrap = [True, False]

         # Create the random grid
         random_grid = {'n_estimators': n_estimators,
                        'max_features': max_features,
                        'max_depth': max_depth,
                        'min_samples_split': min_samples_split,
                        'min_samples_leaf': min_samples_leaf,
                        'bootstrap': bootstrap,
                        'random_state': [42]}

         random_grid
```

```
Out[67]: {'n_estimators': [10, 231, 452, 673, 894, 1115, 1336, 1557, 1778, 2000],
          'max_features': ['auto', 'sqrt'],
          'max_depth': [2, 12, 23, 34, 45, 56, 66, 77, 88, 99, 110, None],
          'min_samples_split': [2, 5, 10],
          'min_samples_leaf': [1, 2, 4],
          'bootstrap': [True, False],
          'random_state': [42]}
```

```
In [68]: # Use the random grid to search for best hyperparameters
         # First create the base model to tune
         rf = RandomForestClassifier()
         # Random search of parameters, using 3 fold cross validation,
         # search across 100 different combinations, and use all available cores
         rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
                                        n_iter = 100, cv = 5, verbose=2, random_state=42,
                                        n_jobs = -1, scoring=make_scorer(roc_auc_score))

         # Fit the random search model
         rf_random.fit(X_train, labels_train)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits



[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  33 tasks      | elapsed:   50.6s
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed:  3.8min
[Parallel(n_jobs=-1)]: Done 357 tasks      | elapsed:  7.6min
[Parallel(n_jobs=-1)]: Done 500 out of 500 | elapsed: 10.3min finished
/home/ravila/Software/miniconda/lib/python3.6/site-packages/sklearn/model_selection/_search.py
  DeprecationWarning)
```

```
Out[68]: RandomizedSearchCV(cv=5, error_score='raise-deprecating',
                   estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criteri
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False),
                   fit_params=None, iid='warn', n_iter=100, n_jobs=-1,
                   param_distributions={'n_estimators': [10, 231, 452, 673, 894, 1115, 1336, 15
                   pre_dispatch='2*n_jobs', random_state=42, refit=True,
                   return_train_score='warn', scoring=make_scorer(roc_auc_score),
                   verbose=2)
```

```
In [69]: rf_random.best_params_
```

```
Out[69]: {'random_state': 42,
          'n_estimators': 10,
          'min_samples_split': 2,
          'min_samples_leaf': 1,
          'max_features': 'sqrt',
          'max_depth': 45,
          'bootstrap': True}
```

```
In [70]: rf_random.best_score_
```

```
Out[70]: 0.62751831501831501
```

Default model

```
In [71]: base_model = RandomForestClassifier(n_estimators = 10, random_state = 42)
         base_model.fit(X_train, labels_train)
         pred = base_model.predict(X_test)

         balanced_accuracy = balanced_accuracy_score(labels_test, pred)
         accuracy = accuracy_score(labels_test, pred)
         auc = roc_auc_score(y_true=labels_test, y_score=[j for i, j in base_model.predict_prob
         print("Accuracy_score:", accuracy)
         print("Balanced accuracy score:", balanced_accuracy)
         print("AUC score:", auc)
```

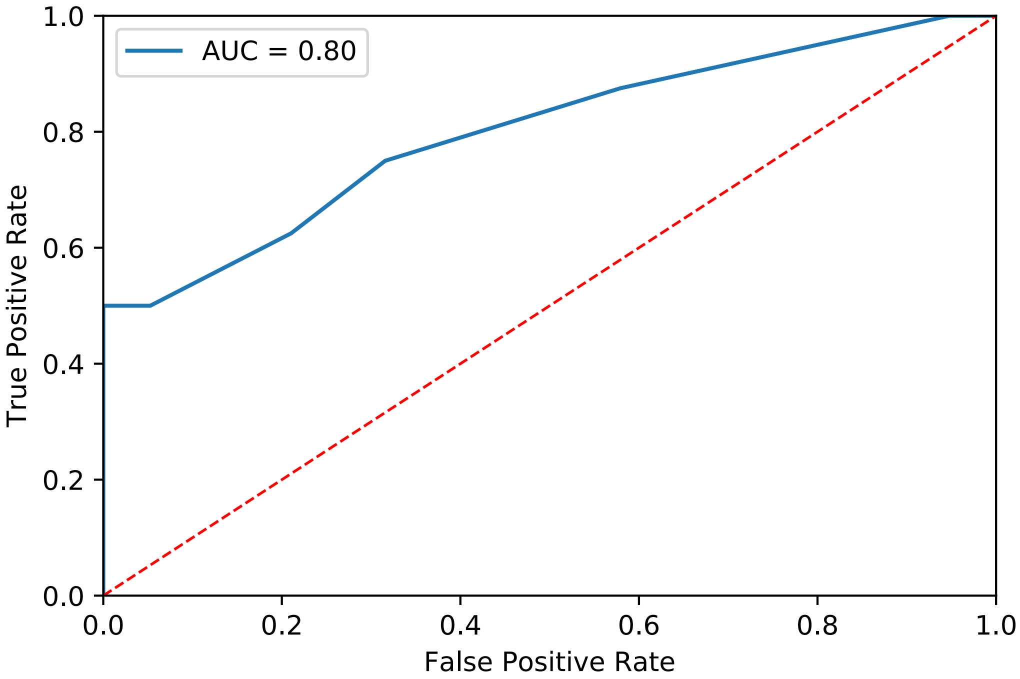
```
Accuracy_score: 0.851851851852
Balanced accuracy score: 0.75
AUC score: 0.799342105263
```

```
In [72]: rf_best = rf_random.best_estimator_
         rf_best.fit(X_train, labels_train)
         pred = rf_best.predict(X_test)

         balanced_accuracy = balanced_accuracy_score(labels_test, pred)
         accuracy = accuracy_score(labels_test, pred)
         auc = roc_auc_score(y_true=labels_test, y_score=[j for i, j in rf_best.predict_proba(
         print("Accuracy_score:", accuracy)
         print("Balanced accuracy score:", balanced_accuracy)
         print("AUC score:", auc)
```

```
Accuracy_score: 0.851851851852
Balanced accuracy score: 0.75
AUC score: 0.799342105263
```

```
In [73]: # Plot an ROC curve
         fpr, tpr, _ = roc_curve(y_true=labels_test, y_score=[j for i, j in rf_best.predict_pro
         auc = roc_auc_score(y_true=labels_test, y_score=[j for i, j in rf_best.predict_proba(
         plt.plot(fpr, tpr, label="AUC = {0:.2f}".format(auc))
         plt.plot([0, 1], [0, 1], color='red', lw=1, linestyle='--')
         plt.xlim([0.0, 1.0])
         plt.ylim([0.0, 1.0])
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.legend()
         plt.show()
```


AUC = 0.80
True Positive Rate
False Positive Rate