```python
# imports
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score, precision_score, recall_score
from bootstrap import bootstrap

def run_random_forests(X, y, n_trials=5):
    """
    Runs random forests for a dataset
    5 trials of random forests with 5-fold cross validation and a gridsearch over hyperparameters:
        max_depth, max_features
    and computes mean accuracy over metrics
        accuracy, f1, roc, precision, recall

    parameters
    ----------
    X: feature vector
    y: target vector
    n_trials: number of trials to run

    returns
    --------
    train_metrics: average of each metric on training set across 5 trials
    test_metrics: average of each metric on test set across 5 trials
    hyperp: dataframe of hyperparameters tried during hyperparameter search,
            of metrics (acc, f1, roc) and their mean performance during cross-validation
    """

    # hyperparameters
    depth_list = [2,4,6,8,16,20,30,None]
    feature_list = [1,2,4,6,8,12,16,20]
    params = {'max_depth':depth_list, 'max_features':feature_list}

    # metric evaluation scores
    scores = ['accuracy', 'f1', 'roc_auc', 'precision', 'recall']

    # to hold calculated metric performances
    train_metrics = []
    test_metrics = []

    for trial in range(n_trials):

        # initialize model for cross validation grid search
        RF = RandomForestClassifier()
        GS = GridSearchCV(RF, params, scoring=scores, refit=False)
```

```
        GS = GridSearchCV(RF, params, scoring=scores, refit=False)

        # bootstrap training and testing sets
        X_train, X_test, y_train, y_test = bootstrap(X,y)
        GS.fit(X_train, y_train)

        # collect results and get dataframe to visualize hyperparameter
search
        res = GS.cv_results_
        hyperp = pd.DataFrame(res['params'])
        hyperp['acc'] = res['mean_test_accuracy']
        hyperp['f1'] = res['mean_test_f1']
        hyperp['roc'] = res['mean_test_roc_auc']

        test_per = [] # test set performances
        train_per = [] # train set performances

        # get best hyperparameters for each metric and use on test set
        for s in scores:

            # train rf with best hyperparameters for metric
            best_p = res['params'][np.argmax(res['mean_test_{}'.format(s)])]
            RF = RandomForestClassifier(max_depth=best_p['max_depth'],
max_features=best_p['max_features'])
            RF.fit(X_train, y_train)

            # predictions for train and test sets
            y_pred = RF.predict(X_test)
            y_pred_train = RF.predict(X_train)

            # evaluate metric on test set
            if s == 'accuracy':
                test_per.append(accuracy_score(y_test, y_pred))
                train_per.append(accuracy_score(y_train, y_pred_train))
            elif s == 'f1':
                test_per.append(f1_score(y_test, y_pred))
                train_per.append(f1_score(y_train, y_pred_train))
            elif s == 'roc_auc':
                test_per.append(roc_auc_score(y_test, y_pred))
                train_per.append(roc_auc_score(y_train, y_pred_train))
            elif s == 'precision':
                test_per.append(precision_score(y_test, y_pred))
                train_per.append(precision_score(y_train, y_pred_train))
            elif s == 'recall':
                test_per.append(recall_score(y_test, y_pred))
                train_per.append(recall_score(y_train, y_pred_train))

        train_metrics.append(train_per)
        test_metrics.append(test_per)

```

```
95          print('Trial {} done'.format(trial+1))
96
97      # take mean of each metric across 5 trials
98      train_metrics = np.mean(np.array(train_metrics), axis=0)
99      test_metrics = np.mean(np.array(test_metrics), axis=0)
100
101     return train_metrics, test_metrics, hyperp
```