```python
# imports
import pandas as pd
import numpy as np
from sklearn.svm import SVC
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_svm(X, y, n_trials=5):
    """
    Runs svm for a dataset
    5 trials of SVM with 5-fold cross validation and a gridsearch over hyperparameter: C
    and computes mean accuracy over metrics
        accuracy, f1, roc, precision, recall
    No cross-val hyperparameters will be returned because of the high dimensionality of
    hyperparameters tried here

    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

    """

    # hyperparameters
    C_list = [1e-7,1e-6,1e-5,1e-4,1e-3,1e-2,1e-1,1e0,1e1,1e2,1e3,1e4]
    kernel_list = ('rbf', 'linear', 'poly')
    degree_list = [2,3]
    gamma_list = [1e-6,1e-5,1e-4,1e-3,1e-2,1e-1]
    params = {'C': C_list, 'gamma':gamma_list, 'kernel':kernel_list, 'degree':degree_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
        SVM = SVC(max_iter=2000)
        GS = GridSearchCV(SVM, 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
        res = GS.cv_results_

        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)])]
            SVM = SVC(max_iter=2000, kernel=best_p['kernel'], C=best_p['C'],
degree=best_p['degree'], gamma=best_p['gamma'])
            SVM.fit(X_train, y_train)

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

        print('Trial {} done'.format(trial+1))

```

```
    # take mean of each metric across 5 trials
    train_metrics = np.mean(np.array(train_metrics), axis=0)
    test_metrics = np.mean(np.array(test_metrics), axis=0)

    return train_metrics, test_metrics
```