```python
    :param clf: classifacador que se deseja usar no aprendizado
    :return: acurácias de treino e teste, f-beta scores de treino e teste e o objeto classificador
    '''

    import warnings
    warnings.filterwarnings("ignore")   #pra não cagar o meu log :D


    classificacao = []
    for i in range(0, int(df.shape[0] / 300)):  # gambiarra para confirmação binária de acerto
        classificacao += [i + 1] * 300

    classi = pd.DataFrame(classificacao)
    X_train, X_test, y_train, y_test = train_test_split(df, classi, test_size=0.3, random_state=0)
    '''
    classifiers = [DecisionTreeClassifier(random_state=20),AdaBoostClassifier(random_state=20),
                   svm.SVC(kernel='linear', C=1, random_state=20),RandomForestClassifier(random_state=20),
                   GaussianNB(),KNeighborsClassifier(), SGDClassifier(random_state=20),
                   LogisticRegression(random_state=20)]
    '''
    print("\nClassificador: {}\n".format(clf.__class__.__name__))
    clf = clf.fit(X_train, y_train)
    clf_test_predictions = clf.predict(X_test)
    clf_train_predictions = clf.predict(X_train)
    acc_train_results = accuracy_score(y_train, clf_train_predictions)
    acc_test_results = accuracy_score(y_test, clf_test_predictions)

    fscore_train_results = fbeta_score(y_train, clf_train_predictions, beta=0.5, average='macro')
    fscore_test_results = fbeta_score(y_test, clf_test_predictions, beta=0.5, average='macro')
    return(acc_train_results,acc_test_results, fscore_train_results,fscore_test_results,clf)

```