hw_1

October 7, 2019

[13]:
```
from sklearn import tree
X = [[0, 0], [1, 1]]
y = [0, 1]
clf = tree.DecisionTreeClassifier(criterion='entropy')
clf = clf.fit(X, y)
clf.predict([[2, 2]])
```

[13]:
```
array([1])
```

[14]:
```
import numpy as np
from sklearn.linear_model import SGDClassifier


X_train = np.load('madelon/train-X.npy')
y_train = np.load('madelon/train-y.npy')
X_test = np.load('madelon/test-X.npy')
y_test = np.load('madelon/test-y.npy')
cv_train_X0 = np.load('madelon/cv-train-X.0.npy')
cv_train_X1 = np.load('madelon/cv-train-X.1.npy')
cv_train_X2 = np.load('madelon/cv-train-X.2.npy')
cv_train_X3 = np.load('madelon/cv-train-X.3.npy')
cv_train_X4 = np.load('madelon/cv-train-X.4.npy')
cv_heldout_X0 = np.load('madelon/cv-heldout-X.0.npy')
cv_heldout_X1 = np.load('madelon/cv-heldout-X.1.npy')
cv_heldout_X2 = np.load('madelon/cv-heldout-X.2.npy')
cv_heldout_X3 = np.load('madelon/cv-heldout-X.3.npy')
cv_heldout_X4 = np.load('madelon/cv-heldout-X.4.npy')
cv_train_y0 = np.load('madelon/cv-train-y.0.npy')
cv_train_y1 = np.load('madelon/cv-train-y.1.npy')
cv_train_y2 = np.load('madelon/cv-train-y.2.npy')
cv_train_y3 = np.load('madelon/cv-train-y.3.npy')
cv_train_y4 = np.load('madelon/cv-train-y.4.npy')
cv_heldout_y0 = np.load('madelon/cv-heldout-y.0.npy')
cv_heldout_y1 = np.load('madelon/cv-heldout-y.1.npy')
cv_heldout_y2 = np.load('madelon/cv-heldout-y.2.npy')
cv_heldout_y3 = np.load('madelon/cv-heldout-y.3.npy')
cv_heldout_y4 = np.load('madelon/cv-heldout-y.4.npy')
x_train_list = [cv_train_X0, cv_train_X1, cv_train_X2, cv_train_X3, cv_train_X4]
```

```
y_train_list = [cv_train_y0, cv_train_y1, cv_train_y2, cv_train_y3, cv_train_y4]
x_heldout_list = [cv_heldout_X0, cv_heldout_X1, cv_heldout_X2, cv_heldout_X3,␣
 ↪cv_heldout_X4]
y_heldout_list = [cv_heldout_y0, cv_heldout_y1, cv_heldout_y2, cv_heldout_y3,␣
 ↪cv_heldout_y4]
#print(x_train_list[1].shape)
#print(X_train)

#print(sum(y_train_list).shape)
#print((sum(y_train_list)-y_train_list[4]).shape)
#print(y_heldout_list[1].shape)
def train_and_evaluate_sgd(X_train, y_train, X_test, y_test):
    model = SGDClassifier(loss= 'log', max_iter=10000, random_state= 1)
    #model.fit(X_train, y_train)
    #model.fit(sum(x_train_list), sum(y_train_list))
    #print(model.score(X_test, y_test))
    #print(X_train.shape)
    accuracy_heldout_sum = 0
    accuracy_train_sum = 0
    accuracy_train_list = []
    accuracy_heldout_list =[]
    for i in range(5):
        model.fit(x_train_list[i], y_train_list[i])
        train_score = model.score(x_train_list[i], y_train_list[i])
        heldout_score = model.score(x_heldout_list[i], y_heldout_list[i])
        accuracy_train_list.append(train_score)
        accuracy_heldout_list.append(heldout_score)
        accuracy_train_sum += train_score
        accuracy_heldout_sum += heldout_score
    print(accuracy_heldout_list)
    accuracy_train_std = np.std(np.asarray(accuracy_train_list), ddof=1)
    accuracy_heldout_std = np.std(np.asarray(accuracy_heldout_list), ddof =1)
    accuracy_train = accuracy_train_sum/5
    accuracy_heldout = accuracy_heldout_sum/5
    print(accuracy_train)
    print(accuracy_heldout)
    print(accuracy_train_std)
    print(accuracy_heldout_std)
    accuracy_train_interval = (accuracy_train-2.776*accuracy_train_std/np.
 ↪sqrt(5), accuracy_train+2.776*accuracy_train_std/np.sqrt(5))
    accuracy_heldout_interval = (accuracy_heldout-2.776*accuracy_heldout_std/np.
 ↪sqrt(5), accuracy_heldout+2.776*accuracy_heldout_std/np.sqrt(5))
    print(accuracy_train_interval)
    print(accuracy_heldout_interval)
    model.fit(X_train, y_train)
    accuracy_test = model.score(X_test, y_test)
    print(accuracy_test)
```

```
    return (accuracy_train, accuracy_train_std, accuracy_heldout,␣
↪accuracy_heldout_std, accuracy_test)
```

[15]:
```
sgd_result = train_and_evaluate_sgd(X_train, y_train, X_test, y_test)
sgd_train_acc = sgd_result[0]
sgd_train_std = sgd_result[1]
sgd_heldout_acc = sgd_result[2]
sgd_heldout_std = sgd_result[3]
sgd_test_acc = sgd_result[4]
```

```
[0.5175, 0.515, 0.5, 0.5775, 0.5125]
0.548125
0.5245
0.08285670989726786
0.030382972204838696
(0.4452612995269954, 0.6509887004730046)
(0.48678060520103744, 0.5622193947989624)
0.5566666666666666
```

[16]:
```
from sklearn.tree import DecisionTreeClassifier

def train_and_evaluate_decision_tree(X_train, y_train, X_test, y_test):
    model = DecisionTreeClassifier(criterion='entropy', random_state= 1)
    accuracy_heldout_sum = 0
    accuracy_train_sum = 0
    accuracy_train_list = []
    accuracy_heldout_list =[]
    for i in range(5):
        model.fit(x_train_list[i], y_train_list[i])
        train_score = model.score(x_train_list[i], y_train_list[i])
        heldout_score = model.score(x_heldout_list[i], y_heldout_list[i])
        accuracy_train_list.append(train_score)
        accuracy_heldout_list.append(heldout_score)
        accuracy_train_sum += train_score
        accuracy_heldout_sum += heldout_score
    accuracy_train_std = np.std(np.asarray(accuracy_train_list), ddof=1)
    accuracy_heldout_std = np.std(np.asarray(accuracy_heldout_list), ddof =1)
    accuracy_train = accuracy_train_sum/5
    accuracy_heldout = accuracy_heldout_sum/5
    print(accuracy_train)
    print(accuracy_heldout)
    #print(accuracy_train_std)
    #print(accuracy_heldout_std)
    accuracy_train_interval = (accuracy_train-2.776*accuracy_train_std/np.
↪sqrt(5), accuracy_train+2.776*accuracy_train_std/np.sqrt(5))
    accuracy_heldout_interval = (accuracy_heldout-2.776*accuracy_heldout_std/np.
↪sqrt(5), accuracy_heldout+2.776*accuracy_heldout_std/np.sqrt(5))
    print(accuracy_train_interval)
```

```
    print(accuracy_heldout_interval)
    model.fit(X_train, y_train)
    accuracy_test = model.score(X_test, y_test)
    print(accuracy_test)

    return (accuracy_train, accuracy_train_std, accuracy_heldout,␣
↪accuracy_heldout_std, accuracy_test)
```

[17]:
```
dt_result = train_and_evaluate_decision_tree(X_train, y_train, X_test, y_test)
dt_train_acc = dt_result[0]
dt_train_std = dt_result[1]
dt_heldout_acc = dt_result[2]
dt_heldout_std = dt_result[3]
dt_test_acc = dt_result[4]
```

```
1.0
0.7595
(1.0, 1.0)
(0.7239498542337728, 0.7950501457662271)
0.7366666666666667
```

[18]:
```
def train_and_evaluate_decision_stump(X_train, y_train, X_test, y_test):
    model = DecisionTreeClassifier(criterion='entropy', max_depth=4,␣
↪random_state=1)
    accuracy_heldout_sum = 0
    accuracy_train_sum = 0
    accuracy_train_list = []
    accuracy_heldout_list =[]
    for i in range(5):
        model.fit(x_train_list[i], y_train_list[i])
        train_score = model.score(x_train_list[i], y_train_list[i])
        heldout_score = model.score(x_heldout_list[i], y_heldout_list[i])
        accuracy_train_list.append(train_score)
        accuracy_heldout_list.append(heldout_score)
        accuracy_train_sum += train_score
        accuracy_heldout_sum += heldout_score
    accuracy_train_std = np.std(np.asarray(accuracy_train_list), ddof=1)
    accuracy_heldout_std = np.std(np.asarray(accuracy_heldout_list), ddof =1)
    accuracy_train = accuracy_train_sum/5
    accuracy_heldout = accuracy_heldout_sum/5
    print(accuracy_train)
    print(accuracy_heldout)
    #print(accuracy_train_std)
    #print(accuracy_heldout_std)
    accuracy_train_interval = (accuracy_train-2.776*accuracy_train_std/np.
↪sqrt(5), accuracy_train+2.776*accuracy_train_std/np.sqrt(5))
    accuracy_heldout_interval = (accuracy_heldout-2.776*accuracy_heldout_std/np.
↪sqrt(5), accuracy_heldout+2.776*accuracy_heldout_std/np.sqrt(5))
```

```
    print(accuracy_train_interval)
    print(accuracy_heldout_interval)
    model.fit(X_train, y_train)
    accuracy_test = model.score(X_test, y_test)
    print(accuracy_test)
    return (accuracy_train, accuracy_train_std, accuracy_heldout,␣
↪accuracy_heldout_std, accuracy_test)
```

[19]:
```
dt4_result = train_and_evaluate_decision_stump(X_train, y_train, X_test, y_test)
dt4_train_acc = dt4_result[0]
dt4_train_std = dt4_result[1]
dt4_heldout_acc = dt4_result[2]
dt4_heldout_std = dt4_result[3]
dt4_test_acc = dt4_result[4]
```

```
0.783125
0.7645000000000001
(0.766895561084868, 0.7993544389151319)
(0.7410267900789006, 0.7879732099210995)
0.7516666666666667
```

[20]:
```
def train_and_evaluate_sgd_with_stumps(X_train, y_train, X_test, y_test):

    accuracy_heldout_sum = 0
    accuracy_train_sum = 0
    accuracy_train_list = []
    accuracy_heldout_list =[]
    for i in range(5):
        X_train_features_full = x_train_list[i]
        X_test_features_full = x_heldout_list[i]
        X_train_full_half_features_list =[]
        X_test_full_half_features_list = []
        model_list = []
        X_train_full_new = np.zeros([1600,1])
        X_test_full_new = np.zeros([400,1])
        for j in range(50):
            model = DecisionTreeClassifier(criterion='entropy', max_depth= 4)
            X_train_features_full_rot90 = np.rot90(X_train_features_full, k =1)
            X_test_features_full_rot90 = np.rot90(X_test_features_full, k =1)
            randnum = np.random.randint(0, 10000)
            np.random.seed(randnum)
            np.random.shuffle(X_train_features_full_rot90)
            np.random.seed(randnum)
            np.random.shuffle(X_test_features_full_rot90)
            X_train_features_full_half = np.rot90(X_train_features_full_rot90,␣
↪k = -1)
            X_test_features_full_half = np.rot90(X_test_features_full_rot90, k␣
↪= -1)
```

```
            X_train_features_full_half = np.hsplit(X_train_features_full_half,␣
↪2)[0]
            X_test_features_full_half = np.hsplit(X_test_features_full_half,␣
↪2)[0]
            X_test_full_half_features_list.append(X_test_features_full_half)
            X_train_full_half_features_list.append(X_train_features_full_half)
            model = model.fit(X_train_features_full_half, y_train_list[i])
            features = model.predict(X_train_features_full_half)
            test_features = model.predict(X_test_features_full_half)
            features = features.reshape((1600, 1))
            test_features = test_features.reshape(400, 1)
            X_train_full_new = np.append(X_train_full_new, features, axis= 1)
            X_test_full_new = np.append(X_test_full_new, test_features, axis= 1)
        X_train_full_new = X_train_full_new[:,1:]
        X_test_full_new = X_test_full_new[:,1:]
        model = SGDClassifier(loss= 'log', max_iter=10000, random_state= 1)
        model = model.fit(X_train_full_new, y_train_list[i])
        train_score = model.score(X_train_full_new, y_train_list[i])
        heldout_score = model.score(X_test_full_new, y_heldout_list[i])
        accuracy_train_list.append(train_score)
        accuracy_heldout_list.append(heldout_score)
        accuracy_train_sum += train_score
        accuracy_heldout_sum += heldout_score
    accuracy_train_std = np.std(np.asarray(accuracy_train_list), ddof=1)
    accuracy_heldout_std = np.std(np.asarray(accuracy_heldout_list), ddof =1)
    accuracy_train = accuracy_train_sum/5
    accuracy_heldout = accuracy_heldout_sum/5
    print(accuracy_heldout_list)
    print(accuracy_train)
    print(accuracy_heldout)
    print(accuracy_train_std)
    print(accuracy_heldout_std)
    accuracy_train_interval = (accuracy_train-2.776*accuracy_train_std/np.
↪sqrt(5), accuracy_train+2.776*accuracy_train_std/np.sqrt(5))
    accuracy_heldout_interval = (accuracy_heldout-2.776*accuracy_heldout_std/np.
↪sqrt(5), accuracy_heldout+2.776*accuracy_heldout_std/np.sqrt(5))
    print(accuracy_train_interval)
    print(accuracy_heldout_interval)

    X_train_features_full = X_train
    X_test_features_full = X_test
    X_train_full_half_features_list =[]
    X_test_full_half_features_list = []
    model_list = []
    X_train_full_new = np.zeros([2000,1])
    X_test_full_new = np.zeros([600,1])
    for i in range(50):
```

```
        model = DecisionTreeClassifier(criterion='entropy', max_depth=4)
        X_train_features_full_rot90 = np.rot90(X_train_features_full, k = 1)
        X_test_features_full_rot90 = np.rot90(X_test_features_full, k = 1)
        randnum = np.random.randint(0,10000)
        np.random.seed(randnum)
        np.random.shuffle(X_train_features_full_rot90)
        np.random.seed(randnum)
        np.random.shuffle(X_test_features_full_rot90)
        X_train_features_full_half = np.rot90(X_train_features_full_rot90, k =␣
↪-1)
        X_test_features_full_half = np.rot90(X_test_features_full_rot90, k = -1)
        X_train_features_full_half = np.hsplit(X_train_features_full_half, 2)[0]
        X_test_features_full_half = np.hsplit(X_test_features_full_half, 2)[0]
        X_test_full_half_features_list.append(X_test_features_full_half)
        X_train_full_half_features_list.append(X_train_features_full_half)
        model = model.fit(X_train_features_full_half, y_train)
        model_list.append(model)
        features = model.predict(X_train_features_full_half)
        test_features = model.predict(X_test_features_full_half)
        features = features.reshape((2000, 1))
        test_features = test_features.reshape(600, 1)
        X_train_full_new = np.append(X_train_full_new, features, axis= 1)
        X_test_full_new = np.append(X_test_full_new, test_features, axis= 1)
    X_train_full_new = X_train_full_new[:,1:]
    X_test_full_new = X_test_full_new[:,1:]
    model = SGDClassifier(loss= 'log', max_iter=10000)
    model = model.fit(X_train_full_new, y_train)
    accuracy_test = model.score(X_test_full_new, y_test)
    print(accuracy_test)

    return (accuracy_train, accuracy_train_std, accuracy_heldout,␣
↪accuracy_heldout_std, accuracy_test)
```

[21]:
```
stumps_result = train_and_evaluate_sgd_with_stumps(X_train, y_train, X_test,␣
↪y_test)
stumps_train_acc = stumps_result[0]
stumps_train_std = stumps_result[1]
stumps_heldout_acc = stumps_result[2]
stumps_heldout_std = stumps_result[3]
stumps_test_acc = stumps_result[4]
```

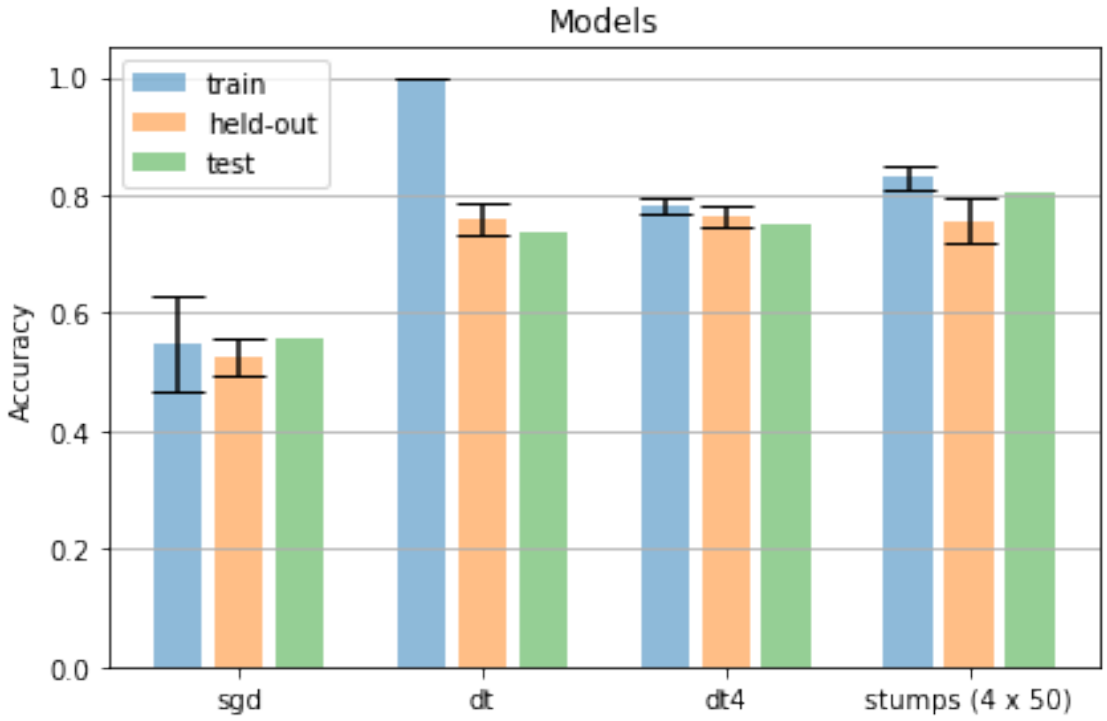
```
[0.745, 0.7975, 0.71, 0.7375, 0.7925]
0.8296250000000001
0.7565000000000001
0.02076938100907198
0.037524991672217595
(0.8038405416287253, 0.8554094583712748)
(0.7099140384235767, 0.8030859615764234)
```

0.8033333333333333

[22]:
```
import os
import matplotlib.pyplot as plt


def plot_results(sgd_train_acc, sgd_train_std, sgd_heldout_acc,␣
↪sgd_heldout_std, sgd_test_acc,
                 dt_train_acc, dt_train_std, dt_heldout_acc, dt_heldout_std,␣
↪dt_test_acc,
                 dt4_train_acc, dt4_train_std, dt4_heldout_acc,␣
↪dt4_heldout_std, dt4_test_acc,
                 stumps_train_acc, stumps_train_std, stumps_heldout_acc,␣
↪stumps_heldout_std, stumps_test_acc):
    """
    Plots the final results from problem 2. For each of the 4 classifiers, pass
    the training accuracy, training standard deviation, held-out accuracy,␣
↪held-out
    standard deviation, and testing accuracy.

    Although it should not be necessary, feel free to edit this method.
    """
    train_x_pos = [0, 4, 8, 12]
    cv_x_pos = [1, 5, 9, 13]
    test_x_pos = [2, 6, 10, 14]
    ticks = cv_x_pos

    labels = ['sgd', 'dt', 'dt4', 'stumps (4 x 50)']

    train_accs = [sgd_train_acc, dt_train_acc, dt4_train_acc, stumps_train_acc]
    train_errors = [sgd_train_std, dt_train_std, dt4_train_std,␣
↪stumps_train_std]

    cv_accs = [sgd_heldout_acc, dt_heldout_acc, dt4_heldout_acc,␣
↪stumps_heldout_acc]
    cv_errors = [sgd_heldout_std, dt_heldout_std, dt4_heldout_std,␣
↪stumps_heldout_std]

    test_accs = [sgd_test_acc, dt_test_acc, dt4_test_acc, stumps_test_acc]

    fig, ax = plt.subplots()
    ax.bar(train_x_pos, train_accs, yerr=train_errors, align='center', alpha=0.
↪5, ecolor='black', capsize=10, label='train')
    ax.bar(cv_x_pos, cv_accs, yerr=cv_errors, align='center', alpha=0.5,␣
↪ecolor='black', capsize=10, label='held-out')
    ax.bar(test_x_pos, test_accs, align='center', alpha=0.5, capsize=10,␣
↪label='test')
```

```
    ax.set_ylabel('Accuracy')
    ax.set_xticks(ticks)
    ax.set_xticklabels(labels)
    ax.set_title('Models')
    ax.yaxis.grid(True)
    ax.legend()
    plt.tight_layout()
```

[23]:
```
plot_results(sgd_train_acc, sgd_train_std, sgd_heldout_acc, sgd_heldout_std,␣
↪sgd_test_acc,
                dt_train_acc, dt_train_std, dt_heldout_acc, dt_heldout_std,␣
↪dt_test_acc,
                dt4_train_acc, dt4_train_std, dt4_heldout_acc,␣
↪dt4_heldout_std, dt4_test_acc,
                stumps_train_acc, stumps_train_std, stumps_heldout_acc,␣
↪stumps_heldout_std, stumps_test_acc)
```



[24]:
```
file = open('badges/train.names.txt', 'r')
names_text = file.read()
file.close
names = names_text.splitlines()

def compute_features(names):
    names_split = []
```

```
    for i in range(len(names)):
        names_split.append(names[i].split())
        names_split[i][0] = list(names_split[i][0])
        names_split[i][1] = list(names_split[i][1])
    charactors = list('abcdefghijklmnopqrstuvwxyz')
    res = []
    for name in names_split:
        first_name = name[0]
        last_name = name[1]
        first_features = []
        while len(first_name) < 5 :
            first_name.append('_')
        for i in range(5):
            for j in range(26):
                if first_name[i] == charactors[j]:
                    first_features.append(1)
                else:
                    first_features.append(0)
        last_features = []
        while len(last_name) < 5 :
            last_name.append('_')
        for i in range(5):
            for j in range(26):
                if last_name[i] == charactors[j]:
                    last_features.append(1)
                else:
                    last_features.append(0)
        first_features.extend(last_features)
        res.append(first_features)
    features_array = np.array(res)
    return features_array
```

[25]:
```
train_names = compute_features(names)
np.save('badges/train_names.npy', train_names)
file = open('badges/test.names.txt', 'r')
names_text = file.read()
file.close
names = names_text.splitlines()
test_names = compute_features(names)
np.save('badges/test_names.npy', test_names)
file = open('badges/hidden-test.names.txt', 'r')
names_text = file.read()
file.close
names = names_text.splitlines()
hidden_test_names = compute_features(names)
np.save('badges/hidden_test_names.npy', hidden_test_names)
```

[ ]:
```

```

[26]:
```
train_labels = np.load('badges/train.labels.npy')
test_labels = np.load('badges/test.labels.npy')
train_names = np.load('badges/train_names.npy')
test_names = np.load('badges/test_names.npy')
hidden_test = np.load('badges/hidden_test_names.npy')
model = SGDClassifier(loss= 'log', max_iter=10000)
model.fit(train_names, train_labels)
sgd_score = model.score(test_names, test_labels)
print(sgd_score)
model = DecisionTreeClassifier(criterion='entropy')
model.fit(train_names, train_labels)
dt_score = model.score(test_names, test_labels)
print(dt_score)
model = DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=1)
model.fit(train_names, train_labels)
dt4_score = model.score(test_names, test_labels)
print(dt4_score)
X_train_features_full = np.copy(train_names)
X_test_features_full = np.copy(test_names)
X_train_full_new = np.zeros([1000,1])
X_test_full_new = np.zeros([1000,1])
for i in range(50):
    model = DecisionTreeClassifier(criterion='entropy', max_depth=4)
    X_train_features_full_transpose = np.transpose(X_train_features_full)
    X_test_features_full_transpose = np.transpose(X_test_features_full)
    randnum = np.random.randint(0,10000)
    np.random.seed(randnum)
    np.random.shuffle(X_train_features_full_transpose)
    np.random.seed(randnum)
    np.random.shuffle(X_test_features_full_transpose)
    X_train_features_full_half = np.transpose(X_train_features_full_transpose)
    X_test_features_full_half = np.transpose(X_test_features_full_transpose)
    X_train_features_full_half = np.hsplit(X_train_features_full_half, 2)[0]
    X_test_features_full_half = np.hsplit(X_test_features_full_half, 2)[0]
    model.fit(X_train_features_full_half, train_labels)
    features = model.predict(X_train_features_full_half)
    test_features = model.predict(X_test_features_full_half)
    features = features.reshape((1000, 1))
    test_features = test_features.reshape(1000, 1)
    X_train_full_new = np.append(X_train_full_new, features, axis= 1)
    X_test_full_new = np.append(X_test_full_new, test_features, axis= 1)
X_train_full_new = X_train_full_new[:,1:]
X_test_full_new = X_test_full_new[:,1:]
model = SGDClassifier(loss= 'log', max_iter=10000)
model.fit(X_train_full_new, train_labels)
stumps_score = model.score(X_test_full_new, test_labels)
print(stumps_score)
```

```
0.649
0.618
0.66
0.643
```

[ ]:

[27]:
```
def compute_new_features(names):
    names_split = []
    for i in range(len(names)):
        names_split.append(names[i].split())
        names_split[i][0] = list(names_split[i][0])
        names_split[i][1] = list(names_split[i][1])
    charactors = list('aeiou')
    res = []
    for name in names_split:
        first_name = name[0]
        last_name = name[1]
        first_features = []
        while len(first_name) < 7 :
            first_name.append('_')
        for i in range(7):
            if first_name[i] in charactors:
                first_features.append(1)
            else:
                first_features.append(0)
        last_features = []
        while len(last_name) < 7 :
            last_name.append('_')
        for i in range(7):
            if last_name[i] in charactors:
                last_features.append(1)
            else:
                last_features.append(0)
        first_features.extend(last_features)
        res.append(first_features)
    features_array = np.array(res)
    print(features_array.shape)
    return features_array
```

[28]:
```
file = open('badges/train.names.txt', 'r')
names_text = file.read()
file.close()
names = names_text.splitlines()
train_names_final = compute_new_features(names)
np.save('badges/train_names_final.npy', train_names_final)
file = open('badges/test.names.txt', 'r')
names_text = file.read()
```

```
file.close()
names = names_text.splitlines()
test_names_final = compute_new_features(names)
np.save('badges/test_names_final', test_names_final)
print(test_names_final.shape)
file = open('badges/hidden-test.names.txt', 'r')
names_text = file.read()
file.close()
names = names_text.splitlines()
hidden_test_names_final = compute_new_features(names)
np.save('badges/hidden_test_names_final', hidden_test_names_final)
```

```
(1000, 14)
(1000, 14)
(1000, 14)
(1000, 14)
```

[29]:
```
train_labels = np.load('badges/train.labels.npy')
test_labels = np.load('badges/test.labels.npy')
train_names = np.load('badges/train_names_final.npy')
test_names = np.load('badges/test_names_final.npy')
hidden_test = np.load('badges/hidden_test_names_final.npy')
model = SGDClassifier(loss= 'log', max_iter=10000, random_state= 1)
model.fit(train_names, train_labels)
sgd_score = model.score(test_names, test_labels)
print(sgd_score)
model = DecisionTreeClassifier(criterion='entropy', random_state= 1)
model.fit(train_names, train_labels)
dt_score = model.score(test_names, test_labels)
print(dt_score)
model = DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=1)
model.fit(train_names, train_labels)
dt4_score = model.score(test_names, test_labels)
print(dt4_score)
X_train_features_full = np.copy(train_names)
X_test_features_full = np.copy(test_names)
X_train_full_half_features_list =[]
X_test_full_half_features_list = []
model_list = []
X_train_full_new = np.zeros([1000,1])
X_test_full_new = np.zeros([1000,1])
for i in range(50):
    model = DecisionTreeClassifier(criterion='entropy', max_depth=4)
    X_train_features_full_transpose = np.transpose(X_train_features_full)
    X_test_features_full_transpose = np.transpose(X_test_features_full)
    randnum = np.random.randint(0,10000)
    np.random.seed(randnum)
```

```
    np.random.shuffle(X_train_features_full_transpose)
    np.random.seed(randnum)
    np.random.shuffle(X_test_features_full_transpose)
    X_train_features_full_half = np.transpose(X_train_features_full_transpose)
    X_test_features_full_half = np.transpose(X_test_features_full_transpose)
    X_train_features_full_half = np.hsplit(X_train_features_full_half, 2)[0]
    X_test_features_full_half = np.hsplit(X_test_features_full_half, 2)[0]
    X_test_full_half_features_list.append(X_test_features_full_half)
    X_train_full_half_features_list.append(X_train_features_full_half)
    model.fit(X_train_features_full_half, train_labels)
    features = model.predict(X_train_features_full_half)
    test_features = model.predict(X_test_features_full_half)
    features = features.reshape((1000, 1))
    test_features = test_features.reshape(1000, 1)
    X_train_full_new = np.append(X_train_full_new, features, axis= 1)
    X_test_full_new = np.append(X_test_full_new, test_features, axis= 1)
X_train_full_new = X_train_full_new[:,1:]
X_test_full_new = X_test_full_new[:,1:]
model = SGDClassifier(loss= 'log', max_iter=10000)
model.fit(X_train_full_new, train_labels)
stumps_score = model.score(X_test_full_new, test_labels)
print(stumps_score)
```

```
0.641
0.769
0.658
0.63
```

[30]:
```
model = DecisionTreeClassifier(criterion='entropy')
model.fit(train_names, train_labels)
label = model.predict(hidden_test)
```

[31]:
```
print(label.shape)
print(train_labels.shape)
np.savetxt('badges/labels.txt', label, fmt='%s' )
```

```
(1000,)
(1000,)
```

[ ]:

[ ]:

[ ]: