CS 498 AML HW 1 REPORT

Pengyu Cheng
pcheng11
673359146

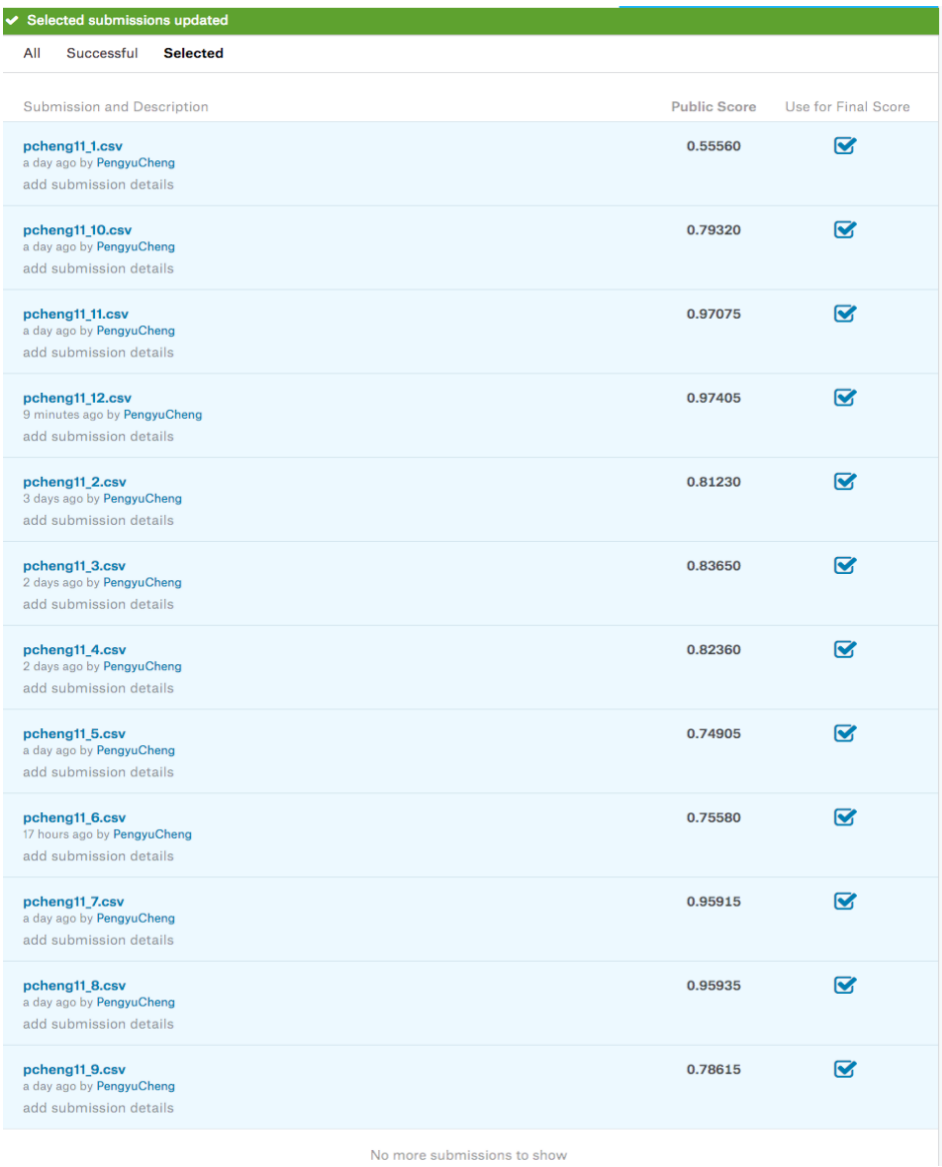
- Part 1 A accuracy: 0.7584415584415585
- Part 1 B accuracy: 0.7441558441558442
- Part 1 D accuracy: 0.7714285714285715

Accuracy subject to change as I used random split for dataset.

- Screenshot of my code:

```
def random_split(df):
    train, test = train_test_split(df, test_size=0.2)
    return train, test

def prior(df):
    return [len(df[df[:, 8]==0])/df.shape[0], len(df[df[:, 8]==1])/df.shape[0]]

def mean_variance(train_set):
    class_dic = {}
    for i in range(2): #class num
        sub_data = train_set[train_set[:, 8] == i]
        means = np.mean(sub_data[:,:-1], axis = 0)
        stds = np.std(sub_data[:,:-1], axis = 0)
        class_dic[i] = [means, stds]#holding mean and std for each feature
    return class_dic

def mean_variance_missing(train_set):
    class_dic = {}
    for i in range(2): #class num
        sub_data = train_set[train_set[:, 8] == i]
        for j in [2,3,5,7]:
            sub_data[sub_data[:, j] == 0] = np.nan
        means = np.nanmean(sub_data[:,:-1], axis = 0)
        stds = np.nanstd(sub_data[:,:-1], axis = 0)
        class_dic[i] = [means, stds]#holding mean and std for each feature
    return class_dic

def compute_prob_accuracy(test_set, class_dic, prior):
    correct_num = 0
    for record in test_set:
        arg_list = []
        for i in range(2):
            max_lh = np.sum(np.log(scipy.stats.norm(class_dic[i][0], class_dic[i][1]).pdf(record[:-1]))) + np.log(prior[i])
            arg_list.append(max_lh)
        pred = np.argmax(arg_list)
        if pred == record[8]:
            correct_num += 1
    return correct_num/test_set.shape[0]

def compute_ten_avg_accuracy(df, deal_missing=False):
    total_accuracy = 0.0
    print('Computing avg accuracy...')
    for i in range(10):
        train, test = random_split(df)
        prior_dist = prior(train)
        if deal_missing:
            cls_dic = mean_variance_missing(train)
        else:
            cls_dic = mean_variance(train)
        total_accuracy += compute_prob_accuracy(test, cls_dic, prior_dist)
    print('Computing avg accuracy done')
    return total_accuracy/10

def svm_compute_and_classify(df):
    #initiate svm
    clf = svm.SVC(kernel='linear')
    accuracy = 0.0
    print('Computing avg accuracy...')
    for i in range(10):
        correct_num = 0
        train_set, test_set = train_test_split(df, test_size=0.2)
        train_X = train_set[:,:-1]
        train_Y = train_set[:, -1]
        test_X = test_set[:,:-1]
        test_Y = test_set[:, -1]
        clf.fit(train_X, train_Y)
        pred_label = clf.predict(test_X)
        incorrect_num = np.sum(np.abs(test_Y - pred_label))
        accuracy += (1 - incorrect_num / len(test_set))
    accuracy /= 10
    print('Computing avg accuracy done')
    return accuracy
```

1:	Gaussian + untouched:	0.55560
2:	Gaussian + stretched:	0.81230
3:	Bernoulli + untouched:	0.83650
4:	Bernoulli + stretched:	0.82360
5:	10 trees + 4 depth + untouched:	0.74905
6:	10 trees + 4 depth + stretched:	0.75580
7:	10 trees + 16 depth + untouched:	0.95915
8:	10 trees + 16 depth + stretched:	0.95935
9:	30 trees + 4 depth + untouched:	0.78615
10:	30 trees + 4 depth + stretched:	0.79320
11:	30 trees + 16 depth + untouched:	0.97075
12:	30 trees + 16 depth + stretched:	0.97405



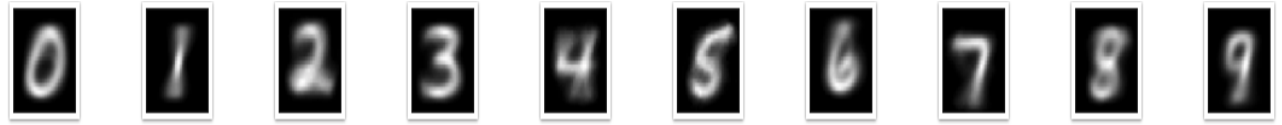
- Gaussian + Untouched

Gaussian_untouched_0.png Gaussian_untouched_1.png Gaussian_untouched_2.png Gaussian_untouched_3.png Gaussian_untouched_4.png Gaussian_untouched_5.png Gaussian_untouched_6.png Gaussian_untouched_7.png Gaussian_untouched_8.png Gaussian_untouched_9.png

- Gaussian + Stretched

Gaussian_stretched_0.png Gaussian_stretched_1.png Gaussian_stretched_2.png Gaussian_stretched_3.png Gaussian_stretched_4.png Gaussian_stretched_5.png Gaussian_stretched_6.png Gaussian_stretched_7.png Gaussian_stretched_8.png Gaussian_stretched_9.png

- Bernoulli + Untouched



Bernoulli_untouched_0.png Bernoulli_untouched_1.png Bernoulli_untouched_2.png Bernoulli_untouched_3.png Bernoulli_untouched_4.png Bernoulli_untouched_5.png Bernoulli_untouched_6.png Bernoulli_untouched_7.png Bernoulli_untouched_8.png Bernoulli_untouched_9.png

- Bernoulli + Stretched

Bernoulli_stretched_0.png Bernoulli_stretched_1.png Bernoulli_stretched_2.png Bernoulli_stretched_3.png Bernoulli_stretched_4.png Bernoulli_stretched_5.png Bernoulli_stretched_6.png Bernoulli_stretched_7.png Bernoulli_stretched_8.png Bernoulli_stretched_9.png

```python
def validate(val_set_x, labels, clf):
    """
    validate on val set
    """
    pred = clf.predict(val_set_x)
    accuracy = sum(labels == pred)/len(labels)
    return accuracy

def test(clf, test_set):
    """
    predict on the test set
    """
    pred = clf.predict(test_set)
    index = np.array(range(len(pred)))
    return np.column_stack((index,pred)), pred

def write_to_csv(res, num):
    """
    write to csv file
    """
    with open('pcheng11_'+str(num)+'.csv', 'w+') as f:
        writer = csv.writer(f)
        writer.writerow(["ImageID","Label"])
        writer.writerows(res)

def plot_img(pred, test_set, dist, stretched=False):
    test_data = np.column_stack((test_set, pred))
    for i in range(10):
        if dist == "Gaussian":
            img = np.mean(test_data[test_data[:,-1] == i][:, :-1]/255, axis=0)
        else:
            img = np.mean(test_data[test_data[:,-1] == i][:, :-1], axis=0)
        if stretched:
            img = img.reshape((20, 20))
            imsave(str(dist) + '_stretched_' + str(i) + '.png', img)
        else:
            img = img.reshape((28, 28))
            imsave(str(dist) + '_untouched_' + str(i) + '.png', img)
```

```python
def naive_bayes_clf(X, Y, dist):
    """
    initialize naive bayes classifer
    """
    if dist == "Gaussian":
        clf = GaussianNB()
    elif dist == "Bernoulli":
        clf = BernoulliNB()
    else:
        print("Not implemented!")

    clf.fit(X, Y)
    return clf

def random_forest_clf(X, Y, n_tree, n_depth):
    """
    initialize random forest classifer

    """
    clf = RandomForestClassifier(max_depth=n_depth, n_estimators=n_tree)
    clf.fit(X, Y)
    return clf
```

```python
print("Start validation...")
#Gaussian + untouched
clf = naive_bayes_clf(train_set_x, train_set_y, "Gaussian")
accuracy_1 = validate(val_set_x, val_set_y, clf)
res, pred = test(clf, test_set)
plot_img(pred, test_set, "Gaussian")
write_to_csv(res, 1)
print("1) Gaussian + untouched: " + str(accuracy_1))
#Gaussian + stretched
clf = naive_bayes_clf(stre_train_set_x, train_set_y, "Gaussian")
accuracy_2 = validate(stre_val_set_x, val_set_y, clf)
res = test(clf, stre_test_set)
print("2) Gaussian + stretched: " + str(accuracy_2))
plot_img(pred, stre_test_set, "Gaussian", True)
write_to_csv(res, 2)
#Bernoulli + untouched
clf = naive_bayes_clf(thres_train_set_x, train_set_y, "Bernoulli")
accuracy_3 = validate(thres_val_set_x, val_set_y, clf)
res = test(clf, thres_test_set)
print("3) Bernoulli + untouched " + str(accuracy_3))
plot_img(pred, thres_test_set, "Bernoulli")
write_to_csv(res, 3)
#Bernoulli + stretched
clf = naive_bayes_clf(stre_thres_train_set_x, train_set_y, "Bernoulli")
accuracy_4 = validate(stre_thres_val_set_x, val_set_y, clf)
res = test(clf, stre_thres_test_set)
print("4) Bernoulli + stretched " + str(accuracy_4))
plot_img(pred, stre_thres_test_set, "Bernoulli", True)
write_to_csv(res, 4)



#decision_forest
print("=====*10)
print("Start validation...")
index_rf = 4
for n_tree in [10,30]:
    for n_depth in [4,16]:
        for kind in ["untouched", "stretched"]:
            index_rf += 1
            if kind == "stretched":
                clf = random_forest_clf(stre_train_set_x, train_set_y, n_tree, n_depth)
                accuracy_rf = validate(stre_val_set_x, val_set_y, clf)
                res = test(clf, stre_test_set)
            else:
                clf = random_forest_clf(train_set_x, train_set_y, n_tree, n_depth)
                accuracy_rf = validate(val_set_x, val_set_y, clf)
                res = test(clf, test_set)
            print(str(index_rf) + ": " + str(n_tree) + " trees + " + str(n_depth) + " depth + " + kind + ":" + str(accuracy_rf))
            write_to_csv(res, index_rf)
```