**Problem 1:**

**1.**

(b) Please reference the relevant code 1b.py in attachment. The accuracy of the train\_data used in learned classifier can be listed as follows, from 1 to 1e+8:

0.8846153846153846

0.8846153846153846

0.8846153846153846

0.8974358974358975

0.8846153846153846

0.8974358974358975

0.9487179487179487

0.9743589743589743

0.9871794871794872

With the increasement of C, the learning parameters are trained more accuracy and fittable to the train data, then the accuracy of the train data are more high.

(c) Please reference the relevant code 1c.py in attachment. The accuracy of the validate\_data used in learned classifier can be listed as follows, from 1 to 1e+8:

0.8620689655172413

0.8620689655172413

0.8620689655172413

0.8103448275862069

0.8103448275862069

0.8448275862068966

0.8103448275862069

0.8275862068965517

0.8103448275862069

The over training of train data can lead to over-fitting. The trained parameters fit train data perfectly but cannot fit the other validation data.

We can see the **best value of c is 1**.

(d) Please reference the relevant code 1d.py in attachment. The accuracy of the test\_data used in learned classifier when c=1 is:

**0.8305084745762712**

**2.**

b) Please reference the relevant code 2b.py in attachment. The accuracy of the training data used in learned classifier can be listed as follows, c from 1 to 1e+8, σ from 1e-1 to 1e+3:

1.0

1.0

0.9358974358974359

0.8333333333333334

0.782051282051282

1.0

1.0

0.9743589743589743

0.8717948717948718

0.782051282051282

1.0

1.0

1.0

0.8974358974358975

0.782051282051282

1.0

1.0

1.0

0.9102564102564102

0.8461538461538461

1.0

1.0

1.0

0.9615384615384616

0.8717948717948718

1.0

1.0

1.0

0.9871794871794872

0.8974358974358975

1.0

1.0

1.0

1.0

0.9102564102564102

1.0

1.0

1.0

1.0

0.9102564102564102

1.0

1.0

1.0

1.0

0.9487179487179487

We can see from the result that with the increasement of c the training result are more accuracy for training data, with the increasement of σ, the accuracy of training tends to decrease at high value.

c) Please reference the relevant code 2c.py in attachment. The accuracy of the validation data used in learned classifier can be listed as follows, c from 1 to 1e+8, σ from 1e-1 to 1e+3:

0.7413793103448276

0.7413793103448276

0.8275862068965517

0.8103448275862069

0.7413793103448276

0.7413793103448276

0.7413793103448276

0.8103448275862069

0.7758620689655172

0.7413793103448276

0.7413793103448276

0.7413793103448276

0.7931034482758621

0.8275862068965517

0.7413793103448276

0.7413793103448276

0.7413793103448276

0.7931034482758621

0.8448275862068966

0.8275862068965517

0.7413793103448276

0.7413793103448276

0.7931034482758621

0.8448275862068966

0.8275862068965517

0.7413793103448276

0.7413793103448276

0.7931034482758621

0.8275862068965517

0.8275862068965517

0.7413793103448276

0.7413793103448276

0.7931034482758621

0.7758620689655172

0.8793103448275862

0.7413793103448276

0.7413793103448276

0.7931034482758621

0.7758620689655172

0.896551724137931

0.7413793103448276

0.7413793103448276

0.7931034482758621

0.7758620689655172

0.8448275862068966

We can see from the result that with the increasement of c the training result are more accuracy for training data, with the increasement of σ, the accuracy of training firstly increases and then decrease.

The best accuracy is 0.896551724137931, it reached **when c = 1e+7 and σ=1e+2**

d) Please reference the relevant code 2d.py in attachment, when c = 1e+7 and σ=1e+2, the accuracy of the test data set is

**0.8305084745762712**

**3.**

Please reference the relevant code 3a.py in attachment. The accuracy of the training data used in KNN classifier can be listed as follows, k = 1, 5, 11, 15, 21:

1.0

0.9230769230769231

0.8461538461538461

0.7948717948717948

0.782051282051282

Please reference the relevant code 3b.py in attachment. The accuracy of the validation data used in KNN classifier can be listed as follows, k = 1, 5, 11, 15, 21:

0.9137931034482759

0.8620689655172413

0.8793103448275862

0.8103448275862069

0.7413793103448276

Please reference the relevant code 3c.py in attachment. The accuracy of the test data used in KNN classifier can be listed as follows, k = 1, 5, 11, 15, 21:

0.8813559322033898

0.8813559322033898

0.8135593220338984

0.7457627118644068

0.7288135593220338

With the increasing of the K, the accuracy is decreasing. This means the result are disturbed by other unrelated neighbors when the number of selected neighbors increasing.

**4.**

From the test above, I prefer KNN classifier for this task. As we can see,

For train data, both KNN and SVM can reach accuracy of 1.0

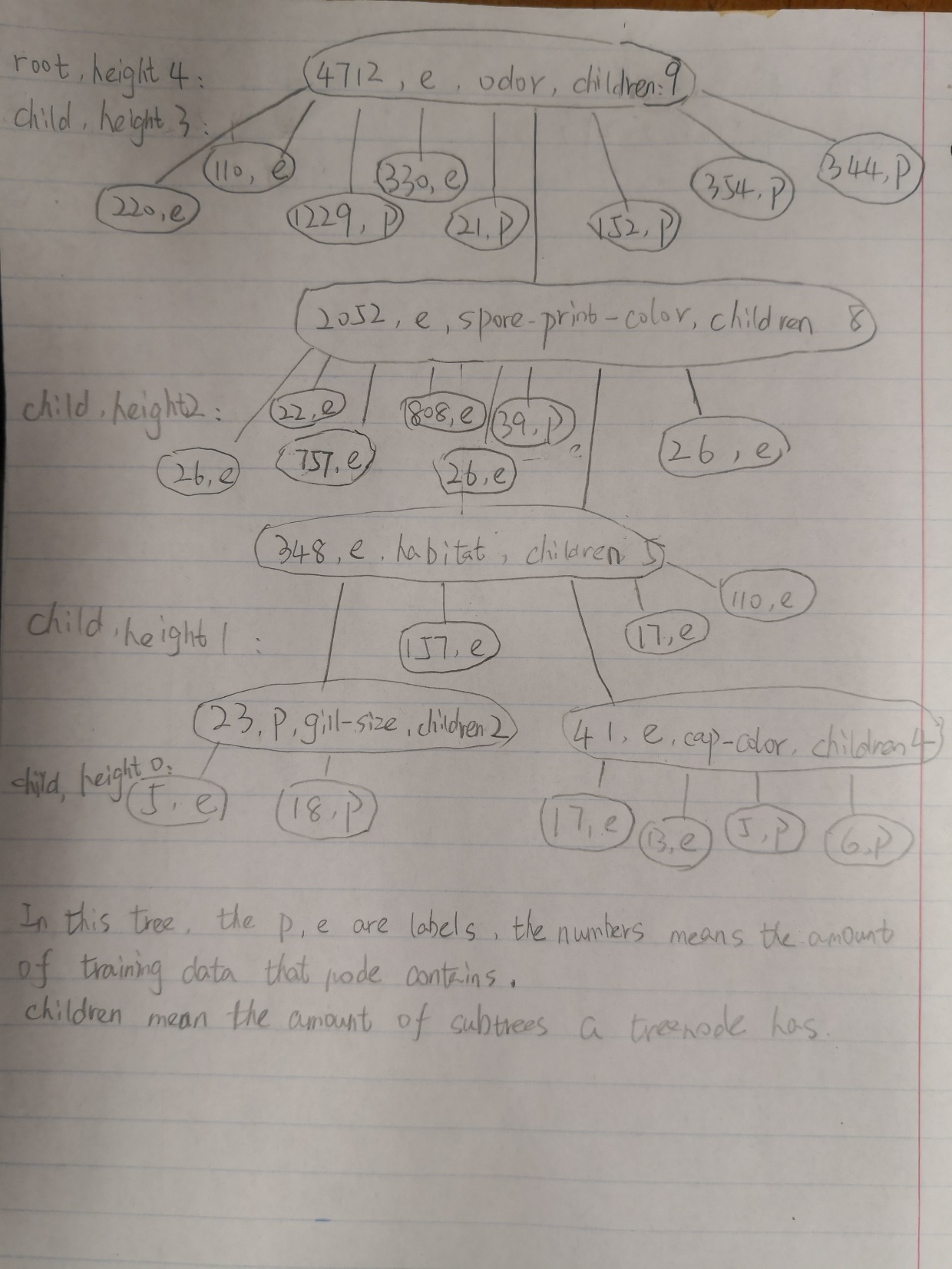
For validation data, the SVM can reach accuracy 0.89655, but the KNN can reach accuracy 0.9137931

For validation data, the SVM can reach accuracy 0.8305, but the KNN can reach accuracy 0.881356

So, from the testing above, the KNN is more accurate than SVM in this classifier task. In addition, the KNN are much simpler than SVM in realization. So, I prefer KNN at here.

**Problem 2:**

**1.**

Please reference the relevant code 2.2.py in attachment. The tree is in the picture below. 

**2.**

Please reference the relevant code 2.2.py in attachment. The size in the learned decision tree is 29.

**3.**

Please reference the relevant code 2.3.py in attachment. The height of the learned decision tree is 4.

**4.**

Please reference the relevant code 2.4.py in attachment. The accuracy of the learned decision tree for training data is 1.0.

**5.**

Please reference the relevant code 2.5.py in attachment. The accuracy of the decision tree for test data is 1.0.

**6.**

From question 5, we can see that the decision accuracy is 1.0 in test. So, the decision tree works completely well for this problem. So, the Audubon Society Field Guide’s statement is not true.

**7.**

The learned decision tree depends on the training/test split quite a lot. When the height of learned decision tree is smaller and the size of decision trees is smaller, this means the data is more divisible. It will lead to higher accuracy.

When the height of trees is smaller, it means there are less decisions (tree nodes) to be made for each test data. So, there are less chance to make mistake.

When the size of decision trees is smaller, this means each tree node contain more same decision node. With more alike nodes, it means high possibility of this decision is correct.

**8.**

**No.**

The foundation of the KNN algorithm is selecting the greatest information gain in each step of construction KNN tree. If we want to prove our assumption, we just need to find a situation that there is situation more than one attribute construct a biggest information gain.

For IG(𝑋) = H(𝑌) − 𝐻(𝑌|𝑋)

= H(𝑌) - -

For a specific dataset, we know the H(𝑌) is static and = 1.

As < 0,

only when the function 🡪 +0, the IG(X) will get biggest value. This means the x completely fit for y, that is the accuracy is 1.0.

So when there are more than attributes that can completely train the data(or do better than one attribute), it will get the best result.

So, it should not be exactly one attribute for the best KNN tree.