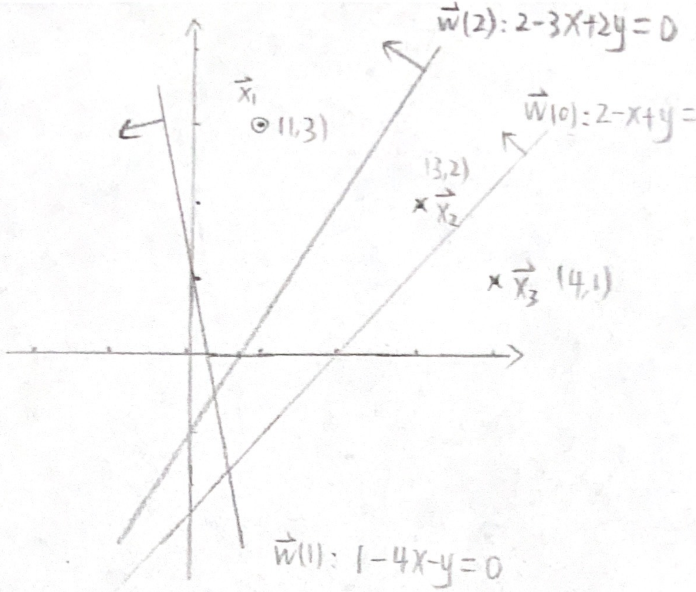
Question 1

(i)



(ii) $\vec{w}(0)^T\vec{x}_1 = 2-1+3 = 4 > 0$ correctly

$\vec{w}(0)^T\vec{x}_2 = 2-3+2 = 1 > 0$ not correctly

$\vec{w}(0)^T\vec{x}_3 = 2-4+1 = -1 < 0$ correctly

(iii) $\vec{w}(1) = \vec{w}(0) + y_2\vec{x}_2 = [2,-1,1]^T + (-1)\begin{bmatrix}1\\3\\2\end{bmatrix} = [1,-4,-1]^T$

$\vec{w}(1)^T\vec{x}_1 = 1-4-3 = -6 < 0$ incorrectly

$\vec{w}(1)^T\vec{x}_2 = 1-12-2 = -13 < 0$ incorrectly

$\vec{w}(1)\vec{x}_3 = 1-16-1 = -16 < 0$ incorrectly

(iv) choose $\vec{x}_s = \vec{x}_1 = \begin{bmatrix}1\\1\\3\end{bmatrix}$

$\vec{w}(2) = \vec{w}(1) + y_1\vec{x}_1 = [1,-4,-1]^T + 1\cdot\begin{bmatrix}1\\1\\3\end{bmatrix} = \begin{bmatrix}2\\-3\\2\end{bmatrix}$

$\vec{w}(2)^T\vec{x}_1 = 2-3+6 = 5 > 0$ correctly

$\vec{w}(2)^T\vec{x}_2 = 2-9+4 = -3 < 0$ correctly

$\vec{w}(2)^T\vec{x}_3 = 2-12+2 = -8 < 0$ correctly

Question2:

(a.i)

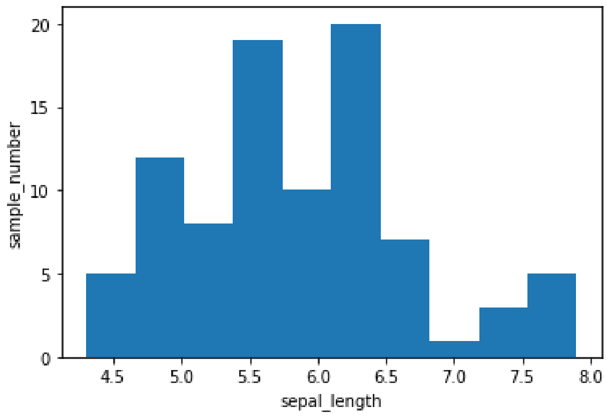
number_setosa: 30

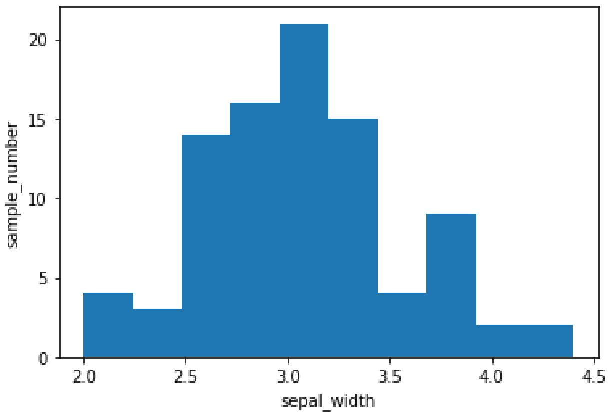
number_versicolor: 30

number_virginica: 30

They are equally distributed.

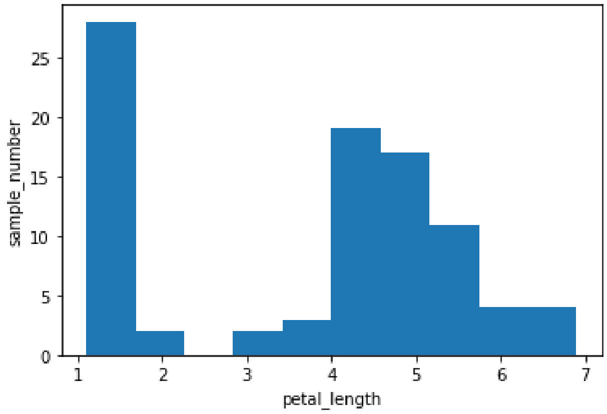
(a.ii)

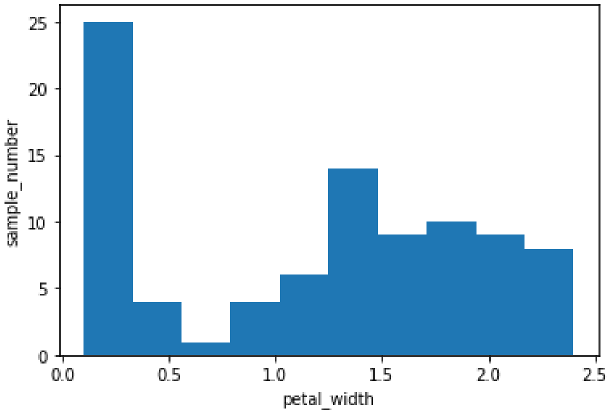




The sepal length feature is distributed multimodal.

The sepal width feature is distributed bimodal.

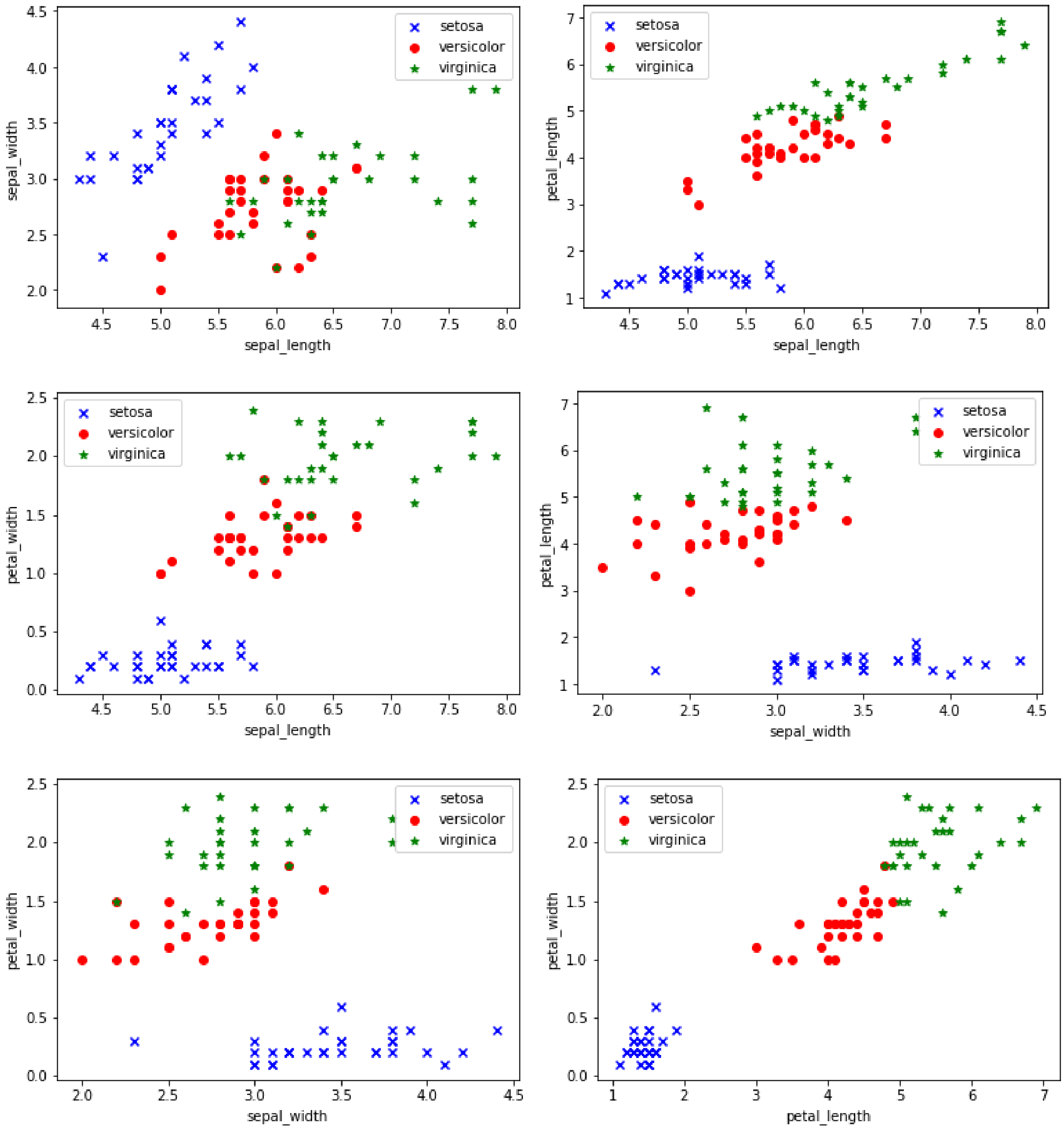




The petal length feature is distributed bimodal.

The petal width feature is distributed bimodal.

(a.iii)



Except the feature combination of sepal length and sepal width, where versicolor and virginica can not separated well. Other feature combinations can separate most of the Iris samples.

Setosa can be easily separated from other two classes. Versicolor and virginica can not be separated by the combination of sepal length and sepal width, but can be separated by other feature combinations, except a few samples on the boundary.

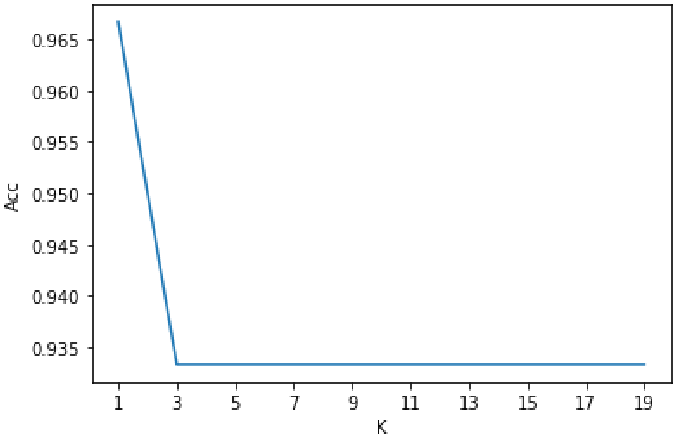
The combination of sepal width and petal length, petal length and petal width are more separable for the three classes.

(b.i)

See the code in the later part of the file.

(b.ii)

L2_Norm



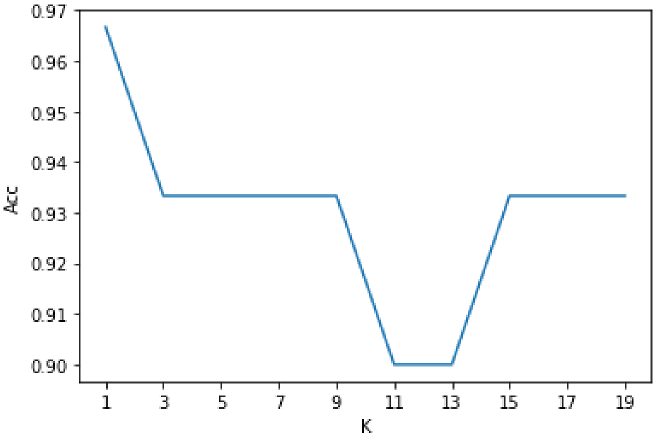
When K=1, we can get the highest Acc value.

(b.iii)

When K=1, the Acc value of this KNN model is 0.9666666666666667. This value is similar to the result from the development set.

(b.iv)

Using the L1_Norm.



When K=1, we can get the highest Acc value.

Apply K=1 to calculating the Acc on test set. The value is also 0.9666666666666667. But from the perspective of the development set, the L1_Norm performance is not so good as the L2_Norm.

Question (a.i)

In [26]:

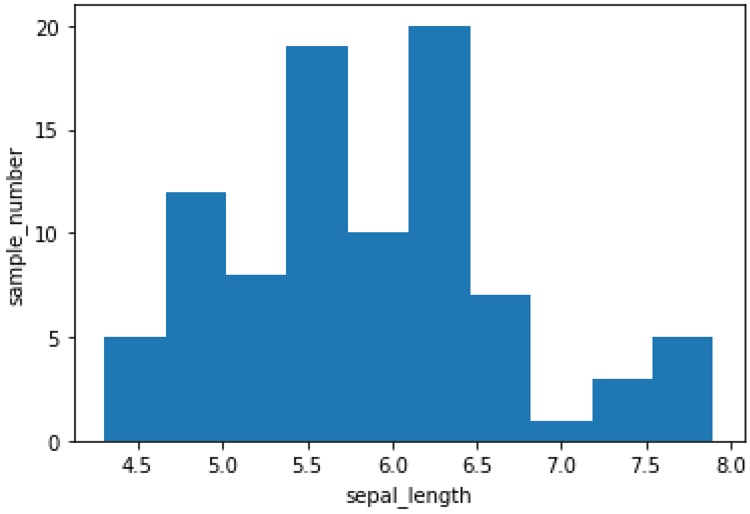
```
class DataPoint(object):
    def __init__(self, feats):
        self.sepal_length = feats['sepal_length']
        self.sepal_width = feats['sepal_width']
        self.petal_length = feats['petal_length']
        self.petal_width = feats['petal_width']
        self.label = feats['label']

def parse_dataset(filename):
    data_file = open(filename, 'r')
    dataset = []
    for index, line in enumerate(data_file):
        if index == 0:
            continue
        sepal_length, sepal_width, petal_length, petal_width, label = line.strip().spli
t(',')
        dataset.append(DataPoint({'sepal_length':float(sepal_length), 'sepal_width':flo
at(sepal_width), \
                                  'petal_length':float(petal_length), 'petal_width':flo
at(petal_width), 'label':label}))
    return dataset

dataset = parse_dataset('iris_train.csv')
number_setosa = 0
number_versicolor = 0
number_virginica = 0
for i in range(len(dataset)):
    if dataset[i].label == "Iris-setosa":
        number_setosa += 1
    if dataset[i].label == "Iris-versicolor":
        number_versicolor += 1
    if dataset[i].label == "Iris-virginica":
        number_virginica += 1

print("number_setosa:", number_setosa)
print("number_versicolor:", number_versicolor)
print("number_virginica:", number_virginica)
```

```
number_setosa: 30
number_versicolor: 30
number_virginica: 30
```

Question (a.ii)

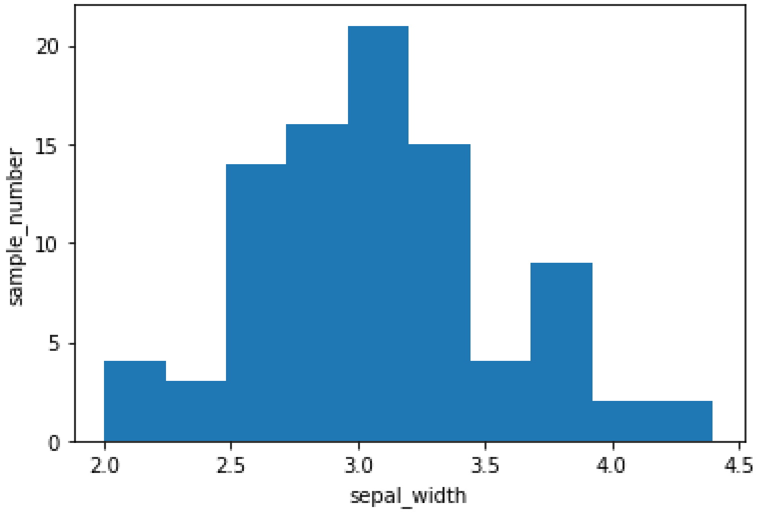
In [27]:

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.pyplot import MultipleLocator

sepal_length = []
sepal_width = []
petal_length = []
petal_width = []
for i in range(len(dataset)):
    sepal_length.append(dataset[i].sepal_length)
    sepal_width.append(dataset[i].sepal_width)
    petal_length.append(dataset[i].petal_length)
    petal_width.append(dataset[i].petal_width)

y_major_locator=MultipleLocator(5)
ax=plt.gca()
ax.yaxis.set_major_locator(y_major_locator)

plt.hist(sepal_length)
plt.xlabel("sepal_length")
plt.ylabel("sample_number")
plt.show()
```



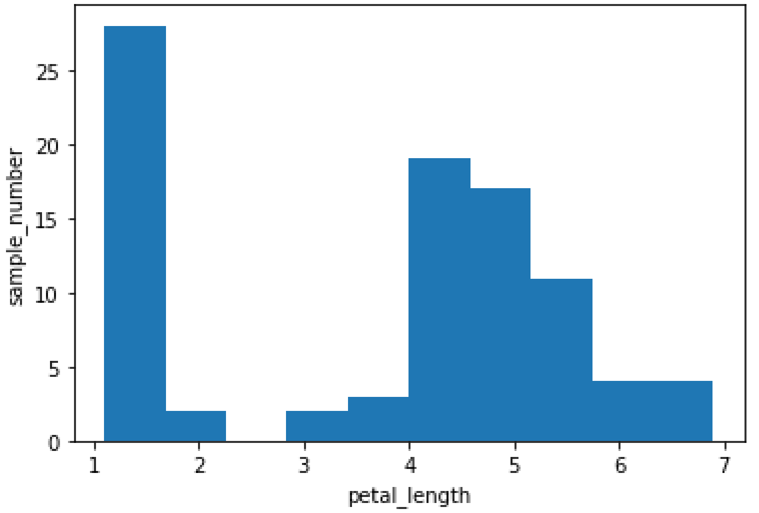
In [25]:

```
y_major_locator=MultipleLocator(5)
ax=plt.gca()
ax.yaxis.set_major_locator(y_major_locator)

plt.hist(sepal_width)
plt.xlabel("sepal_width")
plt.ylabel("sample_number")
plt.show()
```



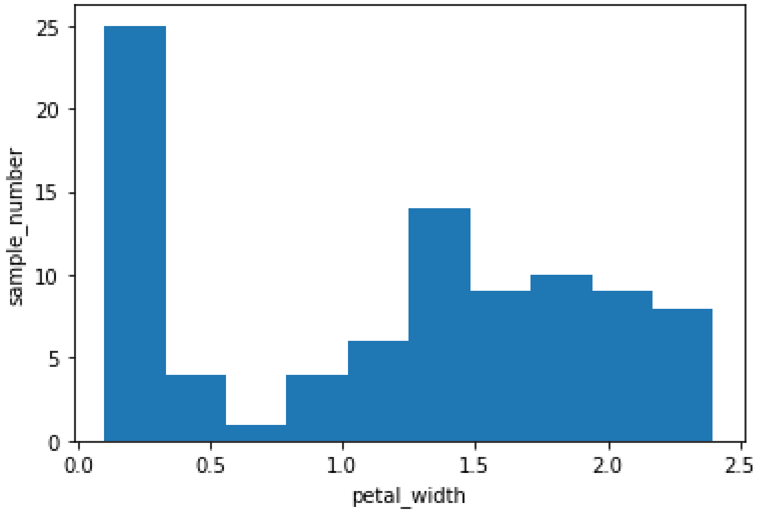
In [21]:

```
plt.hist(petal_length)
plt.xlabel("petal_length")
plt.ylabel("sample_number")
plt.show()
```



In [22]:

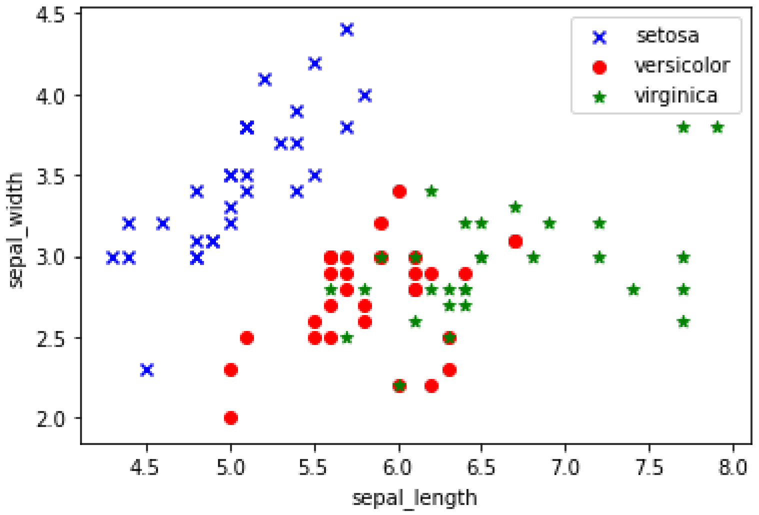
```
plt.hist(petal_width)
plt.xlabel("petal_width")
plt.ylabel("sample_number")
plt.show()
```

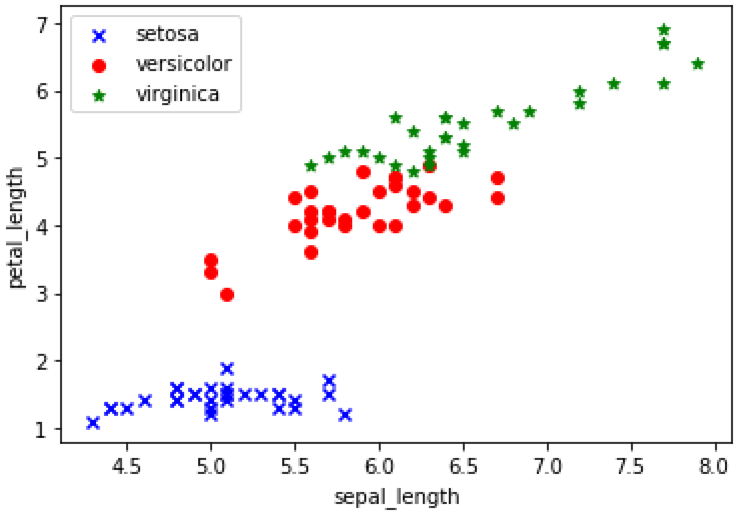


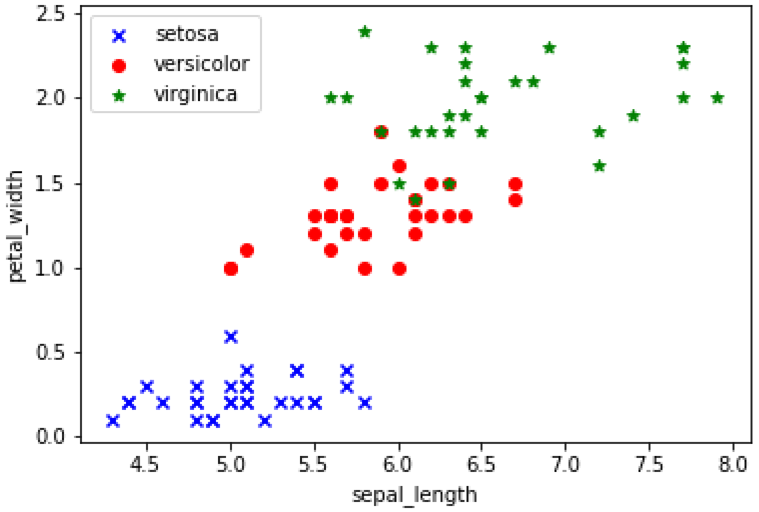
Question (a.iii)

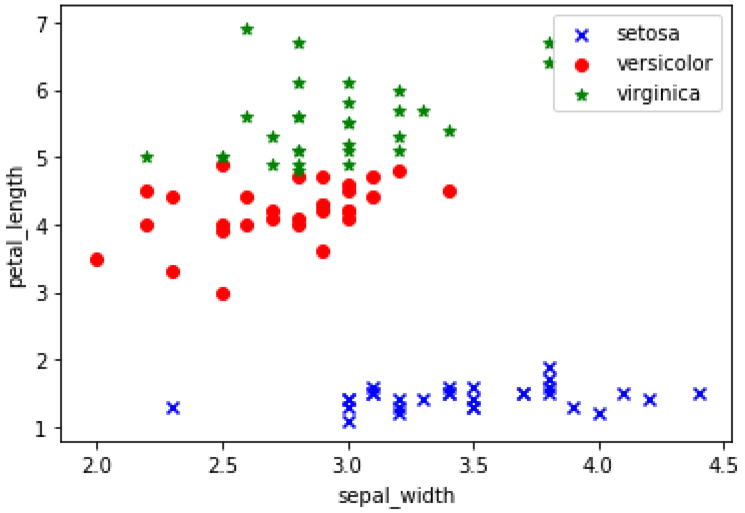
In [29]:

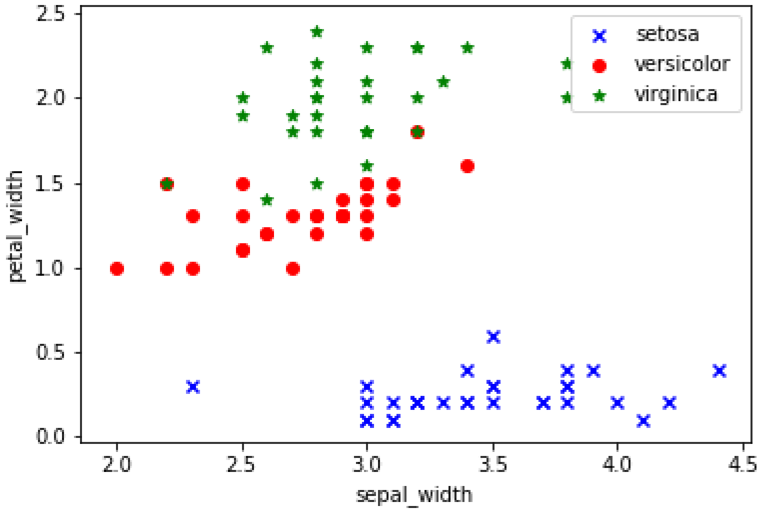
```
def plot_data(dataset):

    setosa_sepal_lengths = [data.sepal_length for data in dataset if data.label == "Iris-setosa"]
    setosa_sepal_widths = [data.sepal_width for data in dataset if data.label == "Iris-setosa"]
    setosa_petal_lengths = [data.petal_length for data in dataset if data.label == "Iris-setosa"]
    setosa_petal_widths = [data.petal_width for data in dataset if data.label == "Iris-setosa"]

    versicolor_sepal_lengths = [data.sepal_length for data in dataset if data.label == "Iris-versicolor"]
    versicolor_sepal_widths = [data.sepal_width for data in dataset if data.label == "Iris-versicolor"]
    versicolor_petal_lengths = [data.petal_length for data in dataset if data.label == "Iris-versicolor"]
    versicolor_petal_widths = [data.petal_width for data in dataset if data.label == "Iris-versicolor"]

    virginica_sepal_lengths = [data.sepal_length for data in dataset if data.label == "Iris-virginica"]
    virginica_sepal_widths = [data.sepal_width for data in dataset if data.label == "Iris-virginica"]
    virginica_petal_lengths = [data.petal_length for data in dataset if data.label == "Iris-virginica"]
    virginica_petal_widths = [data.petal_width for data in dataset if data.label == "Iris-virginica"]


    plt.scatter(setosa_sepal_lengths, setosa_sepal_widths, c='b', marker='x', label='setosa')
    plt.scatter(versicolor_sepal_lengths, versicolor_sepal_widths, c='r', marker='o', label='versicolor')
    plt.scatter(virginica_sepal_lengths, virginica_sepal_widths, c='g', marker='*', label='virginica')
    plt.xlabel("sepal_length")
    plt.ylabel("sepal_width")
    plt.legend()
    plt.show()

    plt.scatter(setosa_sepal_lengths, setosa_petal_lengths, c='b', marker='x', label='setosa')
    plt.scatter(versicolor_sepal_lengths, versicolor_petal_lengths, c='r', marker='o', label='versicolor')
    plt.scatter(virginica_sepal_lengths, virginica_petal_lengths, c='g', marker='*', label='virginica')
    plt.xlabel("sepal_length")
    plt.ylabel("petal_length")
    plt.legend()
    plt.show()

    plt.scatter(setosa_sepal_lengths, setosa_petal_widths, c='b', marker='x', label='setosa')
    plt.scatter(versicolor_sepal_lengths, versicolor_petal_widths, c='r', marker='o', label='versicolor')
    plt.scatter(virginica_sepal_lengths, virginica_petal_widths, c='g', marker='*', label='virginica')
    plt.xlabel("sepal_length")
    plt.ylabel("petal_width")
    plt.legend()
    plt.show()

    plt.scatter(setosa_sepal_widths, setosa_petal_lengths, c='b', marker='x', label='setosa')
    plt.scatter(versicolor_sepal_widths, versicolor_petal_lengths, c='r', marker='o', label='versicolor')
    plt.scatter(virginica_sepal_widths, virginica_petal_lengths, c='g', marker='*', label='virginica')
    plt.xlabel("sepal_width")
    plt.ylabel("petal_length")
    plt.legend()
    plt.show()

    plt.scatter(setosa_sepal_widths, setosa_petal_widths, c='b', marker='x', label='setosa')
    plt.scatter(versicolor_sepal_widths, versicolor_petal_widths, c='r', marker='o', label='versicolor')
    plt.scatter(virginica_sepal_widths, virginica_petal_widths, c='g', marker='*', label='virginica')
    plt.xlabel("sepal_width")
    plt.ylabel("petal_width")
    plt.legend()
    plt.show()

    plt.scatter(setosa_petal_lengths, setosa_petal_widths, c='b', marker='x', label='setosa')
    plt.scatter(versicolor_petal_lengths, versicolor_petal_widths, c='r', marker='o', label='versicolor')
    plt.scatter(virginica_petal_lengths, virginica_petal_widths, c='g', marker='*', label='virginica')
    plt.xlabel("petal_length")
    plt.ylabel("petal_width")
    plt.legend()
    plt.show()

plot_data(dataset)
```

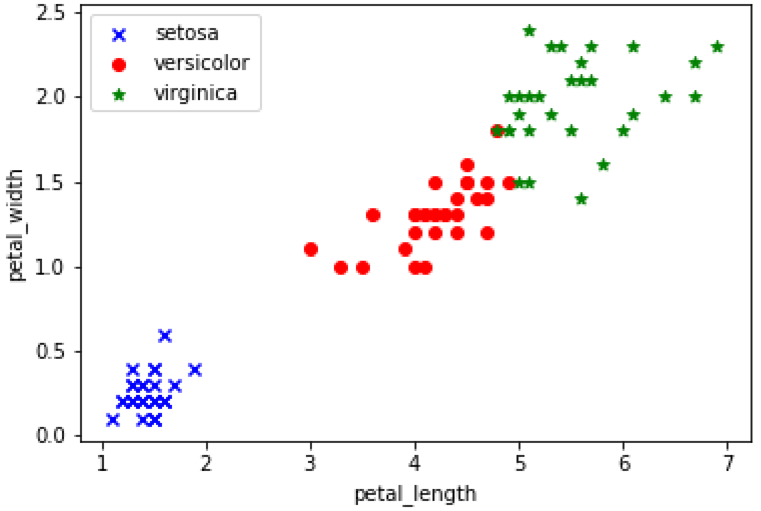

setosa
versicolor
virginica
sepal_width
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
sepal_length


setosa
versicolor
virginica
petal_length
7
6
5
4
3
2
1
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
sepal_length


setosa
versicolor
virginica
petal_width
2.5
2.0
1.5
1.0
0.5
0.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
sepal_length


setosa
versicolor
virginica
petal_length
sepal_width
1
2
3
4
5
6
7
2.0
2.5
3.0
3.5
4.0
4.5


setosa
versicolor
virginica
petal_width
sepal_width
0.0
0.5
1.0
1.5
2.0
2.5
2.0
2.5
3.0
3.5
4.0
4.5


setosa
versicolor
virginica
petal_width
petal_length
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7

Question (b.i)

In [30]:

```
dataset = parse_dataset('iris_train.csv')
train_data = [[data.sepal_length, data.sepal_width, data.petal_length, data.petal_width
] for data in dataset]
train_label = [data.label for data in dataset]
train_data = np.array(train_data)

dataset = parse_dataset('iris_dev.csv')
val_data = [[data.sepal_length, data.sepal_width, data.petal_length, data.petal_width]
for data in dataset]
val_label = [data.label for data in dataset]
val_data = np.array(val_data)

dataset = parse_dataset('iris_test.csv')
test_data = [[data.sepal_length, data.sepal_width, data.petal_length, data.petal_width]
for data in dataset]
test_label = [data.label for data in dataset]
test_data = np.array(test_data)
```

In [31]:

```
def L2_Norm(vec1, vec2):
    vec = (vec1 - vec2) ** 2
    ans = np.sum(vec)
    return np.sqrt(ans)
```

In [32]:

```
def find_topK(t, K):
    dic = {}
    for i in range(len(t)):
        dic[i] = t[i]

    sorted_dic = dict(sorted(dic.items(), key=lambda d: d[1]))
    keys = list(sorted_dic.keys())
    keys = keys[:K]

    return keys
```

In [70]:

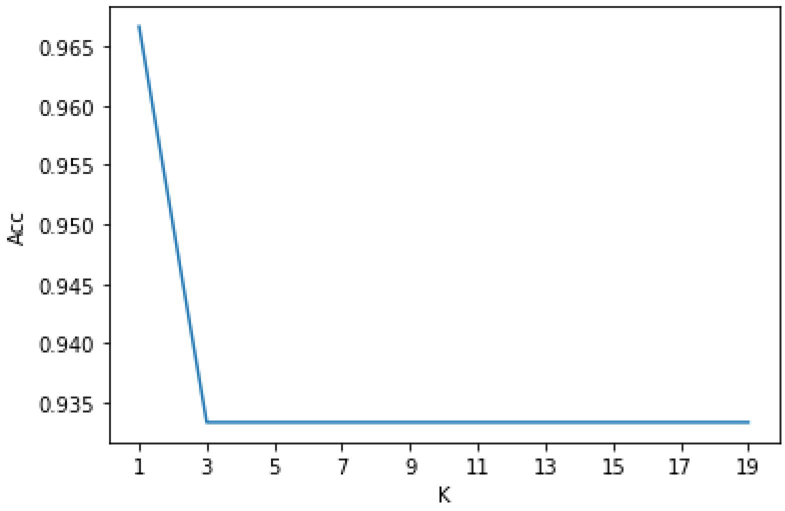
```
def Acc(val_expected, val_label):
    count = 0;
    for i in range(len(val_expected)):
        if val_expected[i] == val_label[i]:
            count += 1
    return count/len(val_expected)
```

In [71]:

```
def knn(train_data, train_label, data, label, K):
    val_expected = []
    for i in range(len(data)):
        t = []
        count_setosa = 0
        count_versicolor = 0
        count_virginica = 0
        for j in range(len(train_data)):
            t.append(L2_Norm(data[i], train_data[j]))
        topK = find_topK(t, K)
        for k in range(K):
            if train_label[topK[k]] == "Iris-setosa":
                count_setosa += 1
            if train_label[topK[k]] == "Iris-versicolor":
                count_versicolor += 1
            if train_label[topK[k]] == "Iris-virginica":
                count_virginica += 1
        dic1 = {"Iris-setosa": count_setosa, "Iris-versicolor": count_versicolor, "Iris
-virginica": count_virginica}
        sorted_dict = dict(sorted(dic1.items(), key=lambda d: d[1], reverse=True))
        keys = list(sorted_dict.keys())
        val_expected.append(keys[0])
    Acc_result = Acc(val_expected, label)
    return Acc_result
```

Question (b.ii)

In [72]:
```
Ks = []
Accs = []
for i in range(1,20,2):
    Ks.append(i)
    Acc_result = knn(train_data, train_label, val_data, val_label, i)
    Accs.append(Acc_result)

plt.xticks(Ks)
plt.plot(Ks, Accs)
plt.xlabel("K")
plt.ylabel("Acc")
plt.show()
print("Acc results of different K:")
print(Accs)
```



```
Acc results of different K:
[0.9666666666666667, 0.9333333333333333, 0.9333333333333333, 0.9333333333333333, 0.93
33333333333333, 0.9333333333333333, 0.9333333333333333, 0.9333333333333333, 0.9333333
333333333, 0.9333333333333333]
```

Question (b.iii)

In [68]:
```
Acc_result = knn(train_data, train_label, test_data, test_label, 1)
print(Acc_result)
```

```
0.9666666666666667
```

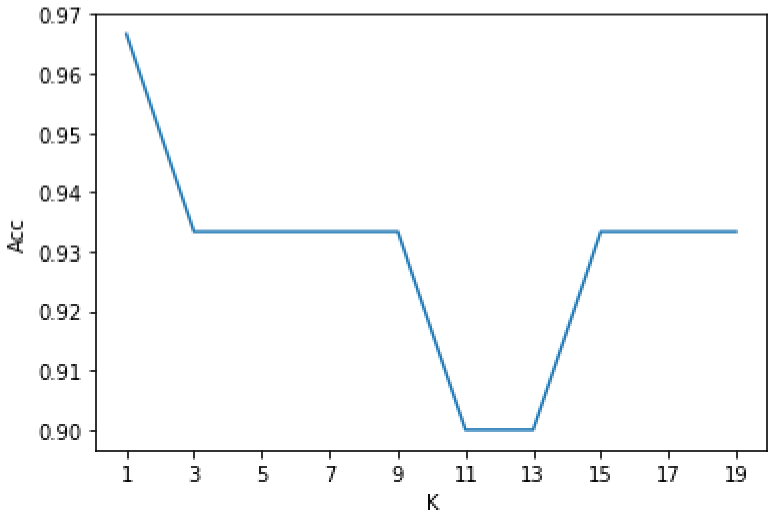
Question (b.iv)

In [37]:
```
def L1_Norm(vec1, vec2):
    vec = vec1 - vec2
    return np.sum(np.abs(vec))
```

In [73]:

```
def knn(train_data, train_label, data, label, K):
    val_expected = []
    for i in range(len(data)):
        t = []
        count_setosa = 0
        count_versicolor = 0
        count_virginica = 0
        for j in range(len(train_data)):
            t.append(L1_Norm(data[i], train_data[j]))
        topK = find_topK(t, K)
        for k in range(K):
            if train_label[topK[k]] == "Iris-setosa":
                count_setosa += 1
            if train_label[topK[k]] == "Iris-versicolor":
                count_versicolor += 1
            if train_label[topK[k]] == "Iris-virginica":
                count_virginica += 1
        dic1 = {"Iris-setosa": count_setosa, "Iris-versicolor": count_versicolor, "Iris-virginica": count_virginica}
        sorted_dict = dict(sorted(dic1.items(), key=lambda d: d[1], reverse=True))
        keys = list(sorted_dict.keys())
        val_expected.append(keys[0])
    Acc_result = Acc(val_expected, label)
    return Acc_result
```

In [76]:
```
Ks = []
Accs = []
for i in range(1,20,2):
    Ks.append(i)
    Accs.append(knn(train_data, train_label, val_data, val_label, i))

plt.xticks(Ks)
plt.plot(Ks, Accs)
plt.xlabel("K")
plt.ylabel("Acc")
plt.show()
print("Acc results of different K:")
print(Accs)
```



```
Acc results of different K:
[0.9666666666666667, 0.9333333333333333, 0.9333333333333333, 0.9333333333333333, 0.93
33333333333333, 0.9, 0.9, 0.9333333333333333, 0.9333333333333333, 0.9333333333333333]
```

In [77]:
```
Acc_result = knn(train_data, train_label, test_data, test_label, 1)
print(Acc_result)
```

```
0.9666666666666667
```

In [ ]: