Chapter 5: Demo KNN Classification

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:
```
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter5_KNN/'
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [4]:
```
iris = pd.read_excel("Iris.xls")
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepallength    150 non-null float64
sepalwidth     150 non-null float64
petallength    150 non-null float64
petalwidth     150 non-null float64
iris           150 non-null object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

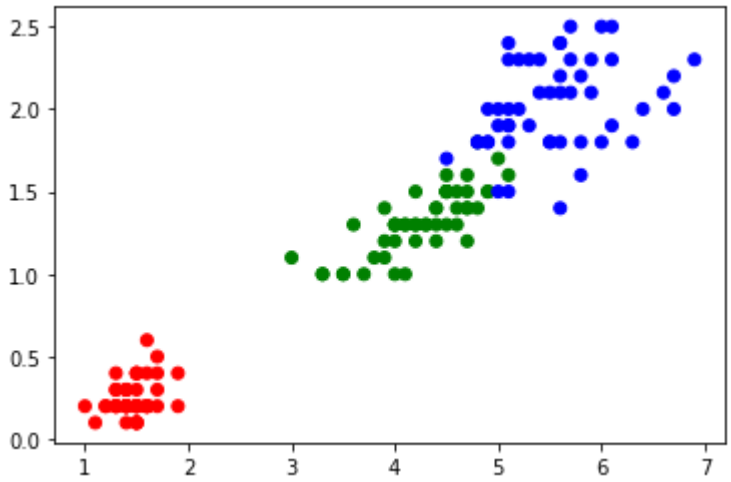
In [5]:
```
iris_class = {'Iris-setosa':0, 'Iris-versicolor':1, 'Iris-virginica':2}
iris['species_num'] = [iris_class[i] for i in iris.iris]
iris.head()
```

Out[5]:

	sepallength	sepalwidth	petallength	petalwidth	iris	species_num
0	5.1	3.5	1.4	0.2	Iris-setosa	0
1	4.9	3.0	1.4	0.2	Iris-setosa	0
2	4.7	3.2	1.3	0.2	Iris-setosa	0
3	4.6	3.1	1.5	0.2	Iris-setosa	0
4	5.0	3.6	1.4	0.2	Iris-setosa	0

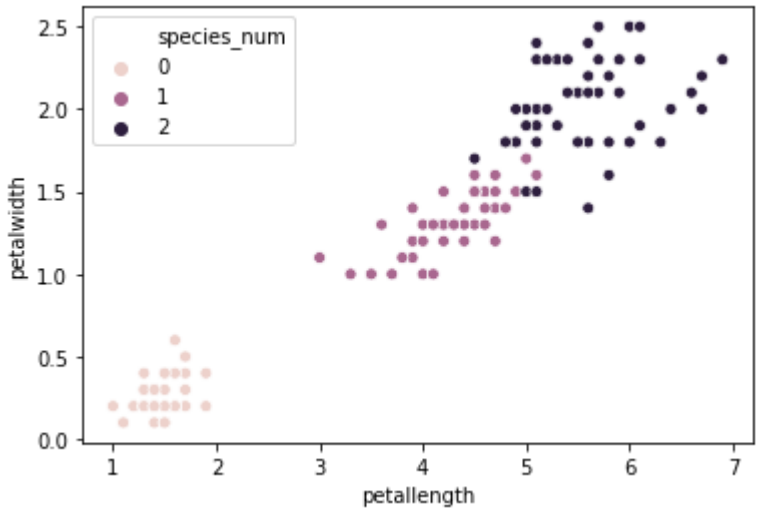
In [6]:
```
def make_color(value):
    color = 'yellow'
    if value == 0:
        color = 'red'
    elif value == 1:
        color = 'green'
    else:
        color = 'blue'
    return color
```

In [7]:
```
pentallength = iris.petallength.values
petalwidth = iris.petalwidth.values
types = iris.species_num.values
color= [make_color(x) for x in types]
plt.scatter(pentallength, petalwidth, color=color)
plt.show()
```



In [8]:
```
import seaborn as sns
```

In [9]:
```
sns.scatterplot(x="petallength", y="petalwidth", data=iris,hue="species_num")
plt.show()
```



In [10]:
```
X = iris.drop(['iris', 'species_num'], axis=1) # input: cac thuoc tinh lien tuc
y = iris.species_num # output: class
```

In [11]:
```
X.head()
```

Out[11]:

	sepallength	sepalwidth	petallength	petalwidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [12]:
```
y.head()
```

Out[12]:
```
0    0
1    0
2    0
3    0
4    0
Name: species_num, dtype: int64
```

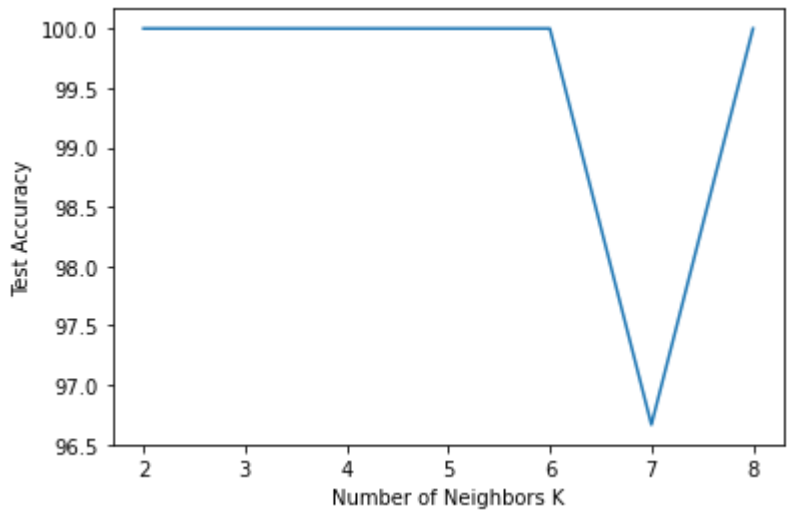
In [13]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
                                                    random_state = 42)
```

In [14]:
```
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
```

In [15]:
```
list_k = []
list_acc = []
# list_score_train
# list_score_test
# |train - test| <<
for K_value in range(2,9):
    list_k.append(K_value)
    neigh = KNeighborsClassifier(n_neighbors = K_value)
    neigh.fit(X_train, y_train)
    y_pred = neigh.predict(X_test)
    acc = accuracy_score(y_test,y_pred)*100
    # model.score(X_train, y_train) & model.score(X_test,y_test)
    # ... |train - test| <<
    list_acc.append(acc)
    print("Accuracy is ", accuracy_score(y_test,y_pred)*100,"% for K-Value:",
          K_value)

vi_tri = list_acc.index(max(list_acc))
k = list_k[vi_tri]

print("\nThe optimal number of neighbors is %d with %0.1f%%" % (k,
                                                    list_acc[vi_tri]))
# chạy nhiều lần, thấy k nào có độ chính xác cao và ít biến động nhất thì chọn
```

```
Accuracy is  100.0 % for K-Value: 2
Accuracy is  100.0 % for K-Value: 3
Accuracy is  100.0 % for K-Value: 4
Accuracy is  100.0 % for K-Value: 5
Accuracy is  100.0 % for K-Value: 6
Accuracy is  96.66666666666667 % for K-Value: 7
Accuracy is  100.0 % for K-Value: 8

The optimal number of neighbors is 2 with 100.0%
```

In [16]:
```
plt.plot(list_k, list_acc)
plt.xlabel('Number of Neighbors K')
plt.ylabel('Test Accuracy')
plt.show()
```



In [17]:
```
# select k => ap dung model
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X_train, y_train)
```

Out[17]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                     weights='uniform')
```

In [18]:
```
y_pred = knn.predict(X_test)
```

In [19]:
```
# Kiểm tra độ chính xác
print("The prediction accuracy is: ", knn.score(X_test,y_test)*100,"%")
```

```
The prediction accuracy is:  100.0 %
```

In [20]:
```
print("The Training R^2 score is: ", knn.score(X_train,y_train))
print("The Testing R^2 score is: ", knn.score(X_test,y_test))
```

```
The Training R^2 score is:  0.9666666666666667
The Testing R^2 score is:  1.0
```

In [21]:
```
df = pd.DataFrame({'Actual': pd.DataFrame(y_test.values)[0].values,
                   'Prediction': pd.DataFrame(y_pred)[0].values})
df.head()
```

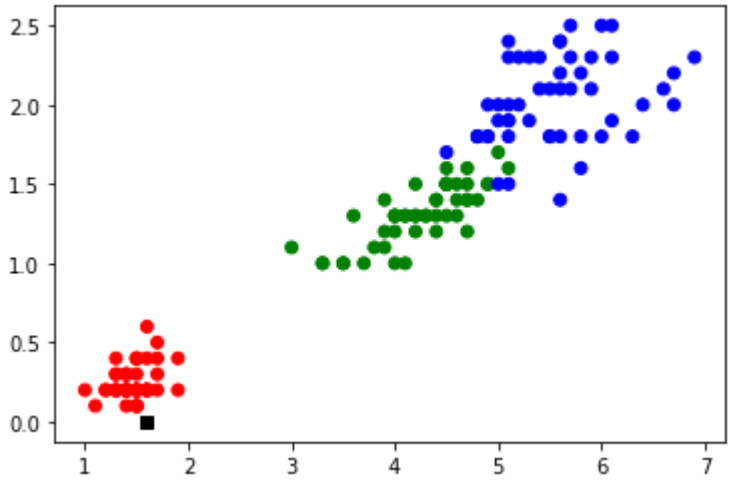
Out[21]:

	Actual	Prediction
0	1	1
1	0	0
2	2	2
3	1	1
4	1	1

In [22]:
```
x_now = [[4.8, 3.3, 1.6, 0.25]]
y_now = knn.predict(x_now)
y_now
```

Out[22]:
```
array([0], dtype=int64)
```

In [23]:
```
types = iris.species_num.values
color= [make_color(x) for x in types]
plt.scatter(pentallength, petalwidth, color=color)
plt.scatter(x_now[0][2], y_now, color='k', marker = 's')
plt.show()
```



In [ ]:
```

```