In [60]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import neighbors, datasets
from sklearn.model_selection import train_test_split

def knearn(n,k):
    np.random.seed(2018) # Set random seed so results are repeatable

## Generate a simple 2D dataset
    X, y = datasets.make_moons(n,'True',0.3)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_si
ze=0.25, random_state=47)

## Create instance of KNN classifier

    classifier = neighbors.KNeighborsClassifier(k,'uniform')
    classifier.fit(X_train, y_train)
    acc = classifier.score(X_test,y_test)
    print("Accuracy n = ",n,"& k = ",k,": ", acc)

## Plot the decision boundary.
# Begin by creating the mesh [x_min, x_max]x[y_min, y_max].
    h = .02  # step size in the mesh
    x_delta = (X[:, 0].max() - X[:, 0].min())*0.05 # add 5% white spa
ce to border
    y_delta = (X[:, 1].max() - X[:, 1].min())*0.05
    x_min, x_max = X[:, 0].min() - x_delta, X[:, 0].max() + x_delta
    y_min, y_max = X[:, 1].min() - y_delta, X[:, 1].max() + y_delta
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min,
 y_max, h))
    #print((np.c_[xx.ravel(), yy.ravel()]).shape)
    #X_blind = np.c_[xx.ravel(), yy.ravel()]
    Z = classifier.predict(np.c_[xx.ravel(), yy.ravel()])
    #print(classifier.score(X_test,y_test))
    # To calculate accuray, we need ground truth. This can be obtaine
d by getting the predictions for NN classifer,
    # where, k = 1, and use it for y_true

# Create color maps
    cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA'])
    cmap_bold = ListedColormap(['#FF0000', '#00FF00'])

# Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

## Plot the training points
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold)
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title("%i-NN classifier trained on %i data points" % (k,n))

## Show the plot
    plt.show()
```

In [54]:

```
for n in ([100,500,1000,5000]):
    for k in list(range(1,51,2)): #using odd k so as to resolve tie
s..
        knearn(n,k)
```

```
Accuracy n =  100 & k =  1 :  0.76
Accuracy n =  100 & k =  3 :  0.92
Accuracy n =  100 & k =  5 :  0.96
Accuracy n =  100 & k =  7 :  0.96
Accuracy n =  100 & k =  9 :  0.92
Accuracy n =  100 & k =  11 :  0.92
Accuracy n =  100 & k =  13 :  0.88
Accuracy n =  100 & k =  15 :  0.88
Accuracy n =  100 & k =  17 :  0.88
Accuracy n =  100 & k =  19 :  0.88
Accuracy n =  100 & k =  21 :  0.88
Accuracy n =  100 & k =  23 :  0.88
Accuracy n =  100 & k =  25 :  0.88
Accuracy n =  100 & k =  27 :  0.84
Accuracy n =  100 & k =  29 :  0.84
Accuracy n =  100 & k =  31 :  0.84
Accuracy n =  100 & k =  33 :  0.84
Accuracy n =  100 & k =  35 :  0.84
Accuracy n =  100 & k =  37 :  0.84
Accuracy n =  100 & k =  39 :  0.84
Accuracy n =  100 & k =  41 :  0.84
Accuracy n =  100 & k =  43 :  0.76
Accuracy n =  100 & k =  45 :  0.76
Accuracy n =  100 & k =  47 :  0.68
Accuracy n =  100 & k =  49 :  0.68
Accuracy n =  500 & k =  1 :  0.912
Accuracy n =  500 & k =  3 :  0.912
Accuracy n =  500 & k =  5 :  0.936
Accuracy n =  500 & k =  7 :  0.952
Accuracy n =  500 & k =  9 :  0.944
Accuracy n =  500 & k =  11 :  0.928
Accuracy n =  500 & k =  13 :  0.928
Accuracy n =  500 & k =  15 :  0.936
Accuracy n =  500 & k =  17 :  0.944
Accuracy n =  500 & k =  19 :  0.936
Accuracy n =  500 & k =  21 :  0.944
Accuracy n =  500 & k =  23 :  0.944
Accuracy n =  500 & k =  25 :  0.952
Accuracy n =  500 & k =  27 :  0.944
Accuracy n =  500 & k =  29 :  0.944
Accuracy n =  500 & k =  31 :  0.944
Accuracy n =  500 & k =  33 :  0.944
Accuracy n =  500 & k =  35 :  0.936
Accuracy n =  500 & k =  37 :  0.944
Accuracy n =  500 & k =  39 :  0.936
Accuracy n =  500 & k =  41 :  0.944
Accuracy n =  500 & k =  43 :  0.928
Accuracy n =  500 & k =  45 :  0.936
Accuracy n =  500 & k =  47 :  0.928
Accuracy n =  500 & k =  49 :  0.936
Accuracy n =  1000 & k =  1 :  0.864
Accuracy n =  1000 & k =  3 :  0.876
Accuracy n =  1000 & k =  5 :  0.904
Accuracy n =  1000 & k =  7 :  0.916
Accuracy n =  1000 & k =  9 :  0.92
Accuracy n =  1000 & k =  11 :  0.916
Accuracy n =  1000 & k =  13 :  0.916
Accuracy n =  1000 & k =  15 :  0.92
Accuracy n =  1000 & k =  17 :  0.908
Accuracy n =  1000 & k =  19 :  0.916
Accuracy n =  1000 & k =  21 :  0.912
Accuracy n =  1000 & k =  23 :  0.908
Accuracy n =  1000 & k =  25 :  0.908
Accuracy n =  1000 & k =  27 :  0.908
Accuracy n =  1000 & k =  29 :  0.916
Accuracy n =  1000 & k =  31 :  0.908
Accuracy n =  1000 & k =  33 :  0.908
Accuracy n =  1000 & k =  35 :  0.912
Accuracy n =  1000 & k =  37 :  0.908
Accuracy n =  1000 & k =  39 :  0.912
Accuracy n =  1000 & k =  41 :  0.912
Accuracy n =  1000 & k =  43 :  0.912
Accuracy n =  1000 & k =  45 :  0.912
Accuracy n =  1000 & k =  47 :  0.912
Accuracy n =  1000 & k =  49 :  0.912
Accuracy n =  5000 & k =  1 :  0.8896
Accuracy n =  5000 & k =  3 :  0.904
Accuracy n =  5000 & k =  5 :  0.9112
Accuracy n =  5000 & k =  7 :  0.9152
Accuracy n =  5000 & k =  9 :  0.9184
Accuracy n =  5000 & k =  11 :  0.92
Accuracy n =  5000 & k =  13 :  0.9192
Accuracy n =  5000 & k =  15 :  0.92
Accuracy n =  5000 & k =  17 :  0.9216
Accuracy n =  5000 & k =  19 :  0.924
Accuracy n =  5000 & k =  21 :  0.924
Accuracy n =  5000 & k =  23 :  0.9264
Accuracy n =  5000 & k =  25 :  0.928
Accuracy n =  5000 & k =  27 :  0.924
Accuracy n =  5000 & k =  29 :  0.9232
Accuracy n =  5000 & k =  31 :  0.9256
Accuracy n =  5000 & k =  33 :  0.9232
Accuracy n =  5000 & k =  35 :  0.9264
Accuracy n =  5000 & k =  37 :  0.928
Accuracy n =  5000 & k =  39 :  0.928
Accuracy n =  5000 & k =  41 :  0.9272
Accuracy n =  5000 & k =  43 :  0.9272
Accuracy n =  5000 & k =  45 :  0.9272
Accuracy n =  5000 & k =  47 :  0.9264
Accuracy n =  5000 & k =  49 :  0.928
```

In [1]:

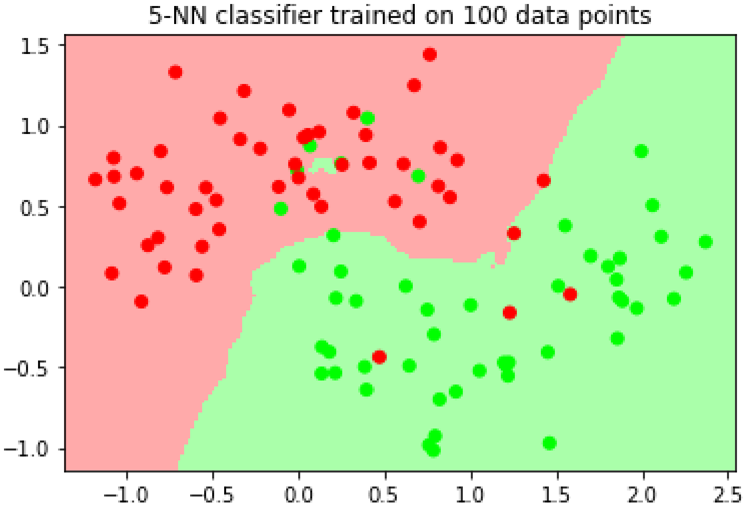
```
'''
As we can see,
n=100, k=5 or 7
n=500, k=7 or 25
n=1000, k=9 or 15
n=5000, k=23 or 35 or 47
These value for n and k yield best accuracy
'''
```

Out[1]:

```
'\nAs we can see,\nn=100, k=5 or 7\nn=500, k=7 or 25\nn=1000, k=9 or
15\nn=5000, k=23 or 35 or 47\nThese value for n and k yield best accu
racy \n'
```

```
In [63]: knearn(100,5)
```

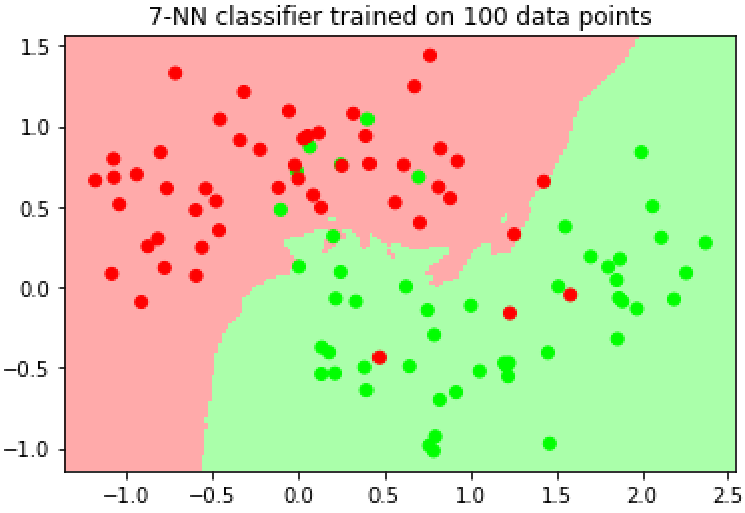
```
Accuracy n =  100 & k =  5 :  0.96
```



5-NN classifier trained on 100 data points

```
In [64]: knearn(100,7)
```

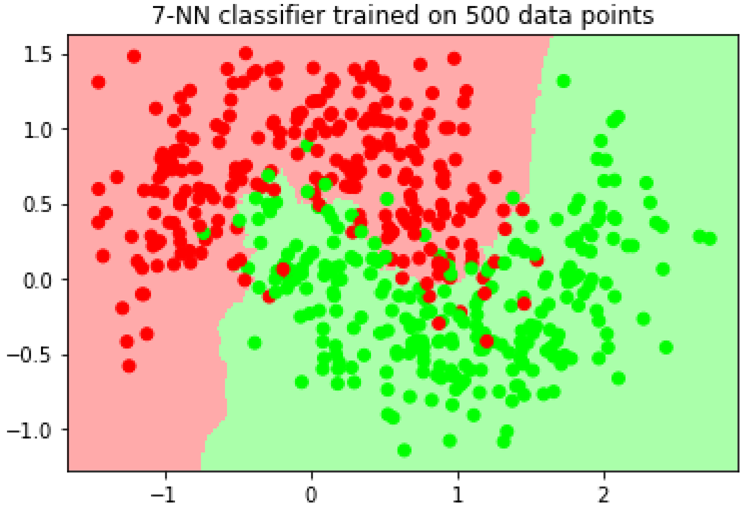
```
Accuracy n =  100 & k =  7 :  0.96
```



7-NN classifier trained on 100 data points

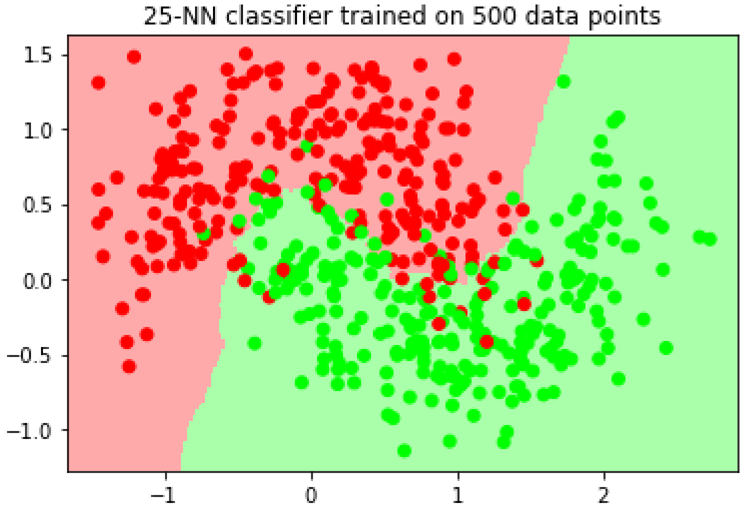
In [65]:
```
knearn(500,7)
```

```
Accuracy n =  500 & k =  7 :  0.952
```


7-NN classifier trained on 500 data points

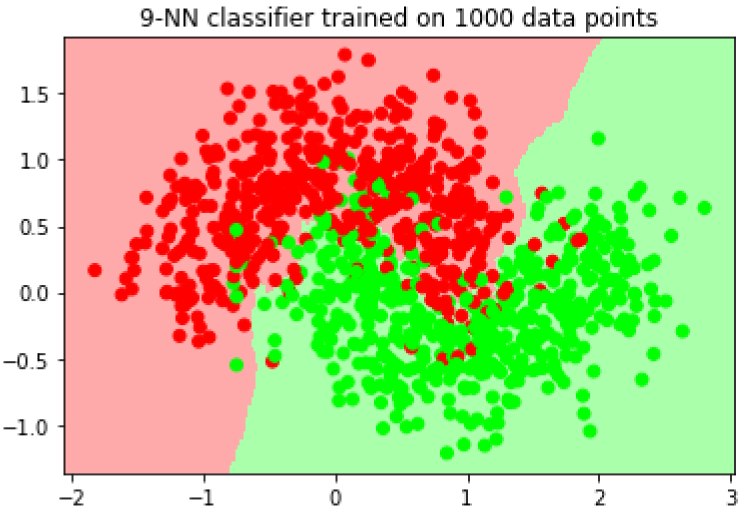
In [66]:
```
knearn(500,25)
```

```
Accuracy n =  500 & k =  25 :  0.952
```


25-NN classifier trained on 500 data points

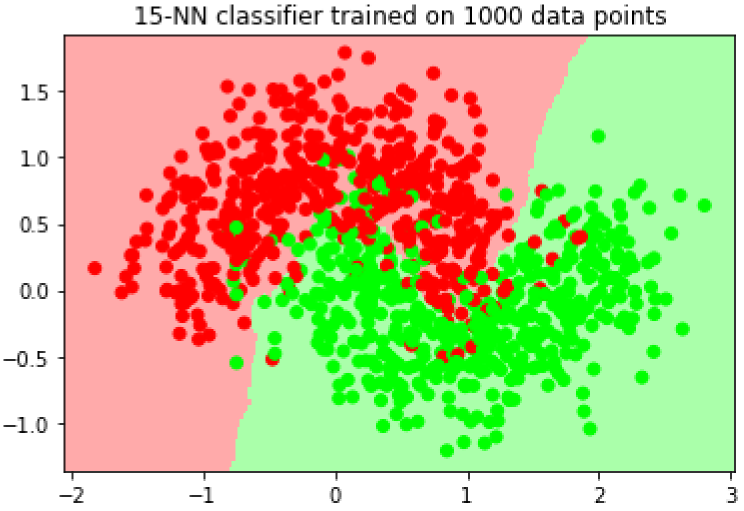
```
In [67]: knearn(1000,9)
```

```
Accuracy n =  1000 & k =  9 :  0.92
```



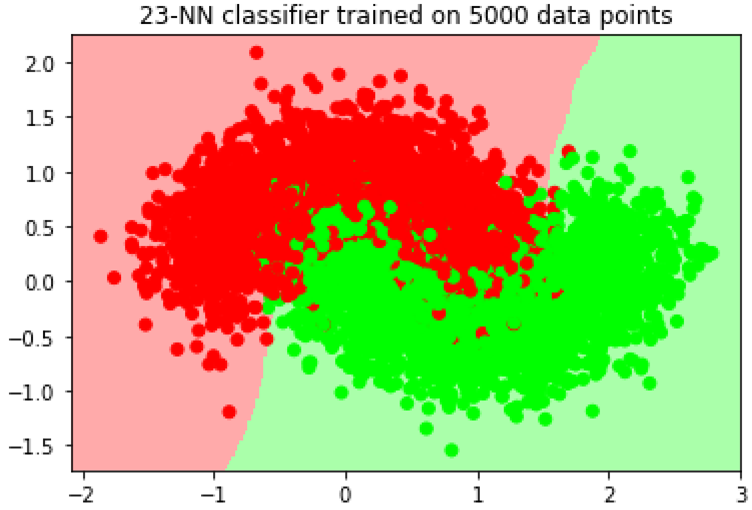
```
In [68]: knearn(1000,15)
```

```
Accuracy n =  1000 & k =  15 :  0.92
```



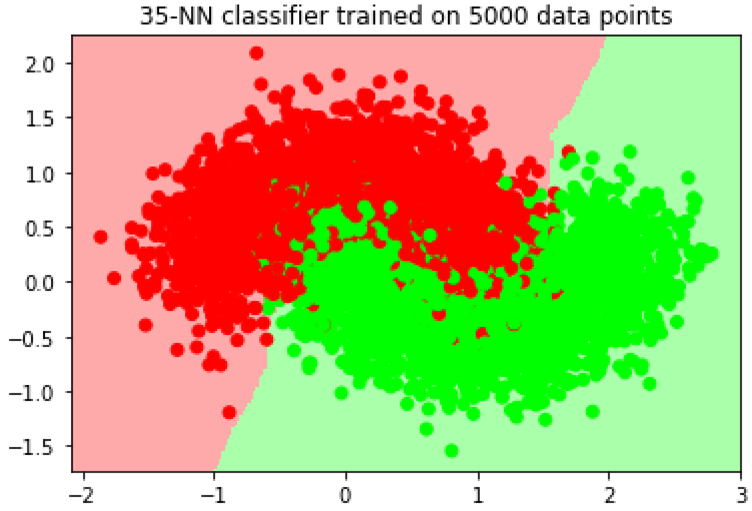
```
In [69]: knearn(5000,23)
```

```
Accuracy n =  5000 & k =  23 :  0.9264
```



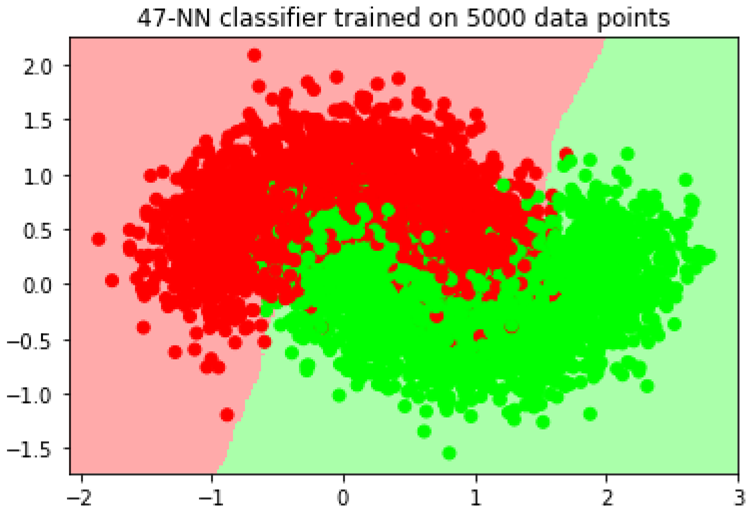
```
In [70]: knearn(5000,35)
```

```
Accuracy n =  5000 & k =  35 :  0.9264
```



```
In [71]: knearn(5000,47)
```

```
Accuracy n =  5000 & k =  47 :  0.9264
```


47-NN classifier trained on 5000 data points