Lab_Report_2

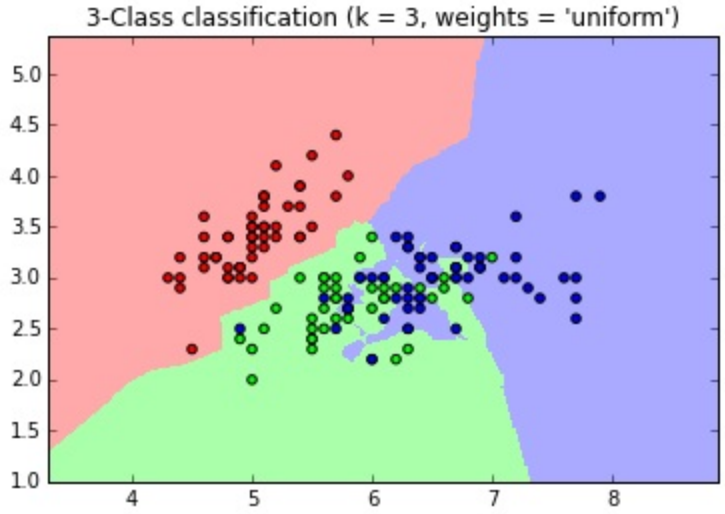
March 23, 2015

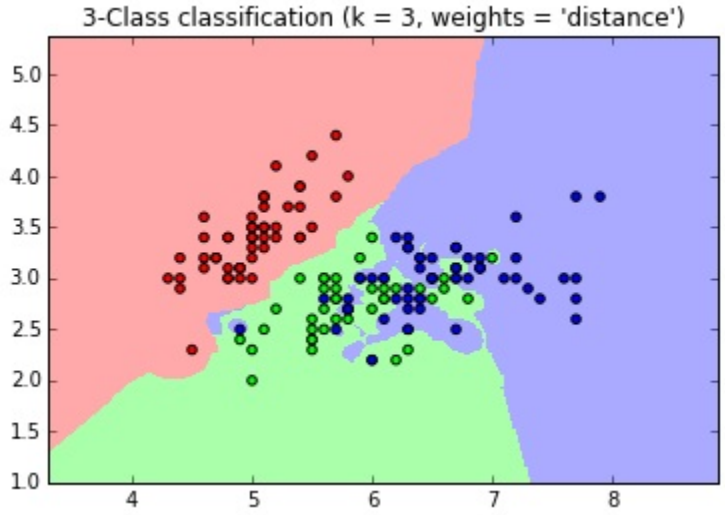
In [24]:
```
#%matplotlib inline
```

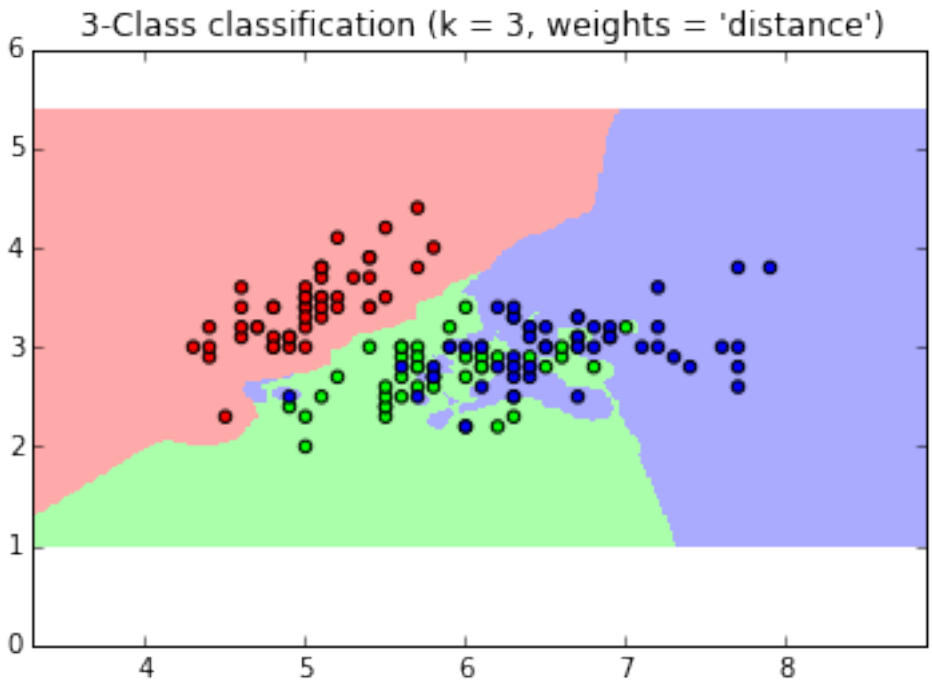
In [25]:
```
"""
Author Muhammed Khan
================================
Nearest Neighbors Classification
================================

Sample usage of Nearest Neighbors classification.
It will plot the decision boundaries for each class.
"""
print(__doc__)

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import neighbors, datasets
from IPython.display import display
from IPython.display import Image

a=Image(filename=('C:/Users/AliSamsung/Documents/Magic Briefcase/Coursera/Python/1.jpg'))
b=Image(filename=('C:/Users/AliSamsung/Documents/Magic Briefcase/Coursera/Python/2.jpg'))

#knn uses Euclidean distance, distance if you could use a ruler
#to connect two points.
#The balance between overfitting and underfitting the data is known as bias-
#variance tradeoff. Choosing a large k reduces the impact of variance caused
#by noisy data but can bias the learner, such that it ignores the smaller but
# important patterns
n_neighbors = 3
# I use a k-value that attemps to balance bias and variance of a large number
# of values by square rooting the total number of values

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2]  # we only take the first two features. We could
                      # avoid this ugly slicing by using a two-dim dataset
y = iris.target
h = .02  # step size in the mesh


# Create color maps
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])
```

```
        for weights in ['uniform', 'distance']:
        # we create an instance of Neighbours Classifier and fit the data.
        # the weights of the NN are given more weight than the further neighbors
        # to offset the bias variance tradeoff somewhat in the distance argument
        # vs the uniform voting rights argument.
            clf = neighbors.KNeighborsClassifier(n_neighbors , weights=weights)
            clf.fit(X, y)

        # Plot the decision boundary. For that, we will assign a color to each
        # point in the mesh [x_min, m_max]x[y_min, y_max].
        x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
        y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
        xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                             np.arange(y_min, y_max, h))
        Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

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

        # Plot also the training points
        plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold)
        plt.xlim(xx.min(), xx.max())
        plt.title("3-Class classification (k = %i, weights = '%s')"
                  % (n_neighbors , weights))

        display(a)
        display(b)
        #Interpretation of results:
        #When k=3, the total number of values in the array, the default sqrt/k=12
        #value was more accurate in differentiating between the gree and blue areas of
        #the visualization, if values were to be imputed; that is to say in a larger
        #area with larger patterns. However between smaller, values the classifier lost
        #a good number of its accuracy improperly classifying many points on the borders
        # when k=12.

        #Conversely, when k=3 the data was highly accurate, but much of the graph
        #that could be filled with more data points more liekly overfit new data.

        #For both k-values it seems like the distance voting was better at classifying
        #than uniform voting

        #plt.show()
```

```
Author Muhammed Khan
================================
Nearest Neighbors Classification
================================

Sample usage of Nearest Neighbors classification.
It will plot the decision boundaries for each class.
```


3-Class classification (k = 3, weights = 'uniform')
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8


3-Class classification (k = 3, weights = 'distance')
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8


3-Class classification (k = 3, weights = 'distance')
6
5
4
3
2
1
0
4
5
6
7
8