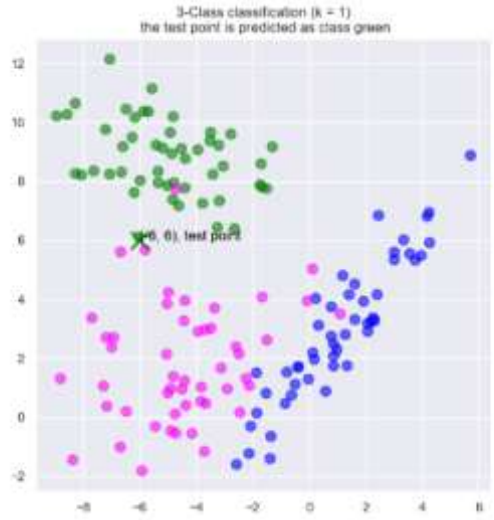
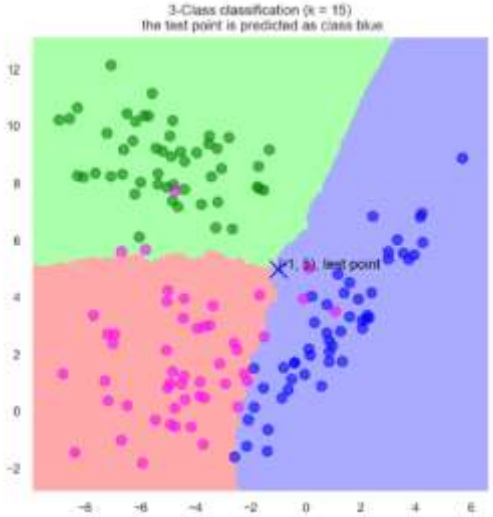
6.1P: kNN classification

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#%matplotlib inline
from matplotlib.colors import ListedColormap
from sklearn import neighbors

sns.set()


dataset=pd.read_csv('task6_1_dataset.csv')
x1=dataset['x1']
x2=dataset['x2']
X_train = np.c_[x1, x2]

Y_train = dataset['y'].values
print(type(Y_train))

k = 1
knn = neighbors.KNeighborsClassifier(k)
knn.fit(X_train, Y_train)

cmap_bold = ListedColormap(['green', 'blue', 'magenta'])

fig,ax = plt.subplots(figsize=(7, 7), dpi=100)

scale = 75
alpha = 0.6
colors = ['green', 'blue', 'magenta']

ax.scatter(X_train[:, 0], X_train[:, 1], c=Y_train, cmap=cmap_bold, alpha=alpha, s=scale)

X_test = [[-6,6]]
Y_pred = knn.predict(X_test)

ax.scatter(X_test[0][0], X_test[0][1], marker="x", s=3*scale, lw=2, c=colors[Y_pred.astype(int)[0]])


ax.set_title("3-Class classification (k = {})\n the test point is predicted as class {}".format(k, colors[Y_pred.astype(int)[0]]
ax.text(X_test[0][0], X_test[0][1], '(%d, %d), %s' % (X_test[0][0], X_test[0][1],'test point') , color='black')


k = 15
knn = neighbors.KNeighborsClassifier(k)
knn.fit(X_train, Y_train)

h = 0.05

cmap_light = ListedColormap(['#AAFFAA', '#AAAAFF', '#FFAAAA'])

x1_min, x1_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
x2_min, x2_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, h), np.arange(x2_min, x2_max, h))

Z = knn.predict(np.c_[xx1.ravel(), xx2.ravel()])


Z = Z.reshape(xx1.shape)

fig,ax = plt.subplots(figsize=(7, 7), dpi=100)
ax.pcolormesh(xx1, xx2, Z, cmap=cmap_light)

ax.scatter(X_train[:, 0], X_train[:, 1], c=Y_train, cmap=cmap_bold, alpha=alpha, s=scale)

plt.xlim(xx1.min(), xx1.max())
plt.ylim(xx2.min(), xx2.max())


X_test = [[-1, 5]]
Y_pred = knn.predict(X_test)
ax.scatter(X_test[0][0], X_test[0][1], alpha=0.95, color=colors[Y_pred.astype(int)[0]], marker='x', s=3*scale)

ax.set_title("3-Class classification (k = {})\n the test point is predicted as class {}".format(k, colors[Y_pred.astype(int)[0]]

ax.text(X_test[0][0], X_test[0][1], '(%d, %d), %s' % (X_test[0][0], X_test[0][1],'test point'), color='black' )
```





This Python software uses a fictitious dataset with two features (x1 and x2) to illustrate k-Nearest Neighbors (k-NN) classification. The script starts by importing the required libraries, which include pandas for data processing, matplotlib and seaborn for visualization, numpy for numerical computations, and sklearn for the k-NN method.

The dataset is loaded into a pandas Data Frame, where the class labels are stored in Y_train and the features are stored in X_train. When the k-NN classifier is originally configured with k=1, it predicts a test point's class by comparing it to its nearest neighbor in the training set. A scatter plot is used to show the training data, with distinct colors used for each class. The predicted class of the test point is highlighted.

After that, the script raises k to 15 and retrains the model, producing a fine grid that shows how the decision boundaries alter as k increases. To demonstrate the impact of having more neighbors on the classification outcome, another test point is classified and shown. This method demonstrates how k-NN functions and how changing k affects the model's bounds for decisions and predictions.

Understanding how the algorithm classifies fresh data points according to their proximity to already labeled data is made simpler by the display. It also emphasizes how crucial it is to select the right k value in order to strike a compromise between decision boundary smoothness and model sensitivity.