k-nearestneighborsclassification

In [ ]:

```
#matplotlib inline
```

Load data: XOR¶

In [5]:

```
from google.colab import drive
drive.mount('/content/gdrive')
!ls "/content/gdrive/My Drive/Colab Notebooks"


import pandas as pd
xor_data = pd.read_csv("/content/gdrive/My Drive/Colab Notebooks/XOR.csv",header=None)
print(xor_data)
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount('
 butterfly_n2000.csv                 Tutorial_PyGRNN.ipynb   XOR.csv
'k-nearest neighbors classification.ipynb'   WordCloud.ipynb
   0  1  2
0  0  0  0
1  0  1  1
2  1  0  1
3  1  1  0
```

In [6]:

```
X = xor_data[[0,1]].to_numpy()
y = xor_data[2].to_numpy()
print(X)
print(y)
```

```
[[0 0]
 [0 1]
 [1 0]
 [1 1]]
[0 1 1 0]
```

Classify XOR dataset¶

In [15]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import neighbors, datasets

n_neighbors = 3  # k parameter
weights = 'distance'  # 'uniform' or 'distance'

clf = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
clf.fit(X, y)

y_pred = clf.predict(X)

print(y_pred)
```

```
[0 1 1 0]
```

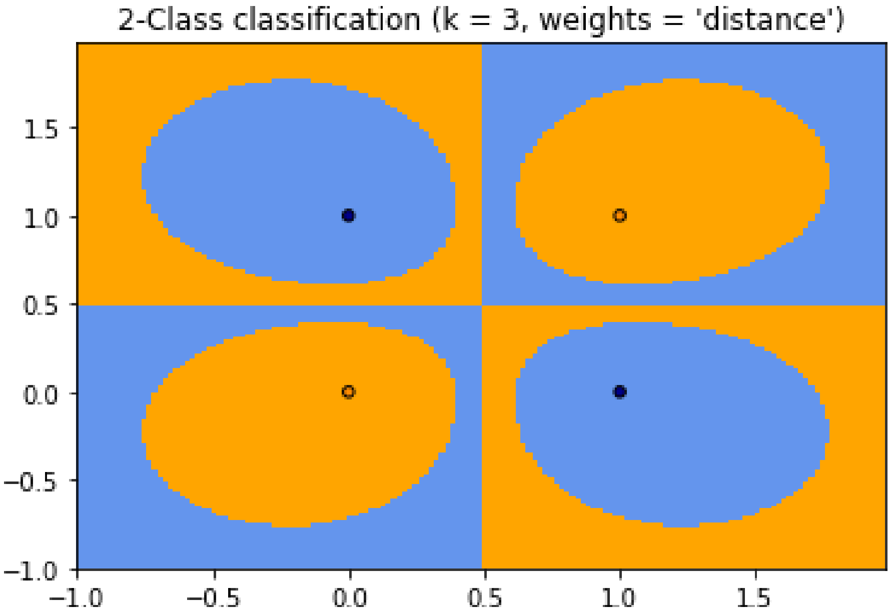
Plot classification surface: decision boundary¶

In [16]:

```
h = .02  # step size in the mesh
# Create color maps
cmap_light = ListedColormap(['orange', 'cyan', 'cornflowerblue'])
cmap_bold = ListedColormap(['darkorange', 'c', 'darkblue'])

# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_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,
                edgecolor='k', s=20)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("2-Class classification (k = %i, weights = '%s')"
              % (n_neighbors, weights))
```

Out[16]:

```
Text(0.5, 1.0, "2-Class classification (k = 3, weights = 'distance')")
```



Nearest Neighbors Classification: IRIS dataset¶

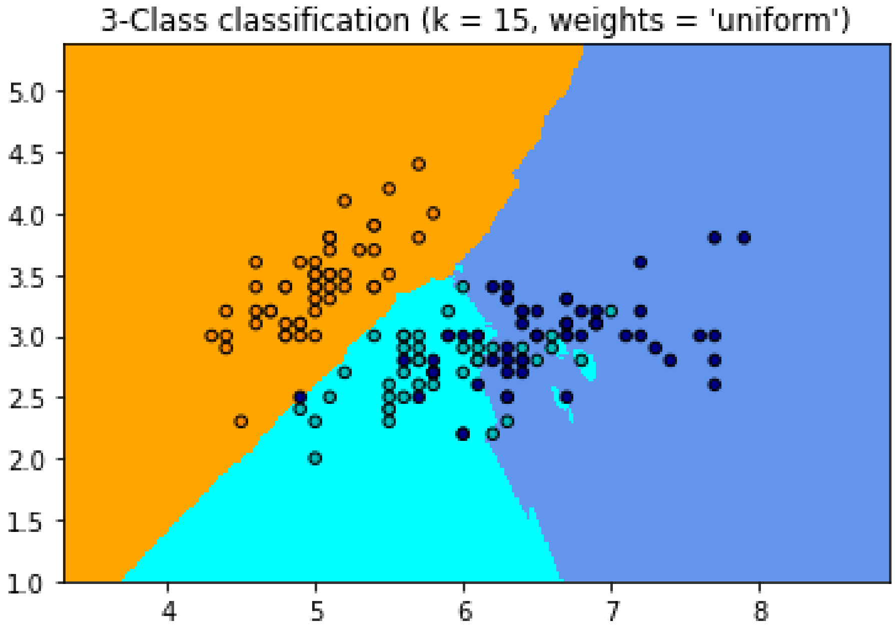
Sample usage of Nearest Neighbors classification. It will plot the decision boundaries for each class.

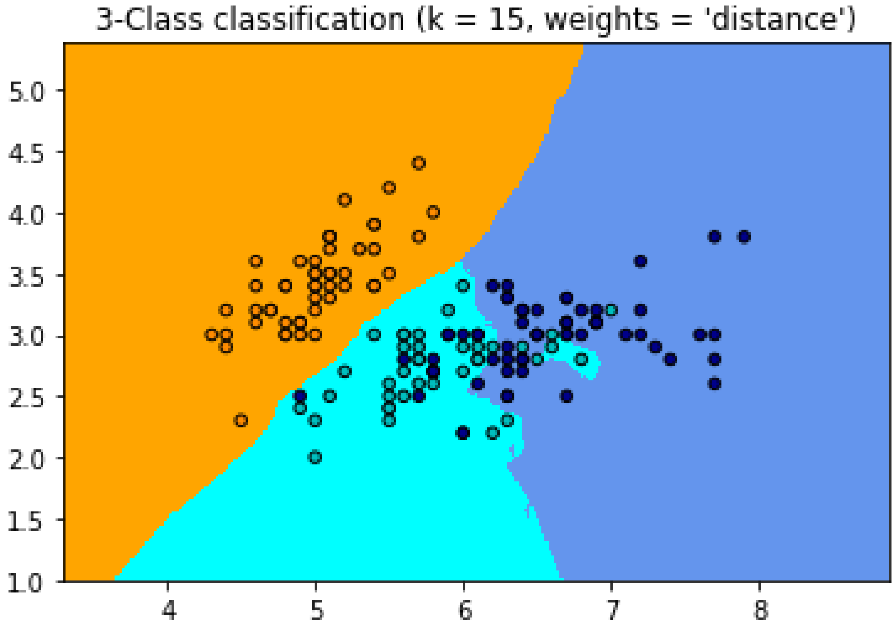
In [ ]:

```
print(__doc__)

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

n_neighbors = 15

# import some data to play with
iris = datasets.load_iris()
```

```
# we only take the first two features. We could avoid this ugly
# slicing by using a two-dim dataset
X = iris.data[:, :2]
y = iris.target

h = .02  # step size in the mesh

# Create color maps
cmap_light = ListedColormap(['orange', 'cyan', 'cornflowerblue'])
cmap_bold = ListedColormap(['darkorange', 'c', 'darkblue'])

for weights in ['uniform', 'distance']:
    # we create an instance of Neighbours Classifier and fit the data.
    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, x_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,
                edgecolor='k', s=20)
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title("3-Class classification (k = %i, weights = '%s')"
              % (n_neighbors, weights))

plt.show()
```

```
Automatically created module for IPython interactive environment
```


3-Class classification (k = 15, 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 = 15, 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