Experiment - 06

k-Nearest Neighbors Classifier Using ScikitLearn

In [1]:
```
print('-------------------------------EXPERIMENT-06--------------------
--------------')
print('NAME:  Pratyush Srivastava')
print('ROLL NO: 18SCSE1010128')
```

```
-------------------------------EXPERIMENT-06--------------------------
-------
NAME:  Pratyush Srivastava
ROLL NO: 18SCSE1010128
```

In [2]:
```
print('-------------------------Nearest Neighbors--------------------
--------------')
```

```
-------------------------Nearest Neighbors--------------------------
-------
```

In [3]:
```
from sklearn.neighbors import NearestNeighbors
import numpy as np
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
nbrs = NearestNeighbors(n_neighbors=2, algorithm='ball_tree').fit(X)
distances, indices = nbrs.kneighbors(X)
indices
```

Out[3]:
```
array([[0, 1],
       [1, 0],
       [2, 1],
       [3, 4],
```

```
       [4, 3],
       [5, 4]], dtype=int64)
```

In [4]:
```
distances
```

Out[4]:
```
array([[0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.41421356],
       [0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.41421356]])
```

In [5]:
```
nbrs.kneighbors_graph(X).toarray()
```

Out[5]:
```
array([[1., 1., 0., 0., 0., 0.],
       [1., 1., 0., 0., 0., 0.],
       [0., 1., 1., 0., 0., 0.],
       [0., 0., 0., 1., 1., 0.],
       [0., 0., 0., 1., 1., 0.],
       [0., 0., 0., 0., 1., 1.]])
```

In [6]:
```
from sklearn.neighbors import KDTree
import numpy as np
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
kdt = KDTree(X, leaf_size=30, metric='euclidean')
kdt.query(X, k=2, return_distance=False)
```

Out[6]:
```
array([[0, 1],
       [1, 0],
       [2, 1],
       [3, 4],
       [4, 3],
       [5, 4]], dtype=int64)
```

In [7]:
```
from sklearn.neighbors import NearestCentroid
import numpy as np
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
y = np.array([1, 1, 1, 2, 2, 2])
```

```
clf = NearestCentroid()
clf.fit(X, y)
```

Out[7]:
```
NearestCentroid()
```

In [8]:
```
print(clf.predict([[-0.8, -1]]))
```

```
[1]
```

In [9]:
```
from sklearn.manifold import Isomap
from sklearn.neighbors import KNeighborsTransformer
from sklearn.pipeline import make_pipeline
estimator = make_pipeline(KNeighborsTransformer(n_neighbors=5, mode='di
stance'),Isomap(neighbors_algorithm='precomputed'),
                          memory='/path/to/cache')
```

In [10]:
```
from sklearn.neighbors import (NeighborhoodComponentsAnalysis,KNeighbor
sClassifier)
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y,stratify=y, te
st_size=0.7, random_state=42)
nca = NeighborhoodComponentsAnalysis(random_state=42)
knn = KNeighborsClassifier(n_neighbors=3)
nca_pipe = Pipeline([('nca', nca), ('knn', knn)])
nca_pipe.fit(X_train, y_train)
```

Out[10]:
```
Pipeline(steps=[('nca', NeighborhoodComponentsAnalysis(random_state=4
2)),
                ('knn', KNeighborsClassifier(n_neighbors=3))])
```

In [11]:
```
print(nca_pipe.score(X_test, y_test))
```

```
0.9619047619047619
```

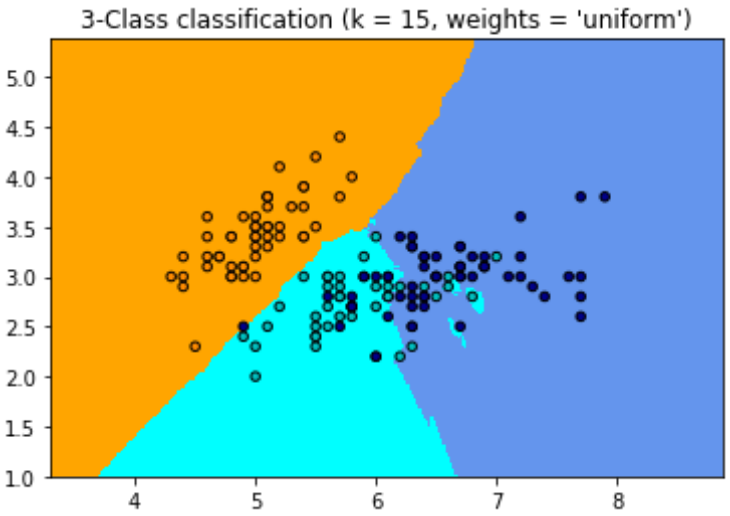
In [12]:
```
print('-------------------------Nearest Neighbors Classification------
```

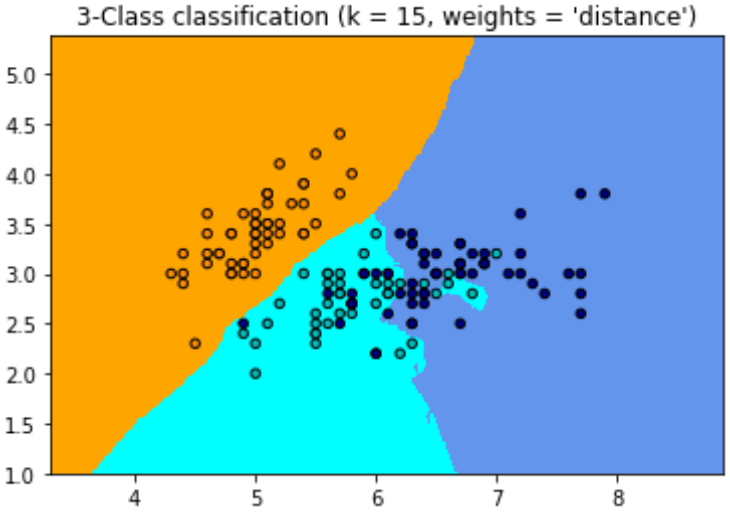
```
-----------------------------')
```

```
-------------------------Nearest Neighbors Classification------------
----------------------
```

In [13]:

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

n_neighbors = 15
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target

h = .02
cmap_light = ListedColormap(['orange', 'cyan', 'cornflowerblue'])
cmap_bold = ListedColormap(['darkorange', 'c', 'darkblue'])

for weights in ['uniform', 'distance']:

    clf = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
    clf.fit(X, y)
    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()])
    Z = Z.reshape(xx.shape)
    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
    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()
```

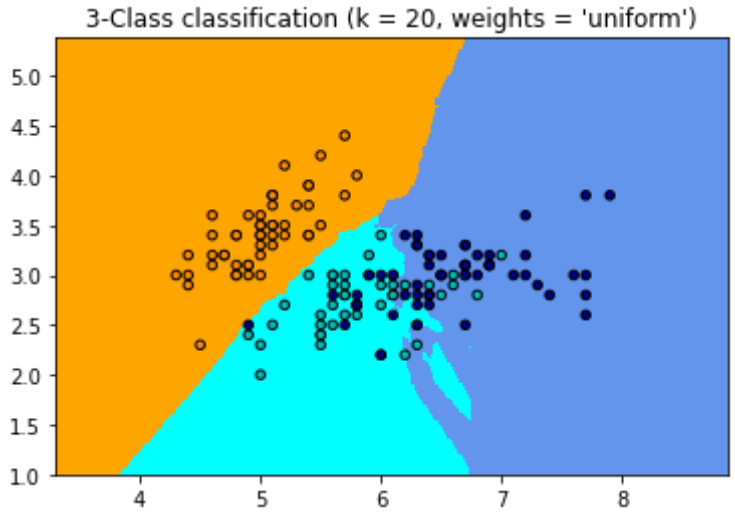


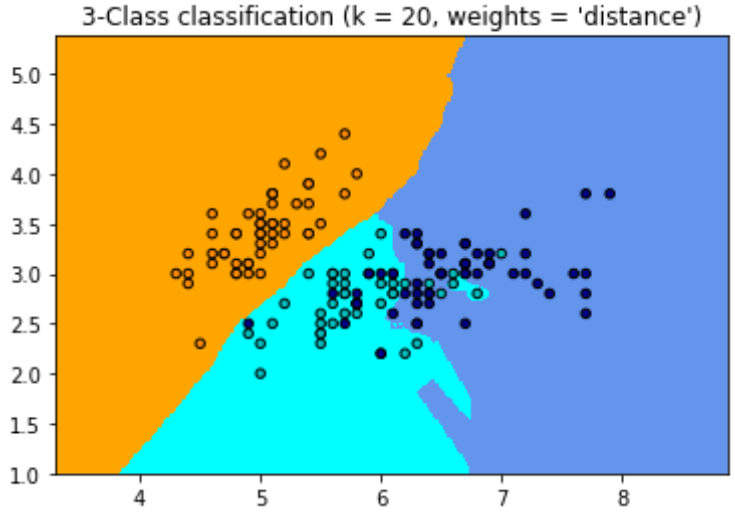


In [14]:

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

n_neighbors = 20
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target

h = .02
cmap_light = ListedColormap(['orange', 'cyan', 'cornflowerblue'])
cmap_bold = ListedColormap(['darkorange', 'c', 'darkblue'])

for weights in ['uniform', 'distance']:

    clf = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
    clf.fit(X, y)
    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()])
    Z = Z.reshape(xx.shape)
    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
    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()
```



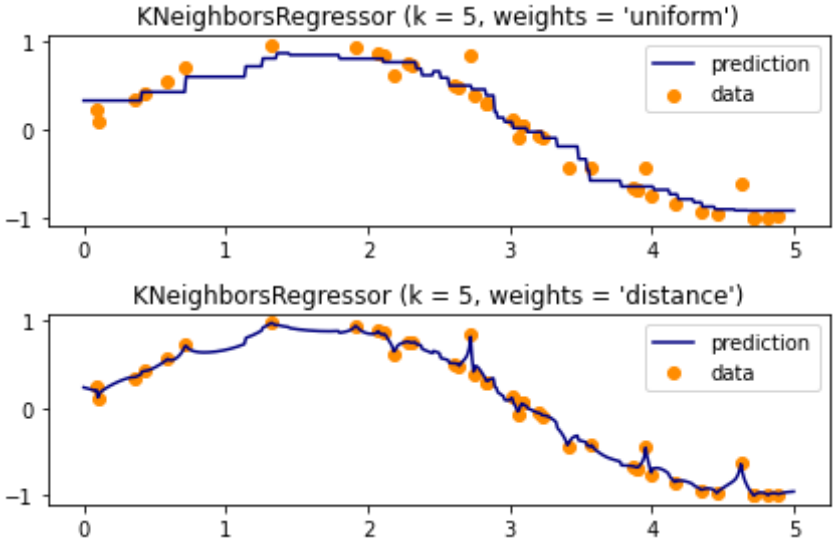


In [15]:
```
print('-----------------------------------Nearest Neighbors regression-
-------------------------------------')
```

```
-----------------------------------Nearest Neighbors regression--------
--------------------------------
```

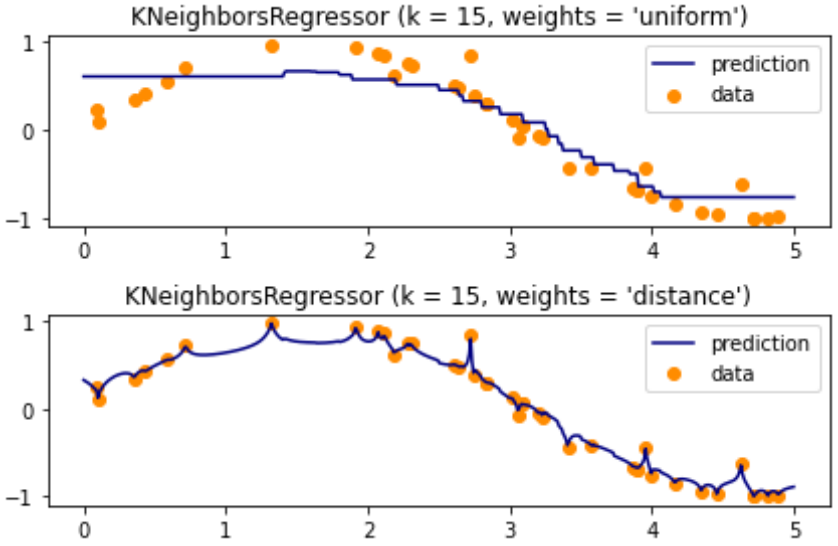
In [16]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import neighbors

np.random.seed(0)
X = np.sort(5 * np.random.rand(40, 1), axis=0)
T = np.linspace(0, 5, 500)[:, np.newaxis]
y = np.sin(X).ravel()

# Add noise to targets
y[::5] += 1 * (0.5 - np.random.rand(8))

n_neighbors = 5

for i, weights in enumerate(['uniform', 'distance']):
    knn = neighbors.KNeighborsRegressor(n_neighbors, weights=weights)
    y_ = knn.fit(X, y).predict(T)

    plt.subplot(2, 1, i + 1)
    plt.scatter(X, y, color='darkorange', label='data')
    plt.plot(T, y_, color='navy', label='prediction')
    plt.axis('tight')
    plt.legend()
    plt.title("KNeighborsRegressor (k = %i, weights = '%s')" % (n_neigh
bors,
                                                                weights
))
```

```
plt.tight_layout()
plt.show()
```



In [17]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import neighbors

np.random.seed(0)
X = np.sort(5 * np.random.rand(40, 1), axis=0)
T = np.linspace(0, 5, 500)[:, np.newaxis]
y = np.sin(X).ravel()

# Add noise to targets
y[::5] += 1 * (0.5 - np.random.rand(8))

n_neighbors = 15

for i, weights in enumerate(['uniform', 'distance']):
    knn = neighbors.KNeighborsRegressor(n_neighbors, weights=weights)
    y_ = knn.fit(X, y).predict(T)

    plt.subplot(2, 1, i + 1)
```

```
    plt.scatter(X, y, color='darkorange', label='data')
    plt.plot(T, y_, color='navy', label='prediction')
    plt.axis('tight')
    plt.legend()
    plt.title("KNeighborsRegressor (k = %i, weights = '%s')" % (n_neigh
bors,
                                                                weights
))

plt.tight_layout()
```



In [18]:
```
print('-------------------------EXPERIMENT-06 Ended--------------------
-------------')
```

```
-------------------------EXPERIMENT-06 Ended--------------------------
------
```

In [ ]:
```

```