KNN

Imports

```
import os
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

from mlxtend.plotting import plot_decision_regions
```

KNN visualization

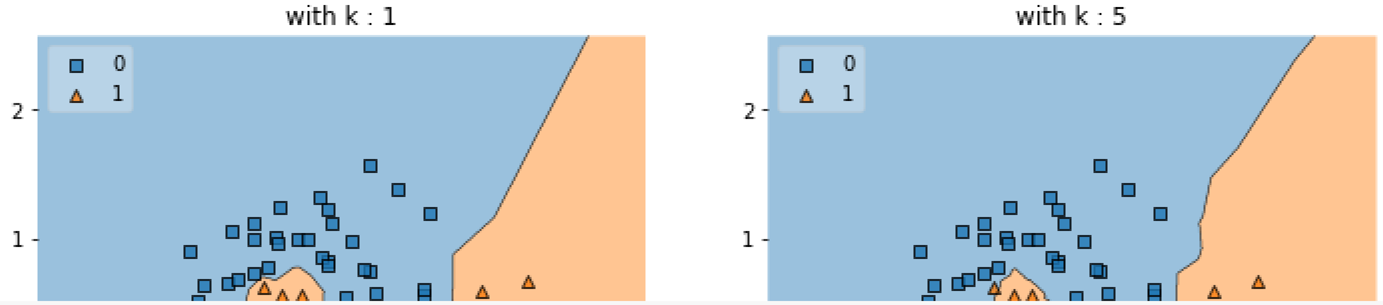
```
base_path = "/content/drive/MyDrive/AAIC/Datasets/Toy Data/"
files = [file for file in os.listdir(base_path)]
files
```

```
['1.ushape.csv',
 '2.concerticcir1.csv',
 '3.concertriccir2.csv',
 '4.linearsep.csv',
 '5.outlier.csv',
 '6.overlap.csv',
 '7.xor.csv',
 '8.twospirals.csv',
 '9.random.csv']
```

```
def knn_plot(data, ax, k_val = 3, use_plt_xtend = False):
  x = data[:,0:2]
  y = data[:, 2]
  model = KNeighborsClassifier(n_neighbors=k_val)
  model.fit(x,y)

  if use_plt_xtend == True:
    plot_decision_regions(x, y.astype(int), clf=model, legend=2, ax=ax)
    ax.set_title("with k : {}".format(k_val))
  else:
    h = 0.1
    min_x = np.min(x[:,0])-1
    max_x = np.max(x[:,0])+1
    max_x = np.max(x[:,0])+1
    min_y = np.min(x[:,1])-1
    max_y = np.max(x[:,1])+1

    xx,yy = np.meshgrid(np.arange(min_x, max_x, h), np.arange(min_y, max_y, h))
    z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    z = z.reshape(xx.shape)

    cmap_light = ListedColormap(['#FFAAAA',  '#AAAAFF'])
    cmap_bold = ListedColormap(['#FF0000', '#0000FF'])
    ax.set_title("with k : {}".format(k_val))
    ax.pcolormesh(xx, yy, z, cmap=cmap_light)
    ax.scatter(x[:,0], x[:,1], c=y, cmap=cmap_bold)

def knn_file(fname, use_plt_xtend = False):
  print("For the data :::: ", fname)
  k_vals = [1, 5, 15, 30]
  data = np.genfromtxt(base_path+fname, delimiter=",")
  fig, axes = plt.subplots(2, 2, figsize=(12,12))
  for i,k in enumerate(k_vals):
    knn_plot(data, axes[i//2, i%2] ,k_val=k, use_plt_xtend=use_plt_xtend)
  plt.show()
```

```
knn_file(files[0], use_plt_xtend=True)
```

```
For the data ::::  1.ushape.csv
/usr/local/lib/python3.7/dist-packages/mlxtend/plotting/decision_regions.py:244: Matplotl
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(), y_max=yy.max())
/usr/local/lib/python3.7/dist-packages/mlxtend/plotting/decision_regions.py:244: Matplotl
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(), y_max=yy.max())
/usr/local/lib/python3.7/dist-packages/mlxtend/plotting/decision_regions.py:244: Matplotl
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(), y_max=yy.max())
/usr/local/lib/python3.7/dist-packages/mlxtend/plotting/decision_regions.py:244: Matplotl
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(), y_max=yy.max())
```

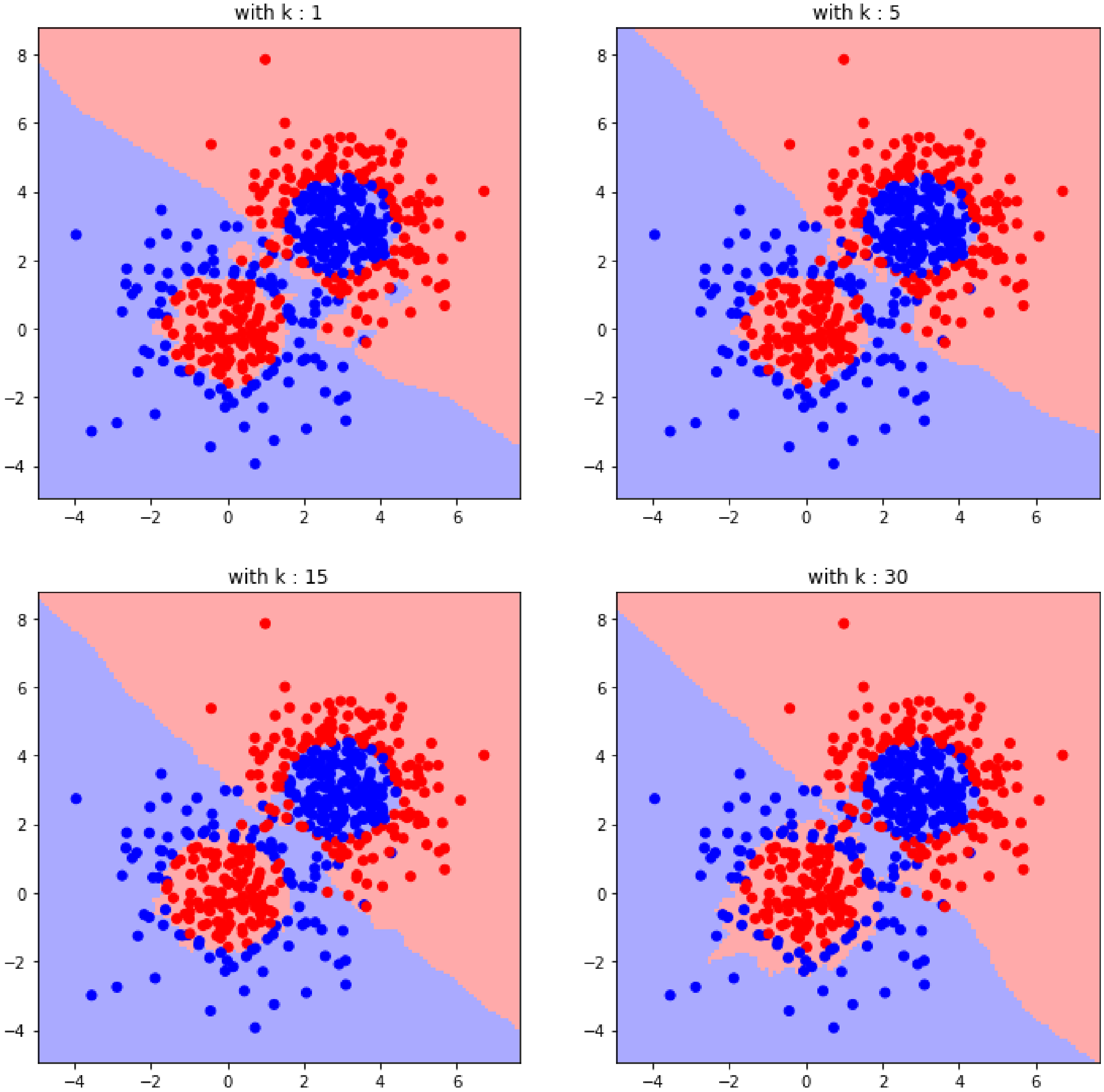


```
knn_file(files[1])
```

```
For the data ::::  2.concerticcir1.csv
```

```
knn_file(files[2])
```

```
For the data ::::  3.concertriccir2.csv
```



```
knn_file(files[3])
```

```
knn_file(files[4])
```

```
knn_file(files[5])
```

```
knn_file(files[6])
```

```
knn_file(files[7])
knn_file(files[7])
```

```
knn_file(files[8])
```

Accuracy Score (KNN With 1-fold cross validation)

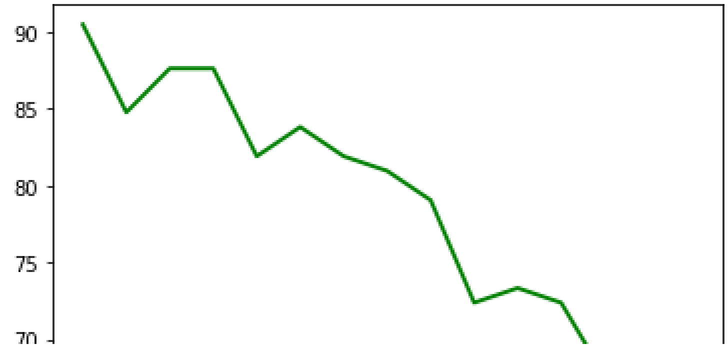
```
fname = '3.concertriccir2.csv'
df = pd.read_csv(base_path+fname, names=["X", "Y", "Class"])
print(df.shape)
X = np.array(df.iloc[:,0:2])
Y = np.array(df.iloc[:,2])
print(X[0:2])
print(Y[0:2])
```

```
(500, 3)
[[ 0.70033457 -0.24706758]
 [-3.95001869  2.74007953]]
[0. 1.]
```

```
X_1, X_test, Y_1, Y_test = train_test_split(X, Y, test_size=0.3, random_state=0)
X_train, X_cv, Y_train, Y_cv = train_test_split(X_1, Y_1, test_size=0.3, random_state=0)
print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(245, 2)
(105, 2)
(150, 2)
```

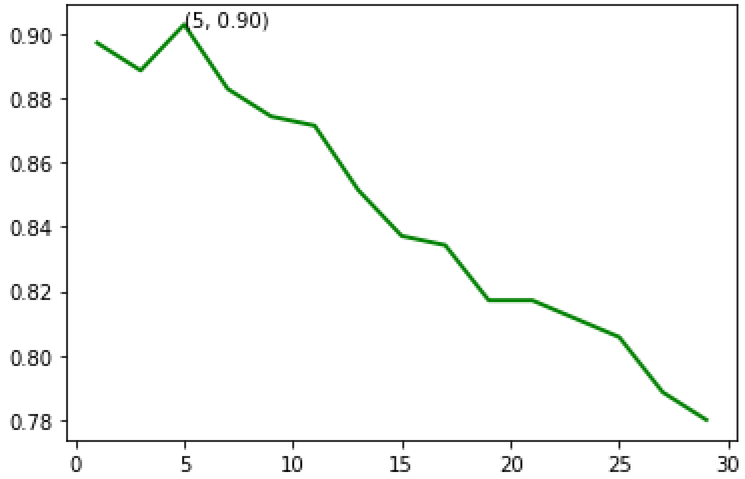
```
k_vals = np.arange(1, 30, 2)
accuracy = []

for k in k_vals:
  model = KNeighborsClassifier(n_neighbors=k)
  model.fit(X_train, Y_train)
  predict = model.predict(X_cv)
  acc = accuracy_score(Y_cv, predict, normalize=True) * float(100)
  accuracy.append(acc)

plt.plot(k_vals, accuracy, linewidth=2, color='green')
plt.show()

k_acc = k_vals[np.argmax(accuracy)] #K value with max accuracy
y_pred = model.predict(X_test)
accuracy_test = accuracy_score(Y_test, y_pred, normalize=True)*100.0
print("For the k={} value, the accuracy of X_test is {}%".format(k_acc, accuracy_test))
```



Accuracy Score (KNN With 10-fold cross validation)

```
k_vals = np.arange(1, 30, 2)
accuracy = []

for k in k_vals:
  model = KNeighborsClassifier(n_neighbors=k)
  model.fit(X_train, Y_train)
  predict = model.predict(X_cv)
  #acc = accuracy_score(Y_cv, predict, normalize=True) * float(100)
  acc_scores = cross_val_score(model, X_1, Y_1, cv=10, scoring="accuracy")
  accuracy.append(np.mean(acc_scores))

scores = { x:y for x,y in zip(k_vals, accuracy)}

k_acc = k_vals[np.argmax(accuracy)] #K value with max accuracy
print((k_acc, scores[k_acc]))
plt.plot(k_vals, accuracy, linewidth=2, color='green')
plt.annotate("({}, {:0.2f})".format(k_acc, scores[k_acc]), xy=(k_acc, scores[k_acc]), textcoor
plt.show()

y_pred = model.predict(X_test)
accuracy_test = accuracy_score(Y_test, y_pred, normalize=True)*100.0
print("For the k={} value, the accuracy of X_test is {}%".format(k_acc, accuracy_test))
```

```
(5, 0.9028571428571428)
```



```
For the k=5 value, the accuracy of X_test is 67.33333333333333%
```

✓ 0s completed at 6:55 PM ● ✕