import itertools

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.ticker import NullFormatter

import pandas as pd

import numpy as np

import matplotlib.ticker as ticker

from sklearn import preprocessing

%matplotlib inline

df['custcat'].value\_counts()

df.hist(column='income', bins=50)

X = df[['region', 'tenure','age', 'marital', 'address', 'income', 'ed', 'employ','retire', 'gender', 'reside']] .values #.astype(float)

X[0:5]

y = df['custcat'].values

y[0:5]

X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))

X[0:5]

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.2, random\_state=4)

print ('Train set:', X\_train.shape, y\_train.shape)

print ('Test set:', X\_test.shape, y\_test.shape)

from sklearn.neighbors import KNeighborsClassifier

k = 4

#Train Model and Predict

neigh = KNeighborsClassifier(n\_neighbors = k).fit(X\_train,y\_train)

neigh

yhat = neigh.predict(X\_test)

yhat[0:5]

from sklearn import metrics

print("Train set Accuracy: ", metrics.accuracy\_score(y\_train, neigh.predict(X\_train)))

print("Test set Accuracy: ", metrics.accuracy\_score(y\_test, yhat))

<!-- Your answer is below:

k = 6

neigh6 = KNeighborsClassifier(n\_neighbors = k).fit(X\_train,y\_train)

yhat6 = neigh6.predict(X\_test)

print("Train set Accuracy: ", metrics.accuracy\_score(y\_train, neigh6.predict(X\_train)))

print("Test set Accuracy: ", metrics.accuracy\_score(y\_test, yhat6))

-->

Ks = 10

mean\_acc = np.zeros((Ks-1))

std\_acc = np.zeros((Ks-1))

ConfustionMx = [];

for n in range(1,Ks):

#Train Model and Predict

neigh = KNeighborsClassifier(n\_neighbors = n).fit(X\_train,y\_train)

yhat=neigh.predict(X\_test)

mean\_acc[n-1] = metrics.accuracy\_score(y\_test, yhat)

std\_acc[n-1]=np.std(yhat==y\_test)/np.sqrt(yhat.shape[0])

mean\_acc

plt.plot(range(1,Ks),mean\_acc,'g')

plt.fill\_between(range(1,Ks),mean\_acc - 1 \* std\_acc,mean\_acc + 1 \* std\_acc, alpha=0.10)

plt.legend(('Accuracy ', '+/- 3xstd'))

plt.ylabel('Accuracy ')

plt.xlabel('Number of Nabors (K)')

plt.tight\_layout()

plt.show()

print( "The best accuracy was with", mean\_acc.max(), "with k=", mean\_acc.argmax()+1)