KNN Alogrithm Classification Example

```
#Objective is to build classifier and predict class of unknown classes using K-nearest n

import itertools
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import matplotlib.ticker as ticker

from matplotlib.ticker import NullFormatter
from sklearn import preprocessing
from matplotlib import rc,font_manager

ticks_font = font_manager.FontProperties(family='Times New Roman', style='normal',
    size=12, weight='normal', stretch='normal')
ax=plt.gca()

## Loading Data ##
df=pd.read_csv('D:\Python\edx\Machine Learning\Classification\elecom_customer_data.csv')
with open('KNN.txt','a') as f:
    print(df.head(),file=f)
    print(df.describe(),file=f)
    print('Classes are: ', df['custcat'].value_counts(),file=f)

#1- Basic Service 2- E-Service 3- Plus Service 4- Total Service#


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

##Feature Set X,Y##

print(df.columns)
X=df[['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed',
       'employ', 'retire', 'gender', 'reside',]].values
X[0:5]

y=df['custcat'].values
y[0:5]

# Normalize Data #
X=preprocessing.StandardScaler().fit(X).transform(X.astype(float))
X[0:5]
```

```
# Train Test Split#
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)
with open('KNN.txt','a') as f:
    print('Train set: ', X_train.shape,y_train.shape,file=f)
    print('Test set: ', X_test.shape,y_test.shape,file=f)

# Classification - K nearest neighbour#

from sklearn.neighbors import KNeighborsClassifier

k=9 # Train model and predict with k=9 (best value for high accuracy, check KNN_k_values
neigh=KNeighborsClassifier(n_neighbors=k).fit(X_train,y_train)
yhat=neigh.predict(X_test)
with open('KNN.txt','a') as f:
    print(yhat[0:5],file=f)

#Predict using accuracy_score which is similar to jaccard_similarity_score function#

from sklearn import metrics
with open('KNN.txt','a') as f:
    print('Train set accuracy: ', metrics.accuracy_score(y_train,neigh.predict(X_train))
    print('Test set accuracy: ',metrics.accuracy_score(y_test,yhat),file=f)

plt.show()
```

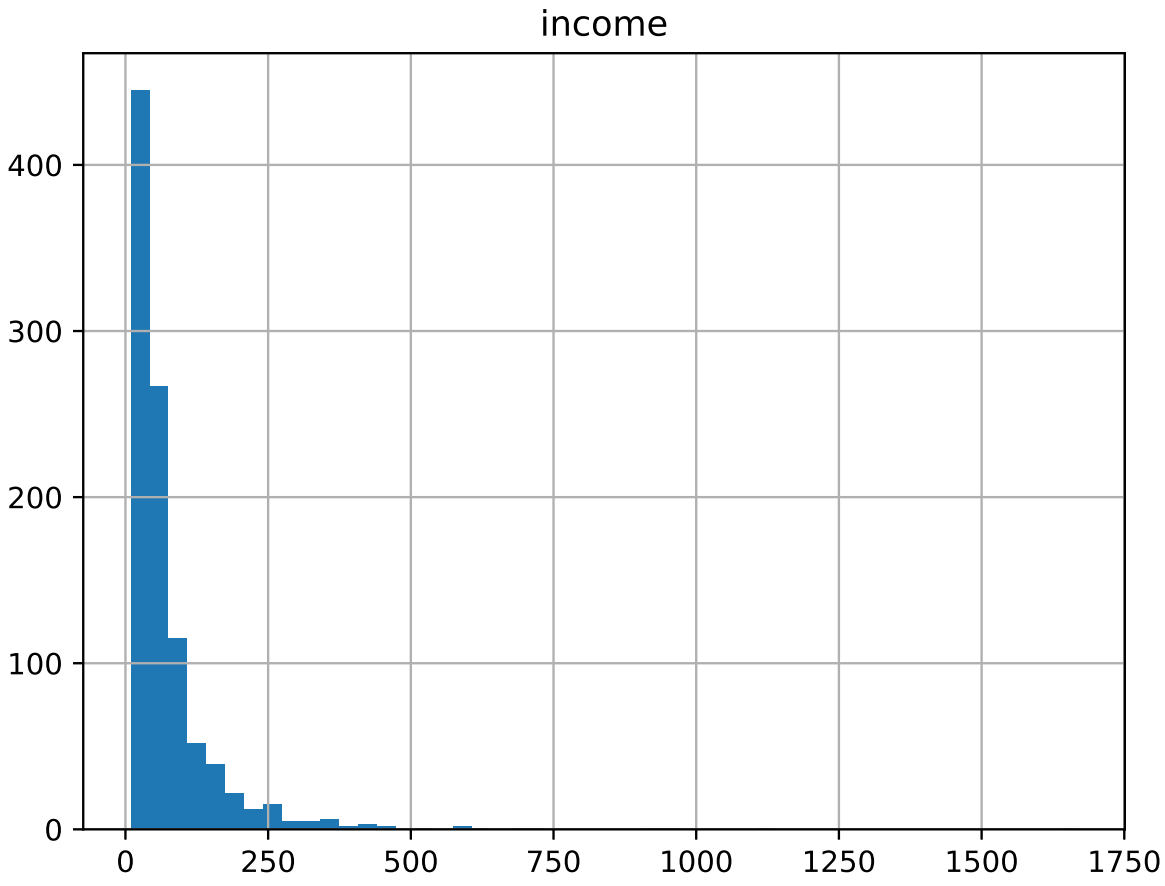
Solution:

```
   region  tenure  age  marital  address  income  ed  employ  retire  gender  reside
custcat
0       2      13   44        1        9    64.0   4       5     0.0       0
2        1
1       3      11   33        1        7   136.0   5       5     0.0       0
6        4
2       3      68   52        1       24   116.0   1      29     0.0       1
2        3
3       2      33   33        0       12    33.0   2       0     0.0       1
1        1
4       2      23   30        1        9    30.0   1       2     0.0       0
4        3
          region       tenure          age      marital      address       income
ed       employ       retire       gender       reside      custcat
count  1000.0000  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000
1000.000000  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000
```

```
mean       2.0220    35.526000    41.684000     0.495000    11.551000    77.535000
2.671000    10.987000     0.047000     0.517000     2.331000     2.487000
std        0.8162    21.359812    12.558816     0.500225    10.086681   107.044165
1.222397    10.082087     0.211745     0.499961     1.435793     1.120306
min        1.0000     1.000000    18.000000     0.000000     0.000000     9.000000
1.000000     0.000000     0.000000     0.000000     1.000000     1.000000
25%        1.0000    17.000000    32.000000     0.000000     3.000000    29.000000
2.000000     3.000000     0.000000     0.000000     1.000000     1.000000
50%        2.0000    34.000000    40.000000     0.000000     9.000000    47.000000
3.000000     8.000000     0.000000     1.000000     2.000000     3.000000
75%        3.0000    54.000000    51.000000     1.000000    18.000000    83.000000
4.000000    17.000000     0.000000     1.000000     3.000000     3.000000
max        3.0000    72.000000    77.000000     1.000000    55.000000  1668.000000
5.000000    47.000000     1.000000     1.000000     8.000000     4.000000
Classes are:  3    281
1    266
4    236
2    217
Name: custcat, dtype: int64
Train set:  (800, 11) (800,)
Test set:  (200, 11) (200,)
[3 1 3 2 4]
Train set accuracy:  0.5025
Test set accuracy:  0.34
```

KNN Alogrithm k Value program

```
#Objective is to find best value of K in KNN classifier to achieve maximum accuracy

import itertools
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import matplotlib.ticker as ticker

from matplotlib.ticker import NullFormatter
from sklearn import preprocessing
from matplotlib import rc,font_manager

ticks_font = font_manager.FontProperties(family='Times New Roman', style='normal',
    size=12, weight='normal', stretch='normal')
```

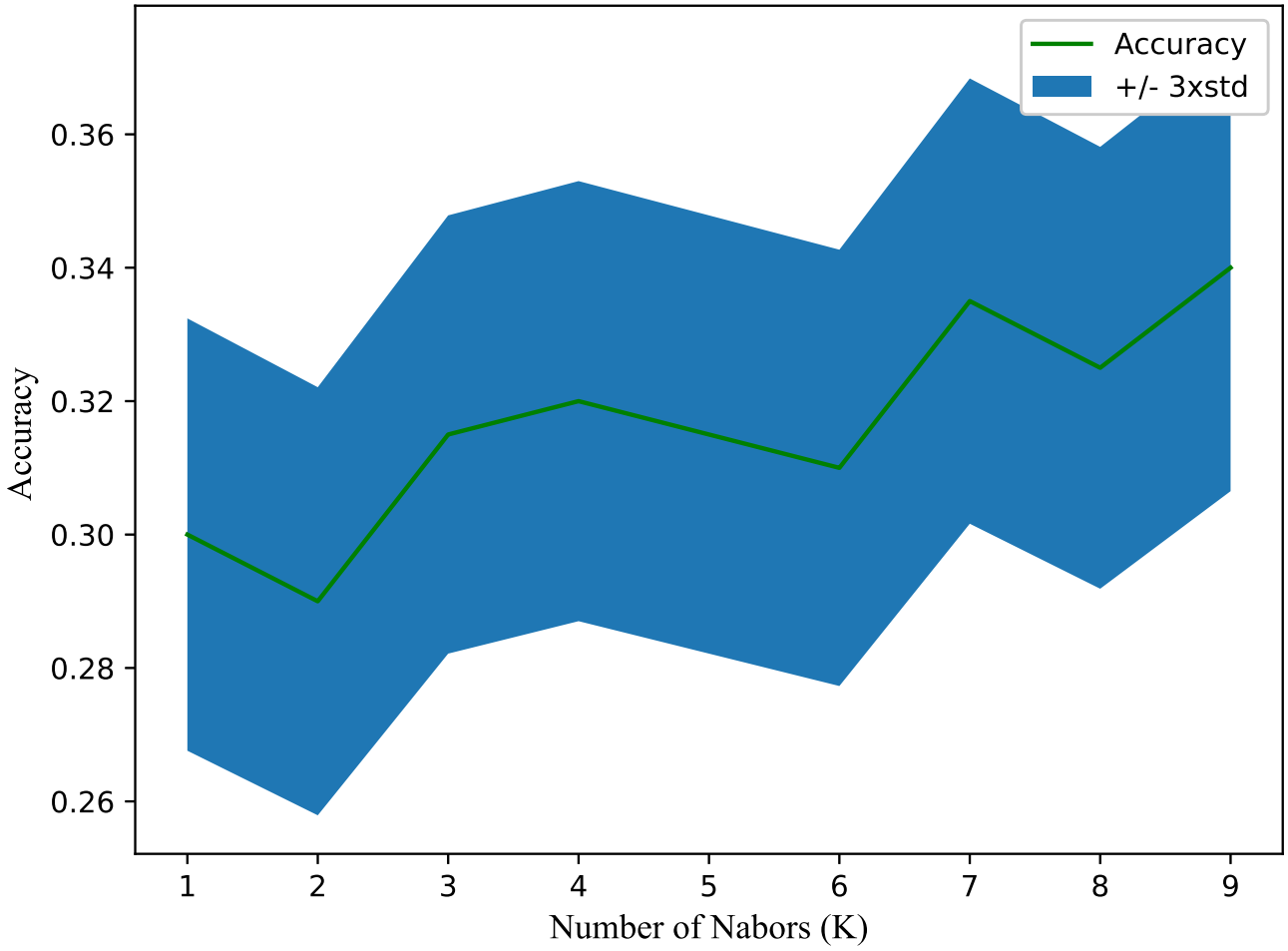

income
400
300
200
100
0
0
250
500
750
1000
1250
1500
1750

```
ax=plt.gca()

## Loading Data ##
df=pd.read_csv('D:\Python\edx\Machine Learning\Classification\elecom_customer_data.csv')

#1- Basic Service 2- E-Service 3- Plus Service 4- Total Service#
##Feature Set X,Y##

print(df.columns)
X=df[['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed',
       'employ', 'retire', 'gender', 'reside',]].values
X[0:5]
y=df['custcat'].values
y[0:5]

# Normalize Data #
X=preprocessing.StandardScaler().fit(X).transform(X.astype(float))
X[0:5]

# Train Test Split#
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)

#Finding the value of k to achieve maximum accuracy via Confusion matrix and then use th
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
Ks=10
acc_mean=np.zeros((Ks-1))
acc_std=np.zeros((Ks-1))
Cmat=[]
for n in range(1,Ks):
    neigh=KNeighborsClassifier(n_neighbors=n).fit(X_train,y_train)
    yhat=neigh.predict(X_test)
    acc_mean[n-1]=metrics.accuracy_score(y_test,yhat)

    acc_std[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])
with open('KNN_kvalue.txt','a') as f:
    print(acc_mean,file=f)
    print('The best accuracy was with: ', acc_mean.max(), 'with k= ',acc_mean.argmax()+1

#Plot of k values vs accuracy #
plt.figure()
plt.plot(range(1,Ks),acc_mean,'g')
plt.fill_between(range(1,Ks),acc_mean - 1 * acc_std,acc_mean + 1 * acc_std, alpha=0.10)
plt.legend(('Accuracy ', '+/- 3xstd'))
plt.ylabel('Accuracy ',fontname='Times New Roman',fontsize=12)
```

```
plt.xlabel('Number of Nabors (K)',fontname='Times New Roman',fontsize=12)
plt.tight_layout()
plt.show()
```

Solution:

```
[0.3   0.29  0.315 0.32  0.315 0.31  0.335 0.325 0.34 ]
The best accuracy was with:  0.34 with k=  9
```


Accuracy
+/- 3xstd
0.36
0.34
0.32
0.30
0.28
0.26
Accuracy
1
2
3
4
5
6
7
8
9
Number of Nabors (K)