In [28]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import random
import math
import matplotlib.pyplot as plt
from sklearn import datasets,linear_model
from sklearn.model_selection import train_test_split
import operator
```

In [29]:
```
data1 = pd.read_csv('train_final.csv')
data2=pd.read_csv('test_final.csv')
data1.head()
```

Out[29]:

	id	Type	LifeStyle	Vacation	eCredit	salary	property	label
0	1	student	spend>saving	6	40	13.62	3.2804	C1
1	2	student	spend>saving	11	21	15.32	2.0232	C1
2	3	student	spend>saving	7	64	16.55	3.1202	C1
3	4	student	spend>saving	3	47	15.71	3.4022	C1
4	5	student	spend>saving	15	10	16.96	2.2825	C1

In [30]:
```
data1_numeric=data1[['Vacation','eCredit','salary','property']]
data1_numeric.head()
```

Out[30]:

	Vacation	eCredit	salary	property
0	6	40	13.62	3.2804
1	11	21	15.32	2.0232
2	7	64	16.55	3.1202
3	3	47	15.71	3.4022
4	15	10	16.96	2.2825

In [31]:
```
data2_numeric=data2[['Vacation','eCredit','salary','property']]
data2_numeric.head()
```

Out[31]:

	Vacation	eCredit	salary	property
0	12	19	14.7900	3.7697
1	29	10	16.1900	2.4839
2	28	60	15.4600	1.1885
3	15	41	21.2600	1.4379
4	2	9	19.7207	0.6913

In [32]:
```
data1_norm = (data1_numeric-data1_numeric.min())/(data1_numeric.max()
data1_norm.head()
```

Out[32]:

	Vacation	eCredit	salary	property
0	0.079365	0.107558	0.219960	0.183167
1	0.158730	0.052326	0.293102	0.112797
2	0.095238	0.177326	0.346023	0.174200
3	0.031746	0.127907	0.309882	0.189984
4	0.222222	0.020349	0.363663	0.127311

In [33]:
```
data2_norm = (data2_numeric-data2_numeric.min())/(data2_numeric.max()
data2_norm.head()
```

Out[33]:

	Vacation	eCredit	salary	property
0	0.20	0.058824	0.104637	0.398926
1	0.54	0.021008	0.175059	0.243041
2	0.52	0.231092	0.138339	0.085992
3	0.26	0.151261	0.430086	0.116229
4	0.00	0.016807	0.352657	0.025714

In [34]:
```
data1=data1.drop(['Vacation','eCredit','salary','property'],axis=1)
data1.head()
```

Out[34]:

	id	Type	LifeStyle	label
0	1	student	spend>saving	C1
1	2	student	spend>saving	C1
2	3	student	spend>saving	C1
3	4	student	spend>saving	C1
4	5	student	spend>saving	C1

In [35]:
```
data2=data2.drop(['Vacation','eCredit','salary','property'],axis=1)
data2.head()
```

Out[35]:

	id	Type	LifeStyle	label
0	1	student	spend<saving	C1
1	2	student	spend>>saving	C1
2	3	student	spend<<saving	C1
3	4	engineer	spend>saving	C1
4	5	librarian	spend<saving	C1

In [36]:
```
data1_label=data1[['label']]
data1_label.head()
```

Out[36]:

	label
0	C1
1	C1
2	C1
3	C1
4	C1

In [37]:
```
data2_label=data2[['label']]
data2_label.head()
```

Out[37]:

	label
0	C1
1	C1
2	C1
3	C1
4	C1

In [38]:
```
data1=data1.drop(['label'],axis=1)
data1.head()
```

Out[38]:

	id	Type	LifeStyle
0	1	student	spend>saving
1	2	student	spend>saving
2	3	student	spend>saving
3	4	student	spend>saving
4	5	student	spend>saving

In [39]:
```
data2=data2.drop(['label'],axis=1)
data2.head()
```

Out[39]:

	id	Type	LifeStyle
0	1	student	spend<saving
1	2	student	spend>>saving
2	3	student	spend<<saving
3	4	engineer	spend>saving
4	5	librarian	spend<saving

In [40]:
```
xtrain = pd.concat([data1, data1_norm,data1_label], axis=1, sort=Fals
xtrain.head()
```

Out[40]:

	id	Type	LifeStyle	Vacation	eCredit	salary	property	label
0	1	student	spend>saving	0.079365	0.107558	0.219960	0.183167	C1
1	2	student	spend>saving	0.158730	0.052326	0.293102	0.112797	C1
2	3	student	spend>saving	0.095238	0.177326	0.346023	0.174200	C1
3	4	student	spend>saving	0.031746	0.127907	0.309882	0.189984	C1
4	5	student	spend>saving	0.222222	0.020349	0.363663	0.127311	C1

In [41]:
```
xtest = pd.concat([data2, data2_norm,data2_label], axis=1, sort=False
xtest.head()
```

Out[41]:

	id	Type	LifeStyle	Vacation	eCredit	salary	property	label
0	1	student	spend<saving	0.20	0.058824	0.104637	0.398926	C1
1	2	student	spend>>saving	0.54	0.021008	0.175059	0.243041	C1
2	3	student	spend<<saving	0.52	0.231092	0.138339	0.085992	C1
3	4	engineer	spend>saving	0.26	0.151261	0.430086	0.116229	C1
4	5	librarian	spend<saving	0.00	0.016807	0.352657	0.025714	C1

In [42]:
```
xtest.head()
```

Out[42]:

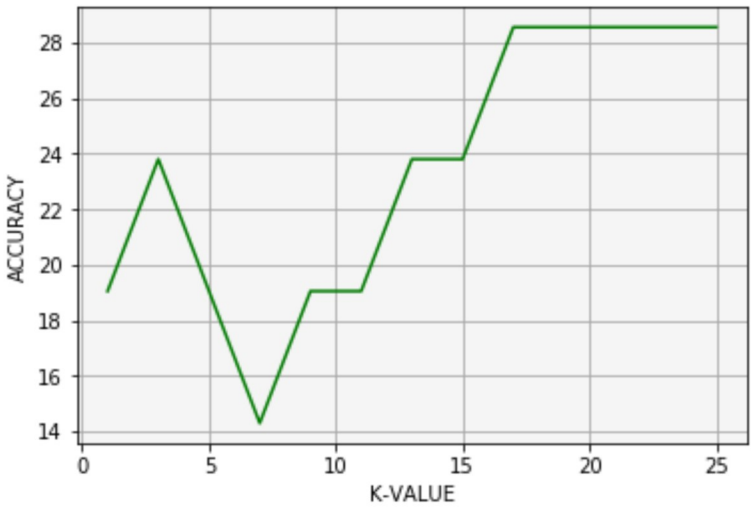
	id	Type	LifeStyle	Vacation	eCredit	salary	property	label
0	1	student	spend<saving	0.20	0.058824	0.104637	0.398926	C1
1	2	student	spend>>saving	0.54	0.021008	0.175059	0.243041	C1
2	3	student	spend<<saving	0.52	0.231092	0.138339	0.085992	C1
3	4	engineer	spend>saving	0.26	0.151261	0.430086	0.116229	C1
4	5	librarian	spend<saving	0.00	0.016807	0.352657	0.025714	C1

In [44]:

```
k_list=[]
acc_list=[]
for k in range(1,26,2):
    k_list.append(k)
    predict=[]
    def euc_distance(testrow,trainrow,length):
        distance=0
        for i in range(1,3):
            if(testrow[i]==trainrow[i]):
                distance+=1
#         for i in range(2):
#             if(testrow[i]==trainrow[i]):
#                 distance+=1
        for i in range(3,length-1):
            distance+=pow((testrow[i]-trainrow[i]),2)
        return math.sqrt(distance)
    def getNeighbours(traindata,testRow,k):
        distance_with_train=[]
        length=len(testRow)
        for x in range(len(traindata)):
            dist=euc_distance(testRow,traindata[x],length)
            distance_with_train.append((traindata[x],dist))
        distance_with_train.sort(key=operator.itemgetter(1))
        neighbors = []
        for x in range(k):
            neighbors.append(distance_with_train[x][0])
        return neighbors
    def getResponse(neighbors):
        votes = {}
        for x in range(len(neighbors)):
            response = neighbors[x][-1]
            if response in votes:
                votes[response] += 1
            else:
                votes[response] = 1
        sortedVotes = sorted(votes.items(), key=operator.itemgetter(1
        return sortedVotes[0][0]
    def getAccuracy(xtest, predict):
        correct = 0
        for x in range(len(xtest)):
            if xtest[x][-1] == predict[x]:
                correct += 1
        return (correct/float(len(xtest))) * 100.0
    for i in range(len(xtest)):
        neighbour=getNeighbours(xtrain.values,xtest.values[i],k)
    #     print(neighbour)
        result = getResponse(neighbour)
        predict.append(result)
#         print('> predicted=' + repr(result) + ', actual=' + repr(xt
    accuracy = getAccuracy(xtest.values, predict)
    acc_list.append(accuracy)
    print('Accuracy: ' + repr(accuracy) + '%','with k=',k)
```

```
Accuracy: 19.047619047619047% with k= 1
Accuracy: 23.809523809523807% with k= 3
Accuracy: 19.047619047619047% with k= 5
```

In [45]:
```
1 plt.plot(k_list,acc_list,color='green')
2 plt.xlabel('K-VALUE')
3 plt.ylabel('ACCURACY')
4 plt.grid(True)
5 plt.show()
```



In [ ]:
```
1
```