In [1]:

```
# here, we implement knn by removing categorical variables(Type,Lifes
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 [2]:

```
data1 = pd.read_csv('train_final.csv')
data2=pd.read_csv('test_final.csv')
# X_train, X_crossVal, X_test = np.split(data, [600, 800])
# X_test
# xtrain,xtest=train_test_split(data,test_size=0.25)
# train_data = data[:50]
# test_data = data[50:]
# train_data[2]
data1.head()

```

Out[2]:

	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 [3]:

```
data2.head()
```

Out[3]:

	id	Type	LifeStyle	Vacation	eCredit	salary	property	label
0	1	student	spend<saving	12	19	14.7900	3.7697	C1
1	2	student	spend>>saving	29	10	16.1900	2.4839	C1
2	3	student	spend<<saving	28	60	15.4600	1.1885	C1
3	4	engineer	spend>saving	15	41	21.2600	1.4379	C1
4	5	librarian	spend<saving	2	9	19.7207	0.6913	C1

In [4]:

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

Out[4]:

	label
0	C1
1	C1
2	C1
3	C1
4	C1

In [5]:

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

Out[5]:

	label
0	C1
1	C1
2	C1
3	C1
4	C1

In [6]:

```
data1=data1.drop(['id','Type','LifeStyle','label'],axis=1)
# data1.head()
data1.head()
```

Out[6]:

	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 [7]:

```
data2=data2.drop(['id','Type','LifeStyle','label'],axis=1)
data2.head()
```

Out[7]:

	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 [8]:

```python
data1_norm = (data1-data1.min())/(data1.max()-data1.min())
data1_norm.head()
```

Out[8]:

	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 [9]:

```python
data2_norm = (data2-data2.min())/(data2.max()-data2.min())
data2_norm.head()
```

Out[9]:

	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 [10]:

```python
data1_norm['label']=data1_label['label']
data1_norm.head()

```

Out[10]:

	Vacation	eCredit	salary	property	label
0	0.079365	0.107558	0.219960	0.183167	C1
1	0.158730	0.052326	0.293102	0.112797	C1
2	0.095238	0.177326	0.346023	0.174200	C1
3	0.031746	0.127907	0.309882	0.189984	C1
4	0.222222	0.020349	0.363663	0.127311	C1

In [11]:

```
data2_norm['label']=data2_label['label']
data2_norm.head()

```

Out[11]:

	Vacation	eCredit	salary	property	label
0	0.20	0.058824	0.104637	0.398926	C1
1	0.54	0.021008	0.175059	0.243041	C1
2	0.52	0.231092	0.138339	0.085992	C1
3	0.26	0.151261	0.430086	0.116229	C1
4	0.00	0.016807	0.352657	0.025714	C1

In [12]:

```
xtrain=data1_norm.copy()
xtest=data2_norm.copy()
```

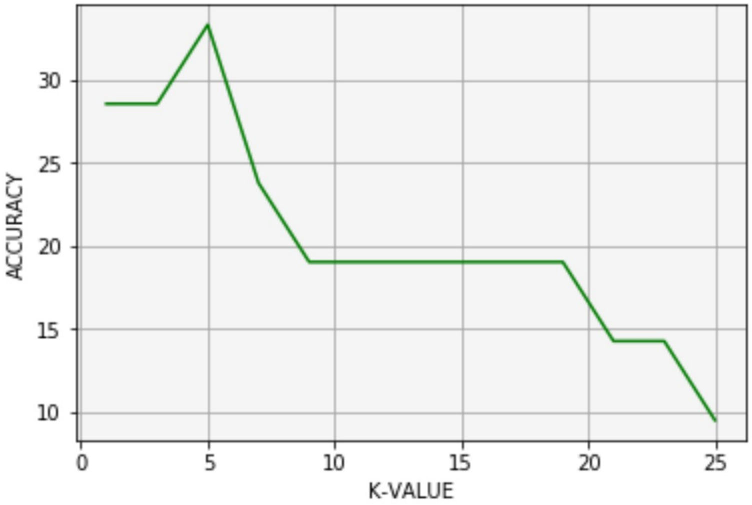
In [13]:

```
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(2):
    #         if(testrow[i]==trainrow[i]):
    #             distance+=1
#         for i in range(2):
#             if(testrow[i]==trainrow[i]):
#                 distance+=1
        for i in range(length):
            distance+=pow((testrow[i]-trainrow[i]),2)
        return math.sqrt(distance)
    def getNeighbours(traindata,testRow,k):
        distance_with_train=[]
        length=len(testRow)-1
        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(
        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(x
    accuracy = getAccuracy(xtest.values, predict)
    acc_list.append(accuracy)
    print('Accuracy: ' + repr(accuracy) + '%','with k=',k)
```

```
Accuracy: 28.57142857142857% with k= 1
Accuracy: 28.57142857142857% with k= 3
Accuracy: 33.33333333333333% with k= 5
```

In [14]:

```
plt.plot(k_list,acc_list,color='green')
plt.xlabel('K-VALUE')
plt.ylabel('ACCURACY')
plt.grid(True)
plt.show()
```



In [16]:

```
bigdata = pd.concat([xtrain, xtest], ignore_index=True)
bigdata.head()
```

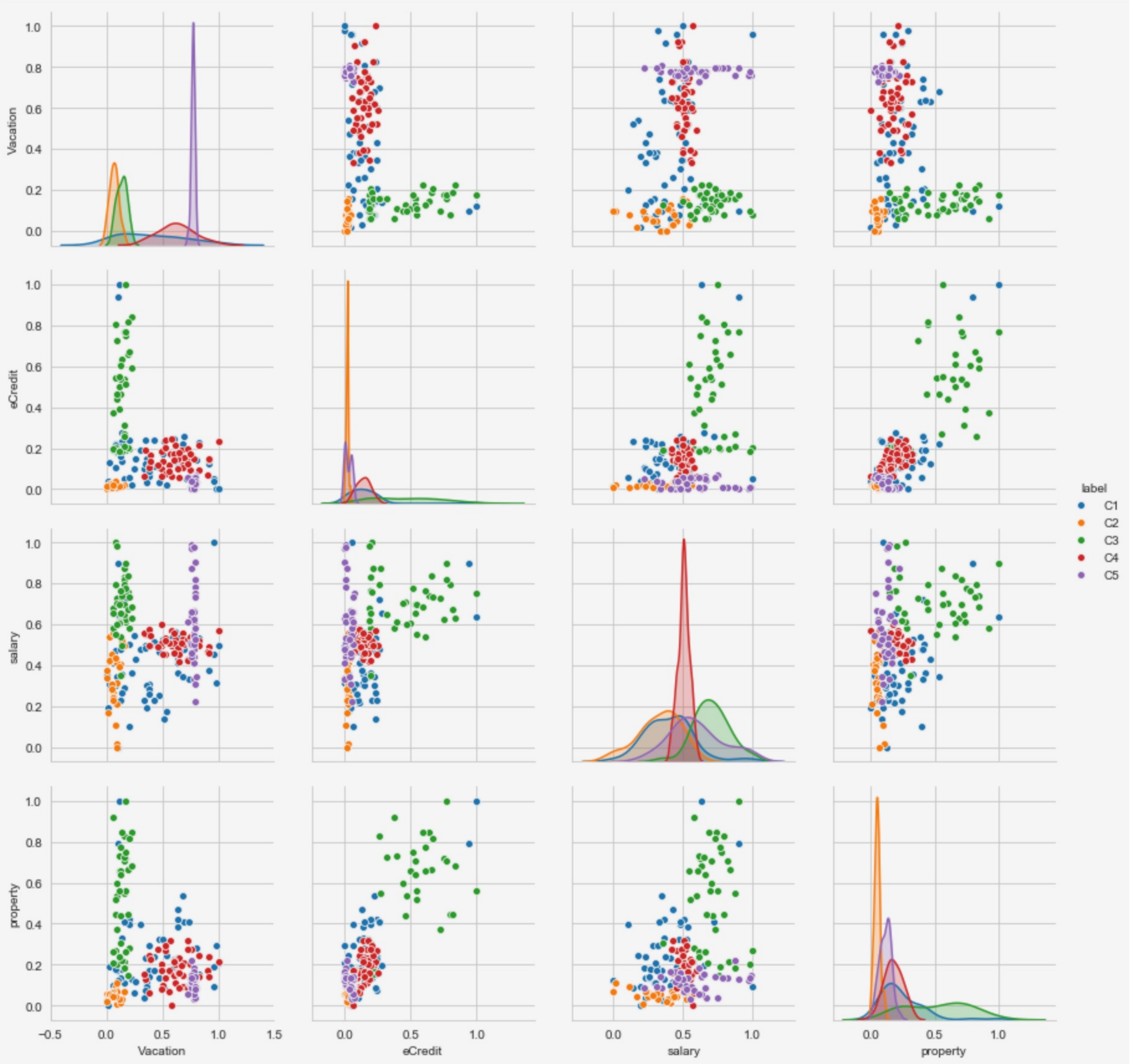
Out[16]:

	Vacation	eCredit	salary	property	label
0	0.079365	0.107558	0.219960	0.183167	C1
1	0.158730	0.052326	0.293102	0.112797	C1
2	0.095238	0.177326	0.346023	0.174200	C1
3	0.031746	0.127907	0.309882	0.189984	C1
4	0.222222	0.020349	0.363663	0.127311	C1

In [17]:

```python
import seaborn as sns
plt.close();
sns.set_style("whitegrid");
sns.pairplot(bigdata, hue="label", size=3);
plt.show()
```

```
C:\Users\prem\Anaconda3\lib\site-packages\seaborn\axisgrid.py:2065: Us
erWarning: The `size` parameter has been renamed to `height`; pleaes u
pdate your code.
  warnings.warn(msg, UserWarning)
C:\Users\prem\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: F
utureWarning: Using a non-tuple sequence for multidimensional indexing
is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the fut
ure this will be interpreted as an array index, `arr[np.array(seq)]`,
which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



In [ ]:

```

```

In [ ]:

```

```