```
from sklearn.feature_extraction import DictVectorizer
import csv
from sklearn import tree
from sklearn import preprocessing
from sklearn.externals.six import StringIO

# Read in the csv file and put features into list of dict and list of class label
allElectronicsData = open(r'E:\DTree\data.csv', 'rt')
reader = csv.reader(allElectronicsData)
#headers = reader.next()
headers=next(reader)

print(headers)
```

```
['RID', 'age', 'income', 'student', 'credit_rating', 'class_buys_computer']
```

```
featureList = []
labelList = []

for row in reader:
    labelList.append(row[len(row)-1])
    rowDict = {}
    for i in range(1, len(row)-1):
        rowDict[headers[i]] = row[i]
    featureList.append(rowDict)
print(featureList)
```

```
[{'credit_rating': 'fair', 'student': 'no', 'age': 'youth', 'income': 'high'}, {'credit_rating': 'excellent', 'student': 'no', 'age': 'youth', 'income': 'high'}, {'credit_rating': 'fair', 'student': 'no', 'age': 'middle_aged', 'income': 'high'}, {'credit_rating': 'fair', 'student': 'no', 'age': 'senior', 'income': 'medium'}, {'credit_rating': 'fair', 'student': 'yes', 'age': 'senior', 'income': 'low'}, {'credit_rating': 'excellent', 'student': 'yes', 'age': 'senior', 'income': 'low'}, {'credit_rating': 'excellent', 'student': 'yes', 'age': 'middle_aged', 'income': 'low'}, {'credit_rating': 'fair', 'student': 'no', 'age': 'youth', 'income': 'medium'}, {'credit_rating': 'fair', 'student': 'yes', 'age': 'youth', 'income': 'low'}, {'credit_rating': 'fair', 'student': 'yes', 'age': 'senior', 'income': 'medium'}, {'credit_rating': 'excellent', 'student': 'yes', 'age': 'youth', 'income': 'medium'}, {'credit_rating': 'excellent', 'student': 'no', 'age': 'middle_aged', 'income': 'medium'}, {'credit_rating': 'fair', 'student': 'yes', 'age': 'middle_aged', 'income': 'high'}, {'credit_rating': 'excellent', 'student': 'no', 'age': 'senior', 'income': 'medium'}]
```

```
# Vetorize features
vec = DictVectorizer()
dummyX = vec.fit_transform(featureList) .toarray()



print("dummyX: " + str(dummyX))
print(vec.get_feature_names())
print("labelList: " + str(labelList))
```

```
dummyX: [[ 0.  0.  1.  0.  1.  1.  0.  0.  1.  0.]
 [ 0.  0.  1.  1.  0.  1.  0.  0.  1.  0.]
 [ 1.  0.  0.  0.  1.  1.  0.  0.  1.  0.]
 [ 0.  1.  0.  0.  1.  0.  0.  1.  1.  0.]
 [ 0.  1.  0.  0.  1.  0.  1.  0.  0.  1.]
 [ 0.  1.  0.  1.  0.  0.  1.  0.  0.  1.]
 [ 1.  0.  0.  1.  0.  0.  1.  0.  0.  1.]
 [ 0.  0.  1.  0.  1.  0.  0.  1.  1.  0.]
 [ 0.  0.  1.  0.  1.  0.  1.  0.  0.  1.]
 [ 0.  1.  0.  0.  1.  0.  0.  1.  0.  1.]
 [ 0.  0.  1.  1.  0.  0.  0.  1.  0.  1.]
 [ 1.  0.  0.  1.  0.  0.  0.  1.  1.  0.]
 [ 1.  0.  0.  0.  1.  1.  0.  0.  0.  1.]
 [ 0.  1.  0.  1.  0.  0.  0.  1.  1.  0.]]
['age=middle_aged', 'age=senior', 'age=youth', 'credit_rating=excellent',
'credit_rating=fair', 'income=high', 'income=low', 'income=medium', 'student=no',
'student=yes']
labelList: ['no', 'no', 'yes', 'yes', 'yes', 'no', 'yes', 'no', 'yes', 'yes', 'yes', 'yes', 'yes', 'no']
```

```
# vectorize class labels
lb = preprocessing.LabelBinarizer()
dummyY = lb.fit_transform(labelList)
#print("dummyY: " + str(dummyY))

# Using decision tree for classification
# clf = tree.DecisionTreeClassifier()
clf = tree.DecisionTreeClassifier(criterion='entropy')
clf = clf.fit(dummyX, dummyY)
print("clf: " + str(clf))
```

```
clf: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
```

```
# Visualize model
with open("allElectronicInformationGainOri.dot", 'w') as f:
    f = tree.export_graphviz(clf, feature_names=vec.get_feature_names(), out_file=f)

oneRowX = dummyX[0, :]
print("oneRowX: " + str(oneRowX))

newRowX = oneRowX
newRowX[0] = 1
newRowX[2] = 0
print("newRowX: " + str(newRowX))
```

```
oneRowX: [ 0.  0.  1.  0.  1.  1.  0.  0.  1.  0.]
newRowX: [ 1.  0.  0.  0.  1.  1.  0.  0.  1.  0.]
```

```
predictedY = clf.predict(newRowX)
print("predictedY: " + str(predictedY))
```

```
predictedY: [1]
```

安装 Graphviz： http://www.graphviz.org/

配置环境变量

转化 dot 文件至 pdf 可视化决策树：

dot -Tpdf allElectronicInformationGainOri.dot -o DTree.pdf