In [1]:
```
import pandas as pd
import numpy as np
import pandas_profiling
import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifie
r
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score

from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
```

```
C:\Users\User\Anaconda3\envs\dataScience1\lib\site-packages\sklearn\externals
\six.py:28: FutureWarning: The module is deprecated in version 0.21 and will
be removed in version 0.23 since we've dropped support for Python 2.7. Please
rely on the official version of six (https://pypi.org/project/six/).
  warnings.warn("The module is deprecated in version 0.21 and will be removed
"
```

In [2]:
```
#data is the variable name for the Pandas default data structure called
# a "Dataframe" in this case its a 2D Array
data = pd.read_csv('Demographic_Data.csv')
```

In [3]:
```
#data.head()
clean_data = data.drop_duplicates()
```

In [4]:
```
clean_data.to_csv(r'D_Data_Clean.csv', index=False)
```

In [5]:
```
data = pd.read_csv('D_Data_Clean.csv')
```

In [6]:
```
data.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
from sklearn.datasets import load_digits
digits = load_digits()
print(digits.data)
digits.target
```

```
[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0. 10. ... 12.  1.  0.]]
```

Out[7]:
```
array([0, 1, 2, ..., 8, 9, 8])
```

In [8]:
```
data.info()
data.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 79979 entries, 0 to 79978
Data columns (total 5 columns):
in-store    79979 non-null int64
age         79979 non-null int64
items       79979 non-null int64
amount      79979 non-null float64
region      79979 non-null int64
dtypes: float64(1), int64(4)
memory usage: 3.1 MB
```

Out[8]:

	in-store	age	items	amount	region
count	79979.000000	79979.000000	79979.000000	79979.000000	79979.000000
mean	0.500006	45.758512	4.505133	835.825727	2.674915
std	0.500003	15.715158	2.061250	721.263650	1.126642
min	0.000000	18.000000	1.000000	5.004700	1.000000
25%	0.000000	33.000000	3.000000	285.120000	2.000000
50%	1.000000	45.000000	4.000000	582.140000	3.000000
75%	1.000000	56.000000	6.000000	1233.400000	4.000000
max	1.000000	85.000000	8.000000	3000.000000	4.000000

In [9]:
```
X = data.iloc[: , 0:4]
print('Summary of feature sample')
X.head()
```

```
Summary of feature sample
```

Out[9]:

	in-store	age	items	amount
0	0	37	4	281.03
1	0	35	2	219.51
2	1	45	3	1525.70
3	1	46	3	715.25
4	1	33	4	1937.50

In [10]:
```
y = data['region']
# iloc[sr:ed, sc:ed]
```

In [33]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .15, ran
dom_state = 123)
```

In [34]:
```
X_train.head()
```

Out[34]:

	in-store	age	items	amount
67626	0	41	3	179.990
33504	1	54	2	1662.300
6736	0	61	7	52.091
55621	0	68	4	57.397
20558	0	59	8	49.716

In [35]:
```
y_train.head()
```

Out[35]:
```
67626    2
33504    1
6736     2
55621    2
20558    2
Name: region, dtype: int64
```

In [37]:
```
from sklearn.model_selection import cross_val_score
```

In [38]:
```
algos_Class = []
#algos_Class.append(('Random Forest Classifier', RandomForestClassifier()))
algos_Class.append(('Decision Tree Classifier', DecisionTreeClassifier()))
```

In [39]:
```
#instanciate
#algo = RandomForestClassifier()
```

In [17]:
```
#information for the group to build the model
#model = algo.fit(X_train, y_train)
```

In [18]:
```
#compare preds to ground truth (y_test - known values)
```

In [40]:
```
#classification
results = []
names = []
for name, model in algos_Class:
     result = cross_val_score(model, X,y, cv=3, scoring='accuracy')
     names.append(name)
     results.append(result)
```

In [41]:
```
for i in range(len(names)):
    print(names[i],results[i].mean())
```

```
Decision Tree Classifier 0.5625101237611064
```

In [42]:
```
#Modeling (Classification)
algo = DecisionTreeClassifier(min_samples_split=10000)
model = algo.fit(X_train,y_train)
```

In [43]:
```
print(cross_val_score(model, X, y, cv=3))
```

```
[0.63612153 0.63390848 0.63712067]
```

In [44]:
```
#Predictions
preds = model.predict(X_test)
```

In [45]:
```
accuracy_score(y_test, preds)
```

Out[45]:
```
0.638659664916229
```

In [46]:
```
print(classification_report(y_test, preds))
```

```
              precision    recall  f1-score   support

           1       0.40      0.54      0.46      2455
           2       0.90      1.00      0.95      3030
           3       0.52      0.21      0.30      2679
           4       0.64      0.71      0.68      3833

    accuracy                           0.64     11997
   macro avg       0.62      0.62      0.60     11997
weighted avg       0.63      0.64      0.62     11997
```

In [47]:
```
region_values = ['1','2','3','4'] #this is just a list specifying the region c
lasses
```

In [48]:
```
dot_data = StringIO()
export_graphviz(model, out_file=dot_data,
    filled=True, rounded=True,
    feature_names=X.columns,
    class_names=region_values,label='all',precision=1,
    special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('tree.png')
Image(graph.create_png())
```

Out[48]: