Lab 4- Decision Trees

This assignment uses 2012 data obtained from the Federal Election Commission on contributions to candidates from committees. The data dictionary is available at http://www.fec.gov/finance/disclosure/metadata/DataDictionaryContributionstoCandidates.shtml (http://www.fec.gov/finance/disclosure/metadata/DataDictionaryContributionstoCandidates.shtml). The file we've given you has been subset to 10,000 randomly sampled rows, wth some columns removed

In [48]:
```
from __future__ import division, print_function
from collections import Counter, defaultdict
from itertools import combinations
import pandas as pd
import numpy as np
import itertools
```

In [49]:
```
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_extraction import DictVectorizer #to turn ca
tegorial variables into numeric arrays
from sklearn import preprocessing #to transform the feature label
s
from sklearn.feature_extraction import DictVectorizer
```

In [50]:
```
df = pd.read_csv('lab4_candidate_contributions.csv')

#convert zip code and transaction date from floats to strings (si
nce we wnat to treat them as categorical)
df.ZIP_CODE = df.ZIP_CODE.astype('int').astype('str')
df.TRANSACTION_DT = df.TRANSACTION_DT.astype('int').astype('str')

df.head()
```

Out[50]:

	CMTE_ID	AMNDT_IND	RPT_TP	ENTITY_TP	NAME	CITY	STA
0	C90011156	N	Q3	IND	ROULAND, PRESTON	CLEVELAND	OH
1	C90011156	N	Q3	IND	DANFORD, KAYLA	OXFORD	OH
2	C90011156	N	YE	IND	CLARK, BARBARA	WHITEHALL	OH
3	C90011156	N	YE	ORG	JENNIFER JANNON	DORMONT	PA
4	C90011156	N	YE	IND	ARNDI, PHILIP	BOSTON	MA

In [ ]:

Calculating Gini Index

Question 1: How many rows are there in the dataset for Obama? For Romney?

In [51]:
```
obama = 0
romney = 0
for i in range(df.CAND_ID.size):
    if df.CAND_ID.get_value(i) == "Obama":
        obama += 1
    else:
        romney += 1
print("Obama: %d, Romney: %d"%(obama, romney))
```

```
Obama: 5761, Romney: 4239
```

Question 2: What is the Gini Index of the this dataset, using Romney and Obama as the target classes?

In [52]:
```
def gini(D):
    obama = sum(D.CAND_ID == "Obama")
    romney = sum(D.CAND_ID == "Romney")
    total = obama+romney
    return 1-((obama/total)**2+(romney/total)**2)

print("gini index: %f"%gini(df))
```

```
gini index: 0.488418
```

Best Split of a Numeric Feature

In [53]:
```
sortd = df.sort(columns="TRANSACTION_AMT")
mingini = 1
mini = 0
ob1 = 0
ob2 = sum(df.CAND_ID == "Obama")
ro1 = 0
ro2 = sum(df.CAND_ID == "Romney")
total = ob2+ro2
for i in range(df.CAND_ID.size-1):
    if sortd.CAND_ID.get_value(i) == "Obama":
        ob1 += 1
        ob2 -= 1
    else:
        ro1 += 1
        ro2 -= 1
    low = df.TRANSACTION_AMT.get_value(i)
    high = df.TRANSACTION_AMT.get_value(i+1)
    if low != high:
        tot1 = ob1+ro1
        tot2 = ob2+ro2
        gini1 = 1-((ob1/tot1)**2+(ro1/tot1)**2)
        gini2 = 1-((ob2/tot2)**2+(ro2/tot2)**2)
        ginit = gini1*((ob1+ro1)/total) + gini2*((ob2+ro2)/total)
        if ginit < mingini:
            mini = i
            mingini = ginit
            minob1 = ob1
            minob2 = ob2
            minro1 = ro1
            minro2 = ro2

print("split after: %d, gini score: %f, gini reduced by: %f, Obam
a below: %d, Obama above: %d, Romney below: %d, Romney above: %d"
%(mini, mingini, (gini(df)-mingini), minob1, minob2, minro1, minr
o2))
```

```
split after: 8040, gini score: 0.488082, gini reduced by: 0.00033
6, Obama below: 4581, Obama above: 1180, Romney below: 3460, Romn
ey above: 779
```

Question 3: What is the best split point of the TRANSACTION_AMT feature.

In [54]:
```
mini
```

Out[54]:
```
8040
```

Question 4: What is the Gini Index of this best split?

```
In [55]: mingini
Out[55]: 0.48808188000763558
```

Question 5: How much does this partitioning reduce the Gini Index over that of the overall dataset?

```
In [56]: (gini(df)-mingini)
Out[56]: 0.00033569999236454651
```

Question 6: How many Romney rows are below your best split point? Obama rows?

```
In [57]: print("Romney Rows: ", minro1)
         print("Obama Rows:", minob1)

         Romney Rows:  3460
         Obama Rows: 4581
```

Question 7: How many Romney rows are above your best split point? Obama rows?

Recall that, to calculate the best split of this numeric field, you'll need to order your data by TRANSACTION AMT, then consider the midpoint between each pair of consecutive transaction amounts as a potential split point, then calculate the Gini Index for that partitioning. You'll want to keep track of the best split point and its Gini Index (remember that you are trying to minimize the Gini Index).

There are a lot of ways to do this. Some are very fast, others very slow. One tip to make this run quickly is, as you consecutively step through the data and calculate the Gini Index of each possible split point, keep a running total of the number of rows for each candidate that are located above and below the split point.

Some Python tips:

- Counter(), from the collections module, is a special dictionary for counting values of a key
- zip() lets you concatenate lists into a list of tuples (for example, if we have a list of the candidates and a list of transaction amounts, zip(candidate_list, transaction_amount) would give us a list of (candidate, transaction amount) pairs

In [58]:
```
sortd = df.sort(columns="TRANSACTION_AMT")
mingini = 1
mini = 0
ob1 = 0
ob2 = sum(df.CAND_ID == "Obama")
ro1 = 0

ro2 = sum(df.CAND_ID == "Romney")
total = ob2+ro2
for i in range(df.CAND_ID.size-1):
    if sortd.CAND_ID.get_value(i) == "Obama":
        ob1 += 1
        ob2 -= 1
    else:
        ro1 += 1
        ro2 -= 1
    low = df.TRANSACTION_AMT.get_value(i)
    high = df.TRANSACTION_AMT.get_value(i+1)
    if low != high:
        tot1 = ob1+ro1
        tot2 = ob2+ro2
        gini1 = 1-((ob1/tot1)**2+(ro1/tot1)**2)
        gini2 = 1-((ob2/tot2)**2+(ro2/tot2)**2)
        ginit = gini1*((ob1+ro1)/total) + gini2*((ob2+ro2)/total)
        if ginit < mingini:
            mini = i
            mingini = ginit
            minob1 = ob1
            minob2 = ob2
            minro1 = ro1
            minro2 = ro2

print("split after: %d, gini score: %f, gini reduced by: %f, Obam
a below: %d, Obama above: %d, Romney below: %d, Romney above: %d"
%(mini, mingini, (gini(df)-mingini), minob1, minob2, minro1, minr
o2))
```

```
split after: 8040, gini score: 0.488082, gini reduced by: 0.00033
6, Obama below: 4581, Obama above: 1180, Romney below: 3460, Romn
ey above: 779
```

Best Split of a Categorial Variable

In [59]:
```
import functools
# question 8
entity_vals = pd.unique(df["ENTITY_TP"])
combinations = functools.reduce(lambda x,y: x+y, [list(itertools.
combinations(entity_vals, r)) for r in range(1, len(entity_vals)/
/2 + 1)])

# question 9
mingini = 1
mincomb = None
for comb in combinations:
    indices = df["ENTITY_TP"].isin(comb)
    split1 = df.loc[indices,:]
    split2 = df.loc[~indices,:]
    cur_gini = (len(split1) * gini(split1) + len(split2) * gini(s
plit2)) / len(df)
    if cur_gini < mingini:
        mingini = cur_gini
        mincomb = comb
        min_split1 = split1
        min_split2 = split2

    #print "%d, %d"%(len(split1), len(split2))
```

Question 8: How many possible splits are there of the ENTITY_TP feature?

In [60]:
```
len(combinations)
# (2**7 - 2) / 2 (because we optimize by throwing out half)
```

Out[60]:
```
63
```

Question 9: Which split of ENTITY_TP best splits the Obama and Romney rows, as measured by the Gini Index?

In [61]:
```
# question 9
mincomb
```

Out[61]:
```
('CCM', 'COM', 'CAN')
```

Question 10: What is the Gini Index of this best split?

In [62]:
```
# question 10
mingini
```

Out[62]:
```
0.48314520835547992
```

Question 11: How much does this partitioning reduce the Gini Index over that of the overall data set?

```
In [63]: # question 11
         gini(df) - mingini

Out[63]: 0.0052723716445202129
```

Question 12: How many Romney rows and Obama rows are in your first partition? How many Romney rows and Obama rows are in your second partition?

```
In [64]: # question 12
         print("Romney: %s, Obama: %s"%(sum(min_split1.CAND_ID == "Romney"
         ), sum(min_split1.CAND_ID == "Obama")))
         print("Romney: %s, Obama: %s"%(sum(min_split2.CAND_ID == "Romney"
         ), sum(min_split2.CAND_ID == "Obama")))

         Romney: 147, Obama: 37
         Romney: 4092, Obama: 5724
```

In this exercise, you will be partitioning the original dataset (as opposed to further partitioning the transaction amount partitions from the previous set of questions).

Python tip: the combinations function of the itertools module allows you to enumerate combinations of a list

Training a decision tree

Question 13: Using all of the features in the original dataframe read in at the top of this notebook, train a decision tree classifier that has a depth of three (including the root node and leaf nodes). What is the accuracy of this classifier on the training data?

In [65]:
```
from random import sample
def trainingSample(n, size):
    rows = sample(range(n), size)
    return rows

def separateRows(training_rows, data):
    """ Return (training set, prediction set) with n% of rows in
training set"""
    training = data.ix[training_rows]
    prediction = data.drop(training_rows)

    return (training, prediction)
```

In [66]:
```
from datetime import datetime
from sklearn import preprocessing
from sklearn.feature_extraction import DictVectorizer

classifier = DecisionTreeClassifier(criterion='gini', splitter='b
est', max_depth=3, min_samples_split=2, min_samples_leaf=1, max_f
eatures=None, random_state=None, min_density=None, compute_import
ances=None, max_leaf_nodes=None)
df_new = df.copy()
df_new.TRANSACTION_DT = df_new.TRANSACTION_DT.apply(lambda x: x i
f len(x) == 8 else "0" + x)
df_new.TRANSACTION_DT = df_new.TRANSACTION_DT.apply(lambda x: dat
etime.strptime(x, "%m%d%Y").toordinal())

CAND_ID = df_new.CAND_ID
X = df_new.drop("CAND_ID", axis = 1)

vec = DictVectorizer()
X = pd.DataFrame(vec.fit_transform(X.to_dict("records")).toarray(
))
X.columns = vec.get_feature_names()

train_size = 0.75
training_rows = trainingSample(X.shape[0], int(train_size * X.sha
pe[0]))
train_rows, pred_rows = separateRows(training_rows, X)
train_Y, pred_Y = separateRows(training_rows, CAND_ID)


train_Y = train_Y == 'Obama'
train_Y = train_Y.astype(int)
pred_Y = pred_Y == 'Obama'
pred_Y = pred_Y.astype(int)
```

In [67]:
```
clf = classifier.fit(train_rows, train_Y)
```

```
In [68]: classifier.score(train_rows, train_Y)
```

```
Out[68]: 0.57906666666666662
```

```
In [69]: classifier.score(pred_rows, pred_Y)
```

```
Out[69]: 0.56720000000000004
```

Question 14: Export your decision tree to graphviz. Please submit a png file of this graphic to bcourses. In your write-up, write down the interpretation of the rule at each node (for example, 'Root node: rows from state AL go the the left, rows from all other states go to the right. Left child of root node: ... etc

```
In [70]: from sklearn.externals.six import StringIO
         with open("obama_romney.dot", 'w') as f:
             f = sklearn.tree.export_graphviz(clf, out_file=f)
```

```
In [71]: print(train_rows.columns[2768])
         print(train_rows.columns[7])
         print(train_rows.columns[745])
         print(train_rows.columns[706])
         print(train_rows.columns[810])
         print(train_rows.columns[2745])
```

```
TRANSACTION_DT
CITY=AKRON
CMTE_ID=C90011156
CMTE_ID=C00521013
ENTITY_TP=IND
STATE=NC
```

The topmost split depends on the date of the contribution. The next highest are whether a contribution was from Akron, and whether CMTE_ID equals C90011156 (which we think has to do with who the contributor is). The final row splits on whether CMTE_ID equals C00521013, whether the donor is an individual, and whether the state is North Carolina.

Question 15: For each of your leaf nodes, specify the percentage of Obama rows in that node (out of the total number of rows at that node).

In [72]:
```
print(3419 / (3079 + 3419))
print(135 / (0 + 135))
print(75 / (56 + 75))
print(68/ (1 + 68))

print(364 / (17 + 364))
print(9 / (7 + 9))
```

```
0.526161895968
1.0
0.572519083969
0.985507246377
0.955380577428
0.5625
```

See this notebook for the basics of training a decision tree in scikit-learn and exporting the outputs to view in graphviz: http://nbviewer.ipython.org/gist/tebarkley/b68c04d9b31e64ce6023 (http://nbviewer.ipython.org/gist/tebarkley/b68c04d9b31e64ce6023)

Scikit-learn classifiers require class labels and features to be in numeric arrays. As such, you will need to turn your categorical features into numeric arrays using DictVectorizer. This is a helpful notebook for understanding how to do this: http://nbviewer.ipython.org/gist/sarguido/7423289 (http://nbviewer.ipython.org/gist/sarguido/7423289). You can turn a pandas dataframe of features into a dictionary of the form needed by DictVectorizer by using df.to_dict('records'). Make sure you remove the class label first (in this case, CAND_ID). If you use the class label as a feature, your classifier will have a training accuracy of 100%! The example notebook link also shows how to turn your class labels into a numeric array using sklearn.preprocessing.LabelEncoder().

We already did this for you at the top of the notebook, but before you convert your features into numeric arrays, you should always make sure they are of the correct type (ie zip code should be a string, not a float, because it is a categorical variable).

Question 14 asks you to interpret the rules at each decision tree node using the graphviz output. The graphviz output looks cryptic (ie it might tell you that X[1014] < 0.5 is the best split for a particular node. To figure out what feature that corresponds to, use the .get_feature_names() function of your DictVectorizer object. If that returns something like 'CITY=PHOENIX', then you know that the left child of the node contains rows not in Phoenix ('CITY=PHOENIX' ==0) and the right child of the node contains rows in Phoenix ('CITY=PHOENIX' == 1).

In [162]:
```
```