In [1]:
```
class Node:
    def __init__(self,pure = False, subset = None,split_rule = None,
                 yes = None,no = None, label = 0 ):
        self.pure = pure
        self.subset = subset
        self.split_rule = split_rule
        self.yes = yes
        self.no = no
        self.label = None
```

In [2]:
```
f1 = open("pa2features.txt","r")
feature_name = [line.strip() for line in f1]

f = open("pa2train.txt","r")
train = [line.strip() for line in f]
train = [[float(i) for i in line.split()] for line in train]
```

Q1

In [3]:
```
import numpy as np
```

In [4]:
```
def entropy(features,index,threshold,amount):
    z_0 = sum([z >= threshold for z,x in features[index]])
    z_1 = amount - z_0

    pz0x0 = sum([z >= threshold and x == 0 for z,x in features[index]])/
z_0
    pz0x1 = 1-pz0x0
    if (pz0x0 == 0 or pz0x1 == 0):
        Hz_0 = 0
    else:
        Hz_0 = -pz0x0*np.log(pz0x0)-pz0x1*np.log(pz0x1)

    pz1x0 = sum([z <= threshold and x == 0 for z,x in features[index]])/
z_1
    pz1x1 = 1-pz1x0
    if (pz1x0 == 0 or pz1x1 == 0):
        Hz_1 = 0
    else:
        Hz_1 = -pz1x0*np.log(pz1x0)-pz1x1*np.log(pz1x1)

    return z_0/amount*Hz_0 + z_1/amount*Hz_1
```

In [5]:
```
def min_entropy(subset):
    features = [[(line[i],line[-1]) for line in subset] for i in range(2
2)]
    features = [sorted(line) for line in features]

    thresholds = []
    for i in range(22):
        a = []
        for j in range(1,len(subset)):
            if features[i][j-1][0] < features[i][j][0]:
                a += [(features[i][j-1][0]+features[i][j][0])/2]
        thresholds += [a]

    entropy_l = []
    for i in range(22):
        for j in thresholds[i]:
            entropy_l += [(entropy(features,i,j,len(subset)),i,j)]

    return min(entropy_l)[1:]
```

In [6]:
```
#Build Decision Tree
queue = []
root = Node(subset = train)
queue.append(root)

while(len(queue) != 0 ):
    node = queue.pop(0)

    node.split_rule = min_entropy(node.subset)
    index,threshold = node.split_rule

    subset1 = [line for line in node.subset if line[index] < threshold]
    node1 = Node(subset = subset1)

    if sum([line[-1] == 0 for line in subset1]) == 0:
        node1.label = 1
        node1.pure = True
    elif sum([line[-1] == 1 for line in subset1]) == 0:
        node1.label = 0
        node1.pure = True
    else:
        queue.append(node1)

    subset2 = [line for line in node.subset if line[index] >= threshold]
    node2 = Node(subset = subset2)
    if sum([line[-1] == 0 for line in subset2]) == 0:
        node2.label = 1
        node2.pure = True
    elif sum([line[-1] == 1 for line in subset2]) == 0:
        node2.label = 0
        node2.pure = True
    else:
        queue.append(node2)

    node.yes = node1
    node.no = node2
```

In [7]:
```
print("Level1:")
print(root.split_rule, len(root.subset))
```

```
Level1:
(4, 0.5) 2000
```

In [8]:
```
print("Level2:")
print(root.yes.split_rule, len(root.yes.subset))
print(root.no.split_rule, len(root.no.subset))
```

```
Level2:
(0, 415000.0) 1319
(4, 1.5) 681
```

In [9]:
```
print("Level3:")
print(root.yes.yes.split_rule, len(root.yes.yes.subset))
print(root.yes.no.split_rule, len(root.yes.no.subset))
print(root.no.yes.split_rule, len(root.no.yes.subset))
print(root.no.no.split_rule, len(root.no.no.subset))
```

```
Level3:
(16, 2506.5) 1284
(20, 208.0) 35
(19, 584.5) 292
(20, 2006.0) 389
```

First three levels decision tree are at the last page:

Q2

In [10]:
```
f_test = open('pa2test.txt', 'r')
test = [line.strip() for line in f_test]
test = [[float(i) for i in line.split()] for line in test]
```

In [11]:
```
def search(root, item):
    curr = root
    while (not curr.pure):
        index, threshold = curr.split_rule
        if (item[index] < threshold):
            curr = curr.yes
        else:
            curr = curr.no
    return curr.label
```

In [12]:
```
train_result = [search(root,l) for l in train]
train_error = sum([train_result[i] != train[i][-1] for i in range(len(tr
ain))])/len(train)
print("train_error", train_error)
```

```
train_error 0.0
```

In [13]:
```
test_result = [search(root,l) for l in test]
test_error = sum([test_result[i] != test[i][-1] for i in range(len(test
))])/len(test)
print("test_error", test_error)
```

```
test_error 0.173
```

Q3

In [14]:
```
f_validate = open('pa2validation.txt', 'r')
validate = [line.strip() for line in f_validate]
validate = [[float(i) for i in line.split()] for line in validate]
```

In [15]:
```
queue = []
queue.append(root)
i = 0
while(not len(queue) == 0):
    node = queue.pop(0)
    validate_result = [search(root,l) for l in validate]
    validate_error = sum([validate_result[i] != validate[i][-1]
                          for i in range(len(validate))])/len(validate)

    if (node.pure):
        continue
    node.pure = True
    node.label = sum([l[-1] == 1 for l in node.subset]) > sum([l[-1] ==
0 for l in node.subset])
    new_validate = [search(root,l) for l in validate]
    new_error = sum([new_validate[i] != validate[i][-1]
                          for i in range(len(validate))])/len(validate)
    if new_error <= validate_error:
        i += 1
        print(i, "validation_error",new_error)
        test_result = [search(root,l) for l in test]
        test_error = sum([test_result[i] != test[i][-1] for i in range(l
en(test))])/len(test)
        print("test_error", test_error)
    else:
        node.pure = False
        node.label = None
        queue.append(node.yes)
        queue.append(node.no)
```

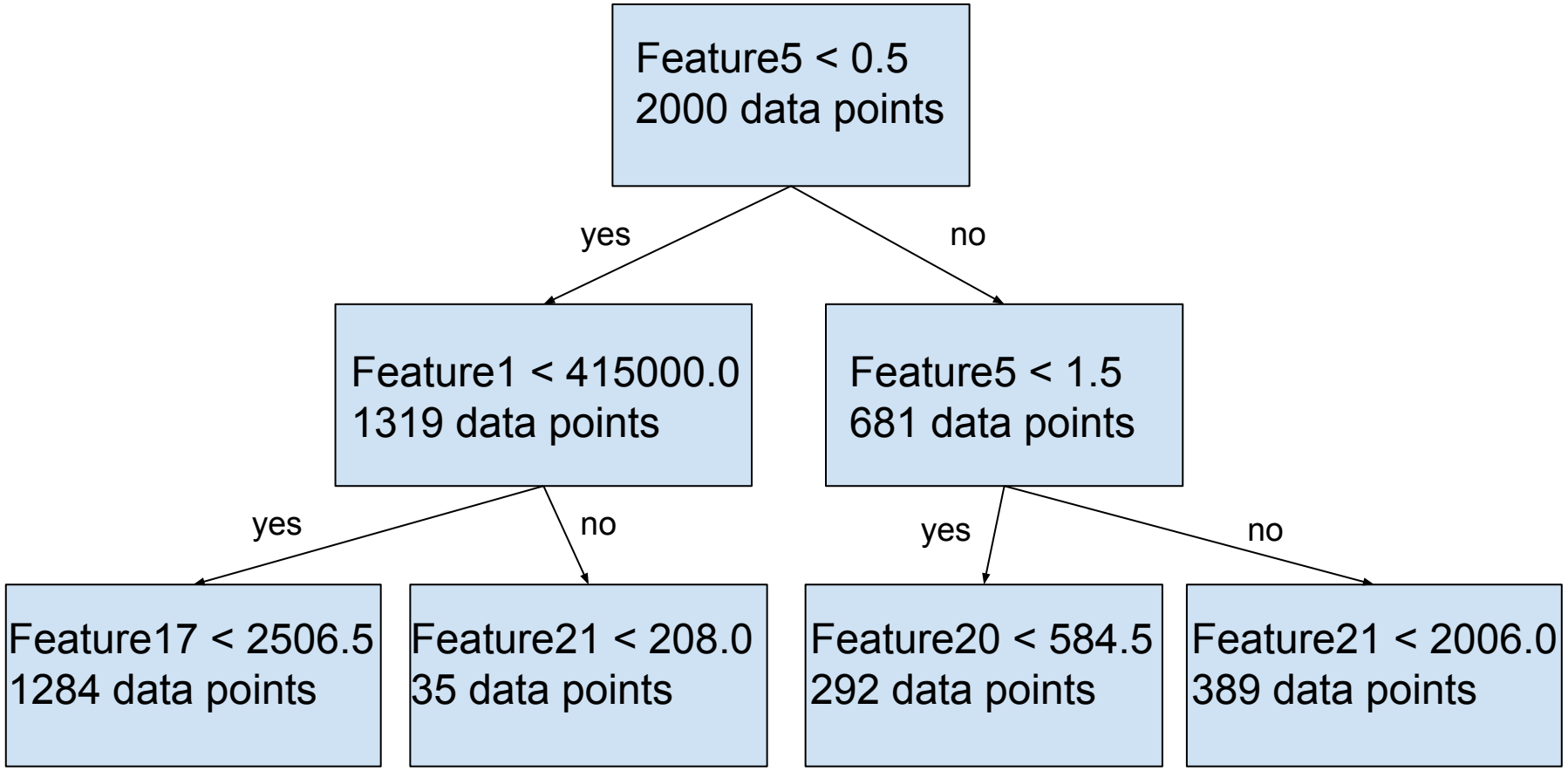
```
1 validation_error 0.122
test_error 0.117
2 validation_error 0.107
test_error 0.103
```

Q4

In [16]:
```
feature_name[4]
```

Out[16]:
```
'PAYMENT_DELAY_SEPTEMBER'
```

PAYMENT_DELAY_SEPTEMBER is the most salient feature that predicts credit card default.


Feature5 < 0.5
2000 data points
yes
no
Feature1 < 415000.0
1319 data points
Feature5 < 1.5
681 data points
yes
no
yes
no
Feature17 < 2506.5
1284 data points
Feature21 < 208.0
35 data points
Feature20 < 584.5
292 data points
Feature21 < 2006.0
389 data points