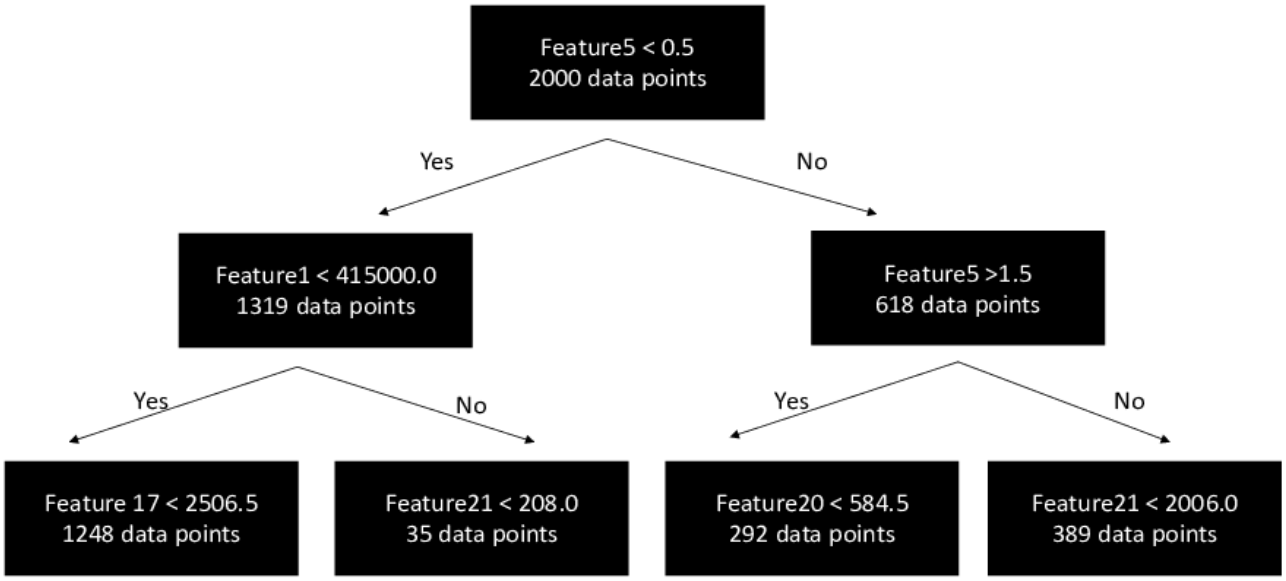
Result

Question 1



Question 2

Train Error: 0.0

Test Error: 0.173

Question 3

1 Round Validation Error: 0.122

1 Round Test Error: 0.117

2 Round Validation Error: 0.107

2 Round Test Error: 0.103

Question 4

PAYMENT_DELAY_SEPTEMBER

Code

Package and Reading File

In [1]:
```
import numpy as np
```

In [2]:
```
f = open("pa2train.txt","r")
train = [l.strip() for l in f]
train = [[float(each) for each in l.split()] for l in train]

f = open('pa2test.txt', 'r')
test = [l.strip() for l in f]
test = [[float(each) for each in l.split()] for l in test]

f = open('pa2validation.txt', 'r')
validate = [l.strip() for l in f]
validate = [[float(each) for each in l.split()] for l in validate]

f = open("pa2features.txt","r")
feature = [l.strip() for l in f]
```

Class and Functions

In [3]:
```
class ID3Node:
    def __init__(self, Set = None, rule = None, pure = False,
                 y = None,n = None, label = 0 ):
        self.Set = Set
        self.rule = rule
        self.pure = pure
        self.y = y
        self.n = n
        self.label = None
```

In [4]:
```
def cal_entropy(f, i, t, num):
    y0 = sum([p1 >= t for p1,p2 in f[i]])
    y1 = num - y0

    pr_y0x0 = sum([p1 >= t and p2 == 0 for p1,p2 in f[i]])/y0
    pr_y0x1 = 1 - pr_y0x0

    pr_y1x0 = sum([p1 <= t and p2 == 0 for p1,p2 in f[i]])/y1
    pr_y1x1 = 1 - pr_y1x0

    if (pr_y0x0 != 0 and pr_y0x1 != 0):
        h_y0 = -pr_y0x0*np.log(pr_y0x0)-pr_y0x1*np.log(pr_y0x1)
    else:
        h_y0 = 0

    if (pr_y1x0 != 0 and pr_y1x1 != 0):
        h_y1 = -pr_y1x0*np.log(pr_y1x0)-pr_y1x1*np.log(pr_y1x1)
    else:
        h_y1 = 0

    h = y0 / num * h_y0 + y1 / num * h_y1

    return h
```

In [5]:
```
def optimal(Set):
    features = [[(l[i],l[-1]) for l in Set] for i in range(22)]
    features = [sorted(l) for l in features]

    l = []
    for i in range(22):
        s = []
        for j in range(1,len(Set), 1):
            if features[i][j-1][0] < features[i][j][0]:
                s += [(features[i][j-1][0] + features[i][j][0])/2]
        l += [s]

    entropy = []
    for i in range(22):
        for j in l[i]:
            entropy += [(cal_entropy(features,i,j,len(Set)),i,j)]
    optimal = min(entropy)[1:]

    return optimal
```

In [6]:
```
def build_tree()->ID3Node:
    root = ID3Node(Set = train)
    tree = [root]

    while(len(tree) != 0 ):
        n = tree.pop(0)

        n.rule = optimal(n.Set)
        i,t = n.rule

        setr = [l for l in n.Set if l[i] >= t]
        nr = ID3Node(Set = setr)

        setl = [l for l in n.Set if l[i] < t]
        nl = ID3Node(Set = setl)

        n.y = nl
        n.n = nr

        if sum([l[-1] == 0 for l in setr]) == 0:
            nr.pure = True
            nr.label = 1
        elif sum([l[-1] == 1 for l in setr]) == 0:
            nr.pure = True
            nr.label = 0
        else:
            tree.append(nr)

        if sum([l[-1] == 0 for l in setl]) == 0:
            nl.pure = True
            nl.label = 1
        elif sum([l[-1] == 1 for l in setl]) == 0:
            nl.pure = True
            nl.label = 0
        else:
            tree.append(nl)
    return root
```

In [7]:
```
def traverse(node, l):
    n = node
    while(not n.pure):
        i, t = n.rule
        if(l[i] >= t):
            n = n.n
        else:
            n = n.y
    return n.label
```

In [8]:
```
def prune():
    i = 0
    root = build_tree()
    q = [root]

    while(not len(q) == 0):
        n = q.pop(0)
        validate_traverse = []
        for each in validate:
            validate_traverse.append(traverse(root, each))
        validate_diff = sum([validate_traverse[i] != validate[i][-1] for
i in range(len(validate))])
        validate_error = validate_diff / len(validate)

        if(n.pure): continue

        n.label = sum([l[-1] == 1 for l in n.Set]) > sum([l[-1] == 0 for
l in n.Set])
        n.pure = True

        validate2_traverse = []
        for each in validate:
            validate2_traverse.append(traverse(root, each))
        validate2_diff = sum([validate2_traverse[i] != validate[i][-1] f
or i in range(len(validate))])
        validate2_error = validate2_diff / len(validate)

        if validate2_error <= validate_error:
            i += 1
            print(i, " Round Validation Error: ", validate2_error)

            test2_traverse = []
            for each in test:
                test2_traverse.append(traverse(root, each))
            test2_diff = sum([test2_traverse[i] != test[i][-1] for i in
range(len(test))])
            test2_error = test2_diff / len(test)
            print(i, " Round Test Error: ", test2_error)
        else:
            n.label = None
            n.pure = False
            q.append(n.y)
            q.append(n.n)
```

Question1

In [9]:
```
root = build_tree()
```

In [10]:
```
print("ID3 Decision Tree Level 1:")
print(root.rule, len(root.Set))
```

```
ID3 Decision Tree Level 1:
(4, 0.5) 2000
```

In [11]:
```
print("ID3 Decision Tree Level 2:")
print(root.y.rule, len(root.y.Set))
print(root.n.rule, len(root.n.Set))
```

```
ID3 Decision Tree Level 2:
(0, 415000.0) 1319
(4, 1.5) 681
```

In [12]:
```
print("ID3 Decision Tree Level 3:")
print(root.y.y.rule, len(root.y.y.Set))
print(root.y.n.rule, len(root.y.n.Set))
print(root.n.y.rule, len(root.n.y.Set))
print(root.n.n.rule, len(root.n.n.Set))
```

```
ID3 Decision Tree Level 3:
(16, 2506.5) 1284
(20, 208.0) 35
(19, 584.5) 292
(20, 2006.0) 389
```

Question2

In [13]:
```
train_traverse = []
for each in train:
    train_traverse.append(traverse(root, each))
test_traverse = []
for each in test:
    test_traverse.append(traverse(root, each))
```

In [14]:
```
train_diff = sum([train_traverse[i] != train[i][-1] for i in range(len(t
rain))])
train_error = train_diff / len(train)
```

In [15]:
```
test_diff = sum([test_traverse[i] != test[i][-1] for i in range(len(test
))])
test_error = test_diff / len(test)
```

In [16]:
```
print("Train Error: ", train_error)
```

```
Train Error:  0.0
```

In [17]:
```
print("Test Error: ", test_error)
```

```
Test Error:  0.173
```

Question3

In [18]:
```
prune()
```

```
1  Round Validation Error:  0.122
1  Round Test Error:  0.117
2  Round Validation Error:  0.107
2  Round Test Error:  0.103
```

Question4

In [19]:
```
feature[root.rule[0]]
```

Out[19]: 'PAYMENT_DELAY_SEPTEMBER'