Mini Project #1: Predicting logic errors of digital circuits

Approach & Reasoning:

Provided data is presented as an extensionless file
File must first be read into python in order to parse it to the appropriate data structure
Once data is in the proper format we will analyze it's shape according to:

X*theta = Y, where X is our input data and Y is our output

X our input data will always be 32 bits long

Input data shape: 15000,32

Y = each of the 32 column

Output data label shape: 15000,1

Based on this information 32 individual models must be created in order to predict all 32 bits

Part 1: Preparing Data

Import Libraries

In [1]:

```
import numpy as np
from datetime import datetime
start_time = datetime.now()
```

In [2]:

```
# function converts any file seperated by newlines to a numpy array
# accounts for any whitespace in the file
def file2Array(fileName, datatype):
    fileData = [line.rstrip('\n') for line in open(fileName)]

    dataList = []
    for i in range(len(fileData)):
        temp = "".join(fileData[i].split())
        dataList.append(temp)
    #print(dataList[0])

    # map() can listify the list of strings individually
    dataList = list(map(list, dataList))
    #print(dataList[0])

    dataArray = np.asarray(dataList, dtype=datatype)
    #print(dataArray.shape)

    return dataArray
```

In [3]:

```
# prepare numpy arrays for machine learning model
trainingData  = file2Array('training_data' ,int)
trainingLabel = file2Array('training_label',int)
testingData   = file2Array('testing_data'  ,int)
testingLabel  = file2Array('testing_label' ,int)

# count the number of feature in the given label file
featureCount    = np.size(trainingLabel,1)
```

Part 2: Build Machine Learning Model using scikit-learn

4 Step Process [I,M,T,P]

1. **Import**
2. **Make**
3. **Train**
4. **Predict**

Example workflow:

Step 1: [I]mport Model

```
from sklearn import tree
```

Step 2: [M]ake Model

```
clf = tree.DecisionTreeClassifier()
```

Step 3: [T]rain Model

```
clf.fit(trainingData,trainingLabel)
```

Step 4: [P]redict with Model

```
predictions = clf.predict(testingData)
```

Gather Metrics

```
 aScore = accuracy_score(testingLabelCol, predictions)
pScore = precision_score(testingLabelCol, predictions)
rScore = recall_score(testingLabelCol, predictions)
```

In [4]:

```
# import Model
from sklearn import tree
# import desired metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

Decision Tree

In [5]:

```
aScoreAvg,pScoreAvg,rScoreAvg = 0,0,0
aScoreTotal,pScoreTotal,rScoreTotal = 0,0,0
# Make Model
clf = tree.DecisionTreeClassifier(max_depth=12)

print("Decision Tree Results: ")

# Train model for all 32 features

for i in range(featureCount):

    # extract column
    trainingLabelCol = (trainingLabel[:, [i]]).ravel()
    testingLabelCol = (testingLabel[:, [i]]).ravel()
    #print(trainingLabel) #debug
    #print(trainingData)  #debug

    if((len(np.unique(trainingLabelCol)) and len(np.unique(testingLabelCol))) == 1):
        aScore = 1.0
        pScore = 1.0
        rScore = 1.0

        aScoreTotal += aScore
        pScoreTotal += pScore
        rScoreTotal += rScore
        #print(i,score,np.unique(trainingLabelCol),np.unique(testingLabelCol)) #print unique
elements
        print(i,aScore,pScore,rScore)
        continue

    # fit model
    clf.fit(trainingData,trainingLabelCol)

    # generate all predictions
    predictions = clf.predict(testingData)

    # obtain metrics (accuracy, precision, recall)
    aScore = accuracy_score(testingLabelCol, predictions)
    pScore = precision_score(testingLabelCol, predictions, average='weighted')
    rScore = recall_score(testingLabelCol, predictions, average='weighted')

    # Sum all scores to average at the end
    aScoreTotal += aScore
    pScoreTotal += pScore
    rScoreTotal += rScore

    #print(i,aScore,pScore,rScore,np.unique(trainingLabelCol),np.unique(testingLabelCol)) #e
xtended print func
    print(i,aScore,pScore,rScore)

# also works also dont forget the rest of the metrics jiao wants
#accuracy_score((testingLabel[:, [4]]).ravel(), predictions)
aScoreAvg = aScoreTotal/featureCount
pScoreAvg = pScoreTotal/featureCount
rScoreAvg = rScoreTotal/featureCount

print("\n")
print("Mean Accuracy: ", aScoreAvg)
print("Mean Precision: ", pScoreAvg)
print("Mean Recall: ", rScoreAvg)
```

```
Decision Tree Results:
0 0.9972 0.9970259425391778 0.9972
1 0.994 0.9939054417945009 0.994
2 0.98728 0.9870550926406854 0.98728
3 0.9709 0.969866998481936 0.9709
4 0.9501 0.9488056865711675 0.9501
5 0.91538 0.9121859462365308 0.91538
6 0.90628 0.9030547051781299 0.90628
7 0.87442 0.872127106788945 7 0.87442
8 0.88076 0.8772736805620921 0.88076
9 0.84832 0.8471847392116618 0.84832
10 0.84798 0.8439515543170978 0.84798
11 0.81954 0.8152767039308854 0.81954
12 0.81392 0.8091706076354079 0.81392
13 0.79466 0.7894819712011323 0.79466
14 0.77596 0.7702700192934141 0.77596
15 0.77918 0.7727125635343316 0.77918
16 0.7648 0.7593233227786733 0.7648
17 0.80242 0.7974104621772419 0.80242
18 0.81712 0.8135441432586242 0.81712
19 0.83072 0.8278640502206966 0.83072
20 0.86442 0.8628015196970772 0.86442
21 0.856 0.852925655846768 0.856
22 0.9051 0.90382754394975 0.9051
23 0.9003 0.8992790108684932 0.9003
24 0.94796 0.9475434118232896 0.94796
25 0.97148 0.9713853887191384 0.97148
26 0.98314 0.983154116218633 0.98314
27 0.98696 0.9870785780987964 0.98696
28 1.0 1.0 1.0
29 1.0 1.0 1.0
30 1.0 1.0 1.0
31 1.0 1.0 1.0


Mean Accuracy:  0.8995718749999999
Mean Precision:  0.8973589363616962
Mean Recall:  0.8995718749999999
```

In [6]:

```
time_elapsed = datetime.now() - start_time
print('Time elapsed (hh:mm:ss.ms) {}'.format(time_elapsed))
```

```
Time elapsed (hh:mm:ss.ms) 0:00:23.499191
```

Final Summary:

Decision tree model was used since it had the best accuracy compared to:

- Neural Networks
- SVM
- Naive Bayes
- Random Forests

One issue was encountered in col[32] all results were 0 which means only one class existed and caused the model to error out, this was corrected by detecting the amount of unique values in the columns if only 1 was detected. It would mean that the prediction would always be true unless proven otherwise.