1 DT_train_binary(X,Y,max_depth)

This function takes the training data as input. As the project description states the labels and the features are binary, but the feature vectors can be of any finite dimension so we have made sure to implement this function in a way were we don't have to check the dimensions or even take that part into account when we are training our data, the program will take care of that during run-time. The function DT train binary(X,Y,max_depth) returns the decision tree trained using information gain, limited by some given maximum depth. If max depth is set to -1 then learning only stops when we run out of features of our information gain is 0. We have decided to store the decision tree is a simple list, hence making it easy to access and modify later.

1.1 Implementation

We start by finding the best feature to split on, we find all the 0's and 1's. we start building the tree by putting the 0's on the left side and 1's on the right side. After that we delete the column, check the depth if we can still go we will do the same thing recursively until we can no longer keep going. The way we choose the best feature to split on is by using information gain and the way we decide to make decisions is by using entropy on the left and right splits, since if the entropy is 0 we should make a decision and not keep building the tree.

2 DT_test_binary(X,Y,DT)

This function takes test data X and test labels Y and a learned decision tree model DT from the first function, and returns the accuracy on the test data using the decision tree for predictions.

2.1 Implementation

Our tree is just a list of 3 variables, root index, left child, right child so we always have to check if we have all those variables. After making sure the tree is correct we go down the tree and start appending the tree to the prediction, compare it to the test label and return the accuracy.

3 DT_train_binary_best(X_train, Y_train, X_val, Y_val)

We Generate decision trees based on information gain on the training data, and return the tree that gives the best accuracy on the validation data.

3.1 Implementation

The way our algorithm works is that we keep track of the current accuracy that the tree has using the training feature. Then we keep building the tree and compare it to the validation data. If we get a better accuracy we update our current accuracy to be that better accuracy and we do that until we can no longer get a better accuracy.

3.2 Testing

3.2.1 DT_train_binary(X,Y,max_depth)

After testing our training Set 1 using max-depth of -1 we get the following results

```
Decision Tree : [1,0,1]
Accuracy 1.0 or 100%
```

After testing our training Set 2 using max-depth of -1 we get the following results

```
Decision Tree : [3,0,2]
Accuracy 8.33334 or 83.33333333333334%
```

3.3 Friends Accuracy

We are using the training samples given to us and our generated DT train binary(X,Y,max depth) function with a max depth=5 generate three decision trees trained with the first, last, and middle 5 samples, respectively. Since the assignment description was a little vague about the middle 5 and that there are 2 middle 5 samples to go with we decided to use 3,4,5,6,7 as the middle 5 samples.

First sample

```
Prediction: [
            "first": 1,
            "second:1,
            "third": 0]
Accuracy: 1/3
```

2nd samples

```
Prediction: [
"first": 0,
"second:0,
"third": 1]
Accuracy: 1/3
```

3rd samples

```
Prediction: [
"first": 0,
"second:0,
"third": 0]
Accuracy: 2/3
```

According to our 3 samples the predictions were 1 out of 3 on the first sample, 1 out of 3 on the 2nd sample and 2 out of 3 on the 3rd sample. the accuracy is definitely not perfect and can be improved and the last sample gave us the best accuracy.

3.3.1 DT_train_binary(X,Y,max_depth)

4 DT_make_prediction(x,DT)

This function takes a single sample and a trained decision tree and returns a single classification. The output is a scalar value.

4.1 Implementation

The way we are calculating the prediction is by traversing the tree going to the left and right childs and checking their values according to the feature data set, so left is 0 and right is 1.

5 DT_train_real(X,Y,max_depth)

The function takes training data as input. The labels and the features are not the same, the features are real values and the labels are binary (so 0's and 1's), The feature vectors can still be of any finite dimension. We are again structuring the training feature data (X) as a 2D numpy array, with each row corresponding to a single sample. The training labels (Y) should be structured as a 2D numpy array, with each row corresponding to a single label. X and Y should have the same number of rows, with each row corresponding to a single sample and its label. max depth is an integer that indicates the maximum depth for the resulting decision tree.

5.1 Implementation

The function starts by finding thresholds by averaging the samples for a feature, and then splitting on the .10 percentile of each in either direction. For example, for the second feature, we would get an average of

3.09375 and multiply that by two getting 6.1875. Then the splits would be .61875, 1.2375, 1.85625...5.568, and 6.1875. Information gain is then calculated for each split which is defined less than or equal to for each threshold. The split with the maximum information gain for all features is chosen as the threshold for a newly added node. This function could be modified to be more more accurate or more efficient. For better accuracy we could use a smaller percentile such as .05 or .01. For better efficiency we could limit the splits looked at to a range such as 40 to 60 percent versus the whole gamut. With this methodology it is possible to split on the same feature multiple times. This process continually builds the tree until the max_depth is reached.

6 DT_test_real(X,Y,DT)

This function takes test data X and test labels Y and a learned decision tree model DT, and returns the accuracy on the test data using the decision tree for predictions.

6.1 Implementation

Follows the same process as DT_test_binary as well as the same list structure. We then count the amount of correctly predicted samples, and then use that to return the overall accuracy.

7 DT_train_real_best(X_train,Y_train,X_val,Y_val)

Generates decision trees based on information gain on the training data, and return the tree that gives the best accuracy on the validation data.

7.1 Implementation

Follows the same process as the binary tree, creating a random forest of trees, and returning the one that provides the best accuracy.