ma438-hw3

Name: Ji Ma
For the following questions, use adult.data as the training file and adult.test as the test file.

In [1]:
```
import decisiontree
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.cm as cm
```

1. Vanilla

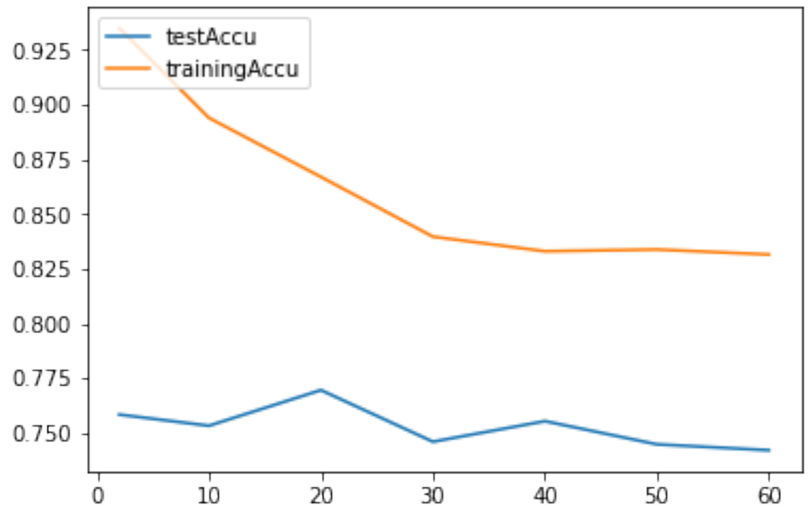
For the full decision tree (vanilla), measure the impact of training set size on the accuracy and size of the tree. (4 points) Consider training set percentages {2%, 10%, 20%, 30%, 40%, 50%, 60%}
Plot a graph of test set accuracy and training set accuracy against training set percentage on the same plot.
Plot another graph of number of nodes vs training set percentage.

In [2]:
```
percentages = [2, 10, 20, 30, 40, 50, 60]
testAccu = []
trainingAccu = []
nodeCounts = []
for p in percentages:
    decisiontree.main('adult.data','adult.test', 'vanilla', p)
    trainingAccu.append(decisiontree.train_accu)
    testAccu.append(decisiontree.test_accu)
    nodeCounts.append(decisiontree.nodeCount)
```

```
Training set accuracy: 0.935
Test set accuracy: 0.7583
Training set accuracy: 0.894
Test set accuracy: 0.7532
Training set accuracy: 0.867
Test set accuracy: 0.7695
Training set accuracy: 0.839666666667
Test set accuracy: 0.7459
Training set accuracy: 0.833
Test set accuracy: 0.7553
Training set accuracy: 0.8338
Test set accuracy: 0.7447
Training set accuracy: 0.8315
Test set accuracy: 0.742
```

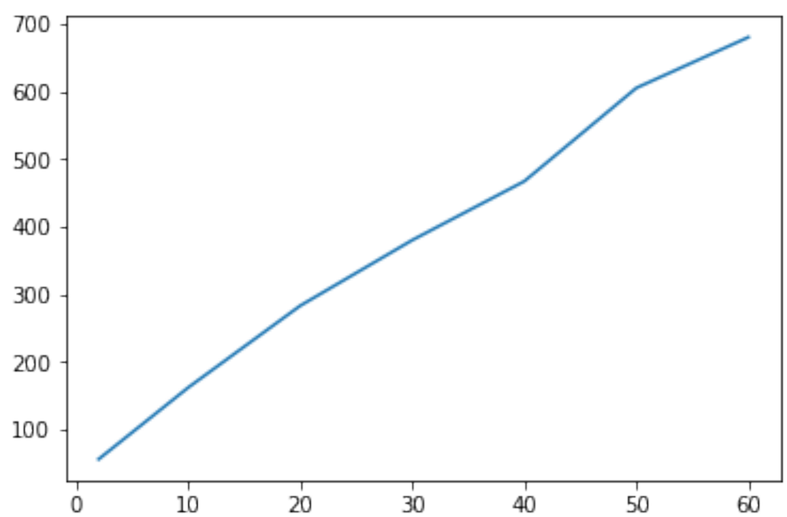
In [3]:

```
# Plot a graph of test set accuracy and training set accuracy against training set percentage on the same plot.
plt.plot(percentages, testAccu)
plt.plot(percentages, trainingAccu)
plt.legend(['testAccu', 'trainingAccu'], loc='upper left')
plt.show()
```



In [4]:

```
# Plot another graph of number of nodes vs training set percentage.
plt.plot(percentages,nodeCounts)
plt.show()
```



2. Depth

Repeat the same analysis for the static-depth case (depth). (7 points) Again, consider values of training set percentage from {2%, 10%, 20%, 30%, 40%, 50%, 60%}. The validation set percentage will remain 40% for all the cases. Consider values of maximum depth from {1, 4, 7, ..., 34} and pick the best value using the validation set accuracy. The accuracies you report will be the ones for this value of maximum depth. So, for example, if the best value of maximum depth for training set 10% is 4, you will report accuracies for 10% using 4; if for 20% it is 16, you will report accuracies for 20% using 16. Plot a graph of test set accuracy and training set accuracy against training set percentage on the same plot. Plot another graph of number of nodes vs training set percentage. Finally, plot the optimal choice of depth against the training set percentage.

In [5]:
```
percentages = [2, 10, 20, 30, 40, 50, 60]
depths = []
i = 1
while i <= 34:
    depths.append(i)
    i += 3
print depths
valid = 40
testAccu = []
trainingAccu = []
nodeCounts = []
depthChoice = []
for p in percentages:
    tempValidAccu = []
    tempTrainingAccu = []
    tempTestAccu = []
    tempNodeCounts = []
    for d in depths:
        decisiontree.depthHelper('adult.data','adult.test', p, 40, d)
        tempTrainingAccu.append(decisiontree.train_accu)
        tempTestAccu.append(decisiontree.test_accu)
        tempValidAccu.append(decisiontree.valid_accu)
        tempNodeCounts.append(decisiontree.nodeCount)
    maxIdx = tempValidAccu.index(max(tempValidAccu))
    trainingAccu.append(tempTrainingAccu[maxIdx])
    testAccu.append(tempTestAccu[maxIdx])
    nodeCounts.append(tempNodeCounts[maxIdx])
    depthChoice.append(depths[maxIdx])
```

```
[1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34]
Training set accuracy: 0.525
Validation set accuracy: 0.49325
Test set accuracy: 0.5016
Training set accuracy: 0.59
```

```
Validation set accuracy: 0.57575
Test set accuracy: 0.5749
Training set accuracy: 0.735
Validation set accuracy: 0.674
Test set accuracy: 0.6904
Training set accuracy: 0.9
Validation set accuracy: 0.7385
Test set accuracy: 0.7534
Training set accuracy: 0.87
Validation set accuracy: 0.742
Test set accuracy: 0.7516
Training set accuracy: 0.925
Validation set accuracy: 0.74025
Test set accuracy: 0.7503
Training set accuracy: 0.945
Validation set accuracy: 0.74825
Test set accuracy: 0.7561
Training set accuracy: 0.955
Validation set accuracy: 0.74475
Test set accuracy: 0.7582
Training set accuracy: 0.935
Validation set accuracy: 0.74375
Test set accuracy: 0.7588
Training set accuracy: 0.955
Validation set accuracy: 0.747
Test set accuracy: 0.7549
Training set accuracy: 0.94
Validation set accuracy: 0.74825
Test set accuracy: 0.7588
Training set accuracy: 0.94
Validation set accuracy: 0.74525
Test set accuracy: 0.7583
Training set accuracy: 0.524
Validation set accuracy: 0.51925
Test set accuracy: 0.4935
Training set accuracy: 0.513
Validation set accuracy: 0.5045
Test set accuracy: 0.5051
Training set accuracy: 0.589
Validation set accuracy: 0.5805
Test set accuracy: 0.5673
Training set accuracy: 0.675
Validation set accuracy: 0.63525
Test set accuracy: 0.6271
Training set accuracy: 0.801
Validation set accuracy: 0.71725
Test set accuracy: 0.7206
Training set accuracy: 0.862
Validation set accuracy: 0.7425
Test set accuracy: 0.7438
```

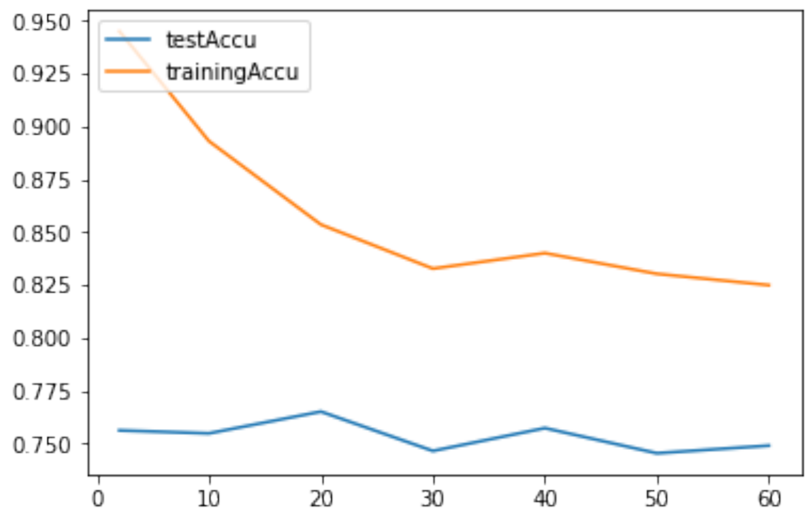
```
Training set accuracy: 0.888
Validation set accuracy: 0.75225
Test set accuracy: 0.7499
Training set accuracy: 0.893
Validation set accuracy: 0.752
Test set accuracy: 0.7578
Training set accuracy: 0.886
Validation set accuracy: 0.757
Test set accuracy: 0.7564
Training set accuracy: 0.885
Validation set accuracy: 0.7555
Test set accuracy: 0.7581
Training set accuracy: 0.894
Validation set accuracy: 0.754
Test set accuracy: 0.7563
Training set accuracy: 0.893
Validation set accuracy: 0.76
Test set accuracy: 0.7547
Training set accuracy: 0.5005
Validation set accuracy: 0.4925
Test set accuracy: 0.5068
Training set accuracy: 0.4935
Validation set accuracy: 0.5205
Test set accuracy: 0.5103
Training set accuracy: 0.5625
Validation set accuracy: 0.57025
Test set accuracy: 0.5626
Training set accuracy: 0.7395
Validation set accuracy: 0.68875
Test set accuracy: 0.6697
Training set accuracy: 0.749
Validation set accuracy: 0.70125
Test set accuracy: 0.6931
Training set accuracy: 0.777
Validation set accuracy: 0.7035
Test set accuracy: 0.7033
Training set accuracy: 0.8075
Validation set accuracy: 0.733
Test set accuracy: 0.7377
Training set accuracy: 0.8235
Validation set accuracy: 0.755
Test set accuracy: 0.754
Training set accuracy: 0.86
Validation set accuracy: 0.76225
Test set accuracy: 0.7717
Training set accuracy: 0.8635
Validation set accuracy: 0.76325
Test set accuracy: 0.7671
Training set accuracy: 0.8535
Validation set accuracy: 0.767
```

```
Test set accuracy: 0.765
Training set accuracy: 0.859
Validation set accuracy: 0.761
Test set accuracy: 0.7637
Training set accuracy: 0.494333333333
Validation set accuracy: 0.494
Test set accuracy: 0.4947
Training set accuracy: 0.516333333333
Validation set accuracy: 0.494
Test set accuracy: 0.5003
Training set accuracy: 0.539666666667
Validation set accuracy: 0.513
Test set accuracy: 0.5132
Training set accuracy: 0.593333333333
Validation set accuracy: 0.5495
Test set accuracy: 0.5454
Training set accuracy: 0.737666666667
Validation set accuracy: 0.70075
Test set accuracy: 0.7019
Training set accuracy: 0.777666666667
Validation set accuracy: 0.72025
Test set accuracy: 0.7216
Training set accuracy: 0.798333333333
Validation set accuracy: 0.7185
Test set accuracy: 0.7308
Training set accuracy: 0.818666666667
Validation set accuracy: 0.73175
Test set accuracy: 0.7379
Training set accuracy: 0.827333333333
Validation set accuracy: 0.7285
Test set accuracy: 0.7468
Training set accuracy: 0.826
Validation set accuracy: 0.73275
Test set accuracy: 0.7389
Training set accuracy: 0.832666666667
Validation set accuracy: 0.737
Test set accuracy: 0.7464
Training set accuracy: 0.828666666667
Validation set accuracy: 0.734
Test set accuracy: 0.7472
Training set accuracy: 0.48075
Validation set accuracy: 0.50675
Test set accuracy: 0.4983
Training set accuracy: 0.489
Validation set accuracy: 0.50575
Test set accuracy: 0.5035
Training set accuracy: 0.5825
Validation set accuracy: 0.58475
Test set accuracy: 0.5915
Training set accuracy: 0.66325
```

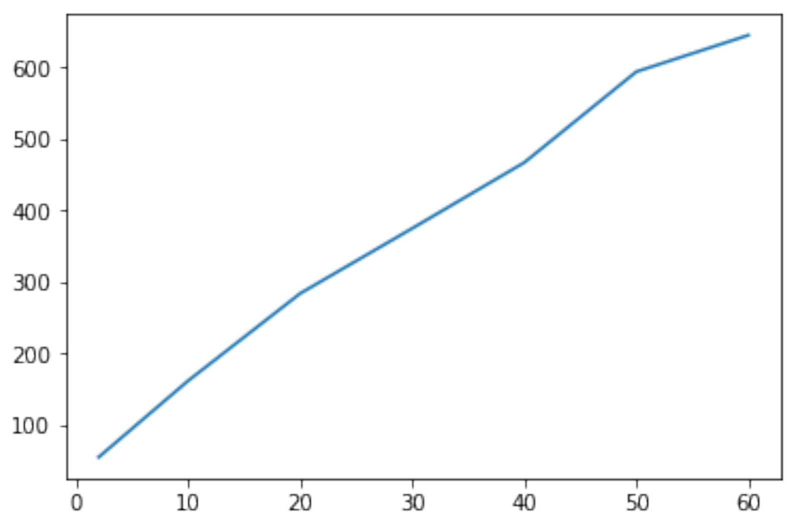
```
Validation set accuracy: 0.648
Test set accuracy: 0.6357
Training set accuracy: 0.70675
Validation set accuracy: 0.68025
Test set accuracy: 0.684
Training set accuracy: 0.75575
Validation set accuracy: 0.69925
Test set accuracy: 0.7088
Training set accuracy: 0.7875
Validation set accuracy: 0.71825
Test set accuracy: 0.725
Training set accuracy: 0.813
Validation set accuracy: 0.736
Test set accuracy: 0.7397
Training set accuracy: 0.82825
Validation set accuracy: 0.75225
Test set accuracy: 0.758
Training set accuracy: 0.83075
Validation set accuracy: 0.751
Test set accuracy: 0.7574
Training set accuracy: 0.832
Validation set accuracy: 0.754
Test set accuracy: 0.7566
Training set accuracy: 0.84
Validation set accuracy: 0.75475
Test set accuracy: 0.7572
Training set accuracy: 0.503
Validation set accuracy: 0.49575
Test set accuracy: 0.5003
Training set accuracy: 0.4986
Validation set accuracy: 0.4955
Test set accuracy: 0.4949
Training set accuracy: 0.599
Validation set accuracy: 0.5975
Test set accuracy: 0.5863
Training set accuracy: 0.6508
Validation set accuracy: 0.6325
Test set accuracy: 0.6314
Training set accuracy: 0.698
Validation set accuracy: 0.6665
Test set accuracy: 0.6711
Training set accuracy: 0.7342
Validation set accuracy: 0.69125
Test set accuracy: 0.6866
Training set accuracy: 0.776
Validation set accuracy: 0.72375
Test set accuracy: 0.7161
Training set accuracy: 0.8036
Validation set accuracy: 0.73025
Test set accuracy: 0.7267
```

```
Training set accuracy: 0.819
Validation set accuracy: 0.74275
Test set accuracy: 0.743
Training set accuracy: 0.8182
Validation set accuracy: 0.7395
Test set accuracy: 0.7469
Training set accuracy: 0.8302
Validation set accuracy: 0.75075
Test set accuracy: 0.7453
Training set accuracy: 0.8282
Validation set accuracy: 0.73875
Test set accuracy: 0.7426
Training set accuracy: 0.492333333333
Validation set accuracy: 0.50075
Test set accuracy: 0.4914
Training set accuracy: 0.504333333333
Validation set accuracy: 0.5015
Test set accuracy: 0.4967
Training set accuracy: 0.517
Validation set accuracy: 0.51075
Test set accuracy: 0.5085
Training set accuracy: 0.592166666667
Validation set accuracy: 0.56325
Test set accuracy: 0.5781
Training set accuracy: 0.676666666667
Validation set accuracy: 0.62425
Test set accuracy: 0.6415
Training set accuracy: 0.748
Validation set accuracy: 0.696
Test set accuracy: 0.6922
Training set accuracy: 0.776666666667
Validation set accuracy: 0.71275
Test set accuracy: 0.7146
Training set accuracy: 0.803333333333
Validation set accuracy: 0.73
Test set accuracy: 0.7359
Training set accuracy: 0.812
Validation set accuracy: 0.73875
Test set accuracy: 0.7431
Training set accuracy: 0.819
Validation set accuracy: 0.7415
Test set accuracy: 0.7414
Training set accuracy: 0.824833333333
Validation set accuracy: 0.7475
Test set accuracy: 0.7489
Training set accuracy: 0.830833333333
Validation set accuracy: 0.74475
Test set accuracy: 0.7471
```

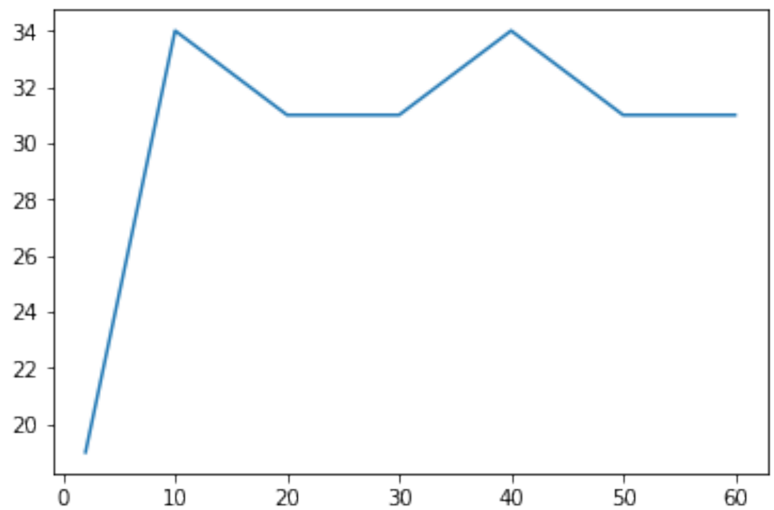
In [6]:
```
# Plot a graph of test set accuracy and training set accuracy against
training set percentage on the same plot.
plt.plot(percentages, testAccu)
plt.plot(percentages, trainingAccu)
plt.legend(['testAccu', 'trainingAccu'], loc='upper left')
plt.show()
```



In [7]:
```
# Plot another graph of number of nodes vs training set percentage.
plt.plot(percentages,nodeCounts)
plt.show()
```



In [8]:
```
# Finally, plot the optimal choice of depth against the training set p
ercentage.
plt.plot(percentages,depthChoice)
plt.show()
```



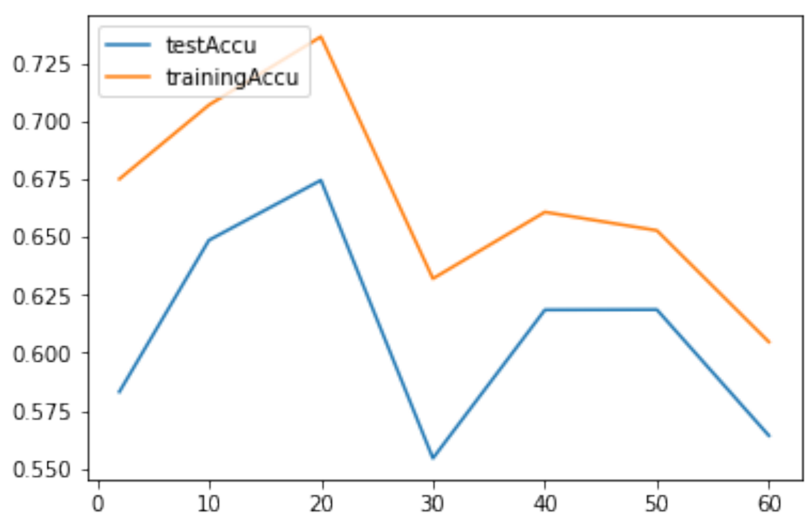
3. Prune

Repeat the above analysis for the pruning case (prune). (7 points) Again, consider values of training set percentage from {2%, 10%, 20%, 30%, 40%, 50%, 60%}. The validation set percentage will remain 40% for all the cases. You will use the validation set when deciding to prune. Plot a graph of test set accuracy and training set accuracy against training set percentage on the same plot. Plot another graph of number of nodes vs training set percentage.

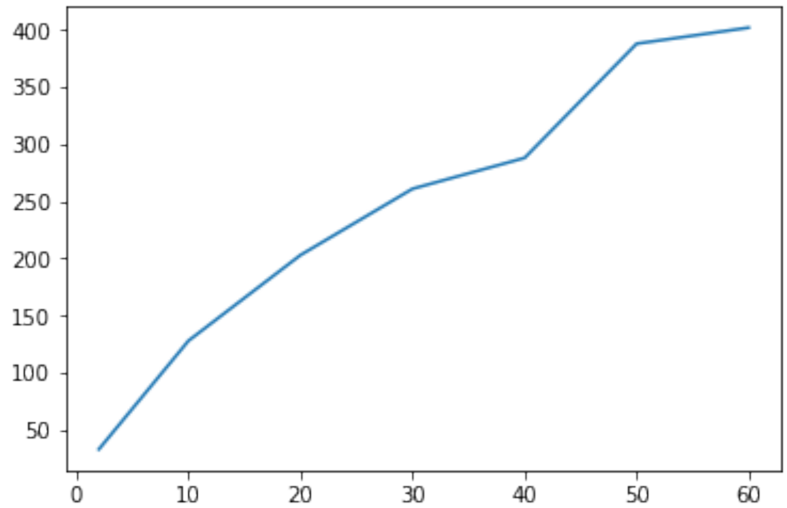
In [2]:
```
percentages = [2, 10, 20, 30, 40, 50, 60]
valid = 40
testAccu = []
trainingAccu = []
nodeCounts = []

for p in percentages:
    decisiontree.pruneHelper('adult.data','adult.test', p, 40)
    nodeCounts.append(decisiontree.nodeCount)
    testAccu.append(decisiontree.test_accu)
    trainingAccu.append(decisiontree.train_accu)
```

```
Training set accuracy: 0.675
Test set accuracy: 0.5831
Training set accuracy: 0.707
Test set accuracy: 0.6486
Training set accuracy: 0.7365
Test set accuracy: 0.6745
Training set accuracy: 0.632
Test set accuracy: 0.5544
Training set accuracy: 0.66075
Test set accuracy: 0.6185
Training set accuracy: 0.6528
Test set accuracy: 0.6186
Training set accuracy: 0.604666666667
Test set accuracy: 0.5642
```

In [3]:
```
# Plot a graph of test set accuracy and training set accuracy against training set percentage on the same plot.
plt.plot(percentages, testAccu)
plt.plot(percentages, trainingAccu)
plt.legend(['testAccu', 'trainingAccu'], loc='upper left')
plt.show()
```



In [4]:
```
# Plot another graph of number of nodes vs training set percentage.
plt.plot(percentages,nodeCounts)
plt.show()
```



4.

Why don’t we prune directly on the test set? Why do we use a separate validation set? (3 points)t

First of all, using testing data for pruning will 'contaminate' the pureness of the purpose of the testing data set. For example, the model will tends to perform well on testing data by adapting to perform well on it without generalizing. So, we need a seperate validation data set for us to test the performance along the pruning process while we still have untouched testing data for the final round benchmarking.

5.

How would you convert your decision tree (in the depth and prune cases) from a classification model to a ranking model? (4 points) That is, how would you output a ranking over the possible class labels instead of a single class label?

So, this is the question about how do we convert the absolute classification tasks like either true or false to the task that identify the probobilities for different labels.

Actually, in my implementation of this decision tree classification algorithm, it could easily adapt to the feature like ranking over the possible class labels instead of single class label because of some of the 'uncertainty' inside of the leaf node of the decision tree. For example, for some leaf nodes, it is not purely classifies as '>=50' or '<50', but a misure of both of the instances like 9/10 chance it's gonna be '>=50', and 1/10 gonna be '<50'. So in the single class label version, I just choose the highest probable answers in the leaf node and return to the decision. So, in order to make the answer lists of the ranking of possible results, we could just make the output not as a list of possible result answers instead of single highest probable result.

In [ ]: