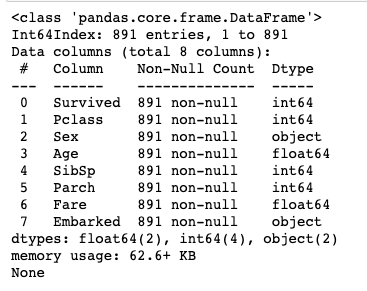
**Data acquisition**

The **FIRST** data acquired in this chapter is the DecTreeAssign1.dat data provided by Dr. Santago. The data provided in the DecTreeAssign1.dat include one class variable together with 30 features named by Fxx where xx is the feature id (00, 01, ...., 29). The class variable is a binary categorical variable with data values “B” and “M”. The remaining features are all continuous variables for the purpose of decision tree building.

The **SECOND** data I used is the Titanic data I downloaded from Kaggle and preprocessed (all missing data imputed). The dataset has 891 rows and 8 columns. Among them, the dataset has 7 features to predict whether or not a person survived on titanic. The categorical features are “Sex” and “Embarked”; while the continuous features are “Pclass”, “Age”, “SibSp”, “Parch”, and “Fare”. Here shown below is a summary of the dataset.



**Program development**

I developed two programs for this chapter:

The **FIRST** program is to build my own (part of a) decision tree classifier on the “DecTreeAssign1.dat” dataset. The most important parts of the decision tree built in this project are listed below:

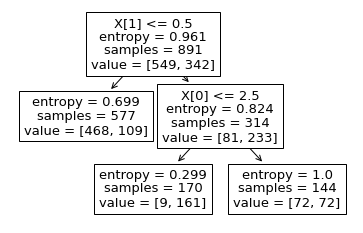
1. Calculate the root node information including the number of observations for each class, the gini index and the entropy value, etc.
2. Split the first root with respect to each of the 30 features at their mean value. Calculate the corresponding split information, including the number of observations, gini index and entropy at each child, altogether with the combined gini and entropy values.
3. Compare the combined gini or entropy values, choose the lowest one (which gives the largest information gain) to be the first split.

The notebook file where I implemented the codes is called “Decision Tree Building.ipynb” and the more detailed implementation can be found in there. There is also one final

**ANOTHER thing** I did in this chapter was to experiment with building an actual decision tree. The codes are implemented in the notebook file “Decision Tree Titanic.ipynb”. What I did in this program is to build an actual decision tree using the given features to conduct binary classification on whether or not people survived in the incident.

There are several interested aspect I need to deal with when building the tree:

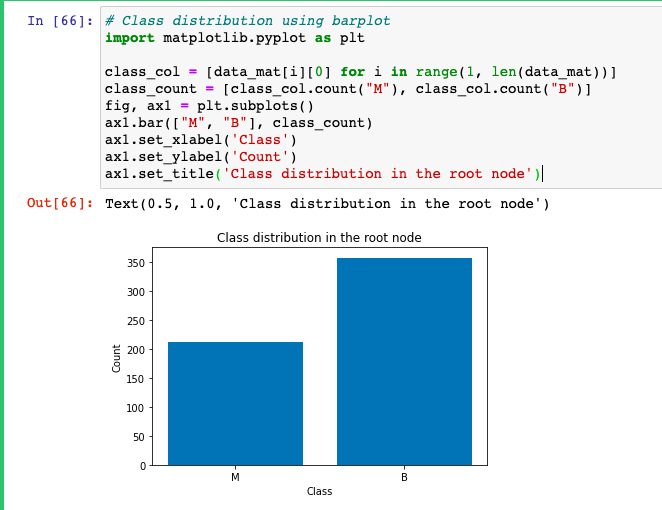
1. The in-built decision tree classifier in scikit-learn package does not support categorical features. Therefore, in order to fit categorical features like “Sex” and “Embarked” into my decision tree classifer, I need to first conduct some sort of transformation to my categorical features. For “sex”, whenever an instance has “male” in this entry, I replace it with 0; and for “female” I replace it with 1. “Embarked” feature is transformed in a similar manner except that I need 0, 1, and 2 to represent “S”, “Q”, and “C”.
2. Use graphs to represent the different decision trees built. The graphs are able to show what feature and what value the decision tree splits on. Also, it shows the gini index on each node and leaf. The graphic representation of one decision tree model I built is shown below:



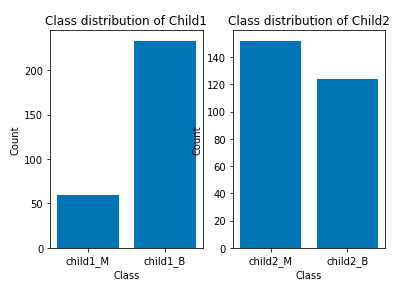
**Data analysis and package use**

I use python in Jupyter Notebook as my programming language and IDE. The packages I used are math, sklearn, and matplotlib in python.

There are some quite interesting findings through working on these program developments. For my **FIRST** program, I experimented with plotting the class distribution. The first plot is the class distribution of the root node before the splitting. The plot shows that the major class in the root node is “B”; nevertheless we cannot classify all instances in the root node to class “B”.



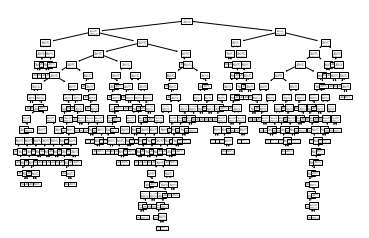
I also plot the class distribution of both child 1 and child 2 after splitting at F23 which is the best splitting point according to both the gini index and the entropy. The resulting class distribution seems more divided, especially for child 1. We have more confidence in classifying all data instances in child1 to class “B”. Still, we don’t have enough evidence to classify the data instances in child2 to any class.



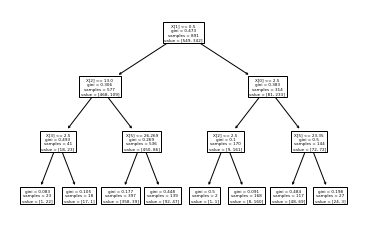
Using the visualization skills, I was able to confirm that my first split is effective in classifying the data instances into different classes.

**For my OTHER program** working on the Titanic dataset, I built several decision trees using the decision tree classifier in package sklearn:

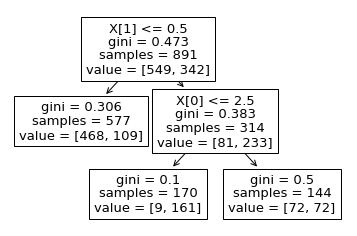
1. I used the default parameter list for the decision tree first. The default parameter list is:
   1. criterion='gini', splitter='best', max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features=None, random\_state=None, max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, class\_weight=None, ccp\_alpha=0.0.
2. Since there is no limit on max\_depth and min\_impurity\_decrease that the tree can extend, the tree keeps growing until every instance can be classified to their desired class. This is really bad because the tree becomes messy and there is definitely overfitting in the tree. The tree built is shown below:



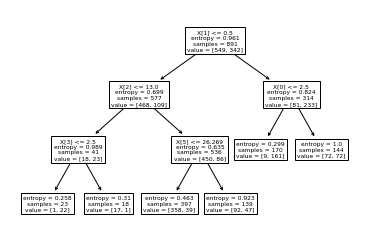
1. In order to improve on such problems, I used different parameter values for max\_depth and min\_impurity\_decrease. First I used max\_depth = 3 to limit the size of the tree:



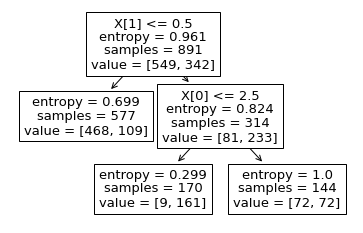
1. Another approach I took is to set min\_impurity\_decrease = 0.02. In other words, stop growing if the growth in impurity measure from a node to its child nodes is less than 0.02, then stop growing:



1. Using the min\_impurity\_decrease to limit the model overfitting, I was able to build a concise model as shown above. In addition to that, I also want to see if changing the splitting criterion to “entropy” will change the model at all. Therefore, I used the classifier “DecisionTreeClassifier(criterion='entropy', min\_impurity\_decrease = 0.02)” to build the decision tree.



1. The tree built is more complex than the tree previously built using gini index as splitting criterion. However, this is only because min\_impurity\_decrease is generally larger for entropy compared to gini index. If we change the min\_impurity\_decrease to 0.05, the newly built tree is exactly the same as the one using min\_impurity\_decrease = 0.02 and gini index as splitting criterion.



**Student learning summary and self-assessment**

Through the study of this chapter, I was able to understand the basic classification techniques a lot better than before. Decision tree is certainly one of the most important classifiers we learned in this chapter. Through studying, applying, and building my own decision tree model, I build solid understanding on the calculation of impurity measures (including entropy, gini index, etc.), and also how to combine the impurity measures of the child nodes in order to decide which node to split on. Model overfitting can be annoying when fitting a decision tree, but in my case I experiment a little bit with avoiding that via modifying the parameters of the decision tree classifier I built using the scikit-learn package.

Questions:

1. I saw that decision tree classifiers are capable of handling missing values using either probabilistic split method or surrogate split method. They are implemented in C4.5 and CART decision tree algorithms respectively, but I still don’t quite understand how they may be implemented in real cases.

Something I want to discover more:

1. I want to explore more about how interactions among different attributes can affect the performance of decision tree classifiers.
2. Algorithms like C4.5 or CART and how they implement the decision tree classifier uniquely.

Self-assessment:  
I believe that my understanding of notions related to basic classification concepts and methods is solid. I am confident in applying decision trees to real world problems, and adjust the decision tree for the best performance of the model. Based upon those understanding, I would like to give myself an A on this chapter.