Project 2: Extracting Patterns from the Titanic Data Set

This project is due November 19. Teams of up to three members can submit jointly. Team members do not have to be in the same section. There will be a help session again on November 16 in lieu of class.

In this project we will go through a first exercise in data science. We are going to look at the chances for survival during the Titanic disaster. The data set is split in two different sets, the training set and the test set. Both are in CSV format. The task is to predict the survival of a passenger based on the data.

1. Data Acquisition and Data Wrangling

Our first task is to obtain and process the data. You will have to write a function that extracts the data from the training file.

When you look at the data, you will find several problems. The cabin numbers are given for few passengers and while the number indicates the class of the passenger and possibly the nearness of an emergency exit, the latter is unlikely to have been a cause in the survival or death of a passenger and the former is already encoded in the class and probably also in the fare. The embarkation harbor has probably little relevance on the survival of a passenger and we can safely disregard it. While names might give hints on the racial origin of the passenger and hence the way they were treated by the crew, this is too difficult for us to use.

However, there is one category that obviously has an impact on the survival chances of a passenger, namely the age. Ages of small children were denoted as fractions, which explains the strange format. However, many ages have not been filled out. Dealing with missing data is part of what is the process of data wrangling. To deal with the age, we first use binning. We establish categories of ages:

- Children : age under 17
- Young adults: age under 25
- Adults: age under 60
- Aged: the rest.

When the age is missing, we can use our *domain knowledge* to make some guesses to fill in missing ages. We use the following rule for assigning missing ages. If the name contains “Master”, this is a child. If the name contains Miss and there are siblings or there are parents present, then it is a child. If the name contains Miss otherwise, we just remove the record from the data set.

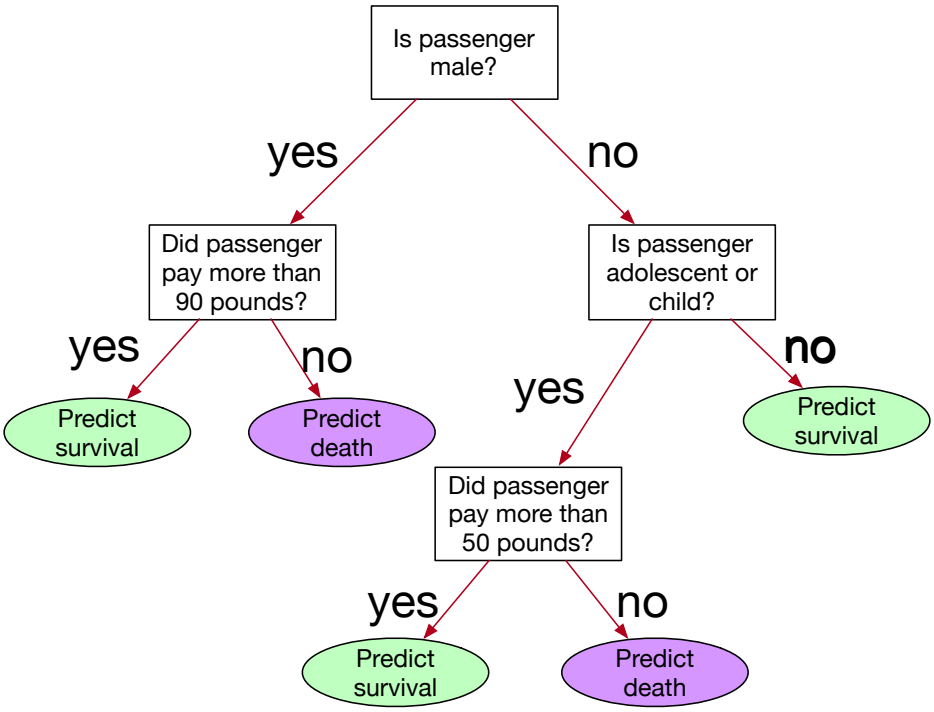
If the fare is not recorded, we just set it to 0.

Task 1: Implement a function that creates a set/list of tuples from the training and the testing data set with codes for:

- Survival: yes or no
- Class booked
- Sex: Male or female, if unknown, assign one
- Age: Use the binning and the rules previously discussed
- Siblings: yes or no
- Parents present: yes or no
- Fare

2. Decision Trees

We assume that survival is a probability depending on a function of the parameters. We want to learn these parameters. A simple, but often effective manner of creating such a function is the use of simple rules. When we look at the survivor data, we notice that children and women fared better as well as the better off. We can come up with a prediction using simple questions, but it works even better if we use a small number of questions. The result is a decision tree:



By asking two or three questions, we now make a prediction whether a passenger survived. The decision tree is not going to give us the correct decision in all cases, but it can achieve an error rate in the ninety percents (so this one probably not). In this case, the type of questions asked might be more of an interest, since we are hardly likely going to embark on the Titanic, even if time-travel were possible. However, in real life, a decision tree like this developed from more recent data could be used to create the rates for a maritime insurance company.

3. Creating Good DecisionTrees

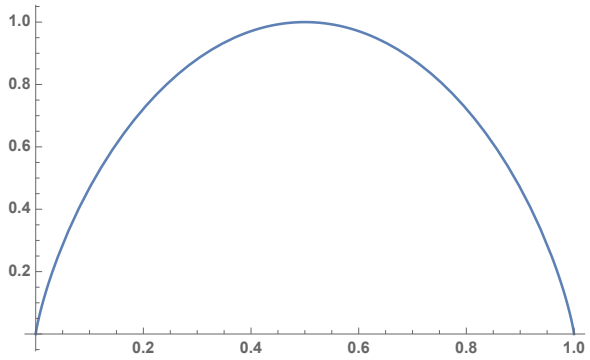
We measure the goodness of a decision tree by trying it out on the test data. Obviously, many possible decision trees could be evaluated. A rather simple algorithm has been most successful. We want to generate a question that will result in a division of a set into two subsets such that the two subsets are more homogeneous than the divided set. After we have split up the dataset into two more homogeneous sets, we try to find two different questions that subdivide these and so on. We stop when the sets are either sufficiently homogeneous or they are small.

We need to measure the homogeneity of a set. A set is perfectly homogeneous, if all its members have either survived or have drowned. For a less perfect homogenous set, there are at least two possibilities. We will use the entropy as a measure. Assume that there are x

survivors and y drowning victims in the set. The homogeneity of the set is measured as its entropy, defined as

$$\text{entropy} = -x\log_2(x) - y\log_2(y).$$

If $x = 0$ or $y = 0$, then the entropy is zero. The higher the entropy, the less homogeneous is the set.



The graph shows the entropy of a set depending on the proportion of elements in the set that survived.

Task 2: Implement a function that calculates the entropy of a set of Titanic passengers.

The best question to be asked is one that yields to more homogeneous sets. We measure the goodness of the question by the increase in homogeneity and for that, we need to calculate the compound homogeneity of the two sets in the partition. We do this by weighing by the cardinalities of the set. Thus, if

$$S = A \cup B, A \cap B = \varnothing$$

is a partition of the set S into two subsets A and B, then the combined homogeneity of the two sets is

$$\text{entropy}(A, B) = \frac{|A|}{|S|}\text{entropy}(A) + \frac{|B|}{|S|}\text{entropy}(B).$$

The best gain is by a question that partitions S into two sets A and B, such that the entropy gain

$$\text{entropy}(S) - \text{entropy}(A, B)$$

is maximized.

Task 3: Implement a function that calculates the entropy of a pair of sets of Titanic passengers.

In the case of the Titanic data set, we need to evaluate any number of possible questions. For binary values such as sex (a binary datum in those days), there is only one possible division, namely putting all elements with one value into one and the ones with the other into the other set. If we have ordinal values such as the age, we can group children vs. everyone else, children and young adults vs adults and aged and finally everyone else vs. aged. If we have a

numeric value such as fare, we have in principle as many questions as there are numbers (almost infinite), but we can limit ourselves to values that appear as fares. One set is then formed by all passengers who paid less than this fare and another is formed by all passengers who paid this fare or more.

Task 4: Implement a function that generates a list of all fares that passengers paid.

Task 5: Write a function that given a set of Titanic passengers will find the fare such that the homogeneity gain of dividing the original set into two subsets (paying less and paying that fare or more) is maximized.

Now we have to try to find the one question of all possible ones that creates the greatest homogeneity gain. Since we do not want to have very small sets, we exclude questions that result in subsets with less than five passengers.

Task 6: Find this question using the functions developed earlier.

Now we would have to continue to subdivide the resulting division of the data set.

Task 7: Create a decision tree such that all leaf nodes have either less than ten elements or have a percentage of at least 80% survivors or 80% drowning victims.

Once we have constructed our decision tree, we can test it on our test set.

Task 8: Use the decision tree in order to create a predictor function.

The predictor function for the example would be:

```
def predict(element):
    if element[4] == male:
        if element[9] > 90:
            return "survived"
        else:
            return "drowned"
    else:
        if element[5] in ["child", "adolescent"]:
            if element[9] > 50:
                return "survived"
            else:
                return "drowned"
        else:
            return "survived"
```

Task 9: Calculate the error rate of your decision tree.