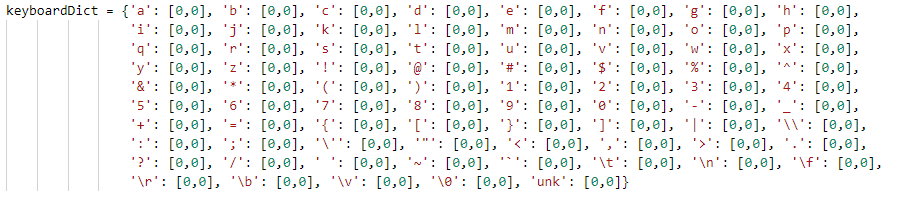
Chapter #

Methodology

The construction of the data set is to be the first and most important part of this research, as if it is not done properly then everything learned from the machine learning algorithm would be without any statistical value. Because of this, we have decided that a number of decisions need to be made. First, the scripts from which all of the comments are to be pulled need to come from a variety of projects that have a higher likely hood of being worked on by programmers with some experience. To this end we have pulled 20 of the topmost trafficked C, C++, C# and Java projects from GitHub. Second, to determine which lines actually contain comments and which do not can be a complex project with the number of different comment formatting options as well as wanting to keep block comments intact for research purposes, to fix this problem we are using srcML, a language parsing tool that converts scripts to XML and is also capable of taking commands so that one can quickly do something such as creating an XML archive of all the comments contained within multiple projects. Now of course potentially the most important piece is the manual verification of comment lines that are being fed into the decision tree algorithm, for this process we have broken down a csv into 7 distinct fields for use in our research.

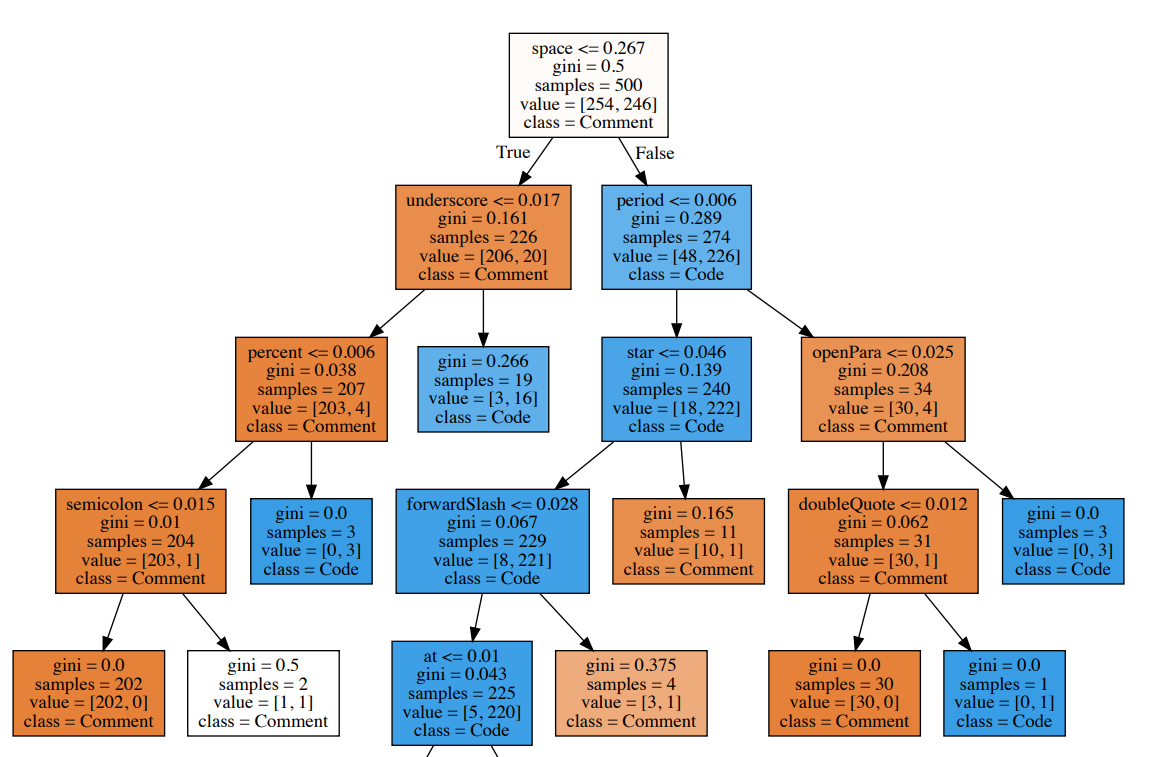
The first of these fields is the comment itself, if the comment is a block comment each like of the csv only contains one line of the comment, but the method we use to denote this is explained shortly. Following the comment field is the file name the comment came from in order to quickly reference both the original file and the original project if need be. The block comment column, which proceeds the file name column, contains either a n to represent a single like comment or two line numbers in parenthesis to represent the outrebounds of the block comment within the csv. For the sake of our research all forms of block comments including doxygen comments are included as block comments. Also included in this csv are two columns related to one another, the language column which displays the original language of the script as well as the contains standard terms column which contains any terms which are considered standard to the language, while this is not currently being utilized in the project, the concept of a bag of words approach is being considered so this data is being preserved for the sake of completeness. The last two columns in the csv are “contains code”, and “is code”. The difference between these two is extremely important as it has been decided that we don’t want to detect lines that reference things such as function names, as this is not truly commented-out code, more an example of code being made into an English term. Both fields are simply marked with either an n or y to note whether the field is a yes or a no. following the construction of the data artifact to be used in this project our next step is to use the information inside a machine learning algorithm to see if we can automate the determination process.

A number of different methods for using this raw string data was considered before coming to the final decision to use a numeric representation. Numeric representation is needed when utilizing the decision tree algorithm present in Scikit Learn, a well-known Python module built for machine learning. To convert the string representations of the comments into useable data frames a multi-step process is used. First a Python dictionary was constructed containing all of the common ASCII characters used in code and in comments shown in Figure #: ASCII Character Dictionary.

Figure #: ASCII character dictionary

The calculation of the values within a line are performed by obtaining a complete count of all values in a comment line and then finding the frequency of that character within the line. This is then stored in the first section of the data frame in the form of a list to be used later by the decision tree. This process was fully automated in the creation of the primary data frame which uses a series of 500 lines of standard comments and 500 lines of commented out code, all of which have been manually verified for the sake of accuracy. Following the section containing all of these values is a targeting section which is created by the automated process as well, denoting a 0 for each true comment line and a 1 for each line of commented-out code. after the targeting section both a target names section is created, and a feature names section is created in order to create an easier to read decision tree model. Once the process of creating the data frame is completed, we can move on to training the decision tree for use in our final analysis.

The process for training the decision tree algorithm is actually quite simple when using Scikit learn. To ensure statistical accuracy of the final output we use a stratified K-fold cross validation method using 5 folds to split our data into a training and testing section at a rate 9 training sections to 1 testing section. Which is then fed into Scikit Learns decision tree and stored for later use. An additional reason for using Scikit Learns decision tree algorithm is that while being user friendly it also prints a very easy to visualize model of the decision tree shown in Figure #: Decision Tree.

 Figure #: Decision Tree