Use of Natural Language Processing and Machine Learning techniques to clinically assess children Autism Spectrum Disorder and Specific Language Impairment

Zhong Kein James Lee*

Faculty of Information and Technology, Monash University, Clayton

October 22, 2016

Abstract

Language Impairment (LI) is characterized by delayed or atypical language development, which can be a prominent disorder in young children. Identifying and quantifying the symptoms and indicators of LI are challenging and often require an experienced practitioner's evaluation. In this paper, we present a method to classify and label children based on their transcribed texts using Natural Language Processing (NLP) and Machine Learning (ML) techniques. NLP is the field of computational linguistics concerned with interactions between computers and natural human language. ML on the other hand is a field of artificial intelligence that enables computers to learn without being explicitly programmed to do so. We propose that the interaction between the features derived from NLP on a child's text and the predictive capabilities of ML will allow us to accurately classify a child based on their text alone.

Keywords: Natural Language Processing, Machine Learning, Autism Spectrum Disorder, Specific Language Impairment, Child Disorder, Ensemble Method
Word Count: 12353

*Electronic address: `zkleee1@student.monash.edu`

Contents

1 Introduction

Language Impairment (LI) is a language disorder usually prominent in younger children. Children with LI exhibit deficits in multiple areas of language proficiency and in some cases these effects may persist into adulthood (Whitehouse et al., 2009). There are a wide variety of tests available that allow clinicians to diagnose LI in children. Nonetheless, these tests are highly dependent on the presence of a clinician and at times a trained linguist for cases that are subtle. LI in children is detected by comparing the scores of children on norm-referenced tests to the mean of a reference population. This means that children who score more than 1.25 times the standard deviation(SD) below the mean of a reference population of children with typical development (TD) on at least 2 types of measures are considered to have LI. However, Campbell et al. (1997) has shown that certain norm-referenced tests are biased against certain groups of children whose population is not accurately represented in the reference group.
The notion of having computer algorithms diagnose clinical disorders is a new and emerging field of studies that is only recently made possible due to the advances made in computer processing power. We propose the use of Natural Language Processing (NLP) and Machine Learning (ML) techniques to take advantage of modern computing prowess to extract indicative linguistic features present in text speech corpora of children that can be potentially useful in a clinical setting to identify children with language disorders.

2 Background

Numerous other studies have been conducted in regards to the usage of NLP and ML to diagnose clinical disorders. The following subsections contains a review on relevant literature as well as research that serve as a basis for this study.

2.1 Literature Review

Up until recently, classification of children with language disorders have mostly been done manually by hand. The downside of this is that this process can be very time consuming and costly as a lot resources have to be committed to each individual diagnosis. Fortunately, the era we live in now provides a more efficient alternative to the old methods of assessing children with language disorders. This can be done through the effective use of machine learning algorithms and natural language processing techniques to classify a transcribed corpus of a child and determine their health based on their linguistic capacity. The literature review consists of 4 sections - Children Language Disorders, NLP, Supervised Classification and Machine Learning Algorithms - where each section provides an overview of the field and reviews of related content. The purpose of this literature review is to research and explore scholarly articles related to the four topics in order to gain a high level comprehension of the topics and their relationships.

2.1.1 Language Impairment in Children

This paper focuses only on two subtypes of LI in children which are Autism Syndrome Disorder(ASD) and SLI. The final goal of this study is to be able to clearly distinguish between a Typically Developing(TD) child, child with SLI and child with ASD.
SLI is a developmental disorder whereby the child with SLI exhibit delayed or disordered language patterns in the absence of other sensory, neurological or other intellectual abnormalities (Solorio, 2013).Solorio (2013) also states that LI can negatively affect social development and academic achievement of a child. Even though children with ASD and SLI share common language phenotypes such as morphology, syntax, phonology (Gabani et al., 2011) and have similar immature lexical-semantic knowledge (Haebig et al., 2015), their definitions differ whereby ASD children also experience pragmatic impairments and broader developmental difficulties (Taylor et al., 2014).
Numerous methods have been explored in order to distinguish between the two groups of LI and TD children. For example, it has been found that both groups of children with LI perform poorly tasks such as Non-Word and Sentence Repetition Task. However only the SLI group performs poorer in the Children's Embedded Figure Test(CEFT) - which required participants to find a simple shape hidden within a more complex geometrical shape - than compared to TD children compared to the relatively similar scores on the test between TD and ASD children (Taylor et al., 2014; Williams et al., 2013). In a Mass Count Test where participants have to judge the quantity of two similar objects in different quantities and sizes, Schaeffer Schaeffer found that SLI children perform significantly better than TD children however High Functioning Autistic(HFA) children do not. Similar results were also produced when testing subject-verb agreement between the three groups. In contrast, Schaeffer Schaeffer found both the HFA group and the SLI group produce correct definite articles significantly less often than the TD group in an Article Choice and Direct Object Scrambling Test.
It's clear that there are many characteristics that define both disorders but current information still lack the precision to produce a definitive clinical classification for both disorders. Further research should be attempted in order to find significant cognitive and linguistic differences between the three groups of children.

2.1.2 Natural Language Processing

NLP is the means by which a computational program analyses, learns and produces human language content (Hirschberg & Manning, 2015). A computer processes a corpus by analyzing its morphology which represents the structure of individual words and its syntax which is the rules and the structure of a sentence of words. Through its syntax it can determine its semantics which is the contextual meaning of the entire sentence. NLP commonly utilizes grammars to represent the syntactic structures of a sentence because they are appropriate for tasks required in NLP such as parsing or syntactic analysis of sentences (Martinez, 2010). Hirschberg and Manning (2015) describes in depth the current state of NLP technologies, their applications and the direction the field is headed into.
NLP was initially known as Natural Language Understanding which was an area of study with the goal of bestowing unto computers the ability to understand natural human lan-

guages. Due to the recent advances in technology that made processing power and a plethora of corpora from the internet to consume available, NLP is now more of a resurgent field in statistics and data analyzing where computer scientists and researchers are constantly finding new ways to utilize its capabilities. NLP is used in many emergent fields such as conducting motivational interviews for substance use disorder patients (Tanana et al., 2016) or in large scale data extraction of echocardiography reports(Nath et al., 2016).
The focus of this paper is on the usage of NLP to identify and distinguish between children who are TD, and those with either ASD or SLI. Similar attempts at classifying children with a linguistic or cognitive disorders have been made by other researchers such as in Prud'hommeaux and Rouhizadeh (2011) study where they attempted to automatically detect pragmatic deficits in children with ASD or in Hassanali's (2013) paper where she uses NLP to analyze child language. A common trait between these articles is that a many features are usually employed to detect LI in children and it is the combination of features that will fine tune the algorithm to give a better precision. A few examples of features are a child's Mean Length Utterance and Total Number of Words in a corpus. In many related works, the number of features can list up to the thousands (Pestian et al., 2010). Most other studies would attempt to derive different NLP related features in preparation for the next phases which are discussed in the following two sections of this paper. A key process in NLP is tokenization and part-of-speech tagging (POS) whereby the program will attempt parse and tokenize natural language in a way for it to understand. Yngve (1960) describes how a parsed tree that has a branching factor towards either the left or right can indicate amount of working memory requirement for a sentence. This can lead to an indication of a sentence's complexity and can be used to indicate other aspects such intelligence of the sentence speaker and more.

2.1.3 Supervised Classification

Detecting patterns is central to NLP as these pattern usually hold new meaning that can be derived from a sentence. A classification is defined as the task of choosing the correct label for a given input. A classifier is called supervised if it is built based on training corpora containing the correct label for each input (Bird et al., 2009).
The choices made in feature selection and how they are implemented plays a heavy role in the effectiveness of a learning method. Features capture the basic information of the input analyzed by the learning method that can be used to classify them. Supervised Classification takes advantage of features derived from NLP as mentioned in the previous section, which through training generate models based on the pairs of labels and feature sets it is passed. After its training it uses the generated models and the corresponding feature sets to predict the labels of new input.
Santafe et al. (2015) have undertaken a study to evaluate the current methods of evaluating a supervised classification algorithms. In the following section we discuss some of the different types of machine learning algorithms used in modern research.

2.1.4 Machine Learning Algorithms

ML is the process by which we train a computational model to produce a classifier based on the data used to train it with. The resulting training module will then be used to make accurate prediction on subsequent data sets (Kotsiantis, 2007; Sharma et al., 2013). In both of these papers they include multiple ML algorithms and a brief review of their usages. A few examples of these algorithms include Decision Trees, Neural Networks, Naive Bayes and k-Nearest-Neighbors among others.

Fundamentally, all of the machine learning algorithms behave differently in terms of how they obtain a prediction. However their end goal is still the same - to predict a classification label based on the given data. To note a few examples, the decision tree classifier builds a tree based on the most significant features from a set of features, to predict a label, it traverses the decision tree until it reaches a leaf node. The choice of which branching path it traverses is based on the specific feature of the data set provided. Compared to the decision tree, the k-Nearest-Neighbors method uses a membership method based on a predefined k nearest neighbors. The principle behind the nearest neighbor methods is to find a predefined number of training samples with the closest distance to the new point, and predict the label from these (Pedregosa et al., 2011). The Naive Bayes classifier is a statistical model based on the Bayes Rule and conditional independence (Russell & Norvig, 2010). The Neural Network model attempts to model a brain and its individual neurons, making a prediction based on the connections between these neurons (Russell & Norvig, 2010). Despite the diversity of machine learning algorithms, Kotsiantis (2007) describes that no one algorithm is ultimately superior compared to the others as they all have their strengths and disadvantages. Polikar (2006) suggests a technique called the ensemble technique or boosting technique which uses multiple learning algorithms to obtain higher predictive capabilities than could be obtained from any of the constituent learning algorithms individually.

In conclusion, taking into account the difficulty of accurately diagnosing certain clinical disorders and the progress that NLP and ML are making. It would certainly makes sense to attempt to use techniques from both NLP and ML to solve not just identifying clinical disorders in children, but a whole range of other real world problems too.

2.2 Relevant Research

This section describes other key researches that play an influential role in affecting the choices made for this study. Most of these in some way or another are related to the applications of NLP and ML on real world problems, or provide a backbone to a reasoning to a particular feature used in this study.
Some good models for this study are papers by Gabani et al. (2011), Sharma et al. (2013) and Kotsiantis (2007). Gabani et al.'s (2011) paper covers a similar study on the classification task of language disorders in children. One key difference between this paper and Gabani et al.'s (2011) is that we include an ensemble method learner, which would be further discussed later in the machine learning section. The papers by Sharma et al. (2013) and Kotsiantis (2007) both give an extensive review on supervised machine learning techniques and their various characteristics. Polikar's (2006) paper on ensemble methods for classification tasks

also discusses the benefits of ensemble based methods such as the ability to take a few weaker classifiers and combine their predictive abilities to give a more accurate classification. This is the reason why we decided to include ensemble methods for this study.

3 Project Requirements

It is hypothesized that through various NLP techniques and training of various ML algorithms we could potentially discover effective forms of screening measures for the assessment of child language disorders. For this project, the specific types of language disorders that we consider are the two widely studied neurodevelopmental conditions amongst young children – Specific Language Impairment (SLI) (Leonard, 1997) and Autism Spectrum Disorder (ASD) (Johnson et al., 2007).

3.1 Functional Requirements

The project is expected to produce classifications using both NLP and ML techniques. This will be achieved by feeding the program with text corpora of a conversation recorded between a child and the investigator(s) from the CHILDES database. The expected output would be a classifier that when passed features of a text-based corpus, is able to predict a label for the data set with an acceptable accuracy rating. The classification task is defined to be a multi-class problem with aim of categorizing children as either TD, SLI or ASD.

The program will pre-process the data sets before any feature measurements, to ensure that all the data sets can be evaluated equally. The pre-processing tasks include removing any unwanted data in the data sets, ensuring that the text is uniformly formatted for unbiased feature measurement. The program is also expected to perform computational tasks including feature measurement calculation, writing extracted features into a file, feature extraction and classifying in an acceptable amount of time, assuming a reasonable size of input data is given.

3.2 Non-Functional Requirements

Below is a list of non-functional requirement for the project:

- The project is required to produce a computational model with adequate accuracy for predicting and diagnosing children with language related disorders. The resulting model could potentially be incorporated into current clinical assessments for the early detection of neurodevelopmental conditions in children in order to enable quick and effective treatment of any disorders. An acceptable baseline accuracy for the classifiers should be the accuracy gained from randomly guessing a classification based on the largest population of the data sets used. For this study we used a total 258 texts, split into 101 SLI children, 33 with ASD and 124 labeled as TD. Therefore if the classifier would randomly guess a label based on the largest population, its expected baseline accuracy would be $\frac{population_{TD}}{population_{Total}} = 48\%$.

- All data sets used should not infringe on any copyright or licensing policies. Only free and legal data sets that are available will be utilized.
- Compatible with the Windows, Mac and Linux environments as long as the hardware and software requirements stated are met.
- In terms of scalability, the program includes feature measurements that utilizes language models built on the available text corpora. A large amount of text for this model to work with will increase it's predictive accuracy as it can better compute features that are distinct to each class. In terms of adding additional features, it is possible for a user to do that given that they have a sufficient background knowledge of python.
- Numerous methods have been implemented to ensure that the program is reliable. What this means is that the program has been designed to avoid errors such as a text corpus being invalid or it has irrelevant data. One of these examples is that we implement a smoothing technique for ratios whereby we add 1 to the numerator and denominator to ensure that the denominator of any fraction would not be zero.

4 Project Plan

4.1 Overview

This projects aims to utilize both NLP and ML techniques in order to bring to light any symptomatic linguistic features of children with language impairment related disorders. The 4 key questions to be tackled in the project include:

- Are we able to identify children with language disorders identifiable as compared to children with normal language development through text-based classification?
- Are children with language disorders distinguishable from its subtypes – SLI or ASD?
- Which of the linguistic patterns, including morphology, phonology, syntax and semantics, are indicators of some form of language disorder?
- Which the machine learning models is best at identifying and classifying children with language disorders compared to their counterparts?

4.2 Risk Analysis

It should be noted that ASD represents 3 separate pervasive developmental disorder types. These are Asperger syndrome, autistic disorder and pervasive developmental disorder-not otherwise specified(American Psychiatric Association, 1994). Even though there are different classifications for each of the 3 disorders, this study considers a child with any of the 3 disorders as a child with ASD and identifies the child as one under a single classification.

Symptoms of ASD and SLI disorders are highly variant and it is not uncommon for clinically diagnosed cases of either disorder to display attributes that are not commonly associated with the disorders at all(Johnson et al., 2007). As a result we cannot be a hundred percent certain of any of the results produced by the machine learners. This paper merely suggests a potential method to identify the two disorders in hopes that it will assist further research into this study in the future.

Other concerns include the lack of a balanced data sets as the ones used in this project do not have a balanced distribution of population mass across the labels. Johnson et al. (2007) also suggests that ASD tends to be exhibited more on male children than compared to female children. The lack of a gender balanced data set is another risk hindering the project. Unfortunately, there are no workarounds to this risk as the amount of quality texts is not high, therefore we cannot afford to be too stringent on the selection criteria for available texts.

Due to the cooperative nature of CHILDES talkbank, many of the corpora in their database have inconsistent transcription practices. One example of this might be that certain clinicians can choose to only use a subset of the transcription symbols, whereas others might use all of them. (For more information on transcriptions rules, refer to The TalkBank Project Tools for Analyzing Talk – Electronic Edition (MacWhinney, 2016)).Due to this, the scores for certain feature measurement might be biased.

4.3 Data sets

This project evaluates text transcripts from the CHILDES database(MacWhinney, 2000). The transcripts include:

1. Hargrove data set(Hargrove et al., 1986): A data set containing transcripts of children with SLI. This data set contains the transcribed texts of 6 children with SLI, aged 3-6 years old. Despite the low number of children, each child has multiple texts transcribed. This data set was recorded when Hargrove et al. (1986) conducted a study on spontaneous speech changes associated with therapy hiatus, making it good for a classification task, as any spontaneous speech that is recorded will definitely be distinct compared to the other classes.

2. Rollins data set(Rollins, 1999): This data set contains multiple texts of 5 children with ASD, aged between 2 to 3 years old. This data set was recorded when (Rollins, 1999) (1999) studied the pragmatic achievements of children with ASD that have at least 1 year of education in pre-school. Because the corpus was originally collected to describe pragmatic skills in children with autism from the prelinguistic to early one-word stage, a good deal of nonverbal information is transcribed. This helps diversify our training set, and provide more concrete indicators of the features of children with ASD. One downside however, is the lack of a control group. Without a control group we have no way knowing how normal (or unusual) the children are behaving. Our best workaround is to pick other data sets with children around the same age.

3. Conti-Ramsden 4 data set (Wetherell et al., 2007): A data set with transcripts of children with SLI and their TD controls. This data set contains 19 children with SLI and a control group of 99 children, aged between 13-15 years old. One of the traits that makes this data set desirable is that they have a large proportion of females. As mentioned earlier, the lack of female texts might be a concern however this corpus

contains the texts of 66 female children, 61 classified as TD and 5 as SLI. Besides that, the large amount of controls is also desirable. Despite the older age of the participants in this data set, this data set is one of the largest available on the CHILDES talkbank. This data set contains text of the children in two activities, a narrative of the wordless picture story book 'Frog, Where Are You?' (Mayer, 1969) and a spontaneous narrative. This study only considers the spontaneous narrative task as the repeated words in the narrative task might give similar measurements for all classes.

4. Nadig data set (Bang & Nadig, 2015): A data set including transcripts of English and French speaking children with ASD and their TD controls. For the purpose of this project we do not consider the French speaking children. The data set contains the transcribed texts of 12 English speaking children with ASD and 25 controls, aged 3 to 7. Despite the possible bilingualism of some of the children, we still selected this data set as they have a good amount of controls and additional texts of ASD children, as we have a low count of ASD texts.

Despite only 1 data set with an acceptable sample size - Conti-Ramsden 4 with 19 SLI children and 99 controls, data sets with smaller sample sizes are selected because of the lack of other relevant data sets. The XML as well as plain text versions of the corpus are used as the XML version already has a precleaned format that is suitable for feature evaluation. The usage of the plain text corpus is included because there are certain transcribed material in the plain text files such as annotations that cannot be accessed through the XML files.

4.4 Resource Requirements

The resources required to run this project is a computer capable of running a Python 3.0 or above environment and has all the required Python toolkits and packages installed. The Python packages used in this project are: matplotlib(Hunter, 2007) version 1.5.2, nltk(Bird et al., 2009) version 3.2.1, NumPy(van der Walt et al., 2011) version 1.11.1, scikit-learn(Pedregosa et al., 2011) version 0.18, spaCy(Honnibal, 2013) version 0.101.0 and SciPy(Jones et al., 2001) version 0.17.1. All of which are available for free on the internet.

- nltk is a Python toolkit used for NLP and classification tasks(Bird et al., 2009). It provides useful functions such as Part-of-Speech (POS) tagging, tokenization, feature extraction that analyses a text corpus and more.

- matplotlib(Hunter, 2007), NumPy(van der Walt et al., 2011) and SciPy(Jones et al., 2001) are mathematical packages for Python which facilitate graphing and advanced mathematical calculations.

- scikit-learn is a machine learning library for Python that has tools to implement various machine learning techniques(Pedregosa et al., 2011).

- spaCy is a Python library that has advanced NLP features. It's features are mostly similar to nltk and is designed for industrial use and is faster than nltk. (Honnibal, 2013). spaCy is included as they offer additional NLP related features on top of what nltk already has that will complement our classification tasks. Additionally, this research also uses the subject object extraction function (Honnibal, 2015) written by the author of spaCy to compute features.

5 Methodology

5.1 Internal Design & Software Architecture

The program is split into 3 phases: feature measurement, the training phase and the prediction phase. In the feature measurement phase, the program will take the cleaned data sets of child text corpora and tally the various features. An extensive list of the features will be covered in the section below. These features are stored in an array with their corresponding data set labels appended to the end where the array takes the form of n_samples by n_features. This phase prepares the data for the next phase. The next phase of the program is the training phase. In this phase the machine learners will be fed the measured features and their corresponding label as inputs. The machine learner will attempt to categorize the labels based on the significance of each given feature in the data set in the training phase. The first round of training and prediction would utilize all of the calculated features. After the first round of training and predictions, the program will calculate the most indicative features and generate a smaller set of features. The training and prediction phases are repeated using this smaller set of more indicative features, and the results will be reported.
The figure 1 is the class diagram of the program. Even though the program is modeled after

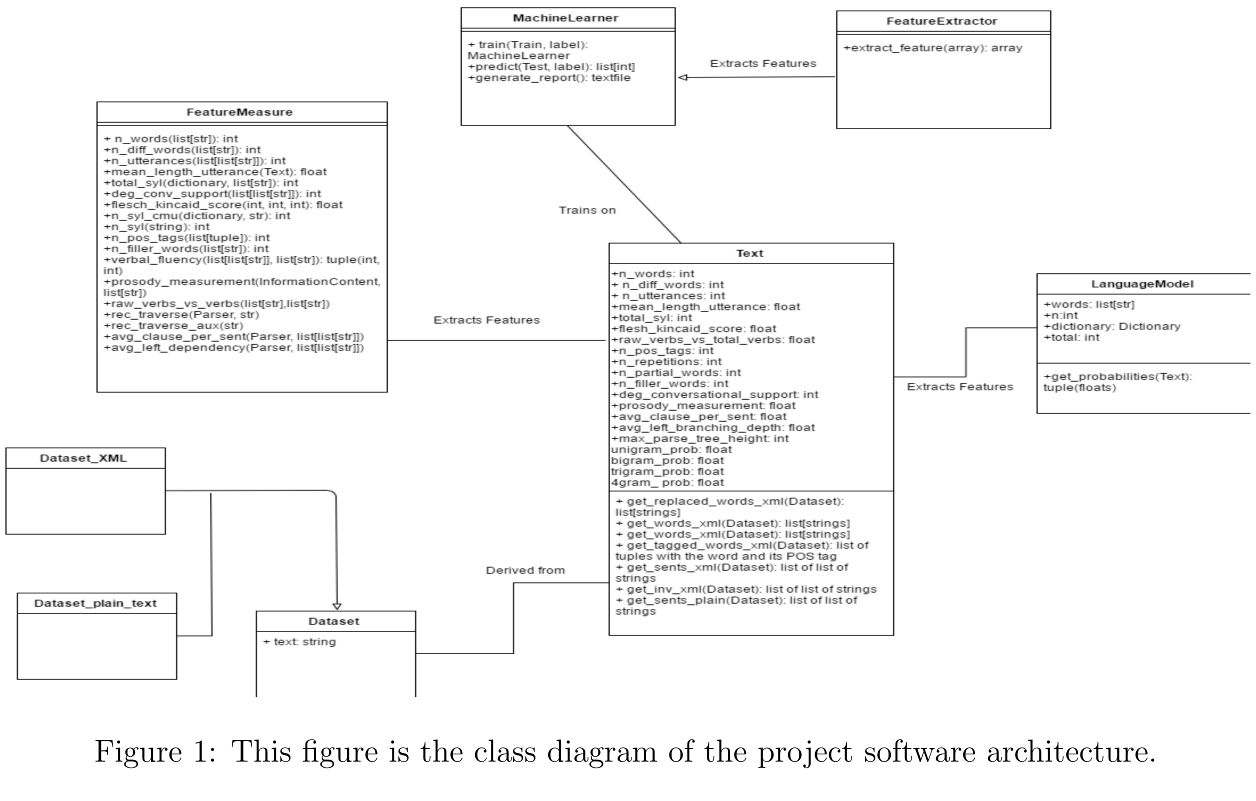

MachineLearner
+ train(Train, label): MachineLearner
+predict(Test, label): list[int]
+generate_report(): textfile
FeatureExtractor
+extract_feature(array): array
Extracts Features
FeatureMeasure
+ n_words(list[str]): int
+n_diff_words(list[str]): int
+n_utterances(list[list[str]]): int
+mean_length_utterance(Text): float
+total_syl(dictionary, list[str]): int
+deg_conv_support(list[list[str]]): int
+flesch_kincaid_score(int, int, int): float
+n_syl_cmu(dictionary, str): int
+n_syl(string): int
+n_pos_tags(list[tuple]): int
+n_filler_words(list[str]): int
+verbal_fluency(list[list[str]], list[str]): tuple(int, int)
+prosody_measurement(InformationContent, list[str])
+raw_verbs_vs_verbs(list[str],list[str])
+rec_traverse(Parser, str)
+rec_traverse_aux(str)
+avg_clause_per_sent(Parser, list[list[str]])
+avg_left_dependency(Parser, list[list[str]])
Trains on
Extracts Features
Text
+n_words: int
+ n_diff_words: int
+ n_utterances: int
+mean_length_utterance: float
+total_syl: int
+flesh_kincaid_score: float
+raw_verbs_vs_total_verbs: float
+n_pos_tags: int
+n_repetitions: int
+n_partial_words: int
+n_filler_words: int
+deg_conversational_support: int
+prosody_measurement: float
+avg_clause_per_sent: float
+avg_left_branching_depth: float
+max_parse_tree_height: int
unigram_prob: float
bigram_prob: float
trigram_prob: float
4gram_ prob: float
+ get_replaced_words_xml(Dataset): list[strings]
+ get_words_xml(Dataset): list[strings]
+ get_words_xml(Dataset): list[strings]
+ get_tagged_words_xml(Dataset): list of tuples with the word and its POS tag
+ get_sents_xml(Dataset): list of list of strings
+ get_inv_xml(Dataset): list of list of strings
+ get_sents_plain(Dataset): list of list of strings
LanguageModel
+words: list[str]
+n:int
+dictionary: Dictionary
+total: int
+get_probabilities(Text): tuple(floats)
Extracts Features
Dataset_XML
Dataset_plain_text
Dataset
+ text: string
Derived from

Figure 1: This figure is the class diagram of the project software architecture.

this class diagram, note that this class diagram is only an approximate representation of the classes in the code and not the actual implementation of it.

5.2 Features

In the machine learning environment, we provide the machine learner with features inspired by similar research as well as additional features derived from NLP. We intend to be able to discern new information that can help clinical practitioners identify and accurately diagnose language related disorders on children. A similar study by Gabani et al.(2011) uses similar features to the ones we use however we have added additional features which we deem to be significant. Below is the comprehensive list of all features that will be used for the classification tasks.

1. Total Number of Words(TNW):

   This feature is commonly used feature when profiling a child's linguistic capabilities for clinical purposes. Children with some form of LI generally have a lower TNW than a child classified as TD(Gabani et al., 2011).

2. Degree of Conversational Support:

   Children LI generally struggle with sustained auditory attention and verbal working memory and hence require constant encouragement and prompts from the interrogator to continue and complete a narration (Duinmeijer et al., 2012). These conversations can turn idle if unprompted by the interrogator and the child may not complete a narration task.
   Children with ASD display a lack of joint attention (JA) at younger ages when communicating(Johnso et al., 2007) and also require a significant amount of prompting to keep their attention on to the conversation at hand. Johnson et al.(2007) mentions that children with Autism disorder tend to exhibit a lack of speech at times or are incapable to follow simple instructions, which will lead to a higher number of prompts from an interrogator. For this feature, the number of all utterances and the total number of words spoken by the interrogator are measured.

3. Ratio of total number of raw verbs compared to the total number of verbs:

   The usage and retention rate of newly learned was explored by Anderson (2001). In his research he suggested that children with LI tend to have difficulty with verb morphology which leads to the incorrect usage of inflected verbs. This ratio will allow us to compare a child's verb morphology.

4. Total Number of POS tags:

   This feature is similar to TNW where we expect children with LI to have a lower Total Number of POS tags.

5. Number of Different Words (NDW):

   As mentioned in the risk analyses subsection and in (Johnson et al., 2007; Gabani et al., 2011), vocabulary based measures might be biased against certain groups of children. However we expect children with LI to have a lower NDW than compared to TD children.

6. Verbal Fluency

   Children with LI have a delay in language learning hence they should have a lower verbal mastery of the a language compared to their age-matched peers. We evaluate word and phrase repetitions, partial words and filler words as they represent a lower verbal fluency. Word and phrase repetition is measured when a child repeats a word or phrase more than once throughout a sentence and throughout the corpus. Johnson et al. (2007) suggests that ASD afflicted children tend to display intense interest towards limited topics of speech and would persistently talk about them in great detail. Measuring the word repetitions alone might not be an strong indicator of ASD however when coupled with other measurements it could be significant.Partial words and filler words are simply summed throughout the text corpus.

7. Language Models:

   Language Models (LM) are a statistical representation of word sequences. They play huge role in NLP related tasks such as POS tagging and natural language generation (Gabani et al., 2011). LMs calculate the probability of a sequence of words in a sentence. For example given a sequence of words $W =< w_1, w_2, \ldots, w_n >$, then its probability would be:

$$p(W) = \prod_{i=1}^{n} < w_i|w_1, \ldots, w_{n-1} > \tag{1}$$

   For this feature we consider N-gram LMs of orders 1-4 as increasing the order too high comes with a heavy computational processing cost. An N-gram LM with an order k is:

$$(W) = \prod_{i=1}^{n} < w_i|w_{i-k+1}, w_{i-k+2}, \ldots, w_{i-1} > \tag{2}$$

   The statistics extracted by language modeling techniques are commonly known as n-gram statistics (Madnani, n.d.). An n-gram is defined as an overlapping sequence of n words. An example of this is the sentence "I have a pet frog". From this sentence, the following sets of n-grams can be obtained (the cardinality of each set is indicated by the corresponding number in parentheses on the right):

   (a) 1-grams (unigrams): I, have, a, pet, frog (5)
   (b) 2-grams (bigrams): I have, have a, a pet, pet frog (4)
   (c) 3-grams (trigrams): I have a, have a pet, a pet frog (3)
   (d) 4-grams: I have a pet, have a pet frog (2)
   (e) 5-grams: I have a pet frog (1)

   The motivation behind n-grams is best explained by applying the chain rule of probability to decompose the desired probability of a sentence. This rule states that the probability of our sentence "I have a pet frog" can be calculated as follows:

$$p(Ihaveapetfrog) = p(I) * p(I|have) * p(a|Ihave) * p(pet|Ihavea) * p(frog|Ihaveapet) \tag{3}$$

Madnani (n.d.) says that equation 3 represents the probability of the sentence as the product of smaller probabilities. The first term in the equation represents the probability of unigram "I". The second term represents the conditional probability of the bigram "I have", which is equivalent to the probability of the word "have" give the previous word "I". The third term is the conditional probability of the trigram "I have a" and so on. Unfortunately, the higher the order of n-grams, the harder it is to get its conditional probability because of data sparsity (Madnani, n.d.). Therefore we utilize approximation to obtain the probabilities of each sentence, which gives us the equation:

$$p(Ihaveapetfrog) \approx p(I) * p(have) * p(a) * p(pet) * p(frog) \tag{4}$$

Similarly, the approximation for the sentence probability using bigrams is:

$$p(Ihaveapetfrog) \approx p(I|have) * p(have|a) * p(a|pet) * p(pet|frog) \tag{5}$$

As we increase the order of the n-grams, we can get better approximations of the sentence probability compared to its true value (Madnani, n.d.). The probabilities for word sequences are obtained from the training data using the cross-validation method, which is to exclude the set that is being compared from the LM. Children with ASD occasionally display the tendency to include "pop-up" words in a sentence. These "pop-up" words are words that are not relevant to the subject or context of the sentence. The probabilities measured by LMs will be affected by these "pop-up" words and other self-invented non-words that may be indicative of a LI disorder. We this research we consider word sequence probabilities of unigrams, bigrams, trigrams and 4-grams. The average probabilities for all sentences in a text for each of the 4 n-grams is considered as a single feature to be used during training and prediction.

8. Mean Length of Utterance (MLU):

   MLU is a commonly evaluated trait in LI studies and many other studies (Rice et al., 2010; Yoder et al., 2011). It continues to prove to be one of the more significant indicators of LI in a child as they tend to produce shorter utterance on average than compared to TD children.

9. Average number of syllables per word and average number of clauses per sentence:

   Like TNW and NDW before this, we expect the average number of syllables per word and average number of clauses per sentence to be lower for children with LI as both these indicators word and sentence complexity. The average number of clauses per sentence was used as a sentence complexity measure by (Gabani et al., 2011) instead of sentence parsers. Gabani et al.(2011) uses the sum of number of non-inflected words in a sentence as an estimate to the number clauses in a sentence with the assumption that each utterance should contain at least 1 clause.

10. Flesch-Kincaid Score:

    Kincaid et al.'s (1975) word has been pivotal in the field of linguistic assessment and is used many times in related researches. The Flesch-Kincaid score is a readability index that scores how hard it is to comprehend a text. The score is defined as:

$$0.39(\frac{totalwords}{totalsentences}) + 11.8(\frac{totalsyllables}{totalwords}) - 15.59 \tag{6}$$

Where a score of 2.0 would mean that a second grader would be able to comprehend the text.

11. Prosody:

    Prosody refers to the method of stressing and the intonation words when delivering speech. Children with ASD display odd conversational behavior such as responding in a way that is not listener responsive or maintaining a unique prosody that tends to disregard listener needs(Johnson et al., 2007). Children with ASD may display the inability to discern and judge the conversational intent of others. Therefore we consider unique and persistent prosody to be a potential marker for a language disorder. McCann and Peppe's (2003) and Wevrick's (1986) research are some that study the prosody of children with ASD. One particular trait mentioned was Echolalia. Echolalia refers to speech in stock phrases that simply involves copying and repeating another person's utterance word for word (Ducenne & Winsler, 2006). For this research, we attempt to measure a child's prosody through the repetition of similar nouns. Due to uneven length of each text, and sometimes the absence of nouns, we only consider the first 10 nouns of each text. Using the WordNet interface on the nltk library, we get the average similarity scores between the set of nouns of each text. The similarities are calculated using the Resnik similarity function, which gives a score denoting how similar two words are, based on the Information Content (IC) of the Least Common Subsumer (most specific ancestor node) (Resnik & Mellish, 1995). The IC is generated based on all the available texts.

12. Parsing

    Fraser et al. (2014) uses features derived from a parse tree in their machine learner. Features such as tree height and Yngve depth of a tree and mean depth are measured. The Yngve depth quantifies the extent of left-branching phrases rather than right-branching phrases contained in the syntactic structure of a sentence. This provides a syntactic complexity comparison metric (Yngve, 1960; Sampson, 1997).

5.3 Machine Learning

Supervised machine learning is a reasoning algorithm that utilizes externally supplied instances to produce a general hypothesis about a classification label, which it then uses to make future predictions on unlabeled data sets(Kotsiantis, 2007; Sharma et al., 2013). The main aim of a supervised learner is to generate a computational model of class label distributions in terms of predictive features. The derived classifier will later be used to predict labels for unlabeled test sets. Both Kotsiantis(2007) and Sharma et al.(2013) have suggested that there is no one superior ML algorithm compared to the rest. There will be instances where one algorithm may perform better than another.

Therefore using the scikit-learn Python toolkit for ML and data mining, this project will attempt use the ensemble method to give the best prediction capabilities to the machine learner. That is to use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone(Polikar, 2006). This section reviews the ML techniques that will be used in this project.

1. Decision Trees: The decision tree classifier builds a tree based on the most indicative features from a set of features, to predict a label, it traverses the decision tree until it reaches a node with no further children. The choice of which branching path it traverses is based on the specific feature of the data set provided (Russell & Norvig, 2010).

2. Instance-based learning - k-nearest neighbors: the k-Nearest-Neighbors method uses a membership method based on a predefined k nearest neighbors. The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these (Pedregosa et al., 2011).

3. Naive Bayes classifier: The Naive Bayes classifier is a statistical model based on the Bayes Rule and conditional independence (Russell & Norvig, 2010). The Neural Network model attempts to model a brain and its individual neurons, making a prediction based on the connections between these neurons (Russell & Norvig, 2010).

4. Neural Network model - Multi-layer Perceptron (MLP): An MLP is the computational modeling of a brain. It consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next layer. Except for the input nodes, each node is a processing element (or a neuron) with a nonlinear activation function. MLP utilizes a supervised learning technique called back-propagation to train the network (Rosenblatt, 1961).

5. Support Vector Machines (SVM) with kernel function: SVMs utilize the notion of a “margin” — either side of a hyperplane that separates two data classes. Maximizing the margin creates the largest possible distance between the separating hyperplane and the instances on either side of it. This results in reducing the upper bound on the expected generalization error, which enables a greater accuracy in predictions. (Kotsiantis, 2007).

6. Soft Voting Ensemble Method: Polikar (2006) suggested a technique called the ensemble technique or boosting technique which uses multiple learning algorithms to obtain higher predictive capabilities than could be obtained from any of the constituent learning algorithms individually. The soft voting ensemble method (also known as weighted average probabilities (Pedregosa et al., 2011)) gives the class label as an argument of the maxima of the sum of predicted probabilities. Simply put, the predicted class probabilities for each classifier are collected, and subsequently multiplied by the classifier weight, and then averaged. The final class label is simply obtained from the class label with the highest average probability.

5.4 Feature Extraction

Once the learners have been prepared, we use a forest of decision trees to extract the significant features. This works the same way how a decision tree works, only on a larger scale, where the significance of each feature is calculated. Once the features have been extracted the machine learners are retrained using the new set of features.

6 Test Plan

6.1 Test Coverage

Our testing for the project covers the functional as well as non-functional requirements and checks if they have been fulfilled. Additionally, we will also conduct unit, integration, functional and performance testing on the program. This is also covered in depth in the test report.

1. Unit testing - We test the individual functions of the program in this stage. Most notably are the functions that measure the features. Various inputs will be considered when we conduct unit testing to ensure that the functions are able to handle a diverse range of events.

2. Integration testing - We also test the system as a whole to ensure that it is working as expected.

3. Functional testing - We consider the usability of the program in this stage of testing and we test its compatibility in different environments. Most notably are the three major operating systems. In the functional testing phase we test the program in Windows, Mac and Linux operating systems.

4. Performance testing - In this phase of testing we execute the code multiple times to obtain the average execution times of the program as well as the average classifier accuracy scores. The averages are calculated over 50 iterations of program execution.

6.2 Classification Evaluation Metrics

Of the sample data sets prepared, 70% of the samples will be used to train and test the machine learner. The remaining 30% would be for actual predictions.
After training a classifier, we also test its prediction accuracy in terms of Accuracy, Precision, Recall and F-measure. Where a positive result refers to a positive ASD or SLI categorization and a negative result as a TD categorization. The following equations define each of the testing measures:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{10}$$

In addition to the methods above, the support will also be tallied. The support represents the number of actual labels in a specific class. The scoring metrics mentioned above will be tallied using the stratified k folds cross validation method. This means that the test data set will be divided into k number sets, each set will have the ratio between classes maintained. Each of the k sets will then be tested against the other $k-1$ sets. This ensures a good estimate and it also helps prevent overfitting. A k value of 4 is selected.
A confusion matrix will also be tabulated. By definition a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i but predicted to be in group j (Pedregosa et al., 2011). Once training on a machine learner is complete, the results of the testing metrics will be tallied and compared.

6.3 Parameter Settings

Each of the machine learning classifiers can be configured with different parameter setting. Whereby different parameters work better for different classification tasks.
This section briefly covers the different parameter settings applied to the classifiers as well as the best performing settings. The optimal parameters are finalized based on testing the classifier with them and scoring its prediction accuracy.

- Decision Tree - settings tested:

  1. criterion - the function that measures the quality of a split. Criterion functions tested are the Gini impurity and entropy information gain.
  2. splitter - the strategy used to select a split point at each node. Available options are random or best split.
  3. max_features - the number of features considered when looking for the best split. Options tested are $log_2(n\_features)$, $sqrt(n\_features)$ and $n\_features$.
  4. max_depth - the maximum depth when building the decision tree. Values applied are max_depth = 5-30.

  Optimal parameters for the decision tree are: criterion = entropy, splitter = best split, max_features = $sqrt(n\_features)$ and max_depth = 10.

- K-Nearest-Neighbors:

  1. k - represents the number of neighbors to consider. Values tested are 3 ¡ k ¡ 20.
  2. weights - weights of each neighbor. Uniform or by distance.
  3. algorithm - Algorithm used to compute nearest neighbor. Options are ball tree, kd tree, brute force or auto.

  Optimal parameters: k = 8, weights = uniform and algorithm = auto

- Naive Bayes: No parameters available.
- Multilayer Perceptron Neural Network:
  1. solver - solver used for weight optimization. Parameters are lbfgs which is the optimizer in the family of quasi-Newton methods, stochastic gradient descent 'adam' and the stochastic gradient descent proposed Kingma and Ba (2015).
  2. alpha - the regularization term penalty. Values tested are 0.00001 to 0.0001.
  3. hidden layer sizes - represents the size of the hidden layers. Values tested are 15-150.
  4. tolerance - which represents the tolerance to consider when the algorithm is not improving and how early it will force a convergence. Values are 0.00001 - 0.0001

  Parameters selected for the MLP Neural Network are: solver = lbfgs, alpha = 0.00005, hidden layer size = 50 and tolerance = 0.00001
- Support Vector Machine:
  1. C - represents the penalty parameter of the error term. Values tested are 0.01 - 2.0
  2. kernel - the kernel function that is used in the algorithm. Linear, polynomial, radial basis, sigmoid or precomputed functions.
  3. class weights - the weights of the class labels. Used when making a prediction. Balanced or a manually entered dictionary of weights for the class labels.
  4. decision function shape - whether to use a one-vs-rest or a one-vs-one decision function shape
  5. probability - whether to enable probability estimates.

  Optimal parameter settings for SVM are: $C = 2.0$, kernel = linear, class weights = balanced, decision function shape = one vs rest and probability = True.
- Soft Voting Ensemble Method - the weights assigned to this classifier are $[1, 2, 1, 2, 3]$ which corresponds to the order in which they appear in this section. Those weights were chosen based on the individual performance of each machine learning algorithm. SVM has the highest weight because its accuracy was the highest. On the other hand decision trees and Naive Bayes methods have low weights because their individual performance was bad.

7 Results

The mean accuracy is calculated from only one instance of predictions as a rough estimate of its accuracy. A stratified k fold cross validation (CV) method is used to score the classifiers as mentioned in section 6.2 to get an averaged score. After the classification task has computed once, we extract the significant features and retrain the machine learners with the new set of features. We compare the results before and after the feature extraction to each other as well as to the baseline performance.

7.1 Decision Tree Results

Pre-feature extraction:

Class	precision	recall	f-measure	support
SLI	0.77	0.96	0.86	28
ASD	0.56	0.62	0.59	8
TD	0.88	0.71	0.79	42
avg/total	0.81	0.79	0.79	78

Mean accuracy: 0.79
Accuracy from CV scoring: 0.77 (+/- 0.07)

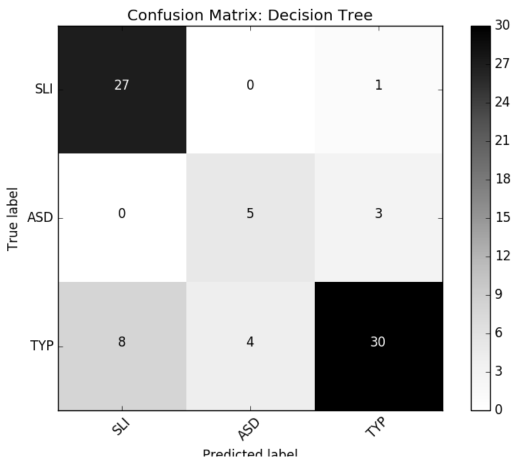


Figure 2: This figure shows the confusion matrix of the decision tree classifier before feature extraction.

Post-feature extraction:

Class	precision	recall	f-measure	support
SLI	0.83	0.89	0.86	28
ASD	0.50	0.75	0.60	8
TD	0.92	0.79	0.85	42
avg/total	0.84	0.82	0.83	78

Mean accuracy: 0.82
Accuracy from CV scoring: 0.82 (+/- 0.06)

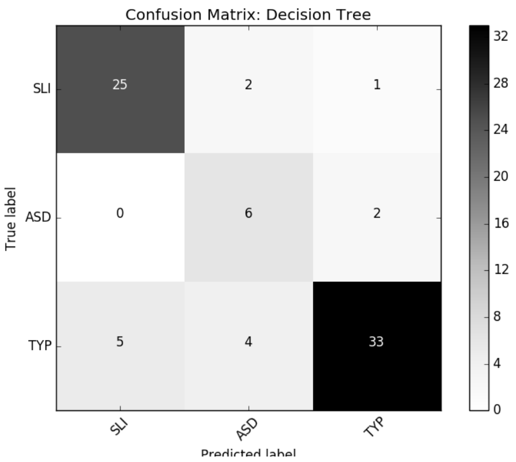


Figure 3: This figure shows the confusion matrix of the decision tree classifier after feature extraction.

7.2 K-Nearest Neighbors Results

Pre-feature extraction:

Class	precision	recall	f-measure	support
SLI	0.81	0.89	0.85	28
ASD	0.80	0.50	0.62	8
TD	0.88	0.88	0.88	42
avg/total	0.85	0.85	0.84	78

Mean accuracy: 0.85
Accuracy from CV scoring:0.78 (+/- 0.05)

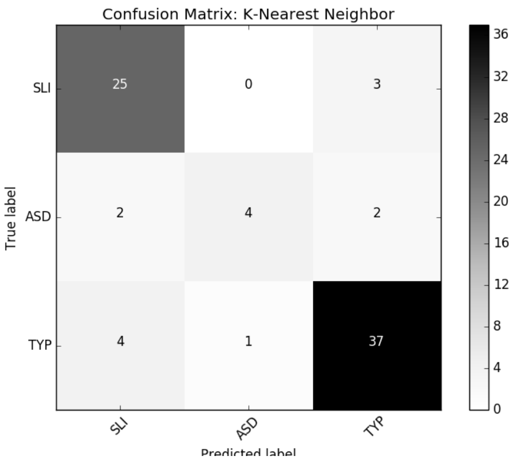


Figure 4: This figure shows the confusion matrix of the k-nearest-neighbors classifier before feature extraction.

Post-feature extraction:

Class	precision	recall	f-measure	support
SLI	0.83	0.89	0.86	28
ASD	1.00	0.62	0.77	8
TD	0.88	0.90	0.89	42
avg/total	0.88	0.87	0.87	78

Mean accuracy: 0.87
Accuracy from CV scoring:0.82 (+/- 0.03)

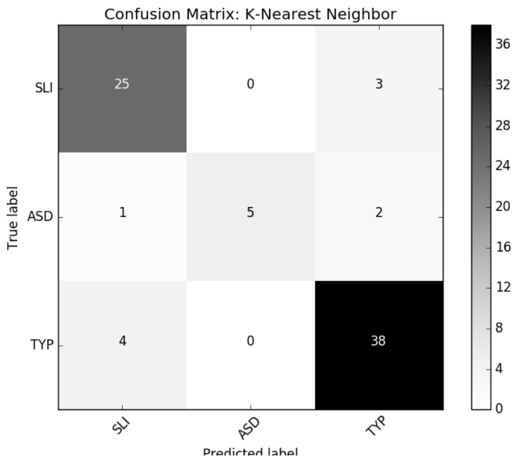


Figure 5: This figure shows the confusion matrix of the k-nearest-neighbors classifier after feature extraction.

7.3 Naive Bayes Results

Pre-feature extraction:

Class	precision	recall	f-measure	support
SLI	0.85	0.82	0.84	28
ASD	0.53	1.00	0.70	8
TD	0.94	0.81	0.87	42
avg/total	0.87	0.83	0.84	78

Mean accuracy: 0.83
Accuracy from CV scoring: 0.85 (+/- 0.03) Post-feature extraction:

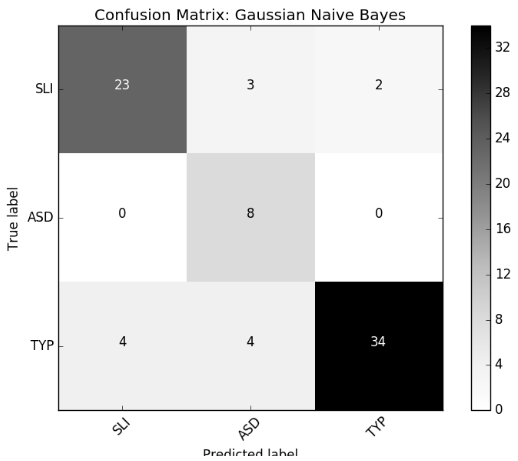


Figure 6: This figure shows the confusion matrix of the Naive Bayes classifier before feature extraction.

Class	precision	recall	f-measure	support
SLI	0.79	0.96	0.87	28
ASD	0.53	1.00	0.70	8
TD	0.97	0.67	0.79	42
avg/total	0.86	0.81	0.81	78

Mean accuracy: 0.81
Accuracy from CV scoring: 0.83 (+/- 0.02)

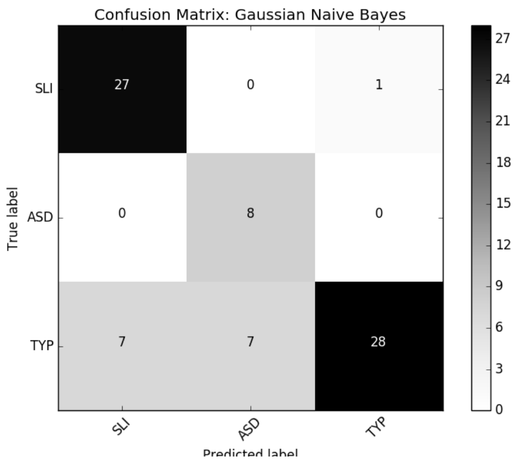


Figure 7: This figure displays the confusion matrix of the Naive Bayes classifier after feature extraction.

7.4 Multilayer Perceptron Neural Network Results

Pre-feature extraction:

Class	precision	recall	f-measure	support
SLI	0.59	0.68	0.63	28
ASD	0.44	0.50	0.47	8
TD	0.70	0.62	0.66	42
avg/total	0.64	0.63	0.63	78

Mean accuracy: 0.63
Accuracy from CV scoring: 0.85 (+/- 0.06)

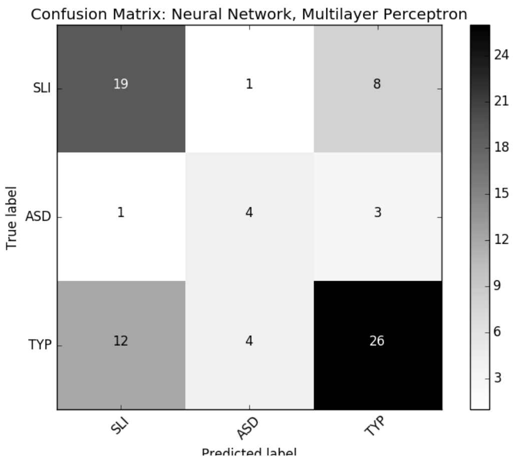


Figure 8: This figure presents the confusion matrix of the Multilayer Perceptron Neural Network classifier before feature extraction.

Post-feature extraction:

Class	precision	recall	f-measure	support
SLI	0.87	0.93	0.90	28
ASD	0.86	0.75	0.80	8
TD	0.93	0.90	0.92	42
avg/total	0.90	0.90	0.90	78

Mean accuracy: 0.90
Accuracy from CV scoring: 0.70 (+/- 0.21)

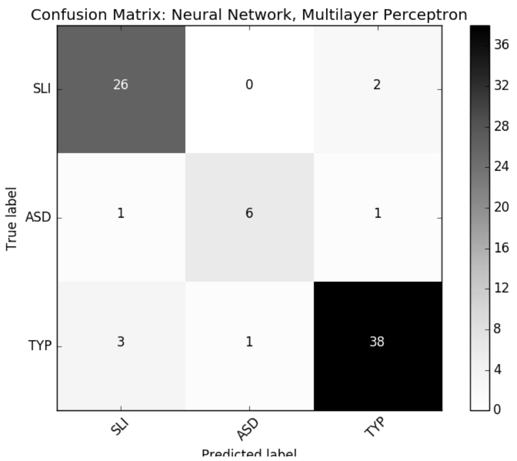


Figure 9: This figure shows the confusion matrix of the Multilayer Perceptron Neural Network classifier after feature extraction.

7.5 Support Vector Machine Results

Pre-feature extraction:

Class	precision	recall	f-measure	support
SLI	0.93	0.89	0.91	28
ASD	0.58	0.88	0.70	8
TD	0.90	0.83	0.86	42
avg/total	0.88	0.86	0.86	78

Mean accuracy: 0.86
Accuracy from CV scoring: 0.87 (+/- 0.03)

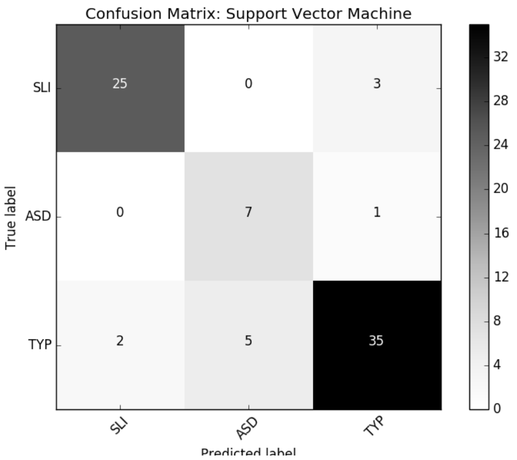


Figure 10: This figure displays the confusion matrix of the decision tree classifier before feature extraction.

Post-feature extraction:

Class	precision	recall	f-measure	support
SLI	0.86	0.89	0.88	28
ASD	0.47	1.00	0.64	8
TD	0.94	0.71	0.81	42
avg/total	0.86	0.81	0.82	78

Mean accuracy: 0.81
Accuracy from CV scoring: 0.87 (+/- 0.08)

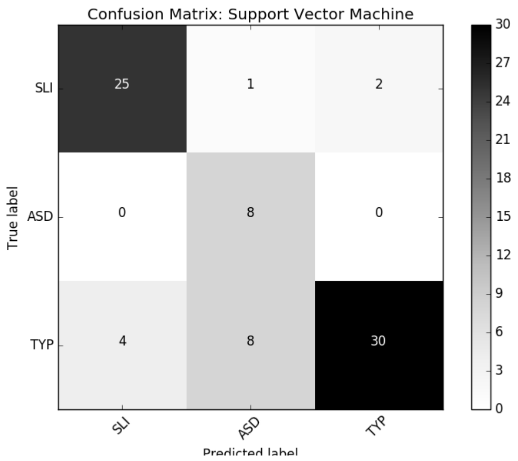


Figure 11: This figure presents the confusion matrix of the SVM classifier after feature extraction.

7.6 Soft Voting Ensemble Method Results

Pre-feature extraction:

Class	precision	recall	f-measure	support
SLI	0.83	0.86	0.84	28
ASD	0.71	0.62	0.67	8
TD	0.86	0.86	0.86	42
avg/total	0.83	0.83	0.83	78

Mean accuracy: 0.83
Accuracy from CV scoring: 0.90 (+/- 0.00) Post-feature extraction:

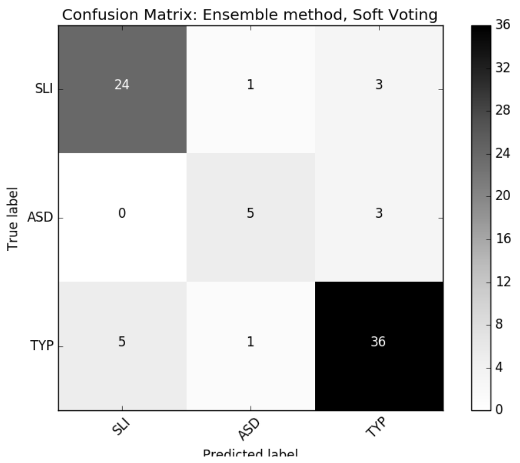


Figure 12: This figure shows the confusion matrix of the Soft Voting Ensemble Method classifier before feature extraction.

Class	precision	recall	f-measure	support
SLI	0.93	0.93	0.93	28
ASD	0.54	0.88	0.67	8
TD	0.92	0.81	0.86	42
avg/total	0.88	0.86	0.87	78

Mean accuracy: 0.86
Accuracy from CV scoring: 0.82 (+/- 0.04)

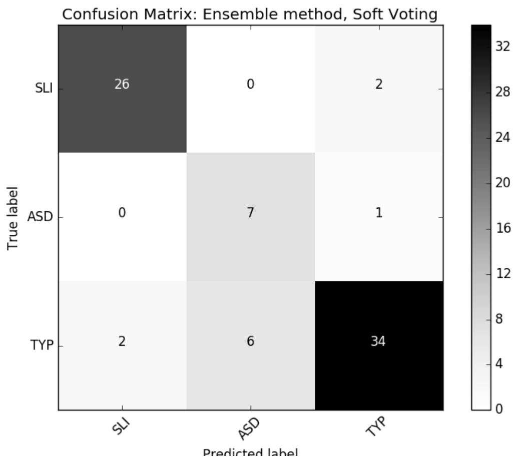


Figure 13: This figure presents the confusion matrix of the Soft Voting Ensemble Method classifier after feature extraction.

The decision plots of all 6 classifiers are also attached in the appendix. In order for easier illustration of the decision points of each classifier, only two features are used. These two features are the Flesch Kincaid score and average left branching depth of the parse trees.

7.7 Feature Analysis

With the goal of identifying features indicative of either ASD or SLI, we ran the extractor on the provided set of features to obtain a smaller, more indicative set of features. The figure 25 shows the analysis of the features with 2 of the more significant features (Flesch Kincaid score and average left branching depth).

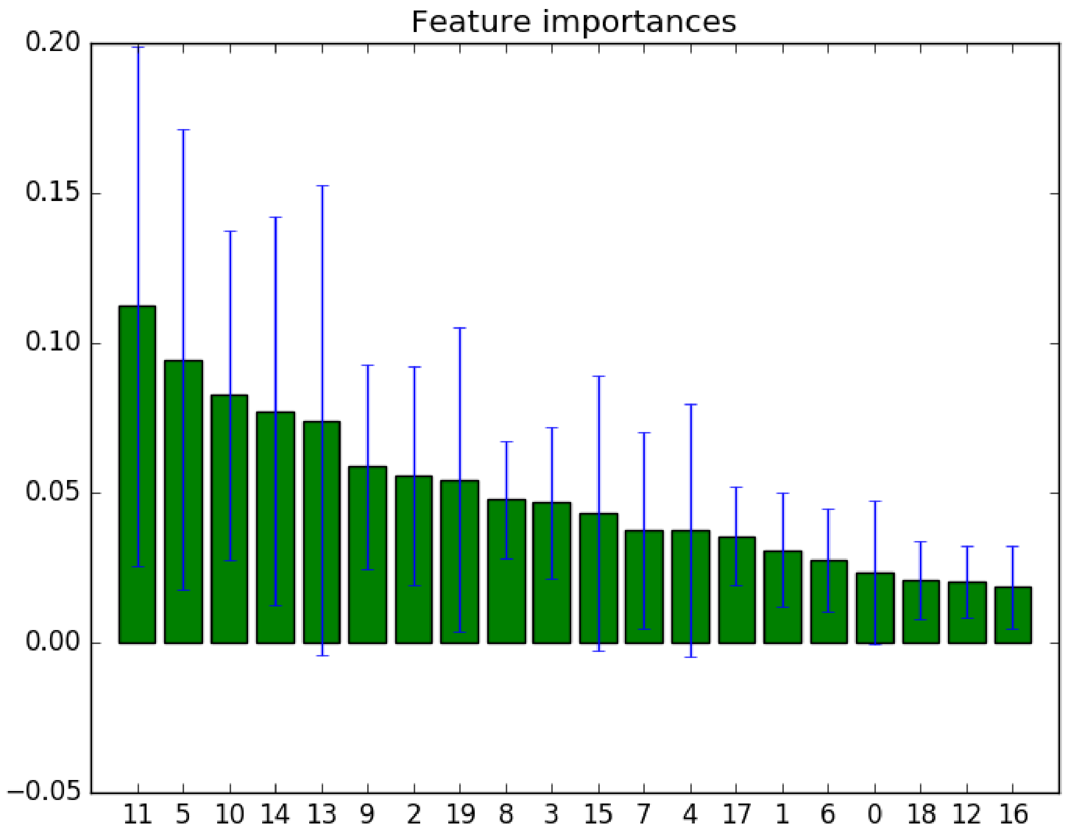


Figure 14: This figure feature importance scores after feature extraction.

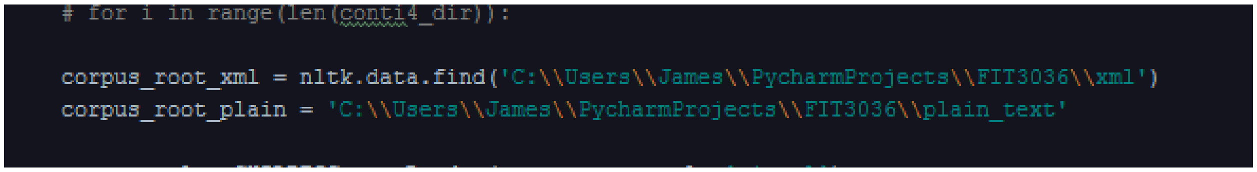
Figure 14 shows the feature importance ranking based on the indicative strength of each feature. The indices on the x-axis of the plot correspond to the following list of feature rankings, with their corresponding scores denoted by the value in the parentheses:

1. feature 11: degree of conversational support (0.112443)
2. feature 5: Flesch Kincaid score (0.094526)
3. feature 10: number of filler words (0.082708)
4. feature 14: average left branching depth (0.077165)

5. feature 13: average clauses per sentence (0.074029)
6. feature 9: number of partial words (0.058679)
7. feature 2: total utterances (0.055597)
8. feature 19: language model average 4-gram probability (0.054239)
9. feature 8: number of repeated words/phrases (0.047796)
10. feature 3: mean length of utterances (0.046835)
11. feature 15: max parse tree height (0.043267)
12. feature 7: number of different pos tags (0.037661)
13. feature 4: average syllables per word (0.037481)
14. feature 17: language model average bi-gram probability (0.035573)
15. feature 1: number of different words (0.030922)
16. feature 6: raw verbs vs total verbs (0.027695)
17. feature 0: total words (0.023442)
18. feature 18: language model average tri-gram probability (0.020959)
19. feature 12: prosody (0.020403)
20. feature 16: language model average uni-gram probability (0.018581)

7.8 Externally Observable Features

The feature measurement program does not take any inputs, the path directories for the text corpora files are expected to be in the same working directory as the program file. All XML text corpus are expected to be in folder called 'XML' and plain text corpora should be located in a folder called 'plain_text'. Within each of these folders, there should be three more subfolders named 'asd', 'sli' and 'typ' where the 'asd' folder contains all text of ASD labeled children, 'sli' contains text of children with SLI and 'typ' for texts of TD children. The path to text directory can be manually changed by changing this line of code in the driver_m.py file: Once the classification program is ran, a report would created and written into a file. This report includes the result data of the classification tasks such as the results covered above - the measurement metrics, the confusion matrices, the evaluated significance of features as well as the the classifier scores after feature extraction has been conducted.



```
# for i in range(len(conti4_dir)):

corpus_root_xml = nltk.data.find('C:\\Users\\James\\PycharmProjects\\FIT3036\\xml')
corpus_root_plain = 'C:\\Users\\James\\PycharmProjects\\FIT3036\\plain_text'
```

Figure 15: This figure displays where the path directory to the text database can be changed. After the features have been calculated, their values will be stored in an output file called 'output_file'.

7.9 Performance

For feature measurements, the dominating value is the n where n is the number of all words in the entire collection of data sets. This is because the program would have to loop through each word in a corpus to first obtain its data in a cleaned format. An example of this would be to get all the words without annotations and so on. During the feature measurements phase, most features would have to iterate over all words in a text to be able to measure the feature. An example of this would be the modeling of the language model, where it calculates the occurrences of all words that appear in the entire collection of data sets. A number of other feature measurements have computational complexities higher than O(m), where m is the number of words in a single text. However, this is negligible as our data sets are of a large enough size so that n ¿ m. Therefore the computational complexity of feature measurements is O(n) where n is the total number of words over all the data sets. The space complexity would be O(n) too as n additional space would be required to store the words while the features are calculated.
Most machine learning algorithms do not have fixed complexities, as they are highly dependent on input data and implementation. We will attempt to justify the computational complexity of the program in this section.

1. Decision Tree - The complexity of training the decision tree is dependent on its implementation, for example the entropy based information gain method for selecting splits would incur an O(length(features)) at each node. As it would have to iterate over all available features to find the one with the best gains. However the random split method would only incur an O(1) time complexity as it only needs to randomly select a split. The complexity to predict using a decision tree is much smaller as it merely needs to traverse from the root of the tree down to a leaf node. That gives us a complexity of O(h) for predictions where h is the height of the decision tree. The space complexity for storing a tree would be its maximum allocated depth times the log(max_depth) for each split at a branch, giving us O(d log(d)) where d is the depth of a tree.

2. K-Nearest-Neighbor - According to Thirumuruganathan (2010) the computational complexity of the KNN is O(dn) where d is the computation time needed for each distance and n is the subsequent time to compute the distance between the new observation and each training set observation. Depending on the selected algorithm, the complexity can be either O(dn+kn) or O(dnk) where k is the number of neighbors to evaluate.

However k is often significantly smaller than n therefore we can get a complexity of O(dn).

3. Naive Bayes - The computational complexity of the Naive Bayes classifier is O(n) where n is the number of words in the entire collection of data sets. This is because the Naive Bayes classifier is simply a conditional probability model, therefore it simply loops through all data sets obtaining the conditional probability for each instance.

4. MLP Neural Network - As with most other machine learning algorithms, the complexity of the MLP Neural Network is dependent on the implementation. The approximated complexity is the number of nodes that are spread among the number of layers in the network, and for each node the additional computational cost of obtaining its activation function. A further study on its computational costs is reviewed by ORPONEN (1994).

5. SVM - Bordes (2005) and Bottou and Lin (2006) covers extensively the complexity of an SVM and its optimization. There are typically two lower bounds on the computational cost of an SVM (Bottou & Lin, 2006): O(R^3) and O(Sn). For the former, the coefficients of the R remaining free support vectors are determined by a system of R linear equations representing the derivatives of the objective function. Their calculation amounts to solving such a system. This typically requires a number of operations proportional to R^3. For the latter, merely verifying that a vector of inputs is the gradient of the dual and checking the optimality conditions gives us O(Sn) for S support Vectors n examples (Bottou & Lin, 2006).

6. Soft Voting Ensembled method - Since the voting method has to train and test all classifiers of a document of size n, it is simply the largest computational complexity among the all classifiers in the ensemble.

The total time taken to collect all the features was 1119.64 seconds and 25.04 seconds to train and test the classifiers and generate the report. The memory usage of the computer is roughly 80MB out of 494MB when idle. When executing the program, the memory usage rise up to the range of 112-187MB and peaks at 214MB out of 494MB.

8 Analysis & Discussion

It is important to note that the results generated by the program are unique each time the program is executed. This is because of the random sampling conducted during the scoring tests. Therefore, certain results obtained might differ slightly compared to this paper. The purpose of the stratified-4-fold cross validation is to attempt to get the best estimate of the score values. Based obtained results it is observed before feature extraction, we observed that all ML algorithms performed well above the baseline (48%). The lowest mean accuracy was from the MLP Neural Network (63%). Even though the MLP Neural Network has a low mean accuracy, its accuracy from cross validation of 85% +/- 6% is good. On average all ML algorithms have encouraging prediction scores. We also observed that the best performing and most consistent machine learning algorithm is the SVM with an accuracy score from cross validation of 87% +/- 3%. The rest of the other ML algorithms perform well with their mean accuracy all above 80% on average and only MLP Neural Network, decision tree and Naive Bayes occasionally dipping into the high 70%s. The best predictor is the soft voting ensemble method with a mean accuracy of 83% and an accuracy from the CV sampling of 90% +/- 0%. Based on these scores we adjusted the weight of the classifier in the ensemble method.

After feature extraction, the scores stayed mostly the same except for a huge change in ML Neural Network. Its mean accuracy rose from 63% to 90% and all its scoring metrics (precision, recall, f-measure) increased. However, the accuracy from CV scoring decreased significantly and its variance increased (from 85% +/- 6% to 70% +/- 21%). Due to the low accuracy from the CV scoring after feature extraction and the opposite before, we have to assume that the feature extraction failed to extract any significant features for the MLP. However when considering the fact that the scores for the other ML algorithms didn't change as much, we can disregard the possibility that the features extracted are **not** significant, but at the same time we cannot ascertain whether they are significant.
The features extracted and used to retrain the classifiers are:

1. feature 11: degree of conversational support (0.112443)
2. feature 5: Flesch Kincaid score (0.094526)
3. feature 10: number of filler words (0.082708)
4. feature 14: average left branching depth (0.077165)
5. feature 13: average clauses per sentence (0.074029)
6. feature 9: number of partial words (0.058679)
7. feature 2: total utterances (0.055597)
8. feature 19: language model average 4-gram probability (0.054239)
9. feature 8: number of repeated words/phrases (0.047796)

An interesting feature to note is the average left branching depth of a text. As mentioned in the methods section, this feature represents the approximated sentence complexity based on the number of left-branched subjects present in the parse tree of a sentence. The Flesch Kincaid score also seems to be an important and indicative feature of a particular classification. The most surprising however is the amount most indicative feature - degree of conversational support. There might be a possibility that certain corpora might naturally have a higher value of this feature however if this is an indicative feature it can be further research in future works.

Comparing the confusion matrices of all classifiers, it is observed that the ASD class has the lowest precision and F-measure across all classes, even after feature extraction. One of the possible reason is that the training data might be biased. As the ASD subset of the data set is the smallest, and a large subset of them come from the same people, the machine learners might not be learning the correct patterns that indicate a child with ASD. Thus giving low precision and F-measures.

9 Future work

One of the major obstacles that we faced is finding consistent and quality controlled text transcripts to use. As mentioned, many different researchers have different styles when it comes to recording texts and not all of them adhere completely to the rules in the CHAT manual MacWhinney (2016). On top of that, this study shows that the ML parameters are the dependent variables and the NLP parameters are the independent variables. what can be inferred from this is that the performance of the ML algorithms are dependent on the features derived from training data based on NLP and we are not yet able to discern if the cause of any good (or bad) performance is the result of a good ML algorithm or precise feature measurement. Because of this, we strongly recommend that future work be focused on generating a simple and effective method to record and transcribe texts digitally that is made available freely to others to improve on. This way future studies would have a consistent database to evaluate, giving them consistent scores for NLP based features. Then we can study ML algorithms to a greater precision too.

10 Conclusion

In conclusion, we accept the initial hypotheses which is that NLP and and ML techniques can be used to classify and diagnose children with sLI or ASD. We accept the hypotheses because all of the ML algorithms we employed have performed well above the baseline.

References

American Psychiatric Association. (1994). *Diagnostic and Statistical Manual of Mental Disorders* (4th ed.). Washington, DC: American Psychiatric Publishing.

Anderson, R. T. (2001). Learning an invented inflectional morpheme in Spanish by children with typical language skills and with specific language impairment. *Royal College of Speech & Language Therapists*, *36*(1), p.1–19.

Bang, J., & Nadig, A. (2015). Language learning in autism: Maternal linguistic input contributes to later vocabulary. *Autism Research*, *8*(2), 214 – 233. doi: 10.1002/aur.1440

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language processing with Python.* O'Reilly Media.

Bordes, A. (2005). Fast Kernel Classifiers with Online and Active Learning. *Journal of Machine Learning Research*, *6*.

Bottou, L., & Lin, C.-J. (2006). Support Vector Machine Solvers.

Campbell, T., Dollaghan, C., Needleman, H., & Janosky, J. (1997). Reducing Bias in Language Assessment: Processing-Dependent Measures. , *40*(3), p.519–525.

Ducenne, L., & Winsler, A. (2006). *Echolalia* (Vol. 1).

Duinmeijer, I., de Jong, J., & Scheper, A. (2012). Narrative abilities, memory and attention in children with a specific language impairment. *Royal College of Speech and Language Therapists*, *47*(5), p.542–555. doi: 10.1111/j.1460-6984.2012.00164.x

Fraser, K. C., Meltzer, J. A., Graham, N. L., Leonard, C., Hirst, G., Black, S. E., & Rochon, E. (2014, June). Automated classification of primary progressive aphasia subtypes from narrative speech transcripts. , *55*, p.43–60. doi: 10.1016/j.cortex.2012.12.006

Gabani, K., Solorio, T., Liu, Y., Hassanali, K.-n., & A. Dollaghan, C. (2011). Exploring a corpus-based approach for detecting language impairment in monolingual English-speaking children. , *53*, p.161–170. doi: 10.1016/j.artmed.2011.08.001

Haebig, E., Kaushanskaya, M., & Weismer, S. E. (2015, December). Lexical Processing in School-Age Children with Autism Spectrum Disorder and Children with Specific Language Impairment: The Role of Semantics. *Journal of Autism and Developmental Disorders*, *45*(12), 4109–4123.

Hargrove, P. M., Holmberg, C., & Zeigler, M. (1986). Changes in spontaneous speech associated with therapy hiatus: A retrospective study. *Children Language Teaching and Therapy*, *2*, 266–280.

Hassanali, K.-N. (2013). Using natural language processing for child language analysis. *ProQuest Dissertations and Theses*.

Hirschberg, J., & Manning, C. D. (2015, July). Advances in natural language processing. *Science (New York, N.Y.)*, *349*(6245), 261–266. doi: 10.1126/science.aaa8685

Honnibal, M. (2013, December). *Parsing English in 500 Lines of Python.* Retrieved from `http://spacy.io/blog/parsing-english-in-python`

Honnibal, M. (2015). *subject object extraction.*

Hunter, J. D. (2007). Matplotlib: A 2d Graphics Environment. , *9*, 90–95. doi: 10.1109/MCSE.2007.55

Johnson, C. P., Myers, S. M., & Council on Children With Disabilities. (2007, November). Identification and Evaluation of Children With Autism Spectrum Disorders. *PEDIATRICS*, *120*(5), p.1183–1215. doi: 10.1542/peds.2007-2361

Jones, E., Oliphant, T., Peterson, P., & others. (2001). *SciPy: Open Source Scientific Tools for Python.* Retrieved from `http://www.scipy.org/`

Kincaid, J. P., Fishburne, R. P., Rogers, R. L., & Chissom, B. S. (1975). Derivation of new readability formulas for Navy enlisted personnel. , 49.

Kingma, D., & Ba, J. (2015). Adam: A Method for Stochastic Optimization.

Kotsiantis, S. B. (2007). Supervised Machine Learning: A Review of Classification Techniques. , *31*, 249–268.

Leonard, L. B. (1997). *Children with specific language impairment.* Cambridge, Mass. : The MIT Press.

MacWhinney, B. (2000). *The CHILDES project: tools for analyzing talk* (3rd ed.). Mahwah, NJ : Lawrence Erlbaum.

MacWhinney, B. (2016). *The TalkBank Project tools for Analyzing Talk – Electronic Edition.* Carnegie Mellon University.

Madnani, N. (n.d.). Querying and Serving N-gram Language Models with Python. *University of Maryland, College Park*.

Martinez, A. R. (2010, March). Natural language processing. *wires computational statistics*, *2*(3), 352–357. doi: 10.1002/wics.76

Mayer, M. (1969). *Frog, Where Are You?*

McCann, J., & Peppe, S. (2003). Prosody in autism spectrum disorders: a critical review. , *38*(4), 325–350.

Nath, C., Albaghdadi, M. S., & Jonnalagadda, S. R. (2016, April). A Natural Language Processing Tool for Large-Scale Data Extraction from Echocardiography Reports. *PLoS ONE*, *11*(4). doi: 10.1371/journal.pone.0153749

ORPONEN, P. (1994, October). COMPUTATIONAL COMPLEXITY of NEURAL NETWORKS: A SURVEY.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. , *12*, 2825–2830.

Pestian, J., Nasrallah, H., Matykiewicz, P., Bennett, A., & Leenaars, A. (2010, August). Suicide Note Classification Using Natural Language Processing: A Content Analysis. *Biomed Inform Insights*, *3*, 19–28.

Polikar, R. (2006). Ensemble based systems in decision making. , *6*(3), 21–45.

Prud'hommeaux, E., & Rouhizadeh, M. (2011). Automatic detection of pragmatic deficits in children with autism.

Resnik, P., & Mellish, C. S. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence (IJCAI'95)* (pp. 448–453). organ Kaufmann Publishers Inc.

Rice, M. L., Smolik, F., Perpich, D., Thompson, T., Rytting, N., & Blossom, M. (2010, April). Mean Length of Utterance Levels in 6-Month Intervals for Children 3 to 9 Years With and Without language Impairments. *American Speech-Language-Hearing Association*, *53*, p.333–349.

Rollins, P. R. (1999). Pragmatic accomplishments and vocabulary development in pre-school children with autism. *American Journal of Speech-Language Pathology: A Journal of Clinical Practice*, *8*, 85–94.

Rosenblatt, F. (1961). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.*

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.).

Sampson, G. (1997, March). Depth in English Grammar. , *33*(1), p.131–151.

Santafe, G., Inza, I., & Lozano, J. A. (2015, December). Dealing with the evaluation of supervised classification algorithms. *Artificial Intelligence Review*, *44*(4), 467–508.

Schaeffer, J. (2016, May). Linguistic and cognitive abilities in children with Specific Language Impairment as compared to children with High-Functioning Autism. *Language Acquisition.* doi: 10.1080/10489223.2016.1188928

Sharma, S., Agrawal, J., Agarwal, S., & Sharma, S. (2013). Machine Learning Techniques for Data Mining: A Survey. School of Information Technology,UTD, RGPV, Bhopal, M.P., India..

Solorio, T. (2013). Survey on Emerging Research on the Use of Natural Language Processing in Clinical Language Assessment of Children. , *7*(12), p.633–646. doi: 10.1111/lnc3.12054

Tanana, M., Hallgren, K. A., Imel, Z. E., Atkins, D. C., & Srikumar, V. (2016, June). A Comparison of Natural Language Processing Methods for Automated Coding of Motivational Interviewing. *Journal of Substance Abuse Treatment*, 43–50. doi: 10.1016/j.jsat.2016.01.006

Taylor, L., Maybery, M., Grayndler, L., & Whitehouse, A. (2014). Evidence for Distinct Cognitive Profiles in Autism Spectrum Disorders and Specific Language. *Journal of Autism and Developmental Disorders*, *44*(1), 19–10. doi: 10.1007/s10803-013-1847-2

Thirumuruganathan, S. (2010, May). *A Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm.* Retrieved from `https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction`

van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. , *13*, 22–3–. doi: 10.1109/MCSE.2011.37

Wetherell, D., Botting, N., & Conti-Ramsden, G. (2007). Narrative in adolescent SLI: a comparison with peers across two different narrative genres. *International Journal of Language and Communication Disorders*.

Wevrick, P. (1986). The Role of Echolalia in Children with Various Disorders: An Overview and Treatment Considerations. *Human Communication Canada/Communication Humaine Canada*, *10*(3).

Whitehouse, A. J. O., Line, E. A., Watt, H. J., & Bishop, D. V. M. (2009). Qualitative aspects of developmental language impairment relate to language and literacy outcome in adulthood. , *44*(4), p.489–510. doi: 10.1080/13682820802708080

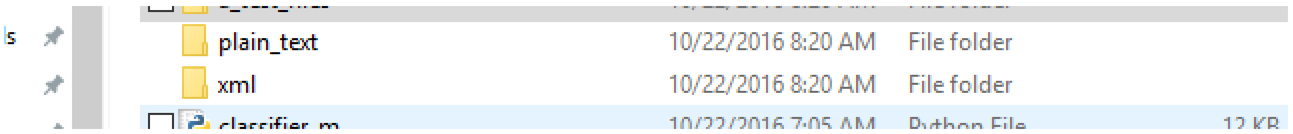
Williams, D., Payne, H., & Marshall, C. (2013). Non-Word Repetition Impairment in Autism and Specific Language Impairment: Evidence for Distinct Underlying Cognitive Causes. *Journal of Autism and Developmental Disorders*, *43*(2), 404–417. doi: 10.1007/s10803-012-1579-8

Yngve, V. H. (1960). A model and an hypothesis for language structure. , *104*, 444–466.

Yoder, P. J., Molfese, D., & Gardner, E. (2011, August). Initial Mean Length of Utterance Predicts the Relative Efficacy of Two Grammatical treatments in Preschoolers With specific Language Impairment. *American Speech-Language-Hearing Association*, *54*, p.1170–1181.

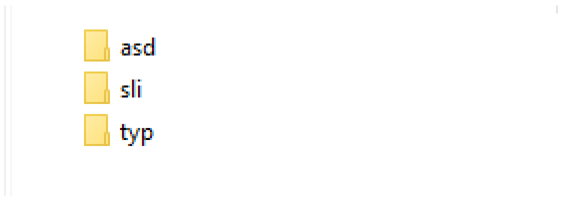
Appendix A Production and Deployment

This section includes the detailed instructions on how to setup and deploy the provided source code. After installing all the required packages as detailed in section 4.4, simply open up the project file in your preferred Python integrated development environment (IDE). We recommend using Pycharm version 4.5.5 and above as this was the IDE that was used to compile the source code. You can get Pycharm Community edition from their website for free.
Before looking at the code, you have to ensure that your data sets are ordered correctly by their type (plain text or XML) and by their labels (ASD, SLI, TYP). Where ASD denotes the folder of texts of children with Autism Spectrum Disorder, SLI is the folder of texts for children Specific Language Impairment and TYP for children classified as typically development. The following figures show how they should be arranged.



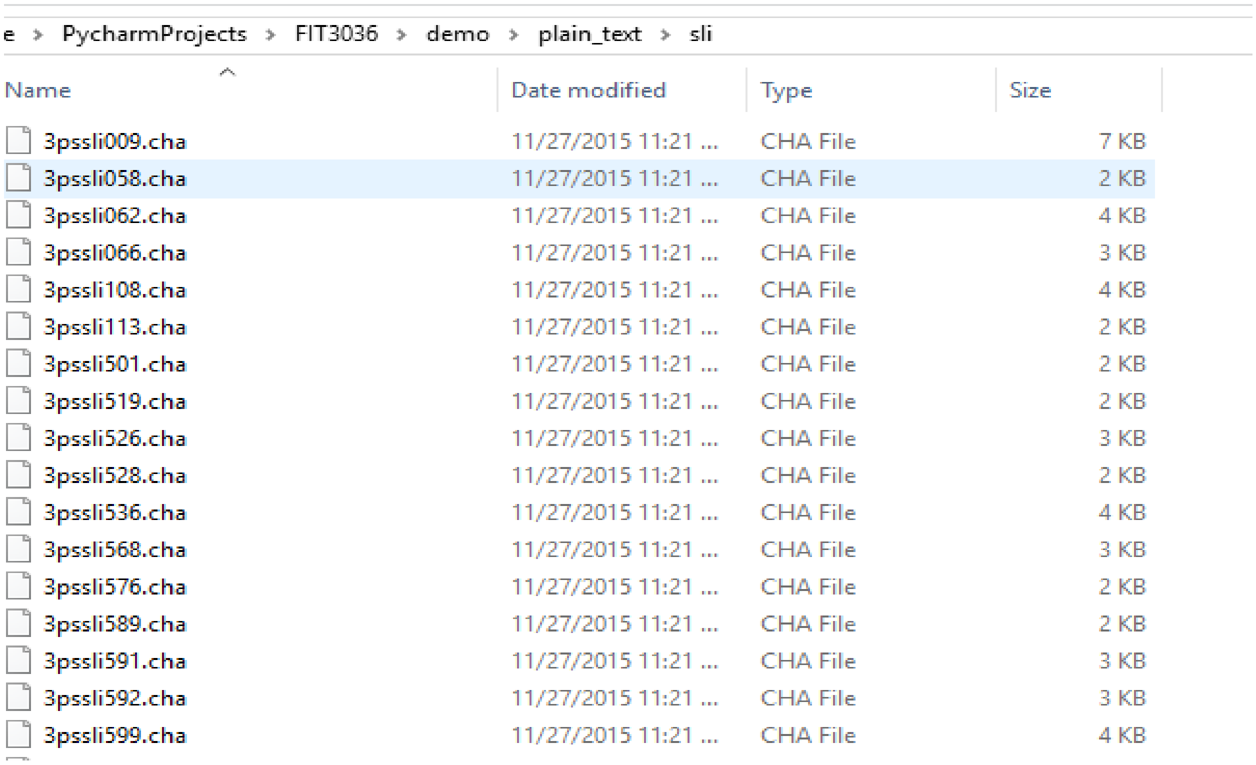
```
plain_text    10/22/2016 8:20 AM    File folder
xml           10/22/2016 8:20 AM    File folder
```

Figure 16: This figure shows how data sets should be stored for the program to read it



```
asd
sli
typ
```

Figure 17: This figure shows the folder structure within the plain_texts and xml folders for data sets


e > PycharmProjects > FIT3036 > demo > plain_text > sli

Name	Date modified	Type	Size
3pssli009.cha	11/27/2015 11:21 ...	CHA File	7 KB
3pssli058.cha	11/27/2015 11:21 ...	CHA File	2 KB
3pssli062.cha	11/27/2015 11:21 ...	CHA File	4 KB
3pssli066.cha	11/27/2015 11:21 ...	CHA File	3 KB
3pssli108.cha	11/27/2015 11:21 ...	CHA File	4 KB
3pssli113.cha	11/27/2015 11:21 ...	CHA File	2 KB
3pssli501.cha	11/27/2015 11:21 ...	CHA File	2 KB
3pssli519.cha	11/27/2015 11:21 ...	CHA File	2 KB
3pssli526.cha	11/27/2015 11:21 ...	CHA File	3 KB
3pssli528.cha	11/27/2015 11:21 ...	CHA File	2 KB
3pssli536.cha	11/27/2015 11:21 ...	CHA File	4 KB
3pssli568.cha	11/27/2015 11:21 ...	CHA File	3 KB
3pssli576.cha	11/27/2015 11:21 ...	CHA File	2 KB
3pssli589.cha	11/27/2015 11:21 ...	CHA File	2 KB
3pssli591.cha	11/27/2015 11:21 ...	CHA File	3 KB
3pssli592.cha	11/27/2015 11:21 ...	CHA File	3 KB
3pssli599.cha	11/27/2015 11:21 ...	CHA File	4 KB

Figure 18: This figure shows the texts files of the SLI children in the sli folder

It important to arrange and name your files exactly as described in figures 16, 17 and 18 as the program is written to read the files in this format. Once you have sorted out the data set directories we can proceed to look at the code.
To start things off first we have to change the directory that the code searches at for the data sets. Open the file driver_m.py and look at the line 24 of the code, displayed in figure 19.

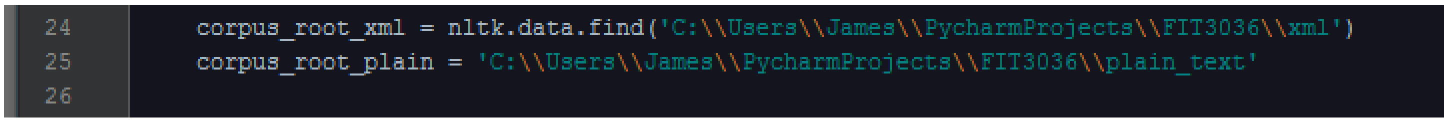

```
24        corpus_root_xml = nltk.data.find('C:\\Users\\James\\PycharmProjects\\FIT3036\\xml')
25        corpus_root_plain = 'C:\\Users\\James\\PycharmProjects\\FIT3036\\plain_text'
26
```

Figure 19: This figure shows the where to change the reference directory in the driver_m.py file

In line 24 of the code, change the directory to the location of your xml folder. Do the same for line 25. It should look something like this:

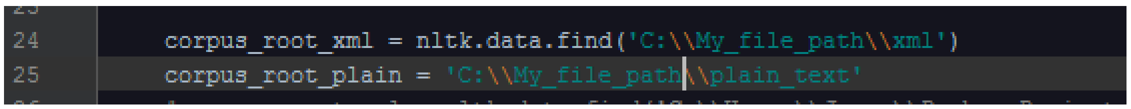
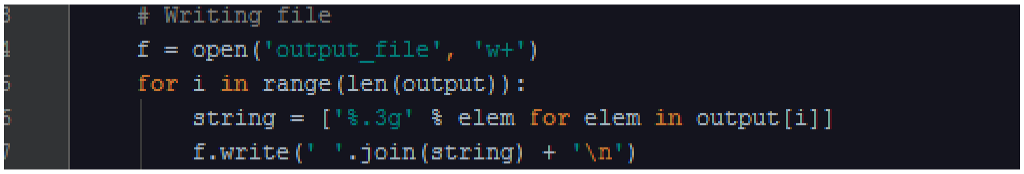

```
24        corpus_root_xml = nltk.data.find('C:\\My_file_path\\xml')
25        corpus_root_plain = 'C:\\My_file_path\\plain_text'
```

Figure 20: This figure shows the where to change the reference directory in the driver_m.py file

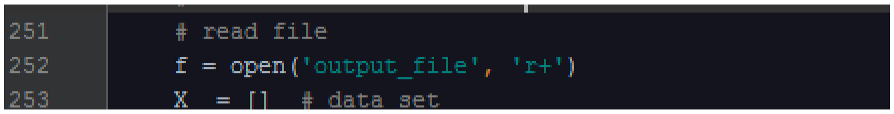
After that the program will be ready to measure the feature for the data sets you have provided. Run the driver_m.py file to get the results of the feature measurements. The code should store the results in a file named output_file. If you wish to change the location of where the features are stored you need to modify line 124 of the driver_m.py file. The default file that the features are stored in is 'output_file'.



```
    # Writing file
    f = open('output_file', 'w+')
    for i in range(len(output)):
        string = ['%.3g' % elem for elem in output[i]]
        f.write(' '.join(string) + '\n')
```

Figure 21: This figure shows where to modify the location of where the feature measurements are stored

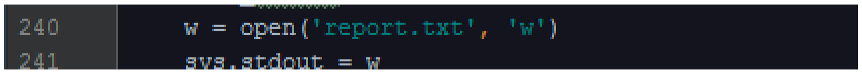
Once you have your features, open and run the classifier_m.py file to train the classifier and generate their prediction report. Be sure to change the name of the file of where the features are stored if you have changed it in classifier_m.py. This can be done at line 252 of the classifier_m.py file.



```
251     # read file
252     f = open('output_file', 'r+')
253     X  = []  # data set
```

Figure 22: This figure shows where to modify the location of where the feature measurements are searched for in classifier_m.py

Run the file and it will generate a report, which is stored in 'report.txt' by default. To change the destination of the report file modify line 240 of the file.



```
240     w = open('report.txt', 'w')
241     sys.stdout = w
```

Figure 23: This figure shows where to modify the location of where the classification task report is stored

Appendix B   Project Schedule Gantt Chart

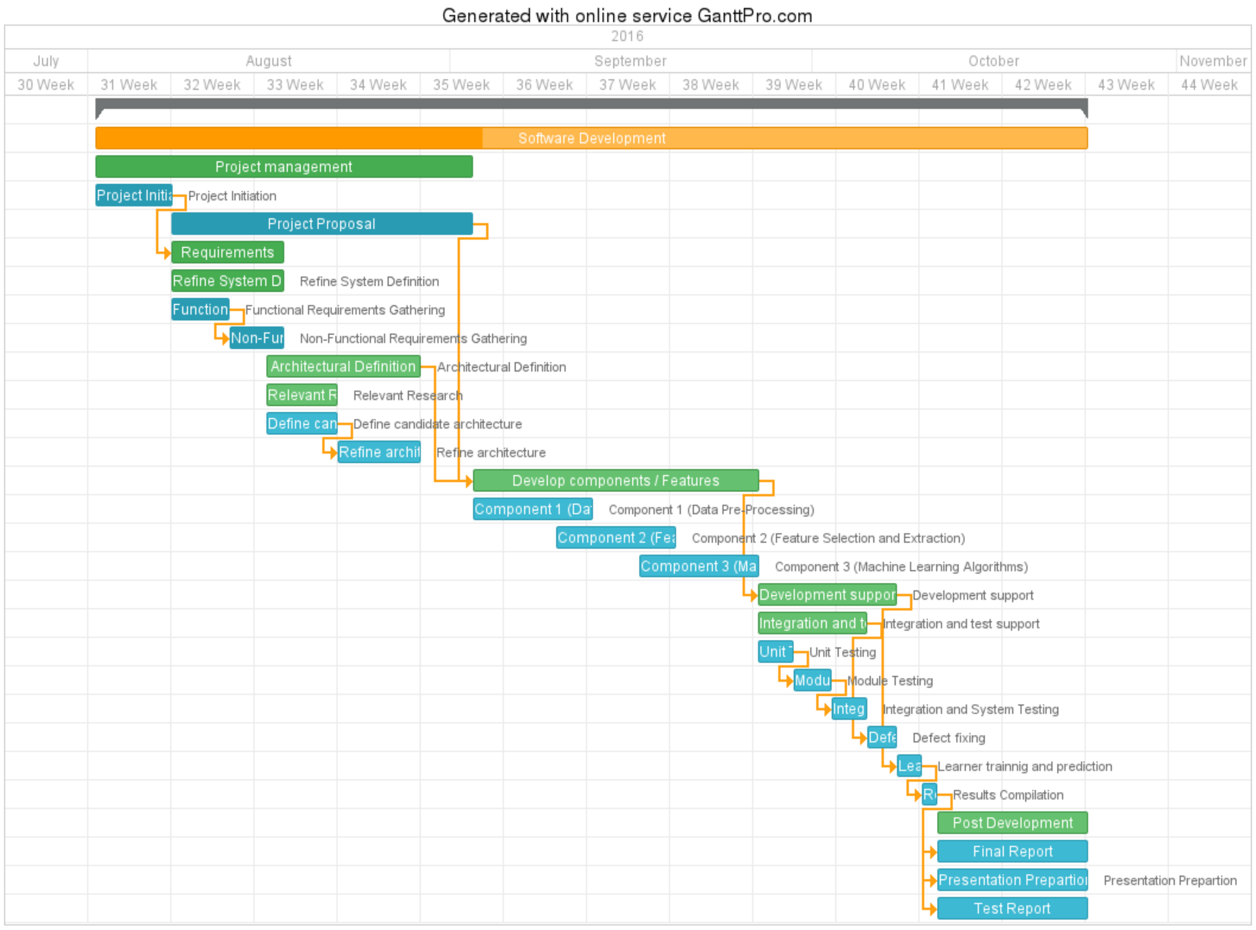


Figure 24: This figure is the Gantt chart representation of the project's schedule.

Appendix C Internal Testing

Each testing phase corresponds to a section in the test report. This section covers a brief overview of each testing phase.

- Unit test - Each function in feature measurements is tested with a set of correct inputs and incorrect inputs, where the test with correct inputs should give the desired outcome and the incorrect inputs should give an error.
- Integration test - The program is tested as two integrated modules. First is the feature measurement code that analyses text and measures features, and the second is the code that trains and test the machine learning classifiers and generates a report.
- Functional testing - The performance of the code in multiple computing environments is tested and reported.
- Performance testing - The performance of the code is tested and the execution time as well as classifier prediction scores are average over 50 iterations of executing the program.

For more information on testing refer to the test report prepared on a separate document.

Appendix D Decision Plots for Machine Learners

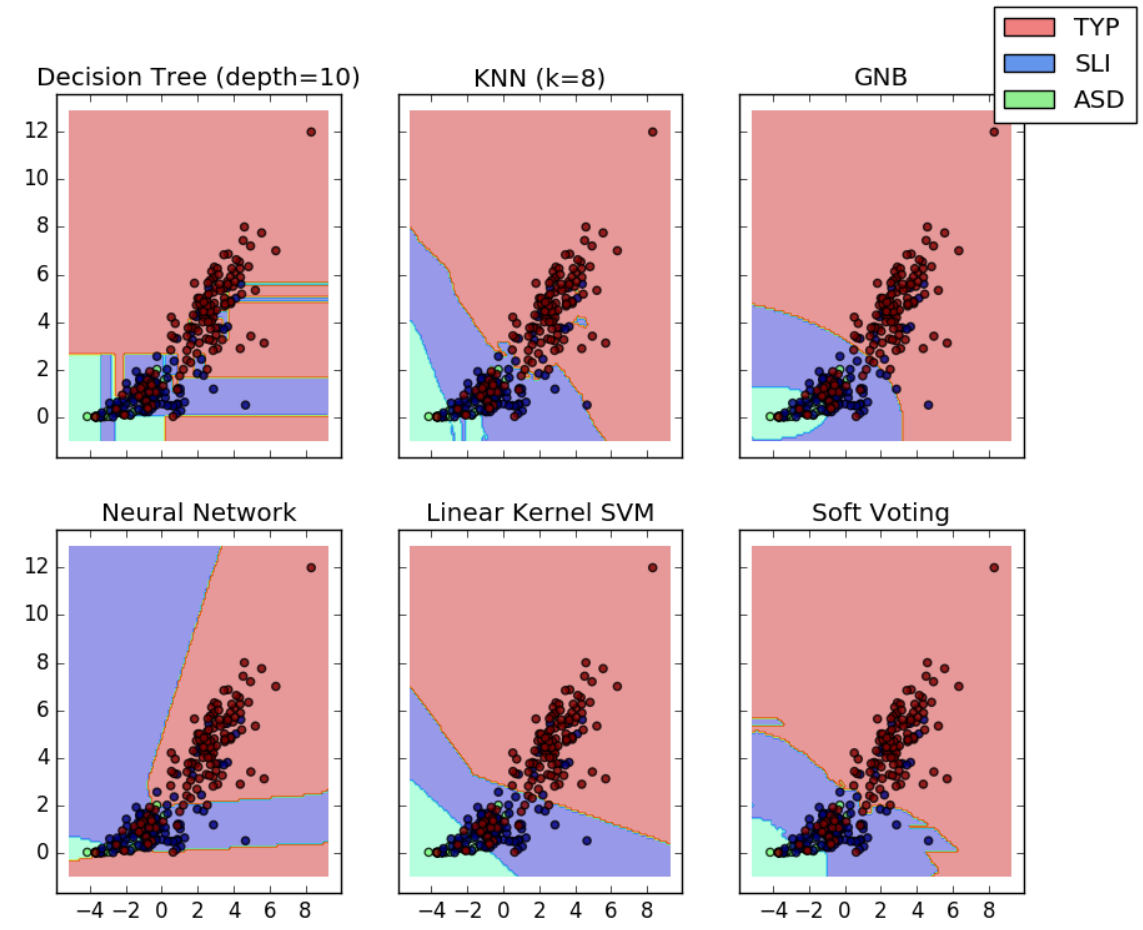


Figure 25: This figure depicts the decision regions based on 2 of the more significant features. The features used are the Flesch Kincaid score and the average left branching depth of sentence parse tree.